Linguistically-Enriched and Context-Aware Zero-shot Slot Filling

A. B. Siddique
University of California, Riverside
msidd005@ucr.edu

Fuad Jamour
University of California, Riverside
fuadj@ucr.edu

Vagelis Hristidis
University of California, Riverside
vagelis@cs.ucr.edu

ABSTRACT

Slot filling is identifying contiguous spans of words in an utterance that correspond to certain parameters (i.e., slots) of a user request/query. Slot filling is one of the most important challenges in modern task-oriented dialog systems. Supervised learning approaches have proven effective at tackling this challenge, but they need a significant amount of labeled training data in a given domain. However, new domains (i.e., unseen in training) may emerge after deployment. Thus, it is imperative that these models seamlessly adapt and fill slots from both seen and unseen domains – unseen domains contain unseen slot types with no training data, and even seen slots in unseen domains are typically presented in different contexts. This setting is commonly referred to as zero-shot slot filling. Little work has focused on this setting, with limited experimental evaluation. Existing models that mainly rely on context-independent embedding-based similarity measures fail to detect slot values in unseen domains or do so only partially. We propose a new zero-shot slot filling neural model, LEONA, which works in three steps. Step one acquires domain-oblivious, context-aware representations of the utterance word by exploiting (a) linguistic features such as part-of-speech; (b) named entity recognition cues; and (c) contextual embeddings from pre-trained language models. Step two fine-tunes these rich representations and produces slot-independent tags for each word. Step three exploits generalizable context-aware utterance-slot similarity features at the word level, uses slot-independent tags, and contextualizes them to produce slot-specific predictions for each word. Our thorough evaluation on four diverse public datasets demonstrates that our approach consistently outperforms the state-of-the-art models by 17.52%, 22.15%, 17.42%, and 17.95% on average for unseen domains on SNIPS, ATIS, MultiWOZ, and SGD datasets, respectively.

ACM Reference Format:
A. B. Siddique, Fuad Jamour, and Vagelis Hristidis. 2021. Linguistically-Enriched and Context-Aware Zero-shot Slot Filling. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Goal-oriented dialog systems allow users to accomplish tasks, such as reserving a table at a restaurant, through an intuitive natural language interface (e.g., Amazon Alexa). For instance, a user may issue the following utterance: *"I would like to book a table at 8 Immortals Restaurant in San Francisco for 5:30 pm today for 6 people"*. For dialog systems to fulfill such a request, they first need to extract the parameter (a.k.a. slot) values of the request. Slots in the restaurant booking domain include `restaurant_name` and `city`, whose values in our example utterance are "`8 Immortals Restaurant`" and "`San Francisco`", respectively. Only after all slot values are filled, the system can call the appropriate API to actually perform the intended action (e.g., reserving a table at a restaurant). Thus, the extraction of slot values from natural languages utterances (i.e., slot filling) is a critical step to the success of a dialog system.

Slot filling is an important and challenging task that tags each word subsequence in an input utterance with a slot label (see Figure 1 for an example). Despite the challenges, supervised approaches have shown promising results for the slot filling task [3, 14, 16, 24, 36, 61, 63, 65]. The disadvantage of supervised methods is the unsustainable requirement of having massive labeled training data for each domain; the acquisition of such data is laborious and expensive. Moreover, in practical settings, new unseen domains (with unseen slot types) emerge only after the deployment of the dialog system, rendering supervised models ineffective. Consequently, models with capabilities to seamlessly adapt to new unseen domains are indispensable to the success of dialog systems. Note that unseen slot types do not have any training data, and the values of seen slots may be present in different contexts in new domains (rendering their training data from other seen domains irrelevant). Filling slots in settings where new domains emerge after deployment is referred to as zero-shot slot filling [2]. `Alexa Skills` and `Google Actions`, where developers can integrate their novel content and services into a virtual assistant are a prominent examples of scenarios where zero-shot slot filling is crucial.

There has been little research on zero-shot slot filling, and existing works presented limited experimental evaluation results. To the best of our knowledge, existing models were evaluated using a single public dataset. Recently, the authors in [51] proposed a cross-domain zero-shot adaptation for slot filling by utilizing example slot values. Due to the inherent variance of slot values, this framework faces difficulties in capturing the full slot value (e.g., "`8 Immortals Restaurant`" for slot type "`restaurant_name`" in Figure 1) in unseen domains. Coach [31] proposed to address the issues in [2, 51] with a coarse-to-fine approach. Coach [31] uses the seen domain data to learn templates for the slots based on whether the words are slot values or not. Then, it determines a slot type for each identified slot value by matching it with the representation of each slot type

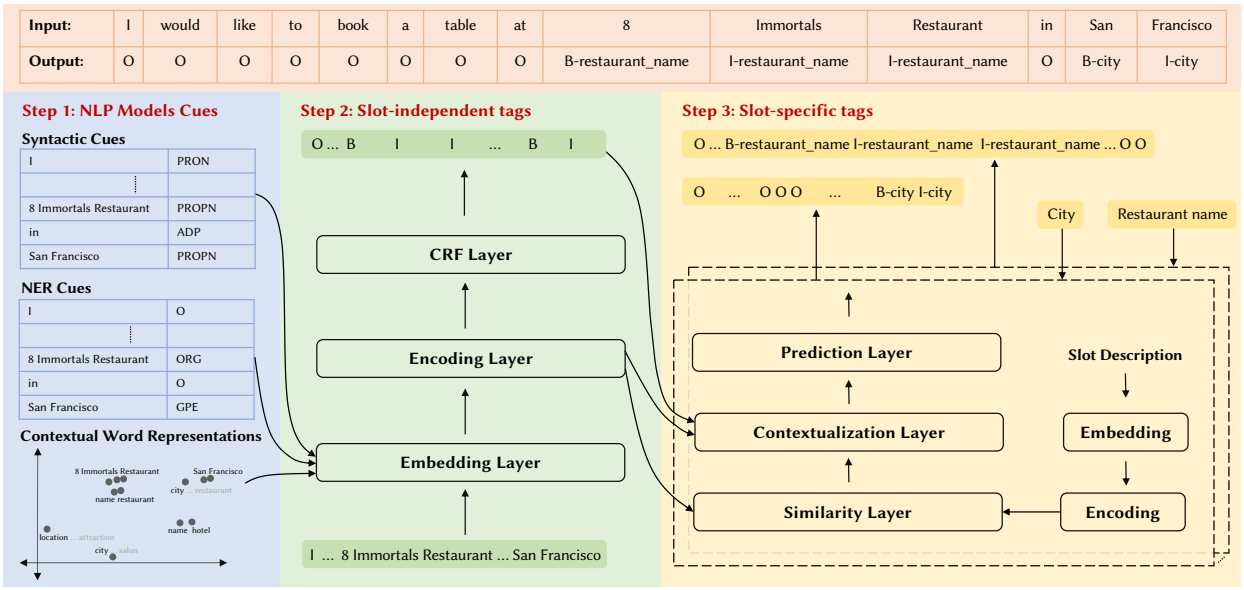


Figure 1: Overview of LEONA with an example utterance and its words' label sequence (following the IOB scheme).

description. The diversity of slot types across different domains makes it practically impossible for Coach to learn general templates that are applicable to all new unseen domains; for example, "book" and "table" can be slot values in an e-commerce domain, but not in the restaurant booking domain.

We propose an end-to-end model LEONA[1,2] that relies on the power of domain-independent linguistic features and contextual representations from pre-trained language models (LM), and context-aware utterance-slot similarity features. LEONA works in three steps as illustrated in Figure 1. Step one leverages pre-trained Natural Language Processing (NLP) models that provide additional domain-oblivious and context-aware information to initialize our embedding layer. Specifically, Step one uses (*i*) syntactic cues through part of speech (POS) tags that provide information on the possibility of a word subsequence being a slot value (e.g., proper nouns are usually slot values); (*ii*) off-the-shelf Named Entity Recognition (NER) models that provide complementary and more informative tags (e.g., geo-political entity tag for "San Francisco"); and (*iii*) a deep bidirectional pre-trained LM (ELMo) [41] to generate contextual character-based word representations that can handle unknown words that were never seen during training. Moreover, the pre-trained ELMo [41] with appropriate fine-tuning has provided state-of-the-art (SOTA) results on many NLP benchmarks [6, 18, 42, 43, 48, 54]. Combined, these domain-independent sources of rich semantic information provide a robust initialization for the embedding layer to better accommodate unseen words (i.e., never seen during training), which greatly facilitates zero-shot slot filling.

