UC Riverside

UC Riverside Electronic Theses and Dissertations

Title

Unsupervised and Zero-Shot Learning for Open-Domain Natural Language Processing

Permalink

https://escholarship.org/uc/item/28h3j571

Author

Siddique, Muhammad Abu Bakar

Publication Date

2021

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA
RIVERSIDE

Unsupervised and Zero-Shot Learning for Open-Domain Natural Language Processing

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Muhammad Abu Bakar Siddique

June 2021

Dissertation Committee:

Dr. Evangelos Christidis, Chairperson
Dr. Amr Magdy Ahmed
Dr. Samet Oymak
Dr. Evangelos Papalexakis

Copyright by
Muhammad Abu Bakar Siddique
2021

The Dissertation of Muhammad Abu Bakar Siddique is approved:

Committee Chairperson

University of California, Riverside

To my family for their *unconditional* love and support.

ABSTRACT OF THE DISSERTATION

Unsupervised and Zero-Shot Learning for Open-Domain Natural Language Processing

by

Muhammad Abu Bakar Siddique

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, June 2021
Dr. Evangelos Christidis, Chairperson

Natural Language Processing (NLP) has yielded results that were unimaginable only a few years ago on a wide range of real-world tasks, thanks to deep neural networks and the availability of large-scale labeled training datasets. However, existing supervised methods assume an unscalable requirement that labeled training data is available for all classes: the acquisition of such data is prohibitively laborious and expensive. Therefore, zero-shot (or unsupervised) models that can seamlessly adapt to new unseen classes are indispensable for NLP methods to work in real-world applications effectively; such models mitigate (or eliminate) the need for collecting and annotating data for each domain. This dissertation addresses three critical NLP problems in contexts where training data is scarce (or unavailable): intent detection, slot filling, and paraphrasing. Having reliable solutions for the mentioned problems in the open-domain setting pushes the frontiers of NLP a step towards practical conversational AI systems.

First, this thesis addresses intent detection — extracting the intents implied in natural language utterances. We propose RIDE: a zero-shot intent detection model that

captures domain-oblivious semantic associations between an utterance and an intent by analyzing how the phrases in an utterance are linked to an intent label via commonsense knowledge. RIDE significantly and consistently outperforms SOTA intent detection models.

The second contribution of this dissertation is a zero-shot model for the *slot filling* task — extracting the required query parameters, given a natural language utterance. Our model, LEONA, exploits domain-independent pre-trained NLP models and context-aware utterance-slot similarity features via attention mechanisms by taking advantage of the fact that slot values appear in similar contexts across domains. LEONA significantly and consistently outperforms SOTA models in a wide range of experimental setups.

Finally, we propose an unsupervised model, PUP, for paraphrasing. Unsupervised paraphrasing has applications in conversational AI, among others. PUP uses a variational autoencoder (trained using a non-parallel corpus) to generate a seed paraphrase that warm-starts a deep reinforcement learning model. Then, it progressively tunes the seed paraphrase guided by a novel reward function that combines semantic adequacy, language fluency, and expression diversity measures. PUP achieves an unprecedented balance across the paraphrase quality metrics.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Artificial Intelligence (AI) has emerged as a leading force in revolutionizing society, economy, governance, and almost every aspect of our lives through AI-driven systems. Such systems augment critical human capabilities including driving trucks and translating languages, among others. Natural Language Processing (NLP), the branch of AI that enables humans to interact with machines through intuitive natural language interfaces such as Amazon Alexa, has yielded results that were unimaginable only a few years ago on a wide range of demanding real-world tasks, thanks to the advances in deep neural networks and the availability of massive labeled training data. Nonetheless, this performance boost is still restricted to a handful of resource-rich domains in most NLP tasks because existing *supervised* NLP solutions typically rely on the critical assumption that the training and test sets are drawn from the same underlying distribution. Such supervised solutions are likely to result in a significantly diminished performance when confronted with new unseen domains. The requirement of having massive labeled training data for each new domain is unsustainable:

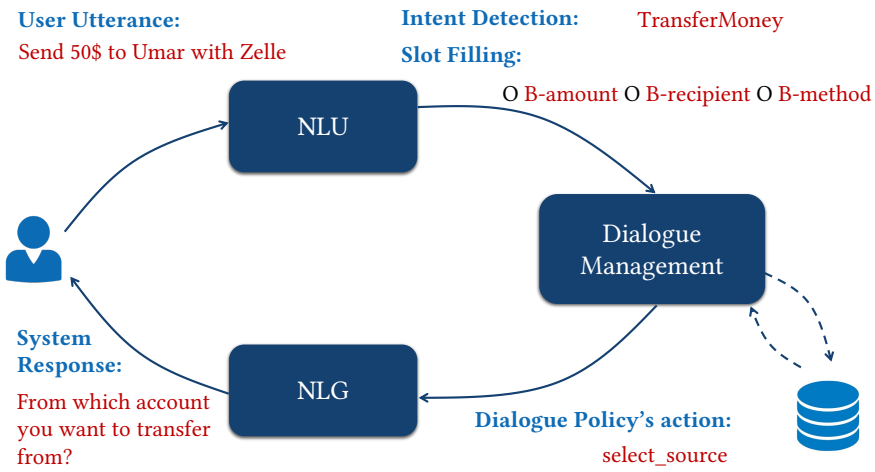


Figure 1.1: A typical pipeline for goal-oriented dialogue systems; NLP tasks are illustrated at each component of the pipeline.

the acquisition of such data is prohibitively laborious and expensive. Therefore, zero-shot (or unsupervised) models that can seamlessly adapt to new unseen domains are indispensable to real-world NLP applications; such models mitigate (or eliminate) the need for collecting and annotating data for each domain, resulting in open-domain NLP solutions.

Open-domain NLP models are critical to the success of real-world conversational AI systems, especially goal-oriented ones. Such systems enable users to instruct machines to perform tasks such as adding a song to a playlist, setting an alarm, and booking a train ticket through a natural language interface. Figure 1.1 presents the typical pipeline of goal-oriented dialogue systems and shows a simple example to highlight the role of each module. The Natural Language Understanding (NLU) module performs intent detection and slot filling, which infer the desired task and extract the required parameters given an utterance. The Dialogue Management module summarizes the state of the dialogue and decides the next action, i.e., the next utterance of the bot at the semantic level; this module usually has access

to structured data source(s) or a set of APIs. The Natural Language Generation (NLG) module converts the selected action into a natural language response. For goal-oriented dialogue systems to perform reliably in real-life settings (when domains are unknown prior to model deployment), each of the mentioned modules needs to seamlessly adapt to new, unseen domains where neither training data nor the full list of classes are available. This dissertation focuses on the NLU module and proposes reliable ways of integrating readily available, domain-oblivious knowledge into neural models in this module. Furthermore, this dissertation proposes an incremental framework for training deep reinforcement learning agents in a way that saves the agent from doing expensive global explorations, which is particularly critical in open-domain settings. Combined, the ideas in this dissertation make it possible to build reliable, open-domain NLP models.

1.1 Dissertation Contributions

Chapter 2 provides a brief introduction to the relevant NLP and conversational AI concepts.

Chapter 3 proposes a generalized zero-shot intent detection model, RIDE. It leverages commonsense knowledge in an unsupervised fashion to overcome the issue of training data scarcity. RIDE computes robust and generalizable *relationship meta-features* that capture deep semantic relationships between utterances and intent labels; these features are computed by considering how the concepts in an utterance are linked to those in an intent label via commonsense knowledge. The motivation for this work is that most of the existing work on intent detection is in the supervised settings. However, in practice, new intents emerge after deploying an intent detection model. Thus, these models should seamlessly adapt

and classify utterances with both seen and unseen intents – unseen intents emerge after deployment and they do not have training data. The few existing models that target this setting rely heavily on the training data of seen intents and consequently overfit to these intents, resulting in a bias to misclassify utterances with unseen intents into seen ones. Our extensive experimental analysis on three widely-used intent detection benchmarks shows that relationship meta-features significantly improve the detection of both seen and unseen intents and that RIDE outperforms the state-of-the-art models. Specifically, RIDE is 30.36% to 58.50% more accurate than the SOTA model for unseen intents.

Chapter 4 proposes a new zero-shot slot filling neural model, LEONA, which works in three steps. Step one acquires domain-oblivious, context-aware representations of utterance words by exploiting (a) linguistic features such as part-of-speech tags; (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. The motivation for this work is that slot filling is one of the most important challenges in modern goal-oriented dialog systems and although supervised approaches have proven effective at tackling this challenge, 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. Our thorough evaluation on four diverse public datasets demonstrates that our approach consistently outperforms 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.

Chapter 5 proposes Progressive Unsupervised Paraphrasing PUP: a novel unsupervised paraphrase generation method based on deep reinforcement learning (DRL). PUP uses a variational autoencoder (trained using a non-parallel corpus) to generate a seed paraphrase that warm-starts the DRL model. Then, PUP progressively tunes the seed paraphrase guided by our novel reward function which combines semantic adequacy, language fluency, and expression diversity measures to quantify the quality of the generated paraphrases in each iteration without needing parallel sentences. The motivation for this work is that paraphrasing can improve the performance of the NLU module of the conversational agents. However, most existing work on paraphrasing use supervised models that are limited to specific domains (e.g., image captions). Such models can neither be straightforwardly transferred to other domains nor generalize well, and creating labeled training data for new domains is expensive and laborious. The need for paraphrasing across different domains and the scarcity of labeled training data in many such domains call for exploring unsupervised paraphrase generation methods. Our extensive experimental evaluation shows that PUP outperforms unsupervised state-of-the-art paraphrasing techniques in terms of both automatic

metrics and user studies on four real datasets. Specifically, our method achieves up to 90% and 34% performance gains for the BLEU and the i-BLEU metrics compared to state-of-the-art unsupervised methods, respectively. We also show that PUP outperforms domain-adapted supervised algorithms on several datasets.

1.2 Dissertation Organization

This dissertation contains published work. Chapter 3 is based on a paper [131] accepted for publication at the International ACM SIGIR Conference on Research and Development in Information Retrieval `(SIGIR'21)`. Chapter 4 is based on a paper [130] published at the International World Wide Web Conference (a.k.a. The Web Conference) `(WWW'21)`. Chapter 5 is based on a paper [132] published at the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining `(KDD'20)`.

Chapter 2

Background and Related Work

2.1 Knowledge Graphs

Knowledge graphs (KG) [12, 138, 155, 2, 141] are structures that capture relationships between entities, and are typically used to capture knowledge in a semi-structured format; i.e., they are used as knowledge bases. Knowledge graphs can be viewed as collections of triples, each representing a fact of the form $\langle$`head`, *relation*, `tail`$\rangle$ where `head` and `tail` describe entities and *relation* describes the relationship between the respective entities (e.g., $\langle$`apple`, *IsA*, `fruit`$\rangle$). Although creating and maintaining knowledge graphs is laborious and time consuming, the immense utility of such graphs (e.g., search engines, question-answering services, commonsense reasoning) has led many researchers and institutions to make the effort of building and maintaining knowledge graphs in many domains, which lifts the burden off of other researchers and developers who utilize these graphs. It turns out that the commonsense knowledge captured in knowledge graphs, which is uniform across domains, can facilitate building open-domain NLP models as we show in Chapter 3.

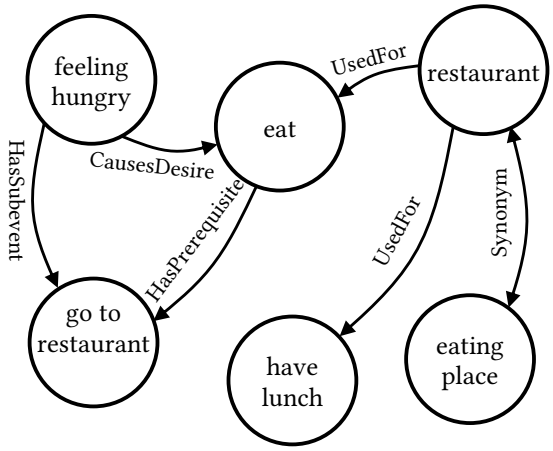


Figure 2.1: A sample subgraph from ConceptNet knowledge graph. The commonsense knowledge captured in knowledge graphs can improve the discriminative power of NLP models as we show in Chapter 3.

We make heavy use of the ConceptNet [138], which is a rich and widely-used commonsense knowledge graph in this dissertation. This KG originated from the crowd-sourcing project Open Mind Common Sense and includes knowledge not only from crowd-sourced resources but also expert-curated resources. It is available in 10 core languages and 68 more common languages. Figure 2.1 presents a subgraph from ConceptNet knowledge graph.

2.2 Link Prediction in Knowledge Graphs

While large knowledge graphs may capture a large subset of knowledge, they are incomplete: some relationships (or links) are missing. Link prediction [86, 63, 150, 94] augments knowledge graphs by predicting missing relations using existing ones. For example, link prediction is used in predicting relationships among friends within a social network or co-authorship relationships in a citation network. Given a knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$,

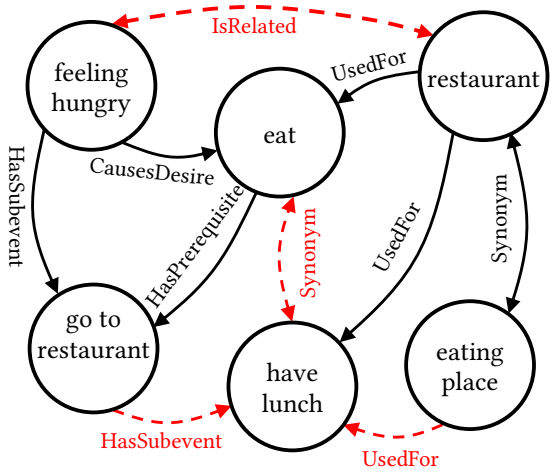


Figure 2.2: An example subgraph from ConceptNet augmented with predicted link (red edges) via link prediction.

where $\mathcal{V}$ represents the entities in the KG and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ represents the set of links among entities in the KG; each item in set $\mathcal{E}$ is of the form $\langle$`head`, *relation*, `tail`$\rangle$, where `head` and `tail` $\in \mathcal{V}$, *relation* $\in \mathcal{R}$, and $\mathcal{R}$ represents the set of relationship types in the KG. The goal of a link predictor is to predict missing (i.e., not present) links based on the existing set of entities $\mathcal{V}$ and the set of links $\mathcal{E}$. For the link prediction task, graph embedding is one of the most effective techniques. Graph embedding methods [46, 63, 45] learn to embed graph nodes in a high-dimensional space where nodes with similar properties or relationships remain close to each other and vector similarity measures (e.g., euclidean distance) hold in the embedding space. It is usually formulated as a binary classification task that can classify novel links of the form $\langle$`head`, *relation*, `tail`$\rangle$ as either positive or negative. Figure 2.2 presents a subgraph from ConceptNet, where a trained link predictor has predicted several missing links. In this dissertation, we use link predictors to overcome the missing links issue in the ConceptNet KG.

2.3 Conditional Random Fields

Conditions Random Fields (CRFs) [145] are discriminative models that are the most appropriate choice for structured prediction tasks (e.g., part of speech tagging [24], named entity recognition [125]) where contextual information or neighboring predictions influence the current prediction [148]. Based on our feature vector $\mathcal{X} = \{x_1, x_2, \cdots, x_T\}$, we want to predict the output vector $\mathcal{Y} = \{y_1, y_2, \cdots, y_T\}$. The goal is to learn a per-token classifier f that maps $x_s \mapsto y_s$ for all s while maximizing the number of labels y_s that are correctly classified by learning inter-dependencies among neighbouring predictions. Linear chain CRFs, one of the most common CRFs, are trained by estimating maximum conditional log-likelihood. Essentially, it estimates a transition cost matrix of size, `num_tags` $\times$ `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. The tasks of Part of Speech (POS) tagging [24] and named entity recognition (NER) [125], among others, have successfully employed CRFs. Figure 2.3 presents a sample sentence along with its POS and NER tags. We utilize CRF for the task of slot filling in Chapter 4.

Part of Speech tagging. The task of POS tagging is the process of assigning special labels to each token in a text, in order to indicate its part of speech, such as `PROPN`, `VERB`, and `ADJ`. Since POS tags have very high interdependence among tags of neighbouring words, CRFs are highly suitable for the task, because CRFs do not only assume that features are interdependent, but also take into account future observations while learning a pattern. In recent years, CRFs have been used as a final layer in deep neural networks for making final predictions for POS tags [89, 60]. We take advantage of existing POS taggers in Chapter 4.

Input Sentence	POS tags	NER tags
I	PRON	O
would	VERB	O
like	VERB	O
to	PART	O
book	VERB	O
a	NOUN	O
table	NOUN	O
at	ADP	O
8	PROPN	B-ORG
Immortals	PROPN	I-ORG
Restaurant	PROPN	I-ORG
in	ADP	O
San	PROPN	B-GPE
Francisco	PROPN	I-GPE

Figure 2.3: Example utterance with its POS and NER tags; NER tags are expressed using the IOB tagging scheme.

Named Entity Recognition. The goal of the NER model is to identify information units from unstructured text, such as names of people, organizations, and locations. The most common case for NER models is to detect named entities for `PER`, `GPE`, `ORG`, and `MISC`. Inside outside beginning (IOB) [114] is a common tagging scheme for NER models when they employ CRFs. Given a sequence of words $\mathcal{X} = \{x_1, x_2, \cdots, x_T\}$, we want to give a tag for each word as follows. The first word of an entity value associated with an entity type (e.g., `PER`) is labeled as `B-Entity`, the other words inside the same entity value are labeled as `I-Entity`, and non-entity words are labeled as `O`. We follow this notation in the remainder of this dissertation and employ pre-trained NER model in Chapter 4.

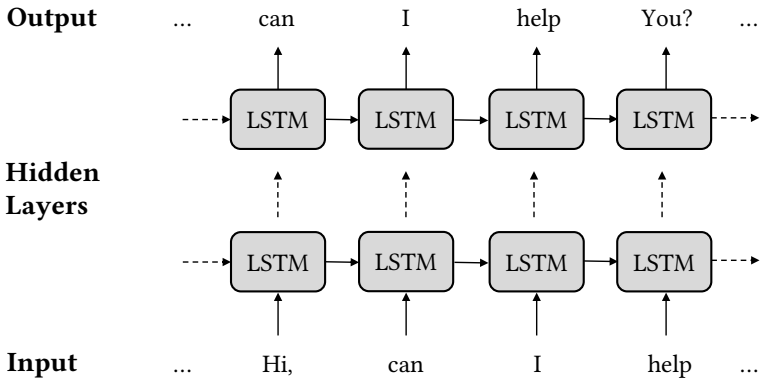


Figure 2.4: Illustration of an LSTM-based language model.

2.4 Language Models

Language modeling [54, 106] is the task of assigning a probability to the next token given all previous ones. For example, in a word-level language model, the probability of the next word, w_i, given previous ones is $p(w_i|w_{i-1}, w_{i-2}, ..., w_0)$. A recurrent neural network (RNN) such as long short-term memory (LSTM) language model [9, 95, 100] can be trained by iteratively passing over the text data in an unsupervised fashion. The goal is to predict word w_i at timestamp i given the previous n tokens. At every word w_i, the negative log-likelihood loss is $\mathcal{L} = -\log p(w_i|w_{i-1}, w_{i-2}, ..., w_{i-n+1})$, where n is the backpropagation through time (BPTT) [48] window size. Figure 2.4 presents such a language model.

Transformers [151] can process sequential input data (e.g., text data), but unlike an RNN, it is not necessary for the data to be processed in order, allowing massive parallelization. Following the transformer architecture, training language models (e.g., BERT [28], ELMo [109]) over huge amounts of text data has become possible. These models have billions of parameters and thereby capture general semantic and syntactic

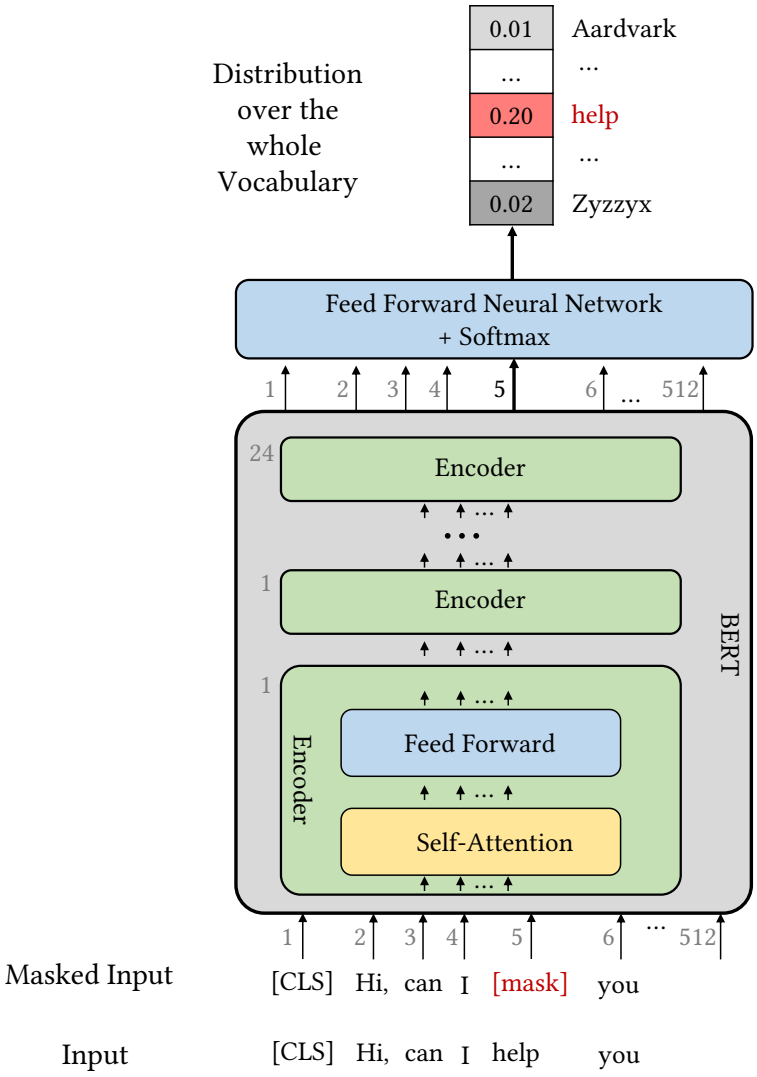


Figure 2.5: Illustration of BERT's Masked Language Modeling task.

information in an effective manner. Unlike GloVe [108] or Word2vec [99] that produce fixed word emebeddings, pre-trained language models (LM) such as BERT [28] and ELMo [109] produce embeddings for the words by considering the context of the words. Pre-trained LMs [28, 109, 84, 62, 167] with appropriate fine-tuning have provided state-of-the-art results on many NLP benchmarks [113, 14, 53, 110, 149, 137].

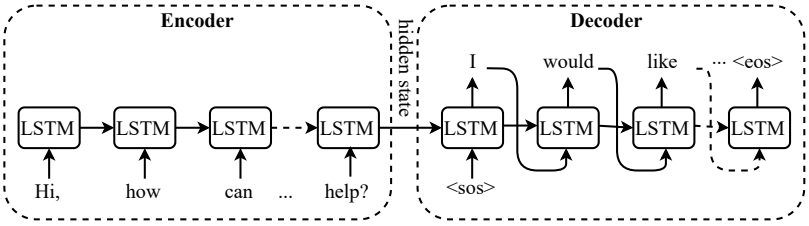


Figure 2.6: Overview of the Encoder-Decoder Framework.

We train ELMo with a bidirectional LSTM-based language models that can capture bidirectional context rather than context in one direction. As a result, contextualized embeddings are produced through concatenating the final and initial hidden states followed by their weighted summation. BERT employs the masked language model and the next sentence prediction tasks for training. The masked language model randomly masks ≈15% of the input tokens and the goal is to predict the masked tokens. Figure 2.5 presents this task. Similarly, the next sentence prediction task is to predict the likelihood that a given sentence B belongs after another sentence A or not. Consequently, the model learns robust, contextual, and generaliazble word embeddings and has shown state-of-the-art results on a wide range of NLU benchamrks. We utilize pre-trained language models in Chapters 3 and 4.

2.5 Encoder-Decoder Framework

Many NLP tasks like neural machine translation, abstractive text summarization, conversational AI, and question answering involve converting an input text sequence to another text sequence. Input and output sequence lengths typically have mismatching lengths in such tasks (commonly referred to as sequence transduction tasks), which makes building

models for such tasks challenging. We can use an encoder-decoder architecture to handle inputs and outputs of this type. An encoder-decoder model (e.g., seq2seq [144]) tries to generate a target sequence $\mathcal{Y} = (y_1, y_2, \cdots, y_m)$ given an input sequence $\mathcal{X} = (x_1, x_2, \cdots, x_n)$, where m and n are target and input sequence lengths, respectively. Figure 2.6 illustrates the encoder and decoder components of the seq2seq model. First, the encoder transforms the input sequence $\mathcal{X}$ into a sequence of hidden states $(h_1, h_2, \cdots, h_n)$ employing RNN units such as LSTM [56]. The encoder reads the input sequence, one token at a time, until the end of the input sequence token occurs and converts it to hidden state $h_i = Encoder(h_{i-1}, emb(x_i))$ by considering the word embedding of the input token x_i and the previous hidden state h_{i-1} at time-step i. $Encoder(.)$ is a non-linear mapping function and $emb(.)$ maps the given word into a high dimensional space. The decoder utilizes another RNN to generate the paraphrased (i.e., target) sequence $\mathcal{Y}$. The decoder is initialized with the last hidden state h_n, and generates one token at a time until the end of sentence token (i.e., $< eos >$) is generated. At time-step i, the generation is conditioned on the previously generated words $\hat{y}_{i-1}, \cdots, \hat{y}_1$ and the current decoder hidden state h'_i:

$$P(y_i|\hat{y}_{i-1}, \cdots, \hat{y}_1, \mathcal{X}) = softmax(Decoder(h'_i, y_{i-1})), \tag{2.1}$$

where $Decoder(.)$ is a non-linear mapping function and $softmax(.)$ converts the given vector into a probability distribution. Such an encoder-decoder model is typically trained by minimizing the negative log-likelihood of the input-target pairs. However, in the task of unsupervised paraphrasing we do not have access to target sequence, so we utilize the deep

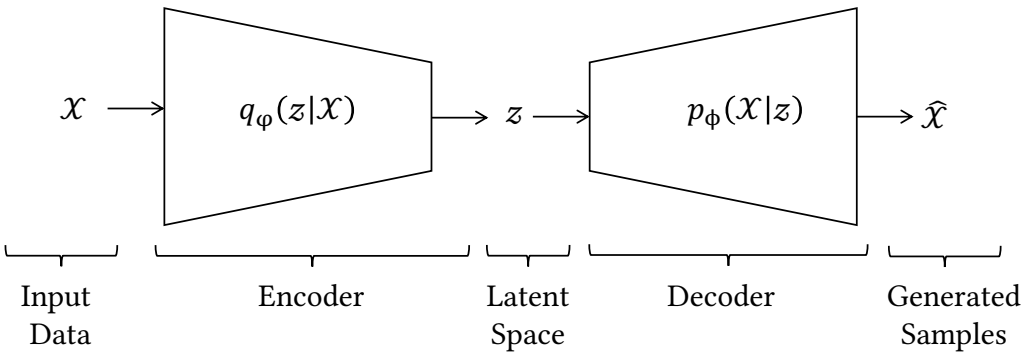


Figure 2.7: VAE learns a latent space z from observations $\mathcal{X}$, then draws samples from the learned latent space z.

reinforcement learning (DRL) paradigm coupled with the encoder-decoder framework in Chapter 5.

2.6 Variational Autoencoder

Variational Autoencoders (VAEs) [65, 119] are powerful generative models and have a wide range of applications from making realistic human-looking faces to creating synthetic music. Moreover, they have been used to generate text [15]. VAEs learn a nonlinear latent representation z from data points $\mathcal{X}$. It is trained in an unsupervised fashion for the following loss function:

$$\mathcal{L}(\varphi, \phi) = -\mathbb{E}_{q_\varphi(z|\mathcal{X})}\big[\log p_\phi(\mathcal{X}|z)\big] + \mathbb{KL}(q_\varphi(z|\mathcal{X})||p(z)), \tag{2.2}$$

where $q_\varphi(z|\mathcal{X})$ is the encoder with parameters φ that encodes the data points $\mathcal{X}$ into a stochastic latent representation z. $p_\phi(\mathcal{X}|z)$ is the decoder with the parameters ϕ that tries to generate an observation $\hat{\mathcal{X}}$ given the random latent code z where $p(z)$ is prior distribution,

i.e., standard normal distribution $\mathcal{N}(0, \mathrm{I})$. The first term in Equation 2.2 is the negative log-likelihood loss for the reconstruction of the data points $\mathcal{X}$. The second term is used to measure Kullback-Leibler ($\mathbb{KL}$) divergence between the encoder's distribution $q_\varphi(z|\mathcal{X})$ and the prior distribution $p(z)$. We can train a VAE model on a corpus in an unsupervised way and learn the corpus' latent representation z. At inference time, sentences can be sampled [15] from the learned latent representation z. VAE is also illustrated in Figure 2.7. In the unsupervised paraphrasing task, we employ a pre-trained VAE to provide a warm-start to the DRL-based paraphrasing model so that it does not start from a random policy in Chapter 5.

Chapter 3

Zero-shot Intent Detection

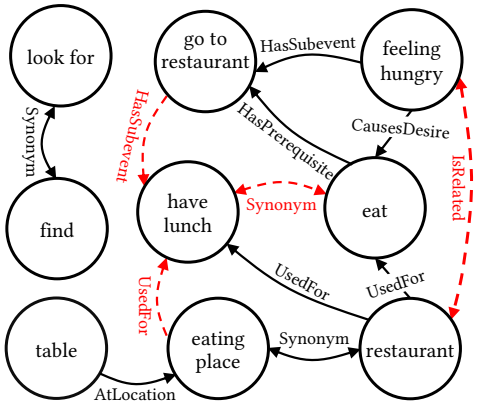
3.1 Introduction

Virtual assistants such as Amazon Alexa and Google Assistant allow users to perform a variety of tasks (e.g., Alexa Skills) through a natural language interface [59]. For example, a user can set an alarm by simply issuing the utterance *"Wake me up tomorrow at 10 AM"* to a virtual assistant, and the assistant is expected to understand that the user's intent (i.e., "AddAlarm") is to invoke the alarm module, then set the requested alarm accordingly. Intent detection is typically the first step towards performing any task in conversational systems and it is a challenging problem due to the vast diversity in user utterances. The challenge is further exacerbated in the more practically relevant setting where intents are added over time. This setting is an instance of the *generalized zero-shot classification problem* [36]: labeled training utterances are available only for seen intents but are unavailable for unseen ones, and at inference time, models do not have prior knowledge

on whether the utterances they receive imply seen or unseen intents. This setting is the focus of this work.

Little research has been conducted on building generalized zero-shot (GZS) models for intent detection, with little success. Earlier works [69, 21, 136] used zero-shot (ZS) learning to train an intent classification model that could classify utterances from unseen intent classes through transferring knowledge from seen classes. The test set in the standard ZS setting is not representative of the real world, as it exclusively includes samples from unseen classes (as opposed to having samples from both seen and unseen classes as in the GZS setting). ZS methods perform poorly in the GZS setting [19, 122], which is primarily caused by their strong bias towards seen classes; ZS intent detection models misclassify almost all test samples from unseen classes into seen ones [163, 162, 81].

To mitigate the issue of training data scarcity for unseen intents and ZS models' inability to effectively handle the GZS setting, we propose incorporating commonsense knowledge into a GZS intent detection model. We argue that such knowledge, if incorporated properly, helps overcome training data scarcity and allows detecting intents regardless of whether they are seen or not, given that commonsense knowledge is uniform across intents. We leverage ConceptNet [138] — a rich and widely-used commonsense knowledge graph (KG) that captures a large subset of knowledge in a semi-structured format (i.e., facts in the form $\langle$`head`, *relation*, `tail`$\rangle$ such as $\langle$`apple`, IsA, `fruit`$\rangle$). Given that ConceptNet is incomplete, similarly to other KGs, we pre-train a link predictor [63] that learns from an existing KG to infer novel edges (i.e., relationships) among nodes (i.e., head/tail) to overcome the missing information challenge.



(a) Link Predictor learns from KG (solid lines) to predict missing edges (dashed lines).

User Utterance:
`Look for something nearby, I am feeling hungry`

Intent Label:
`FindRestaurant`

Relationship Meta-features:

⟨look for, *Synonym*, find⟩	0.99
⋮	⋮
⟨feeling hungry, *IsRelated*, restaurant⟩	0.91
⋮	⋮

(b) Example utterance, intent, and computation of relationship meta-features that facilitate GZS intent detection.

Figure 3.1: A toy example for computing relationship meta-features using link predictor.

Figure 3.1a presents a toy commonsense KG where a link predictor can learn to infer missing facts such as ⟨`feeling hungry`, *IsRelated*, `restaurant`⟩ from existing facts such as ⟨`feeling hungry`, *CausesDesire*, `eat`⟩ and ⟨`restaurant`, *UsedFor*, `eat`⟩. We infuse the knowledge from our link predictor into our model by extracting *relationship meta-features*. These features quantify the level of relevance between an utterance and an intent in the form of relationship weights, where each weight describes the level of relatedness between the phrases in an utterance and an intent label based on a certain relationship type. Figure 3.1b shows an example utterance, an intent label, and an inference about the relationships between the phrases in the utterance and the intent label in the from of a relationship meta-features. Relationship meta-features augment embeddings using commonsense knowledge, which significantly reduces our model's reliance on the scarcely available seen intents training data. Furthermore, these features reduce our model's bias towards seen intents given that they are similarly computed for both seen and unseen intents; i.e., they are domain-oblivious.

Our model, RIDE[1], combines relationship meta-features with contextual word embeddings [109], and feeds the combined feature vectors into a trainable prediction function. RIDE is able to accurately detect both seen and unseen intents in utterances. Our extensive experimental analysis using the three widely used benchmarks `SNIPS` [23], `SGD` [115], and `MultiWOZ` [170] show that our model outperforms the state-of-the-art (SOTA) model in F1 scores on unseen intents in the GZS setting by at least 25.66%.

A secondary contribution of this work is that we managed to further improve the performance of GZS intent detection by employing Positive-Unlabeled (PU) learning [32] to predict if a new utterance belongs to a seen or unseen intent. PU learning assists intent detection models by mitigating their bias towards classifying most utterances into seen intents. A PU classifier is able to perform binary classification after being trained using only positive and unlabeled examples. We found out that the PU classifier also improves the performance of existing intent detection works. Our model, however, outperforms existing ones regardless of the PU classifier's integration.