Step two fine-tunes the semantically rich information from Step one by accounting for the temporal interactions among the utterance words using bi-directional Long Short Term Memory network [19] that effectively transfers rich semantic information from NLP models. This step produces slot-independent tags (i.e., Inside Outside Beginning IOB), which provide complementary cues at the word subsequence level (i.e., hints on which word subsequences constitute slot values) using a Conditional Random Field (CRF) [25]. Step three, which is the most critical step, learns a generalizable context-aware similarity function between the utterance words and those of slot descriptions from seen domains, and exploits the learned function in new unseen domains to highlight the features of the utterance words that are contextually relevant to a given slot. This step also jointly contextualizes the multi-granular information produced at all steps. Finally, CRF is employed to produce slot-specific predictions for the given utterance words and slot type. This step is repeated for every relevant slot type, and the predictions are combined to get the final sequence labels. In our example in Figure 1, the predictions for "restaurant_name" and "city" are combined to produce the final sequence labels shown in the figure.

In summary, this work makes the following contributions:

- We propose an end-to-end model for zero-shot slot filling that effectively captures context-aware similarity between utterance words and slot types, and integrates contextual information across different levels of granularity, leading to outstanding zero-shot capabilities.
- We demonstrate that pre-trained NLP models can provide additional domain-oblivious semantic information, especially for unseen concepts. To the best of our knowledge, this is the first work that leverages the power of pre-trained NLP models for zero-shot slot filling. This finding might have positive implications for other zero-shot NLP tasks.
- We conduct extensive experimental analysis using four public datasets: SNIPS [7], ATIS [30], MultiWOZ [64] and SGD [45], and show that our proposed model consistently outperforms SOTA models in a wide range of experimental evaluations on unseen domains. To the best of our knowledge, this is first work that comprehensively evaluates zero-shot slot filling models on many datasets with diverse domains and characteristics.

[1] **L**inguistically-**E**nriched and c**ON**text-**A**ware
[2] Source code coming soon

2 PRELIMINARIES

2.1 Problem Formulation

Given an utterance with J words $\mathcal{X}_i = (x_1, x_2, \cdots, x_J)$, a slot value is a span of words $(x_e, \cdots, x_f)$ such that $0 \leq e \leq f \leq J$, that is associated with a slot type. Slot filling is a sequence labeling task that assigns the labels $\mathcal{Y}_i = (y_1, y_2, \cdots, y_J)$ to the input $\mathcal{X}_i$, following the IOB labeling scheme [44]. Specifically, the first word of a slot value associated with slot type $\mathcal{S}_r$ is labeled as B-$\mathcal{S}_r$, the other words inside the slot value are labeled as I-$\mathcal{S}_r$, and non-slot words are labeled as O. Let $\mathcal{D}_c = \{\mathcal{S}_1, \mathcal{S}_2, \dots\}$, be the set of slot types in domain c. Let $\mathcal{D}_{\text{SEEN}} = \{\mathcal{D}_1, \cdots, \mathcal{D}_l\}$ be a set of seen domains and $\mathcal{D}_{\text{UNSEEN}} = \{\mathcal{D}_{l+1}, \cdots, \mathcal{D}_z\}$ be a set of unseen domains where $\mathcal{D}_{\text{SEEN}} \cap \mathcal{D}_{\text{UNSEEN}} = \emptyset$. Let $\{(\mathcal{X}_i, \mathcal{Y}_i)\}_{i=1}^{n}$ be a set of training utterances labeled at the word level such that the slot types in $\mathcal{Y}_i$ are in $\mathcal{D}_p \in \mathcal{D}_{\text{SEEN}}$. In traditional (i.e., supervised) slot filling, the domains of test utterances belong to $\mathcal{D}_{\text{SEEN}}$, whereas in zero-shot slot-filling, the domains of test utterances belong to $\mathcal{D}_{\text{UNSEEN}}$; an utterance belongs to a domain if it contains slot values that correspond to slot types from this domain. Note that in zero-shot slot filling, the output slot types belong to either seen or unseen domains (i.e., in $\mathcal{D}_p \in \mathcal{D}_{\text{SEEN}} \cup \mathcal{D}_{\text{UNSEEN}}$). We focus on zero-shot slot filling in this work.

2.2 Pre-trained NLP Models

In this work, we utilize several pre-trained NLP models that are readily available. Specifically, we use: Pre-trained POS tagger, Pre-trained NER model, and Pre-trained ELMo. The cues provided by POS/NER tags and ELMo embeddings are supplementary in our model, and they are further fine-tuned and contextualized using the available training data from seen domains. Next, we provide a brief overview of these models.

Pre-trained POS tagger. This model labels an utterance with part of speech tags, such as PROPN, VERB, and ADJ. POS tags provide useful syntactic cues for the task of zero-shot slot filling, especially for unseen domains. LEONA learns general cues from the language syntax about how slot values are defined in one domain, and transfers this knowledge to new unseen domains because POS tags are domain and slot type independent. For example, proper nouns are usually values for some slots. In this work, we employ SpaCy's pre-trained POS tagger[3], that has shown production level accuracy.

Pre-trained NER model. This model labels an utterance with IOB tags for four entity types: PER, GPE, ORG, and MISC. The NER model provides information at a different granularity, which is generic and domain-independent. Although the NER model provides tags for a limited set of entities and the task of slot filling encounters many more entity types, we observe that many, but not all, slots can be mapped to basic entities supported by the NER model. For instance, names of places or locations are referred to as "GPE" (i.e., geo-political entity or location) by the NER model, whereas in the task of the slot filling, there may be a location of a hotel, restaurant, salon, or some place the user is planing to visit. It remains challenging to assign the name of the location to the correct corresponding entity/slot in the zero-shot fashion. Moreover, NER models can not identify all slots/entities that slot filling intends to extract, resulting in a low recall. Yet, cues from NER model are informative and helpful in reducing the complexity of the task. In this work, we employ SpaCy's pre-trained NER model[4].

Pre-trained ELMo. Pre-trained language models (i.e., ELMo) are trained on huge amounts of text data in an unsupervised fashion. These models have billions of parameters and thereby capture general semantic and syntactic information in an effective manner. In this work, we employ the deep bidirectional language model ELMo to provide contextualized word representations that capture complex syntactic and semantic features of words based on the context of their usage, unlike fixed word embeddings (i.e., GloVe [40] or Word2vec [37]) which do not consider context. Furthermore, these representations are purely character based and are robust for words unseen during training, which makes them suitable for the task of zero-shot slot filling.

2.3 Conditional Random Fields

Conditional Random Fields (CRFs) [57] have been successfully applied to various sequence labeling problems in natural language processing such as POS tagging [8], shallow parsing [50], and named entity recognition [49]. To produce the best possible label sequence for a given input, CRFs incorporate the context and dependencies among predictions. In this work, we employ linear chain CRFs that are trained by estimating maximum conditional log-likelihood. In its simplest form, it estimates a transition cost matrix of size, num_tags × num_tags, where the value at the indices [i, j] represents the likelihood of transitioning from the j-th tag to the i-th tag. Moreover, it allows enforcing constraints in a flexible way (e.g., tag "I" can not be preceded by tag "O").