3.2 Preliminaries

3.2.1 Intent Detection

Let $\mathcal{S} = \{\mathcal{I}_1, \cdots, \mathcal{I}_k\}$ be a set of seen intents and $\mathcal{U} = \{\mathcal{I}_{k+1}, \cdots, \mathcal{I}_n\}$ be a set of unseen intents where $\mathcal{S} \cap \mathcal{U} = \varnothing$. Let $\mathcal{X} = \{\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_m\}$ be a set of labeled training utterances where each training utterance $\mathcal{X}_i \in \mathcal{X}$ is described with a tuple $(\mathcal{X}_i, \mathcal{I}_j)$ such that $\mathcal{I}_j \in \mathcal{S}$. An intent $\mathcal{I}_j$ is comprised of an *Action* and an *Object* and takes the form

[1] RIDE: Relationship Meta-features Assisted Intent DEtection.

"ActionObject"[2] (e.g., "FindRestaurant"); an Action describes a user's request or activity and an Object describes the entity pointed to by an Action [22, 154, 152]. In both the zero-shot (ZS) and the GZS settings, the training examples have intent labels from set $\mathcal{S}$ only, however, the two settings differ as follows.

ZS Intent Detection. Given a test utterance $\mathcal{X}'_i$ whose true label $\mathcal{I}_j$ is known to be in $\mathcal{U}$ a priori, predict a label $\mathcal{I}'_j \in \mathcal{U}$.

GZS Intent Detection. Given a test utterance $\mathcal{X}'_i$, predict a label $\mathcal{I}'_j \in \mathcal{S} \cup \mathcal{U}$. Note that unlike in the ZS setting, it is not known whether the true label of $\mathcal{X}'_i$ belongs to $\mathcal{S}$ or $\mathcal{U}$, which exacerbates the challenge in this setting; we focus on this setting in this work.

3.2.2 ConceptNet Knowledge Graph

In this work, we use ConceptNet [138], which is a rich and widely-used commonsense knowledge graph. For tasks that involve commonsense reasoning such as *generalized zero-shot* intent detection, the ConceptNet [138] commonsense knowledge graph stands out as one of the most popular and freely available resources. It was employed to show state-of-the-art results at SemEval 2017 [139]. In this work, we considered 35 relation types to generate our relationship meta-features. The relation types are: $RelatedTo$, $FormOf$, IsA, $PartOf$, $HasA$, $UsedFor$, $CapableOf$, $AtLocation$, $Causes$, $HasSubevent$, $HasFirstSubevent$, $HasLastSubevent$, $HasPrerequisite$, $HasProperty$, $MotivatedByGoal$, $ObstructedBy$, $Desires$, $CreatedBy$, $Synonym$, $Antonym$, $DistinctFrom$, $DerivedFrom$, $SymbolOf$, $DefinedAs$, $MannerOf$, $LocatedNear$, $HasContext$, $SimilarTo$, $EtymologicallyRelatedTo$,

[2]If intents are described using a complex textual description, Actions and Objects can be extracted using existing NLP tools such as dependency parsers.

EtymologicallyDerivedFrom, *CausesDesire*, $MadeOf$, *ReceivesAction*, *ExternalURL*, and $Self$.

The relationship meta-feature generator produces $35 \times 4 = 140$ dimension vector for each utterance-intent pair. Specifically, we generate relationships: (*i*) from utterance to Object (i.e., Object part in intent label); (*ii*) utterance to Action (i.e., Action part in intent label); and (*iii*) Object to utterance; (*iv*) Action to utterance.

Knowledge Graph augmentation with background knowledge. A knowledge graph may not have redundant, but necessary information. For example, a knowledge graph may have the entry ⟨`movie`, IsA, `film`⟩ but not ⟨`film`, IsA, `movie`⟩ or vice-versa, because one triple can be inferred from the other based on background knowledge (i.e., symmetric nature of the IsA relation). Similarly, the triple ⟨`movie`, $HasA$, `subtitles`⟩ can be used to infer the triple ⟨`subtitles`, $PartOf$, `movie`⟩ based on the background knowledge (i.e., inverse relation between $HasA$ and $PartOf$). So, if this kind of redundant information (i.e., complementing entries for all such triples) is not available in the knowledge graph itself, there is no way for the model to learn these relationships automatically. To overcome this issue, we incorporate the background knowledge that each of the relation types IsA, *RelatedTo*, *Synonym*, *Antonym*, *DistinctFrom*, *LocatedNear*, *SimilarTo*, and *EtymologicallyRelatedTo* is symmetric; and that the relation types $PartOf$ and $HasA$ are inversely related in our link prediction model as described in [63].

3.2.3 Link Prediction

We pre-train a state-of-the-art link prediction model (LP) [63] on the augmented ConceptNet KG to score novel facts that are not necessarily present in the knowledge graph.

Given a triple (i.e., fact) in the form ⟨`head`, *relation*, `tail`⟩, the pre-trained link prediction model scores the triple with a value between 0 and 1, which quantifies the level of validity of the given triple. The training data for a link prediction model is prepared as follows. First, the triples in the input knowledge graph are assigned a label of 1. Then, negative examples are generated by corrupting true triples (i.e., modifying the `head` or `tail` of existing triples) and assigning them a label of -1 [13]. Finally, we train our LP using the generated training data by minimizing the L2 regularized negative log-likelihood loss of training triples [150].

3.2.4 Positive-Unlabeled Learning

Positive-Unlabeled (PU) classifiers learn a standard binary classifier in the unconventional setting where labeled negative training examples are unavailable. The state-of-the-art PU classifier [32], which we integrate into our model, learns a decision boundary based on the positive and unlabeled examples, and thus can classify novel test examples into positive or negative. The aim of the PU classifier is to learn a probabilistic function $f(\mathcal{X}_i)$ that estimates $P(\mathcal{I}_j \in \mathcal{S} \mid \mathcal{X}_i)$ as closely as possible. In this work, we train a PU classifier using our training set (utterances with only seen intents labeled as positive) and validation set (utterances with both seen and unseen intents as unlabeled). We use 512-dimensions sentence embedding as features when using the PU classifier, generated using a pre-trained universal sentence encoder [18].

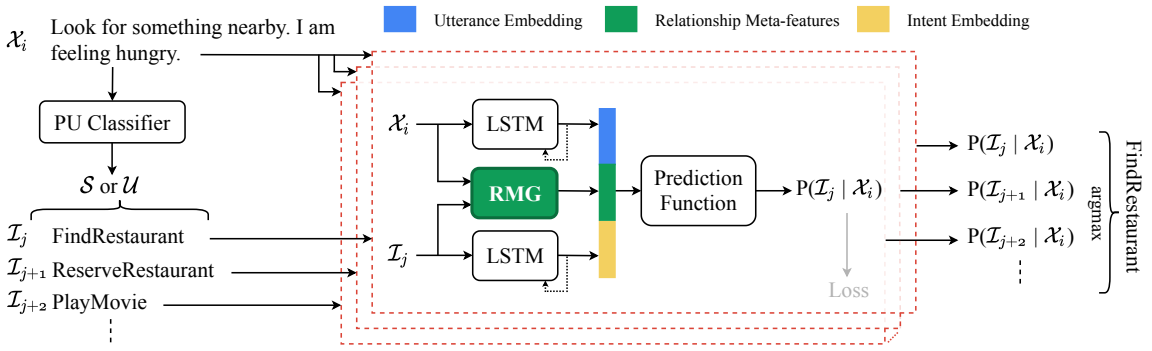


Figure 3.2: Overview of RIDE.

3.3 Our Approach

Figure 3.2 shows an overview of our model: given an input utterance $\mathcal{X}_i$, we first invoke the PU classifier (if it is available) to predict whether $\mathcal{X}_i$ implies a seen or an unseen intent; i.e., whether $\mathcal{X}_i$'s intent belongs to set $\mathcal{S}$ or $\mathcal{U}$. Then, an instance of our core model (the red box in Figure 3.2) is invoked for each intent in $\mathcal{S}$ or $\mathcal{U}$ based on the PU's prediction. Our core model predicts the level of compatibility between the given utterance $\mathcal{X}_i$ and intent $\mathcal{I}_j$, i.e., the probability that the given utterance implies the given intent $P(\mathcal{I}_j|\mathcal{X}_i) \in [0, 1]$. Finally, our model outputs the intent with the highest compatibility probability, i.e., $\text{argmax}_{\mathcal{I}_j} P(\mathcal{I}_j|\mathcal{X}_i)$.

Our core model concatenates relationship meta-features, utterance embedding, and intent embedding and feeds them into a trainable prediction function. The Relationship Meta-features Generator (RMG) is at the heart of our model, and it is the most influential component. Given an utterance and an intent, RMG generates meta-features that capture deep semantic associations between the given utterance and intent in the form of a meta-feature vector.

3.3.1 Relationship Meta-feature Generation

RMG extracts relationship meta-features by utilizing the "ActionObject" structure of intent labels and commonsense knowledge graphs. Relationship meta-features are not only generalizable, but also discriminative: while the example utterance "Look for something nearby. I am feeling hungry." may be related to the intent "ReserveRestaurant", the Action part of this intent is not related to any phrase in the utterance; thus, "ReserveRestaurant" is less related to the utterance than "FindRestaurant".

RMG takes the following inputs: a set of relations in a knowledge graph (35 in the case of ConceptNet) $\mathcal{R} = \{r_1, r_2, .., r_t\}$; the set of n-grams $\mathcal{G}_i = \{g_1, g_2, .., g_q\}$ that correspond to the input utterance $\mathcal{X}_i$, where $|\mathcal{G}| = q$; and an intent label $\mathcal{I}_j = \{\mathcal{A}, \mathcal{O}\}$, where $\mathcal{A}$ and $\mathcal{O}$ are the Action and Object components of the intent, respectively. RMG computes a relationship meta-features vector in four steps, where each step results in a vector of size $|\mathcal{R}|$. The smaller vectors are: $\mathbf{e}^{\overrightarrow{\mathcal{A}}}_{\mathcal{X}_i}$, $\mathbf{e}^{\overrightarrow{\mathcal{O}}}_{\mathcal{X}_i}$, $\mathbf{e}^{\overleftarrow{\mathcal{A}}}_{\mathcal{X}_i}$, and $\mathbf{e}^{\overleftarrow{\mathcal{O}}}_{\mathcal{X}_i}$, where $\mathbf{e}^{\overrightarrow{\mathcal{A}}}_{\mathcal{X}_i}$ captures the Action to utterance semantic relationships and $\mathbf{e}^{\overrightarrow{\mathcal{O}}}_{\mathcal{X}_i}$ captures the Object to utterance relationships. The remaining two vectors capture relationships in the other direction; i.e., utterance to Action/Object, respectively. Capturing bi-directional relationships is important because a relationship in one direction does not necessarily imply one in the other direction – for example, ⟨`table`, *AtLocation*, `restaurant`⟩ does not imply ⟨`restaurant`, *AtLocation*, `table`⟩. The final output of RMG is the relationship meta-features vector $\mathbf{e}_{relationship}$, which is the concatenation of the four aforementioned vectors. We explain next how the smaller vectors is computed.

RMG computes $\mathbf{e}^{\overrightarrow{\mathcal{A}}}_{\mathcal{X}_i}$ by considering the strength of each relation in $\mathcal{R}$ between $\mathcal{A}$ and each n-gram in $\mathcal{G}_i$. That is, $\mathbf{e}^{\overrightarrow{\mathcal{A}}}_{\mathcal{X}_i}$ has $|\mathcal{R}|$ cells, where each cell corresponds to a relation $r \in \mathcal{R}$.

Algorithm 1: RMG

Input: $\mathcal{R} = \{r_1, \cdots, r_t\}$: relations in KG
$\mathcal{G}_i = \{g_1, \cdots, g_q\}$: utterance n-grams
$\mathcal{I}_j = \{\mathcal{A}, \mathcal{O}\}$: intent's Action and Object

Output: $\mathbf{e}_{relationship}$: $\mathcal{X}_i$-$\mathcal{I}_j$ relationship meta-features

Let $\mathbf{e}_{\mathcal{X}_i}^{\overrightarrow{\mathcal{A}}} = \texttt{RM}\ (\mathcal{A}, \mathcal{G}_i, \rightarrow)$ // Action to utterance
Let $\mathbf{e}_{\mathcal{X}_i}^{\overrightarrow{\mathcal{O}}} = \texttt{RM}\ (\mathcal{O}, \mathcal{G}_i, \rightarrow)$ // Object to utterance
Let $\mathbf{e}_{\mathcal{X}_i}^{\overleftarrow{\mathcal{A}}} = \texttt{RM}\ (\mathcal{A}, \mathcal{G}_i, \leftarrow)$ // utterance to Action
Let $\mathbf{e}_{\mathcal{X}_i}^{\overleftarrow{\mathcal{O}}} = \texttt{RM}\ (\mathcal{O}, \mathcal{G}_i, \leftarrow)$ // utterance to Object
Let $\mathbf{e}_{relationship} = [\mathbf{e}_{\mathcal{X}_i}^{\overrightarrow{\mathcal{A}}}, \mathbf{e}_{\mathcal{X}_i}^{\overrightarrow{\mathcal{O}}}, \mathbf{e}_{\mathcal{X}_i}^{\overleftarrow{\mathcal{A}}}, \mathbf{e}_{\mathcal{X}_i}^{\overleftarrow{\mathcal{O}}}]$
return $\mathbf{e}_{relationship}$
Function RM(*concept, phrases, direction*)**:**
    Let $\mathbf{e} = []$
    foreach $r \in \mathcal{R}$ **do**
        if *direction* $= \rightarrow$ **then**
            Let $p = \texttt{Max}\ (LP(concept, r, g))$ for $g \in phrases$
        if *direction* $= \leftarrow$ **then**
            Let $p = \texttt{Max}\ (LP(g, r, concept))$ for $g \in phrases$
        $\mathbf{e}$.append(p)
    return $\mathbf{e}$

Each cell is computed by taking $max(LP(\mathcal{A}, r, g))$ over all $g \in \mathcal{G}_i$. $LP(\texttt{head}, relation, \texttt{tail})$ outputs the probability that the fact represented by the triple $\langle \texttt{head}, relation, \texttt{tail} \rangle$ exists. The vector $\mathbf{e}_{\mathcal{X}_i}^{\overrightarrow{\mathcal{O}}}$ is computed similarly, but with passing $\mathcal{O}$ instead of $\mathcal{A}$ when invoking the link predictor; i.e., taking $max(LP(\mathcal{O}, r, g))$ over all $g \in \mathcal{G}_i$ to compute each cell. The vectors $\mathbf{e}_{\mathcal{X}_i}^{\overleftarrow{\mathcal{A}}}$ and $\mathbf{e}_{\mathcal{X}_i}^{\overleftarrow{\mathcal{O}}}$ are computed similarly, but with swapping the `head` and `tail` when invoking the link predictor; i.e., utterance phrases are passed as `head` and Action/Object parts are passed as `tail`. Algorithm 1 outlines the previous process and Figure 3.3 illustrates the computation $\mathbf{e}_{\mathcal{X}_i}^{\overrightarrow{\mathcal{O}}}$ for an example utterance-intent pair. Finally, the generated meta-features are passed through a linear layer with sigmoid activation before concatenation with the utterance and intent embeddings.

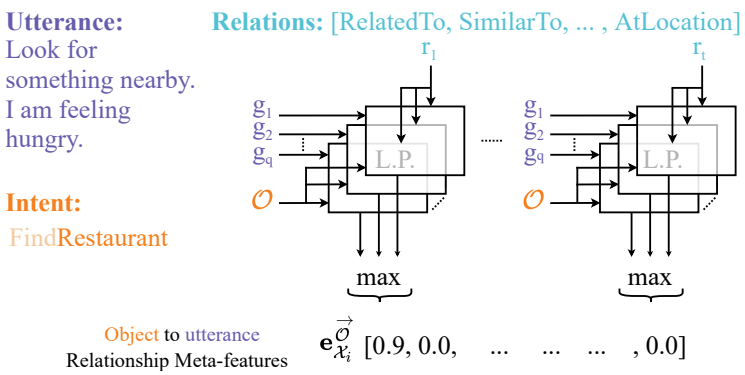


Figure 3.3: Illustration of the computation of $\mathbf{e}^{\vec{\mathcal{O}}}_{\mathcal{X}_i}$ for an example utterance-intent pair. Intent's Object "Restaurant" is related to the word "hungry" in the utterance; thus, the cell corresponding to relation *RelatedTo* (the first cell in $\mathbf{e}^{\vec{\mathcal{O}}}_{\mathcal{X}_i}$) has a high probability value.

3.3.2 Utterance and Intent Encoders

Given an utterance $\mathcal{X}_i = \{w_1, w_2, \cdots, w_u\}$ with u words, first we compute an embedding $emb(w_j) \in \mathbb{R}^{dim}$ for each word w_j in the utterance, where $emb(w_j)$ is the concatenation of a contextual embedding obtained from a pre-trained ELMo model and parts of speech (POS) tag embedding. Then, we use bi-directional LSTM to produce a d-dimensional representation as follows:

$$\overrightarrow{\mathbf{h}}_j = \text{LSTM}_{fw}(\overrightarrow{\mathbf{h}}_{j-1}, emb(w_j)). \tag{3.1}$$

$$\overleftarrow{\mathbf{h}}_j = \text{LSTM}_{bw}(\overleftarrow{\mathbf{h}}_{j-1}, emb(w_j)). \tag{3.2}$$

Finally, we concatenate the output of the last hidden states as utterance embedding $\mathbf{e}_{utterance} = [\overrightarrow{\mathbf{h}}_u; \overleftarrow{\mathbf{h}}_u] \in \mathbb{R}^d$. We encode intent labels similarly to produce an intent embedding $\mathbf{e}_{intent} \in \mathbb{R}^d$.