3 APPROACH

Our model LEONA is an end-to-end neural network with six layers that collectively realize the conceptual three steps in Figure 1. Specifically, the Embedding layer realizes Step one and it also jointly realizes Step two together with the Encoding and the CRF layers. The Similarity, Contextualization, and Predication layers realize Step three. We briefly summarize each layer below, and we describe each layer in detail in the subsequent subsections. The Embedding layer maps each word to a vector space; this layer is responsible for embedding the words from both the utterance and the slot description. The Encoding layer uses bi-directional LSTM networks to refine the embeddings from the previous layer by considering information from neighboring words. This layer encodes utterances as well as slot descriptions. The CRF layer uses utterance encodings and makes slot-independent predictions (i.e., IOB tags) for each word in the utterance by considering dependencies between the predictions and taking context into account. The Similarity layer uses utterance and slot description encodings to compute an attention matrix that captures the similarities between utterance words and a slot type, and signifies feature vectors of the utterance words relevant to the slot type. The Contextualization layer uses representations from different granularities and contextualizes them for slot-specific predictions by employing bi-directional LSTM networks; specifically, it uses representations from the Similarity layer, the Encoding layer, and the IOB predictions produced

[3] https://spacy.io/api/annotation#pos-tagging

[4] https://spacy.io/api/annotation#named-entities

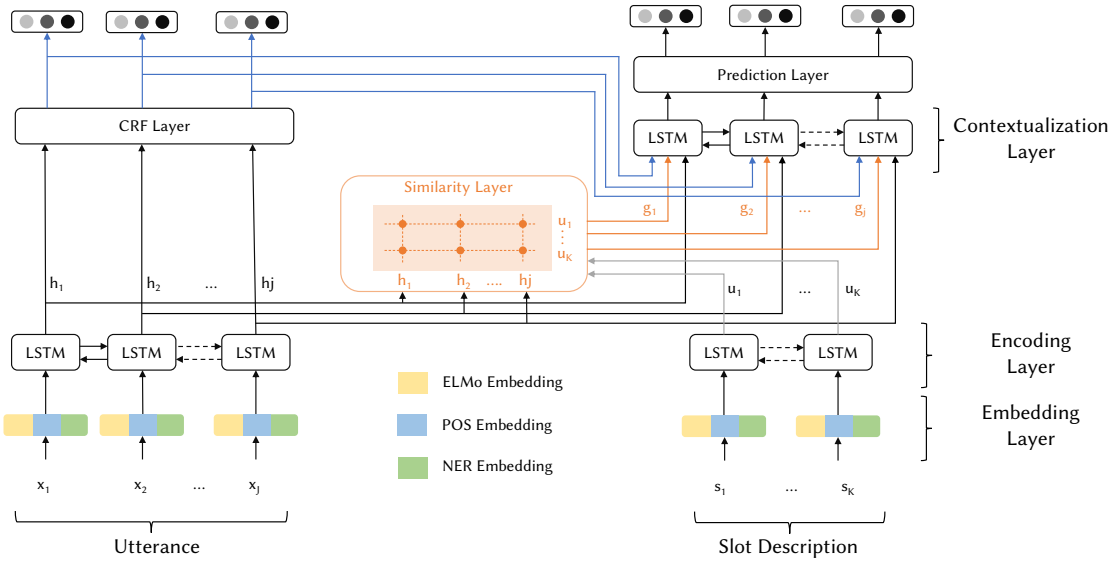


Figure 2: Illustration of the layers in our model LEONA.

by the CRF layer. The Prediction layer employs another CRF to make slot-specific predictions (i.e., `IOB` tags for a given slot type) based on the input from the contextualization layer. Note that the prediction process is repeated for all the relevant slot types and its outputs are combined to produce the final label for each word.

3.1 Embedding Layer

This layer maps each word in the input utterance to a high-dimensional vector space. Three complementary embeddings are utilized: (*i*) word embedding of the POS tags for the input words, (*ii*) word embedding of the NER tags for the input word, and (*iii*) contextual word embedding from the pre-trained ELMo model. Then, we employ a two-layer Highway Network [55] to combine the three embeddings for each word in an effective way. Highway Networks have been shown to have better performance than simple concatenation. They produces a dim-dimensional vector for each word. Specifically, the embedding layer produces $\mathcal{X} \in \mathbb{R}^{dim \times J}$ for the given utterance $\{x_1, x_2, \cdots, x_J\}$ with J words, and $\mathcal{S} \in \mathbb{R}^{dim \times K}$ for the given slot description $\{s_1, s_2, \cdots, s_K\}$ with K words. This representation gets fine-tuned and contextualized in the next layers.

3.2 Encoding Layer

We use a bi-directional LSTM network to capture the temporal interactions between input words. At time-step i, we compute hidden states for the input utterance as follows:

$$\overrightarrow{h}_i = \text{LSTM}(\overrightarrow{h}_{i-1}, \mathcal{X}_{:i})$$
$$\overleftarrow{h}_i = \text{LSTM}(\overleftarrow{h}_{i-1}, \mathcal{X}_{:i})$$

Then, we concatenate the output of the hidden states $\overrightarrow{h}_i$ and $\overleftarrow{h}_i$ to get the bi-directional hidden state representation, $h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i] \in \mathbb{R}^{2d}$. This layer produces $\mathcal{H} \in \mathbb{R}^{2d \times J}$ from the context word vectors $\mathcal{X}$ (i.e., for utterance). Essentially, every column of the matrix represents the fine-tuned context-aware representation of the corresponding word. A similar mechanism is employed to produce $\mathcal{U} \in \mathbb{R}^{2d \times K}$ from word vector $\mathcal{S}$ (i.e., for slot description).

3.3 CRF Layer

The task of the CRF layer is to predict one of three slot-independent tags (i.e., `I`, `O`, or `B`) for each word based on utterance contextual representation $\mathcal{H} = \{h_1, h_2, \cdots, h_J\}$ produced by the encoding layer. Let $\mathcal{Y}$ refer to a sequence label, and the set of all possible state sequences is C. For the given input sequence $\mathcal{H}$, the conditional probability function for the CRF, $P(\mathcal{Y}|\mathcal{H}; W, b)$, over all possible label sequences $\mathcal{Y}$ is computed as follows:

$$P(\mathcal{Y}|\mathcal{H}; W, b) = \frac{\prod_{i=1}^{J} \theta_i(y_{i-1}, y_i, \mathcal{H})}{\sum_{y' \in C} \prod_{i=1}^{J} \theta_i(y'_{i-1}, y'_i, \mathcal{H})}$$

where $\theta_i(y'_{i-1}, y'_i, \mathcal{H}) = \exp(W^T_{y',y} h_i + b_{y',y})$ is a trainable function, that has $W^T_{y',y}$ weight and $b_{y',y}$ bias matrices for the label pair (y', y).

Note that the slot-independent predictions also represent the output of `Step two`; i.e., information about utterance words at a different granularity than the initial cues from NLP models. Essentially, `Step two` learns general patterns of slot values from seen domains irrespective of slot types, and transfers this knowledge to new unseen domains and their slot types. Since it is hard to learn general templates of slot values that are applicable to all unseen domains, we do not use these slot-independent predictions to predict slot-specific tags. Instead, we pass this information to the contextualization layer for further fine-tuning.

3.4 Similarity Layer

The similarity layer highlights the features of each utterance word that are important for a given slot type by employing attention mechanisms. The popular attention methods [1, 29, 60] that summarize the whole sequence into a fixed length feature vector are not suitable for the task at hand, i.e., per word labeling. Alternatively, we compute the attention vector at each time step, i.e., attention vector for each word in the utterance. The utterance encoding $\mathcal{H} \in \mathbb{R}^{2d \times J}$ and slot description encoding $\mathcal{U} \in \mathbb{R}^{2d \times K}$ metrics are input to this layer, that are used to compute a similarity matrix $\mathcal{A} \in \mathbb{R}^{J \times K}$ between the utterance and slot description encodings. $\mathcal{A}_{jk}$ represents the similarity between j-th utterance word and k-th slot description word. We compute the similarity matrix, as follows:

$$\mathcal{A}_{jk} = \alpha(\mathcal{H}_{:j}, \mathcal{U}_{:k}) \in \mathbb{R}$$

where α is a trainable function that captures the similarity between input vectors $\mathcal{H}_{:j}$ and $\mathcal{U}_{:k}$, where $\mathcal{H}_{:j}$ and $\mathcal{U}_{:k}$ are j-th and k-th column-vectors of $\mathcal{H}$ and $\mathcal{U}$, respectively. $\alpha(h, u) = w_{(a)}^{\top}[h \oplus u \oplus h \otimes u]$, where $\oplus$ is vector concatenation, $\otimes$ is element-wise multiplication, and $w_{(a)}$ is a trainable weight vector.

The similarity matrix $\mathcal{A}$ is used to capture bi-directional interactions between the utterance words and the slot type. First we compute attention that highlights the words in the slot description that are closely related to the utterance. At time-step t, we compute it as follows: $\mathcal{U}'_{:t} = \sum_k v_{tk}\mathcal{U}_{:k}$ where $v_t = \text{softmax}(\mathcal{A}_{t:}) \in \mathbb{R}^K$ is the attention weight for slot description computed at time-step t and $\sum v_{tk} = 1$ for all t. $\mathcal{U}' \in \mathbb{R}^{2d \times J}$ represents the attention weights for the slot description with respect to all the words in the utterance. Basically, every column of the matrix represents closeness of the slot description with the corresponding utterance word. Then, attention weights that signify the words in the utterance that have the highest similarity with the slot description are computed as follows: $h' = \sum_j b_j \mathcal{H}_{:j}$ where $b = \text{softmax}(\max_{\text{col}}(\mathcal{A})) \in \mathbb{R}^J$ and max is operated across columns, and $\mathcal{H}' \in \mathbb{R}^{2d \times J}$ is obtained by tiling h' across columns.

We highlight that $\mathcal{U}'$ represents features that highlight important slot description words with closely similar words of utterance, and $\mathcal{H}'$ highlights features of the utterance with high similarity with the slot description, computed based on the similarity matrix $\mathcal{A}$, that itself has been computed based the contextual representations of the utterance ($\mathcal{H}$) and slot description ($\mathcal{U}$) generated by the encoding layer that considers surrounding words (i.e., employing bi-LSTM) to generate the representations. Finally, $\mathcal{U}'$ and $\mathcal{H}'$ are concatenated to produce $\mathcal{G} \in \mathbb{R}^{4d \times J}$, where every column of the matrix represents rich bi-directional similarity features of the corresponding utterance word with the slot description.

Essentially, this layer learns a general context-aware similarity function between utterance words and a slot description from seen domains, and it exploits the learned function for unseen domains. Due to the general nature of the similarity function, this layer also facilitates the identification of slot values in cases when `Step two` fails to correctly identify domain-independent slot values.

3.5 Contextualization Layer

This layer is responsible for contextualizing information from different granularities. Specifically, the utterance encodings from the Encoding layer, the bi-directional similarity between the utterance and the slot description from the Similarity layer, and the slot-independent `IOB` predictions from the CRF layer are passed as input. This layer employs *2* stacked layers of bi-directional LSTM networks to contextualize all the information by considering the relationships among neighbouring words' representations. It generates high quality features for the prediction layer; specifically, the features are $\in \mathbb{R}^{2d \times J}$, where each column represents the $2d$-dimensional features for the given word in the utterance.

3.6 Prediction Layer

The contextualized features are passed as input to this layer, and it is responsible for generating slot-specific predictions for the given utterance and slot type. First, it passes these features through *2* linear layers with ReLU activation. Then a CRF is employed to make structured predictions, as briefly explained in the CRF layer. The prediction process is done for each of the relevant slot types (i.e., slot types in the respective domain) and the resulting label sequences are combined to produces the final label for each word. Note that if the model made two or more conflicting slot predictions for a given sequence of words, we pick the slot type with the highest prediction probability.

3.7 Training the Model

The model has two trainable components: the slot-independent `IOB` predictor and the slot-specific `IOB` predictor. We jointly train both components by minimizing the negative log likelihood loss of both components over our training examples. The training data is prepared as follows. The training examples are of the form $(\mathcal{X}_i, \mathcal{S}_r, \mathcal{Y}'_i, \mathcal{Y}''_{ir})$, where $\mathcal{X}_i$ represents an utterance, $\mathcal{S}_r$ represents a slot type, $\mathcal{Y}'_i$ represents slot-independent `IOB` tags for the given utterance $\mathcal{X}_i$, and $\mathcal{Y}''_{ir}$ represents slot-specific `IOB` tags for the given utterance $\mathcal{X}_i$ and slot type $\mathcal{S}_r$. For a sample from the given dataset of the form $(\mathcal{X}_i, \mathcal{Y}_i)$ that has values for m slot types, first slot-indepedent `IOB` tags $\mathcal{Y}'_i$ are generated by removing slot type information. Then, we generated m positive training examples by setting each of m slot types as $\mathcal{S}_r$ and generating corresponding label $\mathcal{Y}''_{ir}$ (i.e., slot-specific tags for slot type $\mathcal{S}_r$). Finally, q negative examples are generated, where such slot types are chosen which are not present in the utterance. For example, the utterance in Figure 1 "I would like to book a table at 8 Immortals Restaurant in San Francisco" has true labels as "O O O O O O O O B-restaurant_name I-restaurant_name I-restaurant_name O B-city I-city". The positive training examples would be: $(\cdots,$ "`restaurant_name`", "O O O O O O O O B I I O B I", "O O O O O O O O O B I I O O O") and $(\cdots,$ "`city`", $\cdots,$ "O O O O O O O O O O O O O B I"). Whereas the negative examples can be as follows: $(\cdots,$ "`salon_name`", $\cdots,$ "O O O O O O O O O O O O O O"), $(\cdots,$ "`cuisine`", $\cdots, \cdots)$, $(\cdots,$ "`phone_number`", $\cdots, \cdots)$, and so on. Note that slot types are shown in the above example for brevity, the slot descriptions are used in practice.

Table 1: Dataset statistics.

Dataset	SNIPS	ATIS	MultiWOZ	SGD
Dataset Size	14.5K	5.9K	67.4K	188K
Vocab. Size	12.1K	1K	10.5K	33.6K
Avg. Length	9.0	11.1	13.3	13.8
# of Domains	6	1	8	20
# of Intents	7	18	11	46
# of Slots	39	83	61	240

4 EXPERIMENTAL SETUP

In this section, we describe the datasets, evaluation methodology, competing methods, and the implementation details of our model LEONA.

4.1 Datasets

We used four public datasets to evaluate the performance of our model LEONA: SNIPS Natural Language Understanding benchmark (SNIPS) [7], Airline Travel Information System (ATIS) [30], Multi-Domain Wizard-of-Oz (MultiWOZ) [64], and Dialog System Technology Challenge 8, Schema Guided Dialogue (SGD) [45]. To the best of our knowledge, this is first work to comprehensively evaluate zero-shot slot filling models on a wide range of public datasets. Table 1 presents important statistics about the datasets.

SNIPS. A crowd-sourced single-turn Natural Language Understanding (NLU) benchmark widely used for slot filling. It has 39 slot types across 7 intents from different domains. Since this dataset does not have slot descriptions, we used tokenized slot names as the descriptions (e.g., for slot type "playlist_owner", we used "playlist owner" as its description).

ATIS. A single-turn dataset that has been widely used in slot filling evaluations. It covers 83 slot types across 18 intents from a single domain. Many of the intents do not have many utterances, so all the intents having less than 100 utterances are combined into a single intent "Others" in our experiments. Moreover, similarly to SNIPS dataset, we used the tokenized versions of the slot names as slot descriptions.

MultiWOZ. A well-known dataset that has been widely used for the task of dialogue state tracking. In this work, we used the most recent version of the dataset (i.e., MultiWOZ2.2). In its original form, it contains dialogues between users and system. For the task of slot filling, we take all the user utterances and system messages that mention any slot(s), and shuffle the order to make it as if it were a single-turn dataset to maintain consistency with the previous works. For experiments in this work, utterances with intents that have less than 650 (< 1% of the dataset) utterances are grouped into the intent "Others".

SGD. A recently published comprehensive dataset for the eighth Dialog System Technology Challenge; it contains dialogues from 20 domains with a total of 46 intents and 240 slots. SGD was originally proposed for dialogue state tracking. This dataset is also pre-processed to have single-turn utterances labeled for slot filling. Moreover, we merge utterances from domains that have no more than 1850 (< 1% of the dataset) utterances, and we name the resulting domain "Others".

Since not all datasets provide a large enough number of domains, we do the splits in our experiments based on intents instead of domains for datasets that have more intents than domains. That is, we consider intents as domains for SNIPS, ATIS, and MultiWOZ.

4.2 Evaluation Methodology

We compute the slot F1 scores[5] and present evaluation results for the following settings:

Train on all except target intent/domain. This is the most common setting that previous works [2, 31, 51] have used for evaluation. A model is trained on all intents/domains except a single target intent/domain. For example, for SNIPS dataset the model is trained on all intents except a target intent "AddToPlatlist" that is used for testing the model's capabilities in the zero-shot fashion. This setup is repeated for every single intent in the dataset. The utterances at test time only come from a single intent/domain (or "Others") which makes this setting less challenging.

Train on a certain percentage of intents/domains and test on the rest. This is a slightly more challenging setting where test (i.e., unseen in training) intent/domains are usually from multiple unseen new intents/domains. We vary the number of training (i.e., seen) and testing (i.e., unseen) intents/domains to comprehensively evaluate all competing models. In this setting, we randomly select $\approx 25\%$, $\approx 50\%$, and $\approx 75\%$ of the intents/domains for training and the rest for testing, and report average results over five runs.