3.3.3 Training the Model

Our model has two trainable components: the LSTM units in the utterance and intent encoders and the compatibility probability prediction function. We jointly train these components using training data prepared as follows. The training examples are of the form $((\mathcal{X}_i, \mathcal{I}_j), \mathcal{Y})$, where $\mathcal{Y}$ is a binary label representing whether the utterance-intent pair $(\mathcal{X}_i, \mathcal{I}_j)$ are compatible: 1 means they are compatible, and 0 means they are not. For example, the utterance-intent pair ("I want to play this song", "PlaySong") gets a label of 1, and the same utterance paired with another intent such as "BookHotel" gets a label of 0. We prepare our training data by assigning a label of 1 to the available utterance-intent pairs (where intents are seen ones); these constitute positive training examples. We create a negative training example for each positive one by corrupting the example's intent. We corrupt intents by modifying their Action, Object, or both; for example, the utterance-intent pair ("Look for something nearby. I am hungry.", "FindRestaurant") may result in the negative examples (..., "ReserveRestaurant"), (..., "FindHotel"), or (...,"RentMovies"). We train our core model by minimizing the cross-entropy loss over all the training examples.

3.4 Experimental Setup

In this section, we describe the datasets, evaluation settings and metrics, competing methods, and implementation details of our proposed method.

3.4.1 Datasets

Table 3.1 presents important statistics on the datasets we used in our experiments.

Table 3.1: Datasets statistics.

Dataset	SNIPS	SGD	MultiWOZ
Dataset Size	14.2K	57.2K	30.0K
Vocab. Size	10.8K	8.8K	9.7K
Avg. Length	9.05	10.62	11.07
# of Intents	7	46	11

SNIPS [23]. A crowd-sourced single-turn NLU benchmark with 7 intents across different domains.

SGD [115]. A recently published dataset for the eighth Dialog System Technology Challenge, Schema Guided Dialogue (`SGD`) track. It contains dialogues from 16 domains with a total of 46 intents. It is one of the most comprehensive and challenging publicly available datasets. The dialogues contain user intents as part of dialogue states. We only kept utterances where users express an intent by comparing two consecutive dialogue states to check for the expression of a new intent.

MultiWOZ [170]. Multi-Domain Wizard-of-Oz (`MultiWOZ`) is a well-known and publicly available dataset. We used the most recent version 2.2 of `MultiWOZ` in our experiments, which contains utterances spanning 11 intents. Similarly to the pre-processing of `SGD` dataset, we only kept utterances that express an intent to maintain consistency with the previous work.

3.4.2 Datasets Preprocessing

SNIPS Natural Language Understanding benchmark (`SNIPS`) [23] is a commonly used dataset for intent detection, whereas Dialogue System Technology Challenge 8, Schema Guided Dialogue dataset (`SGD`) [115] and Multi-Domain Wizard-of-Oz (`MultiWOZ`) [170]

were originally proposed for the task of dialogue state tracking. For SGD and MultiWOZ, we perform a few trivial preprocessing steps to extract utterances that contain intents, along with their labels, and use them for the task of generalized zero-shot intent detection. First, we provide the details on the preprocessing steps specific to the SGD and MultiWOZ dataset and then describe the preprocessing steps that are common for all datasets.

Steps for SGD and MultiWOZ. To maintain consistency with the previous work on intent detection, we extract only the utterances where user/system expresses an intent, and discard the rest from the original SGD and MultiWOZ datasets. The dialogue state contains a property "active_intent" that keeps track of the user's current intent. After each user utterance, we compare dialogue states to check for the expression of a new user intent, i.e., whether the value of the "active_intent" is modified. Whenever the user expresses a new intent, the value of the "active_intent" is updated. Moreover, sometimes, the bot (i.e., system) also offers new intents to the user (e.g., offering reserving a table to the user, who has successfully searched for a restaurant), which is tracked in the system actions property "act = OFFER_INTENT", and "values = <new_intent>". We also keep such system utterances.

Common Steps. We perform some standard preprocessing steps on all the datasets. We use spaCy to tokenize the sentences. Since intent labels are given in the "ActionObject" format, we tokenize them into "Action Object" phrases before feeding them into our model. For example, the intent labels "FindHotel" and "RateBook", are transformed into "Find Hotel" and "Rate Book", respectively. Note that some Objects parts of intent labels are compound. Consider the intent label "SearchOneWayFlight" whose Action is "Search" and Object is "OneWayFlight". In such cases, our relationship meta-features generator computes

meta-features for each part of the compound object then averages them to produce the Object meta-features vector. In the previous example, "OneWayFlight" meta-features vector is computed as the average of the meta-features of "OneWay" and "Flight".

3.4.3 Evaluation Methodology

We use standard classification evaluation measures: accuracy and F1 score. The values for all the metrics are per class averages weighted by their respective support. We present evaluation results for the following intent detection settings:

ZS intent Detection. In this setting, a model is trained on all the utterances with seen intents – i.e., all samples $(\mathcal{X}_i, \mathcal{I}_j)$ where $\mathcal{I}_j \in \mathcal{S}$. Whereas at inference time, the utterances are only drawn from those with unseen intents; the model has to classify a given utterance into one of the unseen intents. Note that this setting is less challenging than the GZS setting because models know that utterances received at inference time imply intents that belong to the set of unseen intents only, thus naturally reducing their bias towards classifying utterances into seen intents. For each dataset, we randomly place $\approx 25\%$, $\approx 50\%$, and $\approx 75\%$ of the intents in the seen set for training and the rest into the unseen set for testing, and report the average results over 10 runs. It is important to highlight that selecting seen/unseen sets in this fashion is more challenging to models because all the intents get an equal chance to appear in the unseen set, which exposes models that are capable of detecting certain unseen intents only.

GZS intent Detection. In this setting, models are trained on a subset of utterances implying seen intents. At inference time, test utterances are drawn from a set that contains utterances implying a mix of seen and unseen intents (disjoint set from training set) and the

model is expected to select the correct intent from all seen and unseen intents for a given test utterance. This is the most realistic and challenging problem setting, and it is the main focus of this work. For the GZS setting, we decided the train/test splits for each dataset as follows: For `SNIPS`, we first randomly selected 5 out of 7 intents and designated them as seen intents – the remaining 2 intents were designated as unseen intents. We then selected 70% of the utterances that imply any of the 5 seen intents for training. The test set consists of the remaining 30% utterances in addition to all utterances that imply one of the 2 unseen intents. Previous work [80] used the same number of seen/unseen intents, but selected the seen/unseen intents manually. Whereas we picked unseen intents randomly, and we report results over 10 runs resulting in a more challenging and thorough evaluation. That is, each intent gets an equal chance to appear as an unseen intent in our experiments, which allows testing each model more comprehensively. For `SGD`, we used the standard splits proposed by the dataset authors. Specifically, the test set includes utterances that imply 8 unseen intents and 26 seen intents; we report average results over 10 runs. For `MultiWOZ`, we used 70% of the utterance that imply 8 (out of 11) randomly selected intents for training and the rest of the utterances (i.e., the remaining 30% of seen intents' utterances and all utterances implying unseen intents) for testing.

3.4.4 Competing Methods

We compare our model RIDE against the following state-of-the-art (SOTA) models and several strong baselines:

SEG [165]: A semantic-enhanced Gaussian mixture model that uses large margin loss to learn class-concentrated embeddings coupled with a density-based outlier detection algorithm LOF to detect unseen intents.

ReCapsNet-ZS [80]: A model that employs capsule neural network and a dimensional attention module to learn generalizable transformational metrices from seen intents.

IntentCapsNet [161]: A model that utilizes capsule neural networks to learn low-level features and routing-by-agreement to adapt to unseen intents. This model was originally proposed for detecting intents in the standard ZS setting. We extended it to support the GZS setting with the help of its authors.

Other Baseline Models: We extend the following baseline ZS models to support GZS setting.

(i) Zero-shot DDN [69]: A model for ZS intent detection that achieves zero-shot capabilities by projecting utterances and intent labels into the same high dimensional embedding space.

(ii) CDSSM [21]: A model for ZS intent detection that utilizes a convolutional deep structured semantic model to generate embeddings for unseen intents.

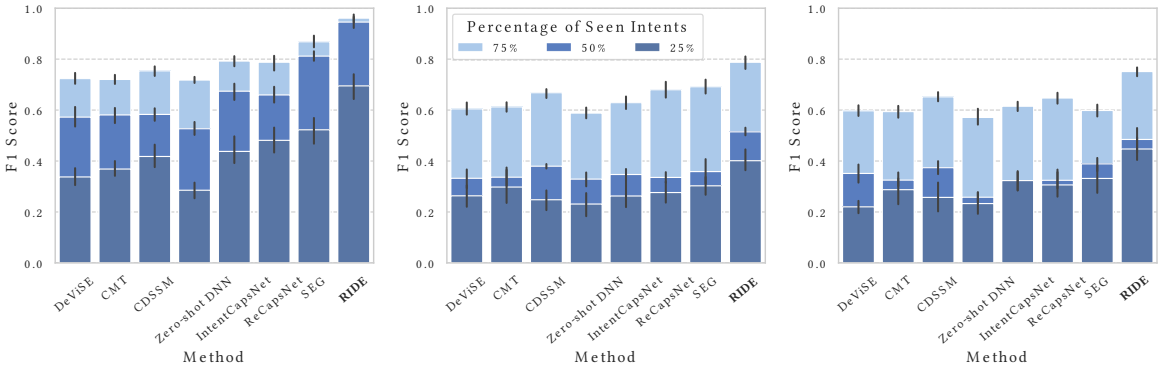
(iii) CMT [136]: A model for ZS intent detection that employs non-linearity in the compatibility function between utterances and intents to find the most compatible unseen intents.

(iv) DeViSE [40]: A model that was originally proposed for zero-shot image

classification that learns a linear compatibility function. Note that baseline ZS models have been extended to support GZS setting.

3.4.5 Implementation Details

We lemmatize ConeptNet KG, that has 1 million nodes (English only after lemmatization), 2.7 million edges, and 35 relation types. The link predictor is trained on the lemmatized version of ConceptNet KG. The link predictor has two 200-dimensional embedding layers and a negative sampling ratio of 10; it is trained for $1,000$ epochs using Adam optimizer with a learning rate of 0.05, L2 regularization value of 0.1, and batch size of 4800. Our relationship meta-features generator takes in an utterance's n-grams with $n \leq 4$ and an intent label, and uses the pre-trained link predictor to produce relationship meta-features with 140 dimensions. Our utterance and intent encoders use pre-trained ELMo contextual word embeddings with 1024 dimension and POS tags embeddings with 300 dimension, and a two-layer RNN with 300-dimensional bidirectional LSTM as recurrent units. Our prediction function has two dense layers with relu and softmax activation. Our core model is trained for up to 200 epochs with early stopping using Adam optimizer and a cross entropy loss with initial learning rate of 0.001 and ReduceLROnPlateau scheduler [112] with 20 patience epochs. It uses dropout rate of 0.3 and batch size of 32. A negative sampling ratio of up to 6 is used. We use the same embeddings generation and training mechanism for all competing models.



(a) SNIPS dataset (b) SGD dataset (c) MultiWOZ dataset

Figure 3.4: F1 scores for competing models in the ZS setting with varying percentages of seen intents. In the ZS setting, utterances with only unseen intents are encountered at inference time, and models are aware of this. Our model RIDE consistently outperforms all other models for any given percentage of seen intents.

3.5 Results

Standard ZS Intent Detection. Figure 3.4 presents the F1 scores averaged over 10 runs for all competing models with varying percentages of seen intents in the ZS setting. The performance of all models improves as the percentage of seen intents increases, which is expected because increasing the percentage of seen intent gives models access to more training data and intents. Our model RIDE consistently outperforms the SOTA model SEG [165] and all other models in the ZS setting with a large margin across all the datasets. Specifically, it is at least 12.65% more accurate on F1 score than the second best model for any percentage of seen intents on all the datasets. Note that all models perform worse on `SGD` and `MultiWOZ` compared to `SNIPS` because these two datasets are more challenging: they contain closely related intent labels such as "FindRestaurant" and "FindHotel".

Table 3.2: Accuracy and F1 scores for competing models in the GZS setting for SNIPS dataset.

Method	**Unseen**		**Seen**	
	Accuracy	F1	Accuracy	F1
DeViSE	0.0311	0.0439	0.9487	0.6521
CMT	0.0427	0.0910	**0.9751**	0.6639
CDSSM	0.0521	0.0484	0.9542	0.7028
Zero-shot DNN	0.0912	0.1273	0.9437	0.6687
IntentCapsNet	0.0000	0.0000	0.9749	0.6532
ReCapsNet	0.1249	0.1601	0.9513	0.6783
SEG	0.6943	0.6991	0.8643	0.8651
RIDE w/o PU	0.8728	0.9103	0.8906	0.8799
RIDE $/w$ PU	**0.9051**	**0.9254**	0.9179	**0.9080**

Table 3.3: Accuracy and F1 scores for competing models in the GZS setting for SGD dataset.

Method	**Unseen**		**Seen**	
	Accuracy	F1	Accuracy	F1
DeViSE	0.0197	0.0177	0.8390	0.5451
CMT	0.0254	0.0621	**0.9014**	0.5803
CDSSM	0.0367	0.0284	0.8890	0.6379
Zero-shot DNN	0.0662	0.1168	0.8825	0.6098
IntentCapsNet	0.0000	0.0000	0.8982	0.5508
ReCapsNet	0.1062	0.1331	0.8762	0.5751
SEG	0.3723	0.4032	0.6134	0.6356
RIDE w/o PU	0.3865	0.4634	0.8126	0.8295
RIDE $/w$ PU	**0.5901**	**0.5734**	0.8315	**0.8298**

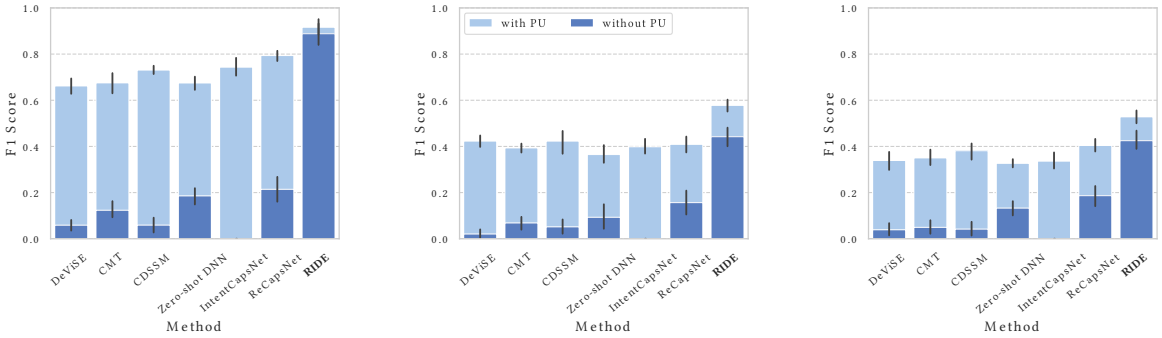
GZS Intent Detection. Table 3.2, 3.3, and 3.4 show accuracy and F1 scores averaged over 10 runs for all competing models in the GZS setting for `SNIPS`, `SGD`, and `MultiWOZ` datasets, respectively. Recall that GZS models receive both seen and unseen intents at inference time, which makes the setting more challenging than the ZS setting) We present results for two variants of our model: RIDE w/o PU which does not use a PU classifier, and RIDE $/w$ PU which uses one. Our model consistently achieves the best F1 score for both seen and unseen intents across all datasets, regardless of whether the PU classifier is integrated or not. For unseen intents, our model RIDE outperforms all other competing

Table 3.4: Accuracy and F1 scores for competing models in the GZS setting for MultiWOZ dataset.

Method	**Unseen**		**Seen**	
	Accuracy	F1	Accuracy	F1
DeViSE	0.0119	0.0270	0.8980	0.5770
CMT	0.0253	0.0679	<u>0.9025</u>	0.6216
CDSSM	0.0373	0.0244	0.8861	0.6515
Zero-shot DNN	0.0802	0.1149	0.8940	0.6012
IntentCapsNet	0.0000	0.0000	**0.9249**	0.6038
ReCapsNet	0.1081	0.1467	0.8715	0.6170
SEG	<u>0.3712</u>	0.4143	0.6523	0.6456
RIDE w/o PU	0.3704	0.4645	0.8558	<u>0.8816</u>
RIDE $/w$ PU	**0.5686**	0.5206	0.8844	**0.8847**

models on accuracy with a large margin. Specifically, RIDE is 30.36%, 58.50%, and 53.18% more accurate than the SOTA model SEG on `SNIPS`, `SGD`, and `MultiWOZ` for unseen intents, respectively. Moreover, our model consistently achieves the highest F1 score on seen as well as unseen intents, which confirms its generalizability. CMT and IntentCapsNet achieve the highest accuracy for utterances with seen intents on all datasets, but their F1 score is among the worst due to their biased-ness towards misclassifying utterances with unseen intents into seen ones. RIDE outperforms the SOTA model SEG regardless of whether a PU classifier is incorporated or not. For `SNIPS`, the role of the PU classifier is negligible as it causes a slight improvement in accuracy and F1 score. For `SGD` and `MultiWOZ`, which are more challenging datasets, the PU classifier is responsible for significant improvements in accuracy. Specifically, it provides 20.36 and 19.82 percentage points improvement for `SGD` and `MultiWOZ`, respectively, on unseen intents.