Train on one dataset and test on the rest of the datasets. This is the most challenging setting, where models are trained on one dataset and tested on the remaining datasets. For example, we train on the SGD dataset and test on SNIPS, ATIS, and MultiWOZ datasets. Similarly, we repeat the process for every dataset. Since datasets are very diverse (i.e., in terms of domains, slot types and user's expressions), this setting can be thought of as a "in the wild" [9] setting, which resembles real-world zero-shot slot filling scenarios to a large degree.

4.3 Competing Methods

We compare against the following state-of-the-art (SOTA) models:

Coach [31]. This model proposes to handle the zero-shot slot filling task with a coarse-to-fine procedure. It first identifies the words that constitute slot values. Then, based on the identified slot values, it tries to assign these values to slot types by matching the identified slot values with the representation of each slot description. We use their best model, i.e., Coach+TR, that employs template regularization but we call it Coach for simplicity.

RZS [51]. This work proposes a zero-shot adaption for slot filling by utilizing example values of each slot type. It employs character and word embedding of the utterance and slot descriptions, which are then concatenated with the averaged slot example embeddings, and passed through a bidirectional LSTM network to get the final prediction for each word in the utterance.

CT [2]. This model fills slots for each slot type individually. Character and word-level representations are concatenated with the slot type representation (i.e., embeddings) and an LSTM network

[5] Standard CoNLL evaluation script is used to compute slot F1 score.

Table 2: SNIPS dataset: Slot F1 scores for all competing models for target intents that are unseen in training.

Target Intent ↓	CT	RZS	Coach	LEONA w/o IOB	LEONA
AddToPlaylist	0.3882	0.4277	0.5090	0.5104	**0.5115**
BookRestaurant	0.2754	0.3068	0.3401	0.3405	**0.4781**
GetWeather	0.4645	0.5028	0.5047	0.5531	**0.6677**
PlayMusic	0.3286	0.3312	0.3201	0.3435	**0.4323**
RateBook	0.1454	0.1643	0.2206	0.2224	**0.2318**
SearchCreativeWork	0.3979	0.4445	0.4665	0.4671	**0.4673**
SearchScreeningEvent	0.1383	0.1225	0.2563	0.2690	**0.2872**
Average	0.3055	0.3285	0.3739	0.3866	**0.4394**

Table 3: ATIS dataset: Slot F1 scores for all competing models for target intents that are unseen in training.

Target Intent ↓	CT	RZS	Coach	LEONA w/o IOB	LEONA
Abbreviation	0.4163	0.5252	0.4804	0.4965	**0.6405**
Airfare	0.6549	0.5410	0.6929	0.7490	**0.9492**
Airline	0.7126	0.6354	0.7212	0.7762	**0.8586**
Flight	0.6530	0.7165	0.8072	0.8521	**0.9070**
Ground Service	0.4924	0.6452	0.7641	0.8463	**0.8490**
Others	0.4835	0.5169	0.6586	0.7749	**0.8337**
Average	0.5688	0.5967	0.6874	0.7492	**0.8397**

is used to make the predictions for each word in the utterance for the given slot type.

Note that we do not compare against simple baselines such as BiLSTM-CRF [26], LSTM-BoE, and CRF-BoE [22] because they have been outperformed by the previous works we compare against.

4.4 Implementation Details

Our model uses 300 dimensional embedding for POS and NER tags, and pre-trained ELMo embedding with 1024 dimensions. The encoding and contextualization layers have two stacked layers of bi-directional LSTMs with hidden states of size 300. The prediction layer has two linear layers with ReLU activation, and the CRF uses the "IOB" labeling scheme. The model is trained with a batch size of 32 for up to 200 epochs with early stopping using Adam optimizer and a negative log likelihood loss with a scheduled learning rate, starting at 0.001, and the model uses dropout rate of 0.3 at every layer to avoid over-fitting. Whereas q is set to three for negative sampling.

5 RESULTS

We present in the next subsections quantitative and qualitative analysis of all competing models. We first present the quantitative analysis in Subsection 5.1 and show that our model consistently outperforms the competing models in all settings. Furthermore, this subsection also has an ablation study that quantifies the role of each conceptual step in our model. We dig deeper into limitations of each competing model in our qualitative analysis in Subsection 5.2.

5.1 Quantitative Analysis

Train on all except target intent/domain. Tables 2 3, 4, and 5 present F1 scores for SNIPS, ATIS, MultiWOZ, and SGD datasets, respectively. All models are trained on all the intents/domains except the target one that is used for zero-shot testing. Our proposed approach is consistently better than SOTA methods. Specifically, it

Table 4: MultiWOZ dataset: Slot F1 scores for all competing models for target intents that are unseen in training.

Target Intent ↓	CT	RZS	Coach	LEONA w/o IOB	LEONA
Book Hotel	0.4577	0.3739	0.5866	0.6181	**0.6446**
Book Restaurant	0.3260	0.4200	0.4576	0.6268	**0.6269**
Book Train	0.4777	0.5269	0.6112	0.6317	**0.7025**
Find Attraction	0.2914	0.3489	0.3029	0.3787	**0.3834**
Find Hotel	0.4933	0.5920	0.7235	0.7673	**0.8222**
Find Restaurant	0.6420	0.6921	0.7671	0.7969	**0.8338**
Find Taxi	0.1459	0.1587	0.1260	0.1682	**0.1824**
Find Train	0.6344	0.4406	0.7754	0.8779	**0.8811**
Others	0.1205	0.0878	0.1201	0.1687	**0.1721**
Average	0.3988	0.4045	0.4967	0.5594	**0.5832**

Table 5: SGD dataset: Slot F1 scores for all competing models for target domains that are unseen in training.

Target Domain ↓	CT	RZS	Coach	LEONA w/o IOB	LEONA
Buses	0.4954	0.5443	0.6280	0.6364	**0.6978**
Calendar	0.5056	0.4908	0.6023	0.6216	**0.7436**
Events	0.5181	0.6324	0.5486	0.7405	**0.7619**
Flights	0.4898	0.4662	0.4898	0.4907	**0.5901**
Homes	0.4542	0.7159	0.6235	0.6927	**0.7698**
Hotels	0.4069	0.5681	0.7216	0.7266	**0.7677**
Movies	0.5100	0.3424	0.5537	0.5687	**0.7285**
Music	0.4111	0.6090	0.5786	0.7466	**0.7613**
RentalCars	0.4138	0.3399	0.6576	0.7344	**0.7389**
Restaurants	0.4620	0.3787	0.7195	0.7451	**0.7574**
RideSharing	0.6619	0.5312	0.7273	0.7656	**0.8172**
Services	0.6380	0.6381	0.7607	0.7628	**0.8180**
Travel	0.6556	0.6464	0.8403	0.9013	**0.9234**
Weather	0.4605	0.5180	0.6003	0.6178	**0.8223**
Others	0.4362	0.5312	0.4921	0.5129	**0.5592**
Average	0.5013	0.5302	0.6363	0.6842	**0.7505**

outperforms SOTA models by 17.52%, 22.15%, 17.42%, and 17.95% on average for unseen intents/domains on SNIPS, ATIS, MultiWOZ, and SGD datasets, respectively. We also present a variant of our model that does not employ "IOB" tags from Step two, we call it LEONA w/o IOB. Even this variant of our model outperforms all other SOTA models. This performance gain over SOTA methods can be attributed to the pre-trained NLP models that provide meaningful cues for the unseen domains, the similarity layer that can capture the closeness of the utterance words with the given slot irrespective of whether it is seen or unseen, and the contextualization layer that uses all the available information to generate a rich context-aware representation for each word in the utterance.

LEONA achieves its best performance on ATIS dataset (see Table 3) as compared to other datasets. It highlights that zero-shot slot filling across different intents within a single domain is relatively easier than across domains, since ATIS dataset consists of a single domain, i.e., airline travel. On the contrary, SGD dataset is the most comprehensive public dataset (i.e., it has 46 intents across 20 domains), yet our proposed method LEONA has better performance on it (see Table 5) than on SNIPS and MultiWoz datasets. This calls attention to another critical point: dataset quality. We observe that SGD dataset is not only comprehensive but also has high quality semantic description for slot types and all the domains have enough training examples with minimal annotation errors (based on a manual study of a small stratified sample from the dataset). For example, the slot types "restaurant_name",

Table 6: Averaged F1 scores for all competing models for seen and unseen slot types in the target unseen domains for SNIPS, ATIS, MultiWOZ, and SGD datasets.