Effect of PU Classifier on Other Models. We observed that one of the main sources of error for most models in the GZS setting is their tendency to misclassify utterances with



(a) SNIPS dataset (b) SGD dataset (c) MultiWOZ dataset

Figure 3.5: F1 scores for unseen intents for the competing models in the GZS setting after integrating a PU classifier.

unseen intents into seen ones due to overfitting to seen intents. We investigated whether existing models can be adapted to accurately classify utterances with unseen intents by partially eliminating their bias towards seen intents. Figure 3.5 presents F1 scores of all the models with and without PU classifier. A PU classifier significantly improves the results of all the competing models. For instance, the IntentCapsNet model with a PU classifier achieves an F1 score of 74% for unseen intents on `SNIPS` dataset in the GZS setting compared to an F1 score of less than 0.01% without the PU classifier. Note that the PU classifier has an accuracy (i.e., correctly predicting whether the utterance implies a seen or an unseen intent) of 93.69 and an F1 score of 93.75 for `SNIPS` dataset; 86.13 accuracy and 83.54 F1 score for `SGD` dataset; and 87.32 accuracy and 88.51 F1 score for `MultiWOZ` dataset. Interestingly, our model RIDE (without PU classifier) outperforms all the competing models even when a PU classifier is incorporated into them, which highlights that the PU classifier is not the main source of the performance of our model. We did not incorporate the PU classifier into SEG model because it already incorporates an equivalent mechanism to distinguish seen intents from unseen ones (i.e., outlier detection).

Table 3.5: Ablation study: F1 scores for unseen intents in GZS setting; the key reason behind our model's astonishing accuracy is our relationship meta-features.

Configuration	SNIPS	SGD	MultiWOZ
UI-Embed *w/o* PU	0.2367	0.1578	0.1723
Rel-M *w/o* PU	0.7103	0.3593	0.3321
RIDE *w/o* PU	<u>0.9103</u>	0.4634	0.4645
UI-Embed */w* PU	0.7245	0.4202	0.4124
Rel-M */w* PU	0.8463	<u>0.5167</u>	<u>0.4781</u>
RIDE */w* PU	**0.9254**	**0.5734**	**0.5206**

Ablation Study. To quantify the effectiveness of each component in our model, we present the results of our ablation study in Table 3.5 (in the GZS setting). Utilizing utterance and intent embeddings only (i.e., UI-Embed) results in very low F1 score, i.e., 23.67% on `SNIPS` dataset. Employing relationship meta-features only (i.e., Rel-M) results in significantly better results: an F1 score of 71.03% on `SNIPS` dataset. When utterance and intent embeddings are used in conjunction with relationship meta-features (i.e., RIDE *w/o* PU), it achieves better F1 score compared to the Rel-M or UI-Embed configurations. A similar trend can be observed for the other datasets as well. Finally, when our entire model is deployed (i.e., including utterance and intent embeddings, relationship meta-features, and the PU classifier, i.e., RIDE */w* PU), it achieves the best results on all the datasets.

Intent Existence Prediction. In real human-to-human or human-to-machine conversations, utterances do not necessarily imply intents. Most existing intent detection models formulate the problem as a classification problem where utterances are assumed to imply an intent, which limits utility of such models in practice. In what follows, we describe a simple method for extending intent detection models (including ours) to accommodate the case when utterances do not necessarily imply intents. We propose to do binary classification as a first step in intent detection, where a binary classifier is used to identify utterances that

Table 3.6: F1 score for intent existence binary classifiers.

Method	SGD	MultiWOZ
CNN	0.9497	0.9512
GRU	**0.9528**	0.9619
LSTM	0.9512	0.9607
Bi-LSTM	0.9525	**0.9621**

do not imply an intent. To validate the viability of this proposal, we experimented with several binary classifiers. To train the classifiers, we created datasets of positive and negative examples from seen intents data; positive examples are utterances that imply intents, and negative examples are utterances that do not have intents (See Section 3.4.2 for details on identifying utterances that imply intents). For the `SGD` dataset, we used the standard train/test splits, and for the `MultiWOZ` dataset, we used the same splits described in the GZS setting. We report in Table 3.6 the average F1 score over 5 runs of several binary classifiers for the `SGD` and the `MultiWOZ` datasets. All classifiers use ELMo [109] and POS tag embeddings. These results show that intent existence classification can be done accurately using the available training data; consequently, intent detection models can be easily and reliably extended to support the case when some input utterances do not imply intents. Pre-trained LMs with appropriate fine-tuning have produced state-of-the-art results on many NLP benchmarks [113, 14, 53, 110, 149, 137]..

3.6 Related Work

We organize the related work on intent detection into three categories: (*i*) supervised intent detection, (*ii*) standard zero-shot intent detection, and (*iii*) generalized zero-shot intent detection.

Supervised Intent Detection. Recurrent neural networks [116] and semantic lexicon-enriched word embeddings [64] have been employed for supervised intent detection. Recently, researchers have proposed solving the related problems of intent detection and slot-filling jointly [79, 171, 164]. Supervised intent classification works assume the availability of a large amount of labeled training data for all intents to learn discriminative features, whereas we focus on the more challenging and more practically relevant setting where intents are evolving and training data is not available for all intents.

Standard Zero-shot Intent Detection. The authors in [168] proposed using label ontologies [37] (i.e., manually annotated intent attributes) to facilitate generalizing a model to support unseen intents. The authors in [26, 69, 158] map utterances and intents to the same semantic vector space, and then classify utterances based on their proximity to intent labels in that space. Similarly, the authors in [41] employ the outlier detection algorithm LOF [16] and likelihood ratios for identifying out-of-domain test examples. While these works showed promising results for intent detection when training data is unavailable for some intents, they assume that all utterances faced at inference time imply unseen intents only. Extending such works to remove the aforementioned assumption is nontrivial. Our model does not assume knowledge of whether an utterance implies a seen or an unseen intent at inference time.

Generalized Zero-shot Intent Detection. To the best of our knowledge, the authors in [80] proposed the first work that specifically targets the GZS intent detection setting. They attempt to make their model generalizable to unseen intents by adding a dimensional attention module to a capsule network and learning generalizable transformation matrices

from seen intents. Recently, the authors in [165] proposed using a density-based outlier detection algorithm LOF [16] and semantic-enhanced Gaussian mixture model with large margin loss to learn class-concentrated embeddings to detect unseen intents. In contrast, we leverage rich commonsense knowledge graph to capture deep semantic and discriminative relationships between utterances and intents, which significantly reduces the bias towards classifying unseen intents into seen ones. In a related, but orthogonal, line of research, the authors in [90, 75, 47] addressed the problem of intent detection in the context of dialog state tracking where dialog state and conversation 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.

3.7 Conclusion

We have presented an accurate generalized zero-shot intent detection model. Our extensive experimental analysis on three intent detection benchmarks shows that our model is 30.36% to 58.50% more accurate than the SOTA model for unseen intents. The main novelty of our model is its utilization of relationship meta-features to accurately identify matching utterance-intent pairs with very limited reliance on training data, and without making any assumption on whether utterances imply seen or unseen intents at inference time. Furthermore, our idea of integrating Positive-Unlabeled learning in GZS intent detection models further improves our models' performance, and significantly improves the accuracy of existing models as well.

Chapter 4

Zero-shot Slot Filling

4.1 Introduction

We discussed in Chapter 3 that goal-oriented dialog systems allow users to accomplish tasks through an intuitive natural language interface. 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, after the successful detection of the user's intent (see Chapter 3), they also 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 4.1 for an example). Despite the challenges, supervised approaches have shown promising results for this task [44, 171, 169, 7, 96, 70, 51, 164]. 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 [5]. `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 previous 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 [127] proposed a cross-domain zero-shot adaptation for slot filling by utilizing example slot values. Due to the inherent variance of slot values across diverse domains, this framework faces difficulties in capturing whole slot values in unseen domains; for instance, it captures "`Immortals Restaurant`" instead of "`8 Immortals Restaurant`" for slot type "`restaurant_name`" in

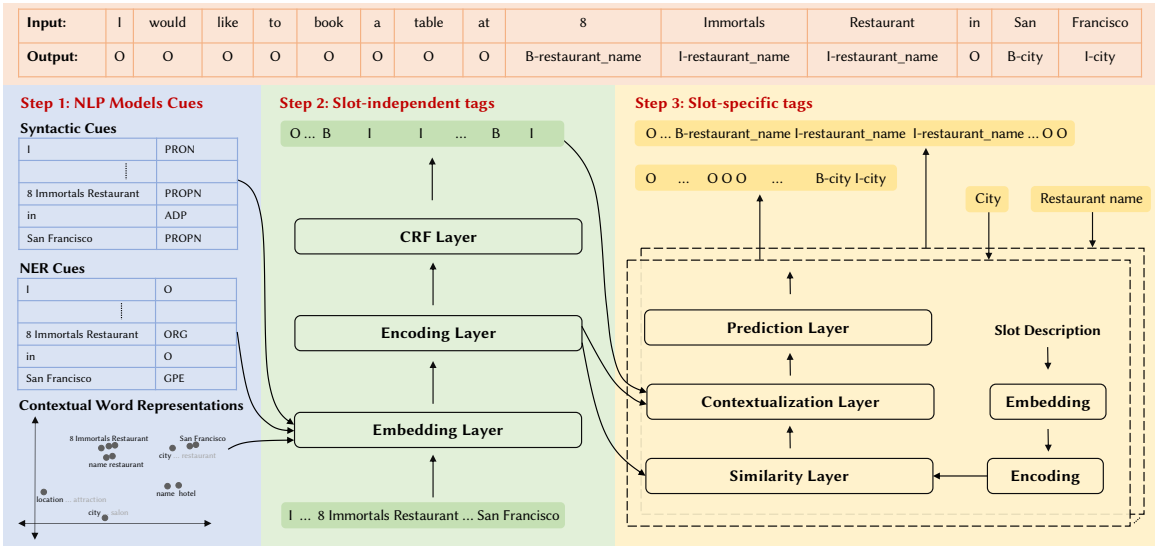


Figure 4.1: Overview of LEONA with an example utterance and its words' label sequence (following the IOB scheme).

Figure 4.1's example utterance. Coach [85] proposed to address the issues in [127, 5] with a coarse-to-fine approach. Coach [85] 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 its representation against that of each slot type 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 zero-shot model LEONA[1] 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 4.1. Step one leverages pre-trained Natural Language

[1] **L**inguistically-**E**nriched and c**ON**text-**A**ware

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) [109] to generate contextual character-based word representations that can handle unknown words that were never seen during training. 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 (LSTM) network [55] that effectively transfers rich semantic information from pre-trained NLP models to the proposed model, LEONA. 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) [71]. 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 4.1, 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` [23], `ATIS` [83], `MultiWOZ` [170] and `SGD` [115], and show that our proposed model consistently outperforms state-of-the-art 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.

4.2 Preliminaries

4.2.1 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 [114]. 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 $\mathcal{D}_{\texttt{SEEN}} = \{\mathcal{D}_1, \cdots, \mathcal{D}_\ell\}$ be a set of seen domains and $\mathcal{D}_{\texttt{UNSEEN}} = \{\mathcal{D}_{\ell+1}, \cdots, \mathcal{D}_z\}$ be a set of unseen domains where $\mathcal{D}_{\texttt{SEEN}} \cap \mathcal{D}_{\texttt{UNSEEN}} = \varnothing$. 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 $\mathcal{D}_p \in \mathcal{D}_{\texttt{SEEN}}$. In traditional (i.e., supervised) slot filling, the domains of test utterances belong to $\mathcal{D}_{\texttt{SEEN}}$, whereas in zero-shot slot filling, the domains of test utterances belong to $\mathcal{D}_{\texttt{UNSEEN}}$; 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 $\mathcal{D}_p \in \mathcal{D}_{\texttt{SEEN}} \cup \mathcal{D}_{\texttt{UNSEEN}}$). We focus on zero-shot slot filling in this work.

4.2.2 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[2], 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. Nonetheless, 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

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

NER model are informative and helpful in reducing the complexity of the task. In this work, we employ SpaCy's pre-trained NER model[3].

Pre-trained ELMo. 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 [108] or Word2vec [99]) which do not consider context. The pre-trained ELMo [109] with appropriate fine-tuning has provided state-of-the-art results on many NLP benchmarks [113, 14, 53, 110, 149, 137]. 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.

4.2.3 Conditional Random Fields

Conditional Random Fields (CRFs) [145] have been successfully applied to various sequence labeling problems in natural language processing such as POS tagging [24], shallow parsing [126], and named entity recognition [125]. 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. Moreover, CRFs allows enforcing constraints in a flexible way (e.g., tag "`I`" can not be preceded by tag "`O`").

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

4.3 Approach

Our model LEONA is an end-to-end neural network with six layers that collectively realize the conceptual three steps in Figure 4.1. 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 following 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 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

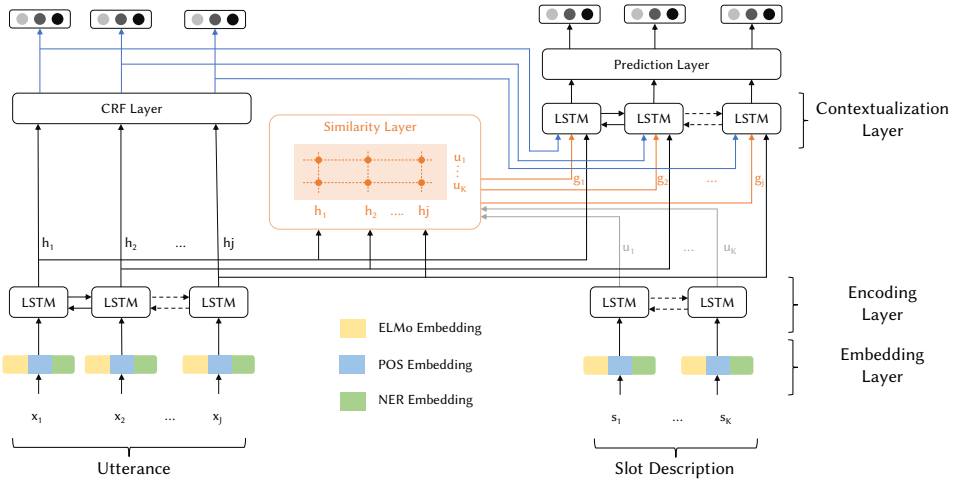


Figure 4.2: Illustration of the layers in our model LEONA.

all the relevant slot types and its outputs are combined to produce the final label for each word.

4.3.1 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 [140] to combine the three embeddings for each word in an effective way; such networks have been shown to perform better than simple concatenation. They produce 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.

4.3.2 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:

$$\overrightarrow{\mathbf{h}}_i = \text{LSTM}(\overrightarrow{\mathbf{h}}_{i-1}, \mathcal{X}_{:i}) \tag{4.1}$$

$$\overleftarrow{\mathbf{h}}_i = \text{LSTM}(\overleftarrow{\mathbf{h}}_{i-1}, \mathcal{X}_{:i}) \tag{4.2}$$

Then, we concatenate the output of the hidden states $\overrightarrow{\mathbf{h}}_i$ and $\overleftarrow{\mathbf{h}}_i$ to get the bi-directional hidden state representation $\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i] \in \mathbb{R}^{2d}$. 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).

4.3.3 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 utterance word based on the utterance's contextual representation $\mathcal{H} = \{\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_J\}$ produced by the encoding layer. Let $\mathcal{Y}$ refer to a sequence label, and the set of all possible state sequences is $\mathcal{C}$. 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:

$$P(\mathcal{Y}|\mathcal{H}; W, b) = \frac{\prod_{i=1}^{J} \theta_i(y_{i-1}, y_i, \mathcal{H})}{\sum_{y' \in \mathcal{C}} \prod_{i=1}^{J} \theta_i(y'_{i-1}, y'_i, \mathcal{H})} \tag{4.3}$$

where $\theta_i(y'_{i-1}, y'_i, \mathcal{H}) = \exp(W^T_{y',y}\mathbf{h}_i + b_{y',y})$ 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.

4.3.4 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 [157, 3, 79] 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 [124]. 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. $\mathcal{A}_{j\mathcal{k}}$ represents the similarity between j-th utterance word and $\mathcal{k}$-th slot description word. We compute the similarity matrix, as follows:

$$\mathcal{A}_{j\mathcal{k}} = \alpha(\mathcal{H}_{:j}, \mathcal{U}_{:\mathcal{k}}) \in \mathbb{R} \tag{4.4}$$

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. $\alpha(\mathbf{h}, u) = w_{(a)}^{\top}[\mathbf{h} \oplus u \oplus \mathbf{h} \otimes u]$, 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 a 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: $\mathbf{h}' = \sum_j b_j \mathcal{H}_{:j}$ where $b = \text{softmax}(\max_{\text{col}}(\mathcal{A})) \in \mathbb{R}^J$ and max operates across columns, and $\mathcal{H}' \in \mathbb{R}^{2d \times J}$ is obtained by tiling $\mathbf{h}'$ across columns.

Intuitively, $\mathcal{U}'$ represents features that highlight important slot description words with closely similar words of an utterance, and $\mathcal{H}'$ highlights features of the utterance with high similarity with the slot description, computed based on the similarity matrix $\mathcal{A}$. Note that $\mathcal{A}$ is computed based the contextual representations of the utterance ($\mathcal{H}$) and slot description ($\mathcal{U}$) generated by the encoding layer that considers surrounding words (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.

4.3.5 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 inputs to this layer. This layer employs *2* stacked 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.

4.3.6 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.

4.3.7 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 the corresponding label $\mathcal{Y}''_{ir}$ (i.e., slot-specific tags for slot type $\mathcal{S}_r$). Finally, q negative samples are generated where slot types that are not present in each respective utterance are selected. For example, the utterance in Figure 4.1 "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 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 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") and ($\cdots$, "`salon_name`", $\cdots$, "O O O O

Table 4.1: Datasets 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

O O O O O O O O O O O"), (· · ·, "`cuisine`", · · ·, · · ·), (· · ·, "`phone_number`", · · ·, · · ·), and so on. Note that slot types are shown in the above example for brevity; slot descriptions are used in practice.

4.4 Experimental Setup

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

4.4.1 Datasets

We used four public datasets to evaluate the performance of our model LEONA: SNIPS Natural Language Understanding benchmark (`SNIPS`), Airline Travel Information System (`ATIS`), Multi-Domain Wizard-of-Oz (`MultiWOZ`), and Dialog System Technology Challenge 8 Schema Guided Dialogue (`SGD`). 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 4.1 presents important dataset statistics.

SNIPS [23]: 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 [83]: 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 have only a small number of 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 [170]: 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 a dialog 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 was 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 [115]: 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`.

4.4.2 Evaluation Methodology

We compute the slot F1 scores[4] and present evaluation results for the following settings:

Train on all except target intent/domain. This is the most common setting that previous works [85, 127, 5] have used in their evaluation. A model is trained on all intents/domains except a single target intent/domain. For `SNIPS` dataset, for example, the model is trained on all intents except a target intent "`AddToPlatlist`" which is used for testing the model's capabilities in the zero-shot fashion. This setup is repeated for every single intent/domain 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 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.

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

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 users' expressions), this setting can be thought of as a "in the wild" [29] setting, which resembles real-world zero-shot slot filling scenarios to a large degree.

4.4.3 Competing Methods

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

Coach [85]: 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 [127]: 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 [5]: This model fills slots for each slot type individually. Character and word-level representations are concatenated with the slot type representation (i.e., embed-

dings) and an LSTM network 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 [72], LSTM-BoE, and CRF-BoE [61] because they have been outperformed by the previous works we compare against.

4.4.4 Implementation Details

Our model uses 300 dimensional embeddings 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 a dropout rate of 0.3 at every layer to avoid over-fitting. Whereas q is set to three for negative sampling.

4.5 Results

We present in the next subsections quantitative and qualitative analysis of all competing models. We first present the quantitative analysis in Subsection 4.5.1 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

Table 4.2: 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 4.3: 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**

our model. We dig deeper into the limitations of each competing model in our qualitative analysis in Subsection 4.5.2.

4.5.1 Quantitative Analysis

Train on all except target intent/domain. Table 4.2, 4.3, 4.4, and 4.5 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. LEONA is consistently better than SOTA methods. Specifically, it outperforms SOTA models by 17.52%, 22.15%, 17.42%, and 17.95% on average for unseen intents/domains on `SNIPS`, `ATIS`,

Table 4.4: 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**

`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 4.3) as compared to other datasets. This 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 with 46 intents across 20 domains, yet our proposed method LEONA has better performance on it (see Table 4.5) than on `SNIPS` and `MultiWoz` datasets. This

Table 4.5: 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**

highlights another critical point: dataset quality. We observes that SGD dataset is not only comprehensive but also has high quality semantic description for slot types. Furthermore, all SGD's domains have enough training examples with minimal annotation error (based on a manual study of a small stratified sample from the dataset). For example, the slot types "restaurant_name", "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 easier 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 4.2) 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 4.4) highlight that transferring knowledge to new unseen intents/domains is easier when some similar intent/domain is present in the training set. For example, our 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 a relatively poor performance. Similar observations can also be made about other competing models.

Comparison for seen and unseen slot types. An unseen target intent/domain may have both unseen and seen slot types. The unseen ones were never 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 diverse 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 unseen and seen slot types individually to test each model's ability in handling completely unseen slot types. Table 4.6 and 4.7 presents results in further detail where results for unseen and seen slot types are reported separately. LEONA is consistently better than other models on unseen as well as unseen slot types. On average, 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 unseen or seen). Moreover, all models have better performance on seen slots than on unseen ones as it is

Table 4.6: Averaged F1 scores for all competing models for seen and unseen slot types in the target unseen intents/domains for SNIPS and ATIS datasets.

Method ↓	**SNIPS**		**ATIS**	
Slot Type →	Seen	Unseen	Seen	Unseen
CT	0.4407	0.2725	0.7552	0.4851
RZS	0.4786	0.2801	0.8132	0.5143
Coach	0.5173	0.3423	0.7742	0.7166
LEONA *w/o* IOB	0.5292	0.3578	0.8155	0.7130
LEONA	**0.6354**	**0.4006**	**0.9588**	**0.7524**

Table 4.7: Averaged F1 scores for all competing models for seen and unseen slot types in the target unseen intents/domains for MultiWOZ and SGD datasets.

Method ↓	**MultiWOZ**		**SGD**	
Slot Type →	Seen	Unseen	Seen	Unseen
CT	0.6062	0.3040	0.7362	0.3940
RZS	0.6604	0.3301	0.7565	0.4478
Coach	0.7034	0.4895	0.7996	0.6614
LEONA *w/o* IOB	0.6651	0.5638	0.7986	0.7424
LEONA	**0.7765**	**0.5962**	**0.9192**	**0.8167**

relatively easier to adapt to a new context (i.e., in a 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 [171], 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 our results to quantify the effect. Table 4.8 and 4.9 present 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

Table 4.8: Averaged F1 scores for all competing models in the target unseen domains of SNIPS and ATIS datasets. The train/test sets have variable number of intents/domains, which makes this setting more challenging.

Method ↓	**SNIPS**			**ATIS**		
% Seen Intents →	25%	50%	75%	25%	50%	75%
CT	0.1043	0.2055	0.2574	0.5018	0.7341	0.6542
RZS	0.1214	0.1940	0.3207	0.6393	0.7727	0.7811
Coach	0.1248	0.2258	0.3081	0.6070	0.7341	0.8104
LEONA *w/o* IOB	0.1550	0.2631	0.4108	0.6495	0.9437	0.9378
LEONA	**0.1710**	**0.2895**	**0.4220**	**0.8093**	**0.9659**	**0.9764**

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, which once again validates our intuition that the availability of 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 perform well 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

Table 4.9: Averaged F1 scores for all competing models in the target unseen domains of MultiWOZ and SGD datasets. The train/test sets have variable number of intents/domains, which makes this setting more challenging.

Method ↓	**MultiWOZ**			**SGD**		
% Seen Intents →	25%	50%	75%	25%	50%	75%
CT	0.2991	0.4371	0.6607	0.4523	0.5389	0.6160
RZS	0.4566	0.4703	0.6951	0.6677	0.6578	0.6741
Coach	0.4408	0.4505	0.6522	0.5888	0.6419	0.6725
LEONA *w/o* IOB	0.5137	0.5529	0.7843	0.6861	0.7315	0.7704
LEONA	**0.5248**	**0.5533**	**0.8581**	**0.7180**	**0.7925**	**0.8324**

Table 4.10: F1 scores for all competing models where the model is trained on one dataset (i.e., either SNIPS or ATIS) and tested on the rest. This setting resembles real-life scenarios.

Method ↓ **Train Dataset →**	**SNIPS**			**ATIS**		
Test Dataset →	ATIS	MultiWOZ	SGD	SNIPS	MultiWOZ	SGD
CT	0.0874	0.1099	0.0845	0.0589	0.0725	0.0531
RZS	0.0915	0.1209	0.1048	0.0819	0.0809	0.0912
Coach	0.1435	0.1191	0.1301	0.0976	0.0962	0.0871
LEONA *w/o* IOB	0.1544	0.1433	0.1504	0.1156	0.1124	0.1359
LEONA	**0.2080**	**0.1832**	**0.1690**	**0.1436**	**0.1394**	**0.1361**

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 4.10 and 4.11 presents the 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

Table 4.11: F1 scores for all competing models where the model is trained on one dataset (i.e., either MultiWOZ or SGD) and tested on the rest. This setting resembles real-life scenarios.

Method ↓ Train Dataset →	**MultiWOZ**			**SGD**		
Test Dataset →	SNIPS	ATIS	SGD	SNIPS	ATIS	MultiWOZ
CT	0.0646	0.0878	0.0616	0.1463	0.2290	0.1529
RZS	0.1496	0.2103	0.0875	0.1905	0.3435	0.2134
Coach	0.1201	0.1730	0.1102	0.1795	0.3383	0.1903
LEONA *w/o* IOB	0.1242	0.1885	0.1258	0.2544	0.4714	0.2743
LEONA	**0.1847**	**0.2662**	**0.1620**	**0.2761**	**0.5205**	**0.2884**

Table 4.12: 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**

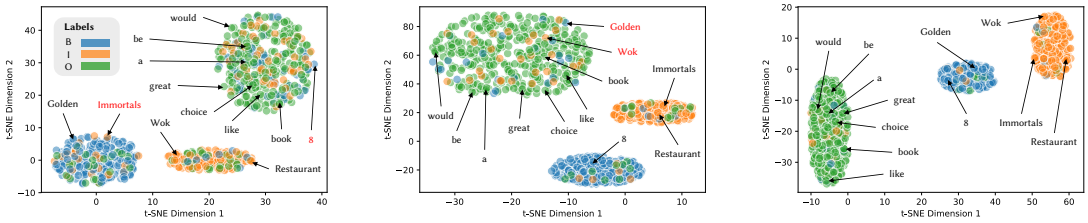
once again highlight the importance of the quality and comprehensiveness of the training dataset(s). Finally, this setting also indicates that current SOTA models are not yet ready to be deployed in real-world scenarios and calls for more exploration and research in the important yet challenging and under-explored task of zero-shot slot filling.

Ablation study. To quantify the role of each component in our model, we present our ablation study results in Table 4.12 over all datasets. First, we study the significance of the pre-trained NLP models in the first three rows in Table 4.12. To produce the results in these rows, we used traditional word [11] and character [52] 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 [85], 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.

4.5.2 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 [91]. Figure 4.3 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 4.3c.



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

Figure 4.3: 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 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 [127] is able to predict the full slot value (i.e., `Golden Wok`) for 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. RZS could extracts "`Immortals Restaurant`" as the slot value 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 4.3a 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 *whole slot values*.

We notice that Coach [85] 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 observe 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 phrase-level representations to detect such slot values in their entirety.

4.6 Related Work

We organize the related work on slot filling into three categories: (*i*) supervised slot filling, (*ii*) few-shot 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 [96, 70] have been employed to tag the input for slots. Similarly, Conditional Random Fields (CRFs) have been integrated with LSTMs/GRUs [60, 118]. The authors in [129, 147] proposed a self-attention mechanism for sequential labeling. More recently, researchers have proposed jointly addressing the related tasks of intent detection and slot filling [44, 51, 79, 171, 164]. The authors in [171] 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 [38, 104, 103] have shown tremendous success for few-shot learning in many tasks such as few-shot image generation [117], image classification [134], and domain adaptation [153]. Following the success of such approaches, few-shot learning in NLP have been investigated for tasks such as text classification [143, 43, 166], entity-relation extraction [88, 42], and few-shot slot filling [87, 39, 57]. The authors in [87] exploited regular expressions for few-shot slot filling, Prototypical Network was employed in [39], and the authors in [57] 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 [68, 10], 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 works has addressed this challenging problem, albeit, with very limited experimental

evaluation. Coach [85] 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 [127] 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 [5] proposed LSTM network and employed slot descriptions to fill the slots for each slot type individually. The authors in [74] 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 [90, 75, 47] 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.

4.7 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 special tokens. We plan to further improve our model by incorporating phrase-level representations to overcome this challenge and allow our model to accurately extract slot values regardless of their length or diversity.

Chapter 5

Unsupervised Paraphrasing

5.1 Introduction

Paraphrasing is the task of generating a fluent output sentence, given an input sentence, to convey the same meaning in different wording. It is an important problem in NLP with a wide range of applications such as summarization [66], information retrieval [67], and question answering. Moreover, paraphrasing can improve the performance of the NLU module (i.e., responsible for intent detection that we discussed in Chapter 3 and slot filling that we discussed in Chapter 4) of conversational agents [93]. Figure 5.1 presents a scenario where the conversational agent fails to understand the user utterance and asks the user to paraphrase it. Whereas ideally, the conversational agent should have paraphrased it by herself. Most of the previous paraphrasing work [111, 76, 49] has focused on *supervised* paraphrasing methods, which require large corpora of parallel sentences (i.e., input and corresponding paraphrased sentences) for training. Unlike large datasets in neural machine translation, there are not many parallel corpora for paraphrasing, and they are often domain-specific;

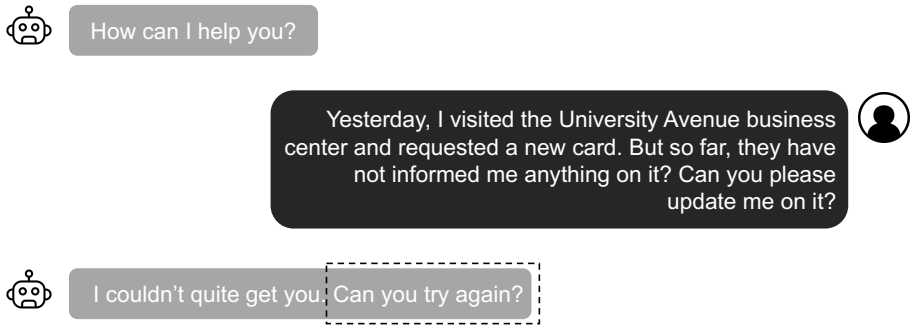


Figure 5.1: A common case in today's state-of-the-art conversational systems: The conversational agent fails to understand a user's utterance and asks the user to try again.

e.g., `Quora` is a questions dataset and `MSCOCO` is an image captioning dataset. Acquiring big parallel datasets for paraphrasing across many domains is not scalable because it is expensive and laborious. Moreover, a model trained in one domain does not generalize well to other domains [77].

The abundance of domains and applications that could benefit from paraphrasing calls for exploring unsupervised paraphrasing, which is still in its infancy. There are relatively few works on unsupervised paraphrasing such as Variational Autoencoder (VAE) [15], Constrained Sentence Generation by Metropolis-Hastings Sampling (CGMH) [98], and Unsupervised Paraphrasing by Simulated Annealing (UPSA) [82]. Although unsupervised approaches have shown promising results, the probabilistic sampling based approaches such as VAE [15] and CGMH [98] are less constrained, and they produce paraphrases that lack semantic similarity to the input. On the other hand, UPSA [82] does not effectively explore the entire sentence space, resulting in paraphrases that are not different enough from the input.

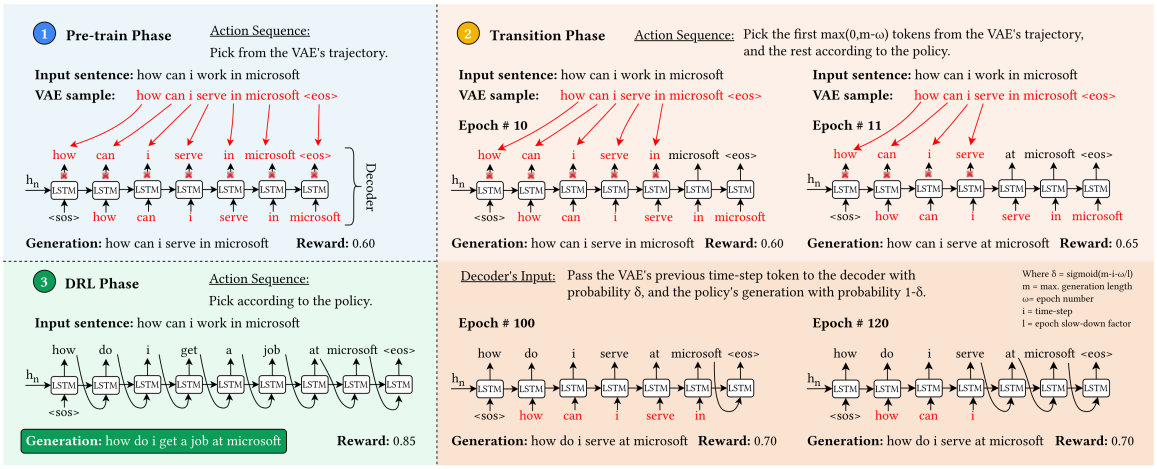


Figure 5.2: Illustration of the decoding process of the proposed unsupervised paraphrasing method: PUP. Red and black color tokens represent the output from VAE and the DRL's chosen action sequences respectively. Whereas the sentence in green is the final paraphrased sentence generated by PUP for the given input sentence.

Given the success of Deep Reinforcement Learning (DRL) [146] in a wide range of applications such as Atari games [102], alphaZero [133], and supervised paraphrasing [76], can DRL also help boost the performance of unsupervised paraphrase generation? To the best of our knowledge, this is the first work to employ DRL in unsupervised paraphrase generation, which is challenging due to the following reasons: (*i*) DRL is known to not work well with large vocabulary sizes when starting with a random policy (i.e., random exploration strategy) [27, 76]; (*ii*) paraphrasing is a multi-step (word-by-word) prediction task, where a small error at an early time-step may lead to poor predictions for the rest of the sentence, as the error is compounded over the next token predictions; and (*iii*) it is challenging to define a reward function that incorporates all the characteristics of a good paraphrase with no access to parallel sentences (i.e., the unsupervised setting).