Method ↓	SNIPS		ATIS		MultiWOZ		SGD	
Slot Type →	Seen	Unseen	Seen	Unseen	Seen	Unseen	Seen	Unseen
CT	0.4407	0.2725	0.7552	0.4851	0.6062	0.3040	0.7362	0.3940
RZS	0.4786	0.2801	0.8132	0.5143	0.6604	0.3301	0.7565	0.4478
Coach	0.5173	0.3423	0.7742	0.7166	0.7034	0.4895	0.7996	0.6614
LEONA w/o IOB	0.5292	0.3578	0.8155	0.7130	0.6651	0.5638	0.7986	0.7424
LEONA	**0.6354**	**0.4006**	**0.9588**	**0.7524**	**0.7765**	**0.5962**	**0.9192**	**0.8167**

Table 7: Averaged F1 scores for all competing models in the target unseen domains of all datasets. The train/test sets have variable number of intents/domains, which makes this setting more challenging.

Method ↓	SNIPS			ATIS			MultiWOZ			SGD		
% Seen Intents →	25%	50%	75%	25%	50%	75%	25%	50%	75%	25%	50%	75%
CT	0.1043	0.2055	0.2574	0.5018	0.7341	0.6542	0.2991	0.4371	0.6607	0.4523	0.5389	0.6160
RZS	0.1214	0.1940	0.3207	0.6393	0.7727	0.7811	0.4566	0.4703	0.6951	0.6677	0.6578	0.6741
Coach	0.1248	0.2258	0.3081	0.6070	0.7341	0.8104	0.4408	0.4505	0.6522	0.5888	0.6419	0.6725
LEONA w/o IOB	0.1550	0.2631	0.4108	0.6495	0.9437	0.9378	0.5137	0.5529	0.7843	0.6861	0.7315	0.7704
LEONA	**0.1710**	**0.2895**	**0.4220**	**0.8093**	**0.9659**	**0.9764**	**0.5248**	**0.5533**	**0.8581**	**0.7180**	**0.7925**	**0.8324**

"hotel_name", and "attraction_name" belong to different domains, but are very similar to one another. The rich semantic description of each slot type makes it easy for the model to transfer knowledge from one domain to new unseen domains with high F1 scores. LEONA shows poor performance on SNIPS dataset (see Table 2) as compared to other datasets, especially for intents: "RateBook" and "SearchScreeningEvent". This poor performance further highlights our previous point (i.e., quality of the dataset) since SNIPS dataset does not provide any textual descriptions for slot types. Moreover, slot names (e.g., "object_name" and "object_type") convey very little semantic information, which exacerbates the challenge for the model to perform well for unseen domains. Finally, the results on MultiWOZ dataset (see Table 4) highlights that transferring knowledge to new unseen intents/domains is easier when some similar intent/domain is there in the training set. For example, the model is able to transfer knowledge for new unseen target intent "Find Hotel" (i.e., not in the training) from other similar intents such as, "Find Restaurant" and "Book Hotel" effectively. However, for the target domain "Find Attraction" that does not have any similar domain in the training set, the model shows relatively poor performance. Similar observations can also be made other competing models.

Comparison for seen and unseen slot types. An unseen target domains may have both unseen and seen slot types. The unseen ones have never been seen during training, and seen ones might have different contexts. For example, "date" is a common slot type that may correspond to many different contexts in different domains such as date of a salon appointment, date of a restaurant booking, return date of a round-trip flight, and so on. We evaluate the performance of the competing models on seen and unseen slot types individually to test each model's ability in handling completely unsen slot types. Table 6 presents results in further detail where results for both seen and unseen slot types are reported separately. LEONA is consistently better than other models on seen as well as unseen slot types. On average, our proposed model LEONA shows 18% and 17% gains in F1 scores over the SOTA model for seen and unseen slots, respectively. These gains are due to our slot-independent IOB predictions (which provide effective templates for seen slot types) and our context-aware similarity function (which works well regardless whether slot types are seen or unseen). Moreover, all the models have better performance on seen slots than on unseen ones as it is relatively easier to adapt to a new context (i.e., in new domain) for seen slots than to new unseen slots in an unseen context. We also note that LEONA achieves a similar performance on ATIS dataset for seen slots in the unseen target domain, when compared with the results reported by SOTA *supervised slot filling* methods in [65], i.e., F1 score of 0.952 vs 0.959 by our method.

Train on a certain percentage of intents/domains and test on the rest. Large labeled training datasets are an important factor in accelerating the progress of supervised models. To investigate whether zero-shot models are affected by the size of training data from different domains, we vary the size of the training data and report results to quantify the effect. Table 7 presents results on all datasets when the training set has data from $\approx$ 25%, $\approx$ 50%, and $\approx$ 75% of the intents/domains and the rest are used for testing. The choice of intents/domains to be in the training or testing sets is done randomly, and average results are reported over five runs. This setting is more challenging in two ways: models have access to less training data and the test utterances come from multiple domains. LEONA is at least 19.06% better (better F1 scores) than other models for any percentage of unseen intents on any dataset. Overall, the performance of LEONA improves as it gets access to training data from more intents/domains, which is a desirable behaviour. Moreover, we also observe that our model achieves 0.72 F1 score on SGD with only 25% of domains in the training data,

Table 8: F1 scores for all competing models where the model is trained on one dataset and tested on the rest. This setting resembles real-life scenarios.

Method ↓ Train Dataset →	SNIPS			ATIS			MultiWOZ			SGD		
Test Dataset →	ATIS	MultiWOZ	SGD	SNIPS	MultiWOZ	SGD	SNIPS	ATIS	SGD	SNIPS	ATIS	MultiWOZ
CT	0.0874	0.1099	0.0845	0.0589	0.0725	0.0531	0.0646	0.0878	0.0616	0.1463	0.2290	0.1529
RZS	0.0915	0.1209	0.1048	0.0819	0.0809	0.0912	0.1496	0.2103	0.0875	0.1905	0.3435	0.2134
Coach	0.1435	0.1191	0.1301	0.0976	0.0962	0.0871	0.1201	0.1730	0.1102	0.1795	0.3383	0.1903
LEONA w/o IOB	0.1544	0.1433	0.1504	0.1156	0.1124	0.1359	0.1242	0.1885	0.1258	0.2544	0.4714	0.2743
LEONA	**0.2080**	**0.1832**	**0.1690**	**0.1436**	**0.1394**	**0.1361**	**0.1847**	**0.2662**	**0.1620**	**0.2761**	**0.5205**	**0.2884**

Table 9: Ablation study of our model LEONA in the zero-shot setting: averaged F1 scores for unseen target domains.

Configuration	SNIPS	ATIS	MultiWOZ	SGD
Step 2	0.3689	0.6719	0.4792	0.6375
Step 3	0.3812	0.6915	0.4999	0.6407
Step 2 + 3	0.4013	0.7605	0.5412	0.6684
Step 1 + 2	0.3820	0.6895	0.4936	0.6471
Step 1 + 3	0.3866	0.7492	0.5594	0.6842
Step 1 + 2 +3	**0.4394**	**0.8397**	**0.5832**	**0.7505**

which once again validates the intuition that having better quality data is very critical to adapt models to new unseen domains. Similar results are observed on ATIS dataset (i.e., single domain dataset), that highlights that knowledge transfer within a single domain is easier, and models can do a very good job on unseen intents even with a small amount of training data (e.g., 25% intents in the training set). Similar conclusions hold true for other methods.

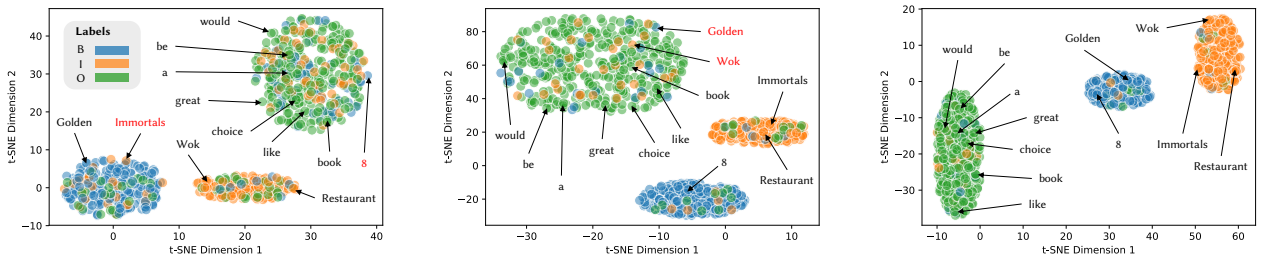
Train on one dataset and test on the rest of the datasets. This setting closely resembles the real-world zero-shot setting, where a model is trained on one dataset and tested on the rest. This is the most challenging setting, since the test datasets come from purely different distributions than those seen during training. Although each domain within a dataset can be thought of as a different distribution, every dataset shows some similarity of expression across different domains. Table 8 presents results of all competing models for this setting. All models show relatively poor performance for this challenging setting. However, LEONA is consistently better than others; specifically, it is up to 56.26% better on F1 score than the SOTA model. Our model achieves the best performance when it is trained on the SGD dataset (relatively better quality dataset) and tested on the rest. On the contrary, it shows the worst performance, when trained on ATIS (i.e., single-domain) and tested on the rest. Similar observations can be made for the other models. These results once again highlight the importance of the quality and comprehensiveness of the training dataset(s). Finally, this setting also indicates that the current SOTA models are not yet ready to be deployed in real-world scenarios and calls for more explorations and research in this important yet challenging and less-explored task of zero-shot slot filling.

Ablation study. To quantify the role of each component in our model, we present an ablation study results in Table 9 over all datasets. First, we study the significance of the pre-trained NLP models in the first three rows in Table 9. To produce the results in these rows, we used traditional word [5] and character [17] embeddings instead of employing powerful pre-trained NLP models. We observe that Step three, i.e., variant of the model that does not use pre-trained NLP models and does not consider "IOB" tags from Step two, is the most influential component in the model, as it alone can outperform the best performing SOTA model Coach [31], but the margin is not significant (i.e., 0.3812 vs. 0.3739 on SNIPS, 0.6915 vs. 0.6874 on ATIS, 0.4999 vs. 0.4967 on MultiWOZ, and 0.6407 vs. 0.6363 on SGD). If "IOB" predictions from Step two are incorporated into it (i.e., row Step 2 + 3) or pre-trained NLP models are employed with it (i.e., row Step 1 + 3), its performance is further improved. Moreover, if we just use Step two by predicting "IOB" tags and assigning these "IOB" tags to the slot type with the highest similarity (i.e., row Step 2), or combine Step one with Step two (i.e., row Step 1 + 2), we note that we do not achieve the best results.

5.2 Qualitative Analysis

In this experiment, we randomly selected 100 utterances in the unseen target domain "Restaurant" from the SGD dataset and visually analyzed the performance of the competing models in extracting the values of the slot type "restaurant_name" from the selected utterances. The goal of this experiment is to visually highlight the strengths/weaknesses of the competing models. We retrieved the multi-dimensional numerical representations of the words in the selected utterances from the final layers of each model and reduced the number of dimensions of each representation to two using t-SNE [35]. Figure 3 shows scatter plots for the resulting 2-dimensional representations for each model. We observe that all models produce clear-cut clusters for each class: B, I, or O, which indicates that all models are able to produce distinguishing representations. However, LEONA produces better representations in the sense that less words are misclassified. That is, there are less violating data point in the clusters of LEONA in Figure 3 (c).

We further analyze the results for two utterances: "Golden Wok would be a great choice in ..." and "I would like to book a table at 8 Immortals Restaurant in ...". RZS [51] is able to predict full slot value (i.e., Golden Wok) of the slot "restaurant_name" in the first utterance. However, we notice that RZS fails to capture the full value (i.e., "8 Immortals Restaurant") for the slot "restaurant_name" in the other utterance, where it could partially extract "Immortals Restaurant" and mistakenly assigns label O to the word "8", which led to subsequent wrong prediction for the word "Immortals" (i.e., predicted label B, whereas the true label is I). This misclassification is also highlighted in Figure 3 (a) by coloring the wrongly predicted words with red. Since RZS relies on the example value(s) and there is a high variability across the lengths of slot values, along with the diversity of expression, this model faces problems in detecting the *full slot values*.



(a) RZS (b) Coach (c) LEONA (this work)

Figure 3: t-SNE visualization of word representations from selected utterances; the selected utterances belong to the unseen domain "Restaurant" in SGD dataset and contain the slot type "restaurant_name". Results are presented for the best performing 3 models.

We notice that Coach [31] fails to detect the value (i.e., `Golden Wok`) for the slot "`restaurant_name`" in the first utterance. However, it successfully captures the slot value in the other utterance. Since Coach relies on learning templates from seen domains and exploits those for unseen domains, it fails to handle the deviation of the unseen domains from the learned templates. LEONA is able to detect full slot values for both utterances successfully, thanks to: the slot-independent `IOB` predictions from `Step two`; the similarity function in `Step three` which is robust to errors from the previous steps; and the contextualization layers of the model. Finally, we observed that our model also fails to fully detect very long slot values. For example, slot values "`Rustic House Oyster Bar And Grill`", "`Tarla Mediterranean Bar + Grill`", and "`Pura Vida – Cocina Latina & Sangria Bar`" for the slot type "`restaurant_name`" are challenging to detect in unseen domains not only because of their long length, but also because of the presence of tokens like &, +, and –, that further exacerbate the challenge. Note that other SOTA models also fail to detect the above example slot values. We plan to overcome this challenge in our future work by learning n-gram phrase-level representations to detect such slot values in their entirety.

6 RELATED WORK

We organize the related work into three categories: (*i*) supervised slot filling, (*ii*) few-shots slot filling, and (*iii*) zero-shot slot filling.

Supervised Slot Filling. Slot filling is an extensively studied research problem in the supervised setting. Recurrent neural networks such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) networks that learn how words within a sentence are related temporally [24, 36] have been employed to tag the input for slots. Similarly, Conditional Random Fields (CRFs) have been integrated with LSTMs/GRUs [21, 47]. The authors in [52, 58] proposed a self-attention mechanism for sequential labeling. More recently, researchers have proposed jointly addressing the related tasks of intent detection and slot filling [14, 16, 29, 61, 65]. The authors in [65] suggested using a capsule neural network by dynamically routing and rerouting information from wordCaps to slotCaps and then to intentCaps to jointly model the tasks. Supervised slot filling methods rely on the availability of large amounts of labeled training data from all domains to learn patterns of slot usage. In contrast, we focus on the more challenging as well as more practically relevant setting where new unseen domains are evolving and training data is not available for all domains.

Few-shot Slot Filling. Few-shot learning requires a small amount of training data in the target domain. Meta-learning based methods [10, 38, 39] have shown tremendous success for few-shot learning in many tasks such as few-shot image generation [46], image classification [53], and domain adaptation [59]. Following the success of such approaches, few-shot learning in NLP have been investigated for tasks such as text classification [13, 56, 62], entity-relation extraction [12, 33], and few-shot slot filling [11, 20, 32]. The authors in [32] exploited regular expressions for few-shot slot filling, Prototypical Network was employed in [11], and the authors in [20] extended the CRF model by introducing collapsed dependency transition to transfer label dependency patterns. Moreover, few-shot slot filling and intent detection have been modeled jointly [4, 23], where model agnostic meta learning (MAML) was leveraged. Few-shot slot filling not only requires a small amount of training data in the target domain, but also requires re-training/fine-tuning. Our model addresses the task of zero-shot slot filling where no training example for the new unseen target domain is available and it can seamlessly adapt to new unseen domains – a more challenging and realistic setting.

Zero-shot Slot Filling. Zero-shot learning for slot filling is less explored, and only a handful of research has addressed this challenging problem, albeit, with very limited experimental evaluation. Coach [31] addressed the zero-shot slot filling task with a coarse-to-fine approach. It first predicts words that are slot values. Then, it assigns the predicted slot value to the appropriate slot type by matching the value with the representation of description of each slot type. RZS [51] utilizes example values of each slot type. It uses character and word embeddings of the utterance and slot types along with the slot examples' embeddings, and passes the concatenated information through a bidirectional LSTM network to get the prediction for each word in the utterance. CT [2] proposed LSTM network and employed slot descriptions to fill the slots for each slot type individually. The authors in [27] also employed LSTM, slot descriptions, and attention mechanisms for individual slot predictions. To tackle the challenge of the zero-shot slot filling, we

leverage the power of the pre-trained NLP models, compute complex bi-directional relationships of utterance and slot types, and contextualize the multi-granular information to better accommodate unseen concepts. In a related, but orthogonal line of research, the authors in [15, 28, 34] tackled the problem of slot filling in the context of dialog state tracking where dialog state and history are available in addition to an input utterance. In contrast, this work and the SOTA models we compare against in our experiments only consider an utterance without having access to any dialog state elements.