Our proposed method, *Progressive Unsupervised Paraphrasing (*PUP*)*, progressively trains a DRL-based model for unsupervised paraphrasing and addresses the aforementioned three challenges using the following techniques:

Unsupervised warm-start of DRL. PUP warm-starts reinforcement learning by an unsupervised pre-trained VAE [15], which acts as an expert [25, 120] in the pre-training phase. The pre-trained VAE saves the DRL model from expensive global exploration during the initial training phase. Remarkably, the proposed technique is the first instance that can successfully warm-start DRL with an unsupervised model. At the end of DRL training, our DRL model achieves up to 54% higher reward compared to the initial VAE model. We expect that our idea of warm-starting DRL models in an unsupervised fashion may have implications on a broader range of NLP problems with limited labels.

Progressive transition for seq2seq DRL. Another major issue DRL models face is the accumulation of error over the predictions of future tokens. This is particularly significant during the initial exploration of the space. To overcome this, we use a progressive transition that takes advantage of the Sequence-to-Sequence (seq2seq) [144] nature of the problem by transitioning between algorithms (e.g., VAE to DRL) token by token, as shown in Figure 5.2. Instead of taking actions according to the initial policy (i.e., random action), the model chooses VAE's output as the action, and then incrementally (i.e., one token per epoch) allows the agent to take actions according to the DRL policy. This technique greatly facilitates the convergence of DRL to models with high rewards and is at the heart of the success of DRL.

Unsupervised reward function for paraphrasing. We propose a novel reward function for the DRL model that can measure the quality of the generated paraphrases when no parallel

sentences are available. This is accomplished by incorporating the most desirable qualities of a good paraphrase , informed on the paraphrasing literature [173, 172, 174, 20, 97, 142]. Our reward function is a combination of semantic adequacy, language fluency, and diversity in expression.

Figure 5.2 provides an illustration of the decoding process of PUP. First, the decoder of the DRL model relies on the VAE's sample to pick its actions in the pre-train phase. Then, in the transition phase, the model gradually starts taking actions according to its policy. Finally, in the DRL phase, the model picks actions entirely according to its policy to maximize the expected reward. For example, when our DRL model is pre-trained with the VAE sample *"how can i serve in microsoft"*, our fully-trained DRL model amazingly generates the paraphrase *"how do i get a job at microsoft"*.

We evaluate PUP on four real datasets and compare it against state-of-the-art unsupervised paraphrasing techniques; we show that PUP outperforms them in all standard metrics. We also conduct a human study, which demonstrates that human evaluators find PUP's paraphrases to be of higher quality compared to other methods' paraphrases across several carefully selected measures. Moreover, we consider comparisons against domain-adapted models – i.e., models trained on one dataset such as `Quora` in a supervised setting and then domain-adapted for another dataset such as `WikiAnswers` in an unsupervised fashion. Remarkably, PUP outperforms domain-adapted supervised paraphrasing methods in datasets where applicable.

5.2 Preliminaries

5.2.1 Problem Formulation

Given an input sequence $\mathcal{X} = (x_1, x_2, \cdots, x_n)$ with the length n, the goal of the proposed model is to generate a target sequence (i.e., paraphrase) $\mathcal{Y} = (y_1, y_2, \cdots, y_m)$, where m is the target sequence length. Note that unlike supervised setting, the proposed model does not have have access to target sentences at training time, which makes the problem challenging.

5.2.2 Overview of PUP

This section provides an overview of the progressive training phases of PUP (Figure 5.2). It consists of three phases: pre-train, progressive transition, and DRL.

Pre-train phase. For tasks like unsupervised paraphrasing, the big vocabulary impedes the learning process of DRL models. It becomes practically infeasible to train such a model based on the reward alone. To address this issue, we employ a pre-trained VAE (trained on a non-parallel corpus) to provide a warm-start to the DRL model — it does not start from a random policy. That is, the output of VAE is used to pick action sequences instead of the agent policy's output. We can think of it as demonstrating the expert's (VAE) actions to DRL, where the expert is an unsupervised model.

Progressive transition phase. The next critical step is to gracefully transition from following the expert's actions to taking actions according to the policy (i.e., DRL decoder's distribution). An abrupt transition can obstruct the learning process due to the nature of the task, i.e., multi-step prediction, where error accumulates. Especially, an inappropriate

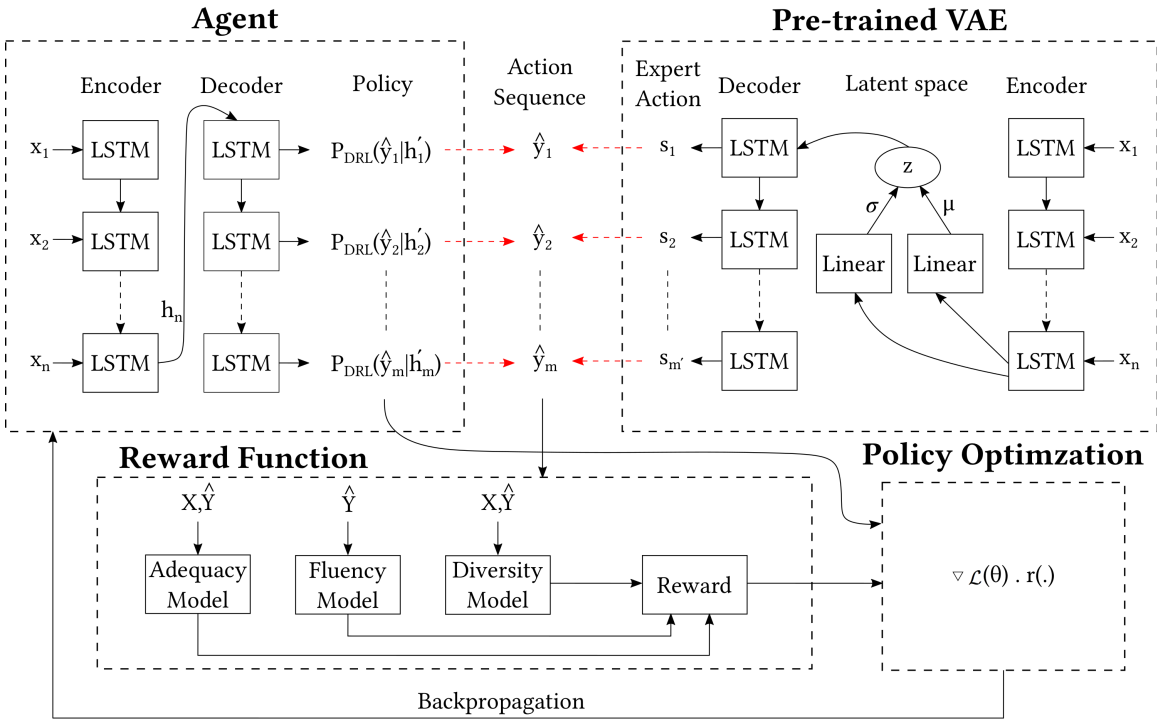


Figure 5.3: Deep reinforcement learning paradigm for unsupervised paraphrase generation.

sample at an early stage of the sentence (i.e., first few words) may lead to a poor eventual paraphrase generation (i.e., ungrammatical or semantically unfaithful). We propose an intuitive way to pick the first $max(0, m - \omega)$ tokens from VAE's output, and pick the rest according to the agent policy, where m is the length of the generated sentence and ω is the epoch number. Moreover, we pass the output of VAE to the decoder's next time-step with a decreasing probability δ (i.e., decreasing with respect to ω), and the DRL's generation otherwise. This helps with mitigating the accumulation of error, especially in the beginning of the transition phase when the model is expected to make mistakes.

DRL phase. Finally, the model is trained to produce an optimized policy by sampling sentences according to its policy and maximizing its expected reward, which is a combination of the semantic adequacy, language fluency, and diversity in expression.

Figure 5.3 presents an overview of the DRL paradigm, where action sequences are picked either from VAE's output or the agent policy (highlighted by red dashed arrows) depending on the different phases.

5.3 Progressive Unsupervised Paraphrasing (PUP)

We first describe how to incorporate DRL for the unsupervised paraphrasing task, then the proposed reward function, and finally we describe the details of PUP.

5.3.1 Reinforcement Learning Paradigm

The reinforcement learning paradigm for unsupervised paraphrasing is presented in Figure 5.3. In DRL terminology, the encoder-decoder model (Section 2.5) acts as an agent, which first encodes the input sentence $\mathcal{X}$ and then generates the paraphrased version $\hat{\mathcal{Y}}$. At time-step i, the agent takes an action $\hat{y}_i \in \mathcal{V}$ according to the policy $P_{DRL}(\hat{y}_i|\hat{y}_{1:i-1}, \mathcal{X})$ (see Equation 2.1), where $\mathcal{V}$ represents the possible action space (i.e., vocabulary for generation). The hidden states of the encoder and the previous outputs of the decoder constitute the state. The agent (i.e., model) keeps generating one token at a time, until the end of sentence token (i.e., $< eos >$) is produced, which completes the action sequence (i.e., trajectory) $\hat{\mathcal{Y}} = (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_m)$. The policy is optimized by maximizing the expected reward r for the action sequences.

5.3.2 Paraphrasing Reward

Automatic quality measures for machine translation (or paraphrasing) such as BLEU [105], Rouge [58], TER [135], and METEOR [4] only work when parallel sentences (i.e., targets or references) are available. We propose a novel reward function that incorporates all the characteristics of a good paraphrase and does not require parallel sentences. The most desired qualities of a good paraphrase [173, 172, 174, 20, 97, 142] include: semantic adequacy (i.e., similarity in meaning), language fluency (i.e., grammatical correctness), and diversity of expression (i.e., sentence dissimilarity). We define the reward $r(\mathcal{X}, \hat{\mathcal{Y}})$ of an output sequence $\hat{\mathcal{Y}}$ generated by the DRL model for input $\mathcal{X}$ as a combination of the above components:

$$r(\mathcal{X}, \hat{\mathcal{Y}}) = \alpha.\, r_{Sim}(\mathcal{X}, \hat{\mathcal{Y}}) + \beta.\, r_F(\hat{\mathcal{Y}}) + \gamma.\, r_{\mathcal{D}}(\mathcal{X}, \hat{\mathcal{Y}}), \tag{5.1}$$

where $r_{Sim}(\mathcal{X}, \hat{\mathcal{Y}})$, $r_F(\hat{\mathcal{Y}})$ and, $r_{\mathcal{D}}(\mathcal{X}, \hat{\mathcal{Y}}) \in [0, 1]$. $r_{Sim}(\mathcal{X}, \hat{\mathcal{Y}})$ is the semantic similarity between input $\mathcal{X}$ and generated paraphrase $\hat{Y}$. $r_F(\hat{\mathcal{Y}})$ captures whether the generated sentence $\hat{\mathcal{Y}}$ is grammatically correct or not. $r_{\mathcal{D}}(\mathcal{X}, \hat{\mathcal{Y}})$ measures the diversity between $\mathcal{X}$ and $\hat{\mathcal{Y}}$. α, β, and $\gamma \in [0, 1]$ are respective weights. Each component is described below.

Semantic Adequacy. The semantic adequacy reward $r_{Sim}(\mathcal{X}, \hat{\mathcal{Y}})$ makes sure that the generated paraphrase $\hat{\mathcal{Y}}$ is similar in meaning to the input sequence $\mathcal{X}$. We use the universal sentence encoder [18], as it has achieved state-of-art results for semantic textual similarity on the STS Benchmark [17] and it provides a straightforward process to incorporate it in any implementation. In a nutshell, it is trained with a deep averaging network (DAN) encoder, and it generates 512-dimension embedding vector for arbitrary length sentence(s). Then, the semantic similarity can be calculated using the cosine similarity of the vectors $v_{\mathcal{X}}$ and

$v_{\hat{\mathcal{Y}}}$, which are embedding vectors for the input sequence $\mathcal{X}$ and the paraphrased sequence $\hat{\mathcal{Y}}$, respectively.

$$r_{Sim}(\mathcal{X}, \hat{\mathcal{Y}}) = \cos(v_{\mathcal{X}}, v_{\hat{\mathcal{Y}}}) = \frac{v_{\mathcal{X}} . v_{\hat{\mathcal{Y}}}}{\|v_{\mathcal{X}}\| \|v_{\hat{\mathcal{Y}}}\|} \tag{5.2}$$

Language Fluency. The fluency reward $r_F(\hat{\mathcal{Y}})$ measures the grammatical correctness of the generated paraphrase $\hat{\mathcal{Y}}$. Since language models such as n-grams [54] and neural models [8] are trained to predict the next token given previous tokens, they can be used to score sentences for fluency. Recently, the Corpus of Linguistic Acceptability (CoLA) [156] has produced the state-of-art results on the grammatical acceptability for in-domain as well as out-of-domain test sets. In its simplest form, CoLA [156] utilizes ELMo-Style (Embeddings from Language Models) and pooling classifier, and it is trained in a supervised fashion. We use a pre-trained CoLA [156] to score our generated paraphrased sequences $\hat{\mathcal{Y}}$.

Expression Diversity. The expression diversity reward $r_{\mathcal{D}}(\mathcal{X}, \hat{\mathcal{Y}})$ encourages the model to generate tokens that are not in the input sequence $\mathcal{X}$. One of the simplest methods to measure the diversity, inverse Jaccard similarity (i.e., $1 -$ Jaccard Similarity), could be used. In this work, we use n-grams dissimilarity. To measure the diversity in expression, we use the inverse BLEU of input sequence $\mathcal{X}$ and the generated sequence $\hat{\mathcal{Y}}$, which is computed using 1 - BLEU($\mathcal{X}$, $\hat{\mathcal{Y}}$). The average of the uni-gram and bi-gram inverse BLEU scores are used in $r_{\mathcal{D}}(\mathcal{X}, \hat{\mathcal{Y}})$.

Combining the three components. In practice, a reward function that can force the DRL model to generate good quality paraphrases must maintain a good balance across the reward components (i.e., semantic similarity, fluency, and diversity). For example, generating

diverse words at the expense of losing too much on the semantic adequacy or fluency is not desirable. Similarly, copying the input sentence as-is to the generation is clearly not a paraphrase (i.e., cosine similarity = 1). To achieve this, we impose strict criteria on the components of the reward function as given below:

$$r_{Sim}(\mathcal{X}, \hat{\mathcal{Y}}) = \begin{cases} r_{Sim}(\mathcal{X}, \hat{\mathcal{Y}}), & \text{if } \tau_{min} \leq r_{Sim}(\mathcal{X}, \hat{\mathcal{Y}}) \leq \tau_{max} \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

$$r_F(\hat{\mathcal{Y}}) = \begin{cases} r_F(\hat{\mathcal{Y}}), & \text{if } r_F(\hat{\mathcal{Y}}) \geq \lambda_{min} \\ 0, & \text{otherwise} \end{cases} \tag{5.4}$$

$$r_{\mathcal{D}}(\mathcal{X}, \hat{\mathcal{Y}}) = \begin{cases} r_{\mathcal{D}}(\mathcal{X}, \hat{\mathcal{Y}}), & \text{if } r_{Sim}(\mathcal{X}, \hat{\mathcal{Y}}) \geq \tau_{min}, r_F(\hat{\mathcal{Y}}) \geq \lambda_{min} \\ 0, & \text{otherwise} \end{cases} \tag{5.5}$$

Equation 5.3 makes sure that the model does not copy the input sentence as-is to the generation (i.e., condition: $r_{Sim}(\mathcal{X}, \hat{\mathcal{Y}}) \leq \tau_{max}$) to enforce the diversity in expression, and does not generate random sentence, which has very low similarity with the input (i.e., condition: $r_{Sim}(\mathcal{X}, \hat{\mathcal{Y}}) > \tau_{min}$). Equation 5.4 penalizes the generations that are not fluent. Finally, diverse words (i.e., Equation 5.5) get rewarded only if the generated sentence achieves a reasonable score on the semantic similarity (i.e., condition: $r_{Sim}(\mathcal{X}, \hat{\mathcal{Y}}) \geq \tau_{min}$) and fluency (i.e., condition: $r_F(\hat{\mathcal{Y}}) \geq \lambda_{min}$). Note that a diversely expressed output sentence, which is not fluent or is not close in meaning to the input sentence needs penalization so that the model may learn a policy that generates not only diverse sentences but also fluent and semantically similar to the input. The objective of combining all the constraints is to ensure competitive

outputs in all metrics and to penalize the model for poor generations on any metric. The weights for each component in the reward (i.e., α, β, and γ), and thresholds (i.e., τ_{min}, τ_{max}, and λ_{min}) for Equations 5.3, 5.4, and 5.5 can be defined based the application needs.

5.3.3 Progressively Training the DRL

The training algorithm optimizes the policy (i.e., encoder-decoder model's distribution $P_{DRL}(.|\mathcal{X})$) to maximize the expected reward $r(.)$ for the generated action sequence $\hat{\mathcal{Y}} = (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_m)$.

The loss for a single sample from the possible action sequences is:

$$\mathcal{L}(\theta) = -\mathbb{E}_{(\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_m)} \sim P_{DRL}(.|\mathcal{X})[r(\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_m)] \tag{5.6}$$

The loss is the negative expected reward for the action sequences. Infinite number of possible samples make the expectation calculations infeasible, thus it is approximated [159]. The gradient for the $\mathcal{L}(\theta)$ is:

$$\nabla\mathcal{L}(\theta) \approx \sum_{i=1}^{m} \nabla \log P_{DRL}(\hat{y}_i|\hat{y}_{1:i-1}, \mathcal{X})[r(\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_m)]. \tag{5.7}$$

The training process for the DRL-based unsupervised paraphrase generation model is outlined in Algorithm 2. We explain each of the training phases below. Note that the pre-trained VAE and the DRL model share the same vocabulary.