7 CONCLUSION

We have presented a zero-shot slot filling model, LEONA, that can adapt to new unseen domains seamlessly. LEONA stands out as the first zero-shot slot filling model that effectively captures rich and context-aware linguistic features at different granularities. Our experimental evaluation uses a comprehensive set of datasets and covers many challenging settings that stress models and expose their weaknesses (especially in more realistic settings). Interestingly, our model outperforms all state-of-the-art models in all settings, over all datasets. The superior performance of our model is mainly attributed to: its effective use of pre-trained NLP models that provide domain-oblivious word representations, its multi-step approach where extra insight is propagated from one step to the next, its generalizable similarity function, and its contextualization of the words' representations. In the most challenging evaluation setting where models are tested on a variety of datasets after being trained on data from one dataset only, our model is up to 56.26% more accurate (in F1 score) than the best performing state-of-the-art model. It remains challenging for all models, including ours, to identify slot values that are very long or that contain certain tokens. We plan to further improve our model by incorporating n-gram phrase-level representations to overcome this challenge and allow our model to accurately extract slot values regardless of their length or diversity.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[2] Ankur Bapna, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2017. Towards zero-shot frame semantic parsing for domain scaling. *arXiv preprint arXiv:1707.02363* (2017).

[3] Jerome R Bellegarda. 2014. Spoken language understanding for natural interaction: The siri experience. In *Natural interaction with robots, knowbots and smartphones*. Springer, 3–14.

[4] Hemanthage S Bhathiya and Uthayasanker Thayasivam. 2020. Meta Learning for Few-Shot Joint Intent Detection and Slot-Filling. In *Proceedings of the 2020 5th International Conference on Machine Learning Technologies*. 86–92.

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[6] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326* (2015).

[7] Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190* (2018).

[8] Douglass Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. 1992. A practical part-of-speech tagger. In *Third Conference on Applied Natural Language Processing*. 133–140.

[9] Abhinav Dhall, Roland Goecke, Shreya Ghosh, Jyoti Joshi, Jesse Hoey, and Tom Gedeon. 2017. From individual to group-level emotion recognition: Emotiw 5.0. In *Proceedings of the 19th ACM international conference on multimodal interaction*. 524–528.

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400* (2017).

[11] Alexander Fritzler, Varvara Logacheva, and Maksim Kretov. 2019. Few-shot classification in named entity recognition task. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. 993–1000.

[12] Tianyu Gao, Xu Han, Ruobing Xie, Zhiyuan Liu, Fen Lin, Leyu Lin, and Maosong Sun. 2020. Neural Snowball for Few-Shot Relation Learning.. In *AAAI*. 7772–7779.

[13] Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. 2019. Induction networks for few-shot text classification. *arXiv preprint arXiv:1902.10482* (2019).

[14] Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. 753–757.

[15] Pavel Gulyaev, Eugenia Elistratova, Vasily Konovalov, Yuri Kuratov, Leonid Pugachev, and Mikhail Burtsev. 2020. Goal-oriented multi-task bert-based dialogue state tracker. *arXiv preprint arXiv:2002.02450* (2020).

[16] Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm.. In *Interspeech*. 715–719.

[17] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2016. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587* (2016).

[18] Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 473–483.

[19] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[20] Yutai Hou, Wanxiang Che, Yongkui Lai, Zhihan Zhou, Yijia Liu, Han Liu, and Ting Liu. 2020. Few-shot Slot Tagging with Collapsed Dependency Transfer and Label-enhanced Task-adaptive Projection Network. *arXiv preprint arXiv:2006.05702* (2020).

[21] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015).

[22] Rahul Jha, Alex Marin, Suvamsh Shivaprasad, and Imed Zitouni. 2018. Bag of experts architectures for model reuse in conversational language understanding. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*. 153–161.

[23] Jason Krone, Yi Zhang, and Mona Diab. 2020. Learning to Classify Intents and Slot Labels Given a Handful of Examples. *arXiv preprint arXiv:2004.10793* (2020).

[24] Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016. Leveraging sentence-level information with encoder lstm for semantic slot filling. *arXiv preprint arXiv:1601.01530* (2016).

[25] John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. (2001).

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

[27] Sungjin Lee and Rahul Jha. 2019. Zero-shot adaptive transfer for conversational language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6642–6649.

[28] Miao Li, Haoqi Xiong, and Yunbo Cao. 2020. The sppd system for schema guided dialogue state tracking challenge. *arXiv preprint arXiv:2006.09035* (2020).

[29] Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454* (2016).

[30] Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2019. Benchmarking natural language understanding services for building conversational agents. *arXiv preprint arXiv:1903.05566* (2019).

[31] Zihan Liu, Genta Indra Winata, Peng Xu, and Pascale Fung. 2020. Coach: A Coarse-to-Fine Approach for Cross-domain Slot Filling. *arXiv preprint arXiv:2004.11727* (2020).

[32] Bingfeng Luo, Yansong Feng, Zheng Wang, Songfang Huang, Rui Yan, and Dongyan Zhao. 2018. Marrying up regular expressions with neural networks: A case study for spoken language understanding. *arXiv preprint arXiv:1805.05588* (2018).

[33] Xin Lv, Yuxian Gu, Xu Han, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2019. Adapting meta knowledge graph information for multi-hop reasoning over few-shot relations. *arXiv preprint arXiv:1908.11513* (2019).

[34] Yue Ma, Zengfeng Zeng, Dawei Zhu, Xuan Li, Yiying Yang, Xiaoyuan Yao, Kaijie Zhou, and Jianping Shen. 2019. An end-to-end dialogue state tracking system

with machine reading comprehension and wide & deep classification. *arXiv preprint arXiv:1912.09297* (2019).
[35] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.
[36] Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, 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 (2014), 530–539.
[37] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
[38] Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999* (2018).
[39] Alex Nichol and John Schulman. 2018. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999* 2, 3 (2018), 4.
[40] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.
[41] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.
[42] Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*. 1–40.
[43] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* (2016).
[44] Lance Ramshaw and Mitch Marcus. 1995. Text Chunking using Transformation-Based Learning. In *Third Workshop on Very Large Corpora*. https://www.aclweb.org/anthology/W95-0107
[45] Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2019. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *arXiv preprint arXiv:1909.05855* (2019).
[46] Scott Reed, Yutian Chen, Thomas Paine, Aäron van den Oord, SM Eslami, Danilo Rezende, Oriol Vinyals, and Nando de Freitas. 2017. Few-shot autoregressive density estimation: Towards learning to learn distributions. *arXiv preprint arXiv:1710.10304* (2017).
[47] Nils Reimers and Iryna Gurevych. 2017. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *arXiv preprint arXiv:1707.06799* (2017).
[48] Erik F Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050* (2003).
[49] Burr Settles. 2004. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*. 107–110.
[50] Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. 213–220.
[51] Darsh J Shah, Raghav Gupta, Amir A Fayazi, and Dilek Hakkani-Tur. 2019. Robust zero-shot cross-domain slot filling with example values. *arXiv preprint arXiv:1906.06870* (2019).
[52] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding. *arXiv preprint arXiv:1709.04696* (2017).
[53] Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*. 4077–4087.
[54] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*. 1631–1642.
[55] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387* (2015).
[56] Shengli Sun, Qingfeng Sun, Kevin Zhou, and Tengchao Lv. 2019. Hierarchical attention prototypical networks for few-shot text classification. 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)*. 476–485.
[57] Charles Sutton and Andrew McCallum. 2006. An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning* 2 (2006), 93–128.
[58] Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2017. Deep semantic role labeling with self-attention. *arXiv preprint arXiv:1712.01586* (2017).
[59] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in neural information processing systems*. 3630–3638.
[60] Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916* (2014).
[61] Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *2013 ieee workshop on automatic speech recognition and understanding*. IEEE, 78–83.
[62] Leiming Yan, Yuhui Zheng, and Jie Cao. 2018. Few-shot learning for short text classification. *Multimedia Tools and Applications* 77, 22 (2018), 29799–29810.
[63] Steve Young. 2002. Talking to machines (statistically speaking). In *Seventh International Conference on Spoken Language Processing*.
[64] Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. 2020. MultiWOZ 2.2 : A Dialogue Dataset with Additional Annotation Corrections and State Tracking Baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*. Association for Computational Linguistics, Online, 109–117. https://doi.org/10.18653/v1/2020.nlp4convai-1.13
[65] Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip S Yu. 2018. Joint slot filling and intent detection via capsule neural networks. *arXiv preprint arXiv:1812.09471* (2018).