Pre-train Phase. Pre-training is a critical step for DRL to work in practice. Since one of the main contributions of this work is to make DRL work in purely unsupervised fashion for

Algorithm 2: Progressively training DRL-based method.

Input: A non-parallel training example $\mathcal{X} = (x_1, x_2, \cdots, x_n)$
Paraphrase generated by VAE $\mathcal{S} = (s_1, s_2, \cdots, s_{m'})$
Probability δ to pass VAE's output as input to decoder
Probability ϵ to sample according to the policy
Epoch number ω
Pre-training status ρ
Learning rate η

1 Initialize $\mathcal{L}(\theta) \leftarrow 0$
2 **for** i=1,$\cdots$, m **do**
3 $vae\_in \leftarrow Uniform(0, 1)$
  if $vae\_in < \delta$ **then**
4 $\hat{y}_{i-1} \leftarrow s_{i-1}$
5 **if** $i \leq m - \omega$ OR $\rho = True$ **then**
6 $\hat{y}_i \leftarrow s_i$
7 **else**
8 $explore \leftarrow Uniform(0, 1)$
  if $explore < \epsilon$ **then**
9 $\hat{y}_i \leftarrow$ Sample $P_{DRL}(\hat{y}_i|h'_i, \hat{y}_{i-1})$
10 **else**
11 $\hat{y}_i \leftarrow$ Argmax $P_{DRL}(\hat{y}_i|h'_i, \hat{y}_{i-1})$
12 $\mathcal{L}(\theta) \leftarrow \mathcal{L}(\theta) + \log P_{DRL}(\hat{y}_i|h'_i, \hat{y}_{i-1})$
13 $\theta \leftarrow \theta + \eta.(\nabla\mathcal{L}(\theta).r(\mathcal{X}, \hat{\mathcal{Y}}))$

the task of paraphrase generation, the pre-training step also has to be unsupervised. We use VAE [15], which is trained in an unsupervised way, and serves as a decent baseline in unsupervised paraphrase generation tasks [98]. The pre-trained VAE (section 2.6) guides as an expert in the pre-train phase to provide a warm-start. Line 6 in Algorithm 2 refers to the pre-train phase. At time-step i, the algorithm picks VAE's sample s_i as the action $\hat{y}_i$. The loss $\mathcal{L}(\theta)$ is computed and accumulated (see line 12 in Algorithm 2). Once, the action sequence is complete (i.e., $(\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_m)$), the reward r is calculated and parameters θ are updated (line 13). This step is a requisite for the DRL model to work in practice for unsupervised paraphrasing.

Transition Phase. Once the model is able to generate sensible sentences, the next critical step is to progressively allow the agent (i.e., encoder-decoder model) to take actions according to its policy. Line 5 in Algorithm 2 refers to whether to take action according to the policy P_{DRL} or to utilize VAE's output $\mathcal{S}$. First $max(0, m - \omega)$ tokens are picked from VAE, and the rest are sampled according to the policy $P_{DRL}(\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_m)$ at time-step i, where m is the length of the generation (i.e., action sequence) and ω is the epoch number. This way, the model picks all tokens from VAE in epoch 0, and in epoch 1, the model is allowed to pick only the last token according to its policy, and so on. Similarly, by epoch m, the model learns to pick all the tokens according to its policy and none from the VAE. The intuition behind this gradual token-by-token transition is that mistakes at earlier tokens (i.e., words at the beginning of the sentence) can be catastrophic, and picking the last few tokens is relatively easy. Moreover, allowing the model to pick according to its policy as soon as possible is also needed, hence we employ gradual transitioning.

Since we allow the DRL model to pick according to its policy at an early stage in the transition phase, the model is expected to make mistakes. However, letting these errors compound over the next predictions may result in never being able to generate sufficiently good samples that can get high rewards. Lines 3-4 in Algorithm 2 attempt to overcome this issue by passing the VAE's previous token S_{i-1} to the decoder as input at time-step i with probability $\delta = sigmoid(m - i - \omega/l)) \in [0, 1]$, where m is the length of the output sentence, ω is the epoch number, and l is the slow-down factor to decay the probability δ as ω grows. It is similar to the above gradual transitioning, but l times slower and probabilistic. The intuition behind the slow progressive transition is that if the DRL model samples wrong

Table 5.1: Statistics about paraphrase datasets

Dataset	Train	Valid	Test	Vocabulary
Quora	117K	3K	20K	8K
WikiAnswers	500K	6K	20K	8K
MSCOCO	110K	10K	40K	10k
Twitter	10K	2K	2K	8K

token, passing the VAE's output to upcoming time-step's decoder would eliminate the accumulation of error in the beginning of the transition phase.

DRL Phase. The DRL phase is the classic reinforcement learning, where the agent takes action $\hat{Y}$ according to its policy P_{DRL}, gets reward r, and optimizes its policy to maximize its expected reward. Greedy decoding impedes the exploration of the space, whereas continuous exploring is also not a desirable behaviour. To keep a balance between exploration (i.e., sample) and exploitation (i.e., argmax), we use a probabilistic decaying mechanism for exploration with probability $\epsilon = \kappa^{\omega}$, where $\kappa \in [0, 1]$ is the constant to control the decay rate of the probability ϵ as ω grows. Lines 7-11 in Algorithm 2 refer to this phase. Pre-trained VAE is used as a baseline model in this phase.

5.4 Experimental Setup

In this section, we describe the datasets, competing approaches, evaluation metrics, and the implementation details of PUP.

5.4.1 Dataset

We use `Quora` [1], `WikiAnswers` [34], `MSCOCO` [78], and `Twitter` [73] datasets to evaluate the quality of the paraphrase generated by PUP and other competing approaches.

Table 5.1 presents key statistics about the datasets. It is important to mention that although these datasets have parallel sentences, we don't use them for training. We only use parallel sentences for validation and to compute the evaluation results on the respective testing sets.

Quora [1]. It is a popular dataset for duplicate question detection annotated by humans which has been used for evaluating paraphrase quality as well, since a pair of duplicate questions can also be considered paraphrases of each other. We follow the training, validation, and testing splits used by [98, 82] for a fair comparison.

WikiAnswers [34]. It contains 2M duplicate question-paraphrase pairs. We use $500K$ non-parallel sentences for training, following previous works [77, 82].

MSCOCO [78]. It is an image captioning dataset that has over 120K images, each captioned by 5 different human annotators. Since all the captions for an image can be thought of as paraphrases, it has also been utilized for the paraphrasing task. We follow the standard splitting [78] and evaluation protocols [82, 111] in our experiments.

Twitter [73]. This dataset is also annotated by humans for duplicate detection. We use the standard train/test split [73], and further split the training set to create a validation set (i.e., 2K sentences).

5.4.2 Competing Methods

We consider the following unsupervised methods and domain-adapted approaches for comparison.

UPSA [82]. UPSA is a simulated annealing based approach that attempts to generate paraphrases using a stochastic search algorithm and achieves state-of-art unsuper-

vised paraphrasing results. We use its open source implementation to generate the paraphrases and compare against our approach.

CGMH [98]. CGMH is a Metropolis-Hastings based approach that generates paraphrase by constraining the decoder at inference time. We use its open source implementation in our comparisons.

Domain-adapted models. We compare against several state-of-the-art supervised methods that are trained in a supervised fashion on one dataset and adapted to another dataset in an unsupervised fashion. For this, we use previously reported results in [77] for `Quora` and `WikiAnswers` datasets. We do not compare with the rule-based approaches such as [93, 6] due to the lack of availability of the rules or any implementation.

5.4.3 Evaluation Metrics

We use well-accepted automatic quantitative evaluation metrics as well as qualitative human studies in order to compare the performance of our method against the competing approaches. For quantitative measures, we use BLEU [105] and ROUGE [58] metrics, which have been widely utilized in the previous work to measure the quality of the paraphrases. Additionally, we use i-BLUE [142] by following the metrics in the most recent work [77, 82]. The metric i-BLUE [142] aims to measure the diversity of expression in the generated paraphrases by penalizing copying words from input sentences.

5.4.4 Implementation Details

The VAE contains two layers with 300-dimensional LSTM units. Our DRL-based model also has two-layers and uses 300-dimensional word embeddings (not pre-trained) and 300-dimensional hidden units. LSTM is utilized as a recurrent unit, and dropout of 0.5 is used. All the sentences are lower cased, and the maximum sentence length is 15 (i.e., we truncate longer sentences to maintain consistency with previous work). The vocabulary size for each dataset is listed in Table 5.1, and infrequent tokens are replaced with $< unk >$ token. We use Adam optimizer with learning rates of 0.15, 10^{-3}, and 10^{-4} in the pre-train, transition, and DRL phases, respectively. The mini-batch size is 32 and gradient clipping of a maximum gradient norm of 2 is used in all the phases. The validation is done after every epoch and the model with the best rewards is saved automatically. Whether to sample or use argmax, $\kappa = 0.9995$ is used. To compute the probability δ, which determines whether to pass VAE's output to the decoder, l is set to 8 during training. At inference time, we utilize beam search [160] with a beam size of $b = 8$ to sample paraphrases for the given input sentences. For the reward function, $\alpha = 0.4$, $\beta = 0.3$, $\gamma = 0.3$, $\tau_{min} = 0.3$, $\tau_{max} = 0.98$, and $\lambda_{min} = 0.3$ are used. All the hypterparameters are picked based on the validation split of the `Quora` dataset, and then consistently used for all the other datasets.

Table 5.2: Performance of the unsupervised and domain-adapted methods on Quora dataset.

	Method	i-BLEU	BLEU	Rouge1	Rouge2
Supervised + domain adapted	Pointer-generator	5.04	6.96	41.89	12.77
	Transformer+Copy	6.17	8.15	44.89	14.79
	Shallow fusion	6.04	7.95	44.87	14.79
	MTL	4.90	6.37	37.64	11.83
	MTL+Copy	7.22	9.83	47.08	19.03
	DNPG	10.39	16.98	56.01	28.61
Unsupervised	VAE	8.16	13.96	44.55	22.64
	CGMH	9.94	15.73	48.73	26.12
	UPSA	12.02	18.18	56.51	**30.69**
	PUP (this work)	**14.91**	**19.68**	**59.77**	30.47

5.5 Results

5.5.1 Automatic Metrics

Table 5.2 and 5.3 present the performance of unsupervised and domain-adapted methods on the `Quora` and `WikiAnswers` datasets, respectively. The best method among all is shown in bold and the best among unsupervised methods is underlined for each metric. Unsupervised methods are trained with non-parallel corpora, and domain-adapted techniques are trained on `Quora` dataset in a supervised fashion and then domain adapted for `WikiAnswers` dataset in an unsupervised fashion (and vice versa). Our proposed method, PUP, outperforms all the unsupervised approaches on all metrics for `Quora` and `WikiAnswers` datasets (except Rouge2 for `Quora` dataset where performance is very competitive with UPSA). Similarly, PUP also outperforms domain-adapted methods for automatic metrics on `Quora` and `WikiAnswers` (except i-BLEU for `WikiAnswers` dataset where the performance is competitive). Although domain-adapted approaches have the advantage of supervised training on one dataset, this advantage does not transfer effectively to the other dataset despite the similarities between the datasets – i.e., `Quora` and `WikiAnswers` are both questions

Table 5.3: Performance of the unsupervised and domain-adapted methods on WikiAnswers dataset.

	Method	i-BLEU	BLEU	Rouge1	Rouge2
Supervised + domain adapted	Pointer-generator	21.87	27.94	53.99	20.85
	Transformer+Copy	23.25	29.22	53.33	21.02
	Shallow fusion	22.57	29.76	53.54	20.68
	MTL	18.34	23.65	48.19	17.53
	MTL+Copy	21.87	30.78	54.10	21.08
	DNPG	**25.60**	35.12	56.17	23.65
Unsupervised	VAE	17.92	24.13	31.87	12.08
	CGMH	20.05	26.45	43.31	16.53
	UPSA	24.84	32.39	54.12	21.45
	PUP (this work)	25.20	**38.22**	**58.88**	**26.72**

datasets. This also highlights that unsupervised approaches are worth exploring for the paraphrasing task as they can be applied to a variety of unlabeled domains or datasets in a flexible way without a need for adaptation. Moreover, the results for VAE (which we use to pre-train our DRL model) are presented in Table 5.2 and Table 5.4 to highlight the performance gain of PUP on each metric.

Table 5.4 presents the results of all unsupervised approaches on `MSCOCO` and `Twitter` datasets, where the best model is shown in bold for each metric. Our proposed method, PUP, is a clear winner on all the metrics among all the unsupervised approaches, which demonstrates the stellar performance of our method as well as the quality of our DRL reward function. The lower performance of unsupervised methods on `Twitter` dataset can be ascribed to the noisy tweets data, however, PUP has significantly better performance (i.e., 90% performance gain on BLEU, and 34% on i-BLEU scores with respect to UPSA) compared to other methods on all of the metrics, which signifies the robustness of the PUP.

Table 5.4: Performance of Unsupervised approaches on MSCOCO and Twitter dataset.

	MSCOCO				Twitter			
Method	i-BLEU	BLEU	Rouge1	Rouge2	i-BLEU	BLEU	Rouge1	Rouge2
VAE	7.48	11.09	31.78	8.66	2.92	3.46	15.13	3.4
CGMH	7.84	11.45	32.19	8.67	4.18	5.32	19.96	5.44
UPSA	9.26	14.16	37.18	11.21	4.93	6.87	28.34	8.53
PUP	**10.72**	**15.81**	**37.38**	**13.87**	**6.62**	**13.03**	**39.12**	**12.91**

5.5.2 Subjective Human Evaluations

To further illustrate the superior quality of the paraphrases generated by PUP, we conduct subjective human evaluations on `Quora` dataset. Table 5.5 presents the average scores along with the confidence intervals of human evaluators for diversity in expression, language fluency, and semantic similarity on randomly selected 300 paraphrases generated by all three unsupervised methods (CGMH, UPSA, and PUP). We used Amazon Mechanical Turk (a widely-used crowd sourcing platform) in our human studies. We selected Mechanical Turk *Masters* from the USA with a HIT approval rate of $\geq 90\%$ to rate the paraphrases on a scale of $1-5$ (1 being the worst and 5 the best) for the three evaluation criteria diversity, fluency, and similarity. Each paraphrase is scored by three different evaluators. Our method PUP outperforms all the competing unsupervised approaches on all criteria. It should also be noted that CGMH is better on diversity of expression than UPSA, and the opposite results are observed for semantic similarity and fluency. In contrast, our reward function facilitates a good balance between the diversity in expression, semantic similarity, and fluency. A similar trend can also be observed in Table 5.6 and Table 5.7, which present automatically calculated reward and a few example paraphrases generated by all three unsupervised approaches, respectively.

Table 5.5: Subjective human studies on paraphrase generations by unsupervised methods on Quora dataset.

Method	Diversity	Fluency	Similarity
CGMH	3.14 ± 0.053	4.1 ± 0.042	2.97 ± 0.055
UPSA	2.96 ± 0.052	4.35 ± 0.033	3.89 ± 0.045
PUP	$\mathbf{3.27} \pm 0.048$	$\mathbf{4.42} \pm 0.027$	$\mathbf{4.09} \pm 0.035$

Table 5.6: Performance of the unsupervised methods for the components of the reward function on Quora dataset.

Method	Diversity	Fluency	Similarity	Reward
VAE	0.31	0.72	0.47	0.497
CGMH	0.29	0.73	0.49	0.502
UPSA	0.25	0.72	0.68	0.563
PUP	**0.53**	**0.95**	**0.81**	**0.768**

5.5.3 Evaluation on Reward Function

Table 5.6 presents the average scores of all the components of our proposed reward function on `Quora` dataset for all the unsupervised approaches. Perhaps not surprisingly, our method outperforms other methods on each individual component of the reward by large margin. Intuitively, this arises from the fact that our DRL-based model is explicitly trained to optimize these reward components. Remarkably, DRL process improves the reward by more than 50% compared to the pre-training phase, i.e., the reward of VAE. This is also visible in Figure 5.4 where PUP starts with a reward value of around 0.5 and is able to achieve up to 0.77 towards the end of the last phase of training.

5.5.4 Ablation Study

Figure 5.4 presents the rewards achieved over the course of different epochs by three models: (*i*) PUP, pre-trained and uses the transition phase; (*ii*) `No Transition`, pre-trained

Table 5.7: Example paraphrase generations by PUP and other unsupervised competing methods on Quora dataset.

Input Sentence	**CGMH Generation**	**UPSA Generation**	**PUP Generation**
how can i work in microsoft	how can i prepare for cpt	how can i get to work at microsoft	how do i get a job at microsoft
which is the best shampoo for dandruff	what is the best shampoo for sciatica	which is the best shampoo for oily skin	what are the proper shampoos for dandruff
which book is the best to learn algo	which programming language is the best to learn algo	which book is best to learn algo	what is a best book for learning algos
what is the best mac game	what is the best video game	what is the best mac app for android games	what are some good mac games
what are the reasons of war	what are the positive aspects of nuclear war	what are the main reasons for a civil war	what is the main reason for war

but does not use the transition phase; and (*iii*) `No Pre-train`, not pre-trained at all. It highlights the need for the distinct phases in our training procedure. It can be observed that without the pre-training phase, `No Pre-train` model is unable to maximize the expected reward. The reward remains small and fluctuates randomly. Similarly, transition phase is also required, as abrupt shift from VAE to DRL derails the training for `No Transition` model, whereas PUP is able to rapidly and consistently improve the reward as the number of epochs grow.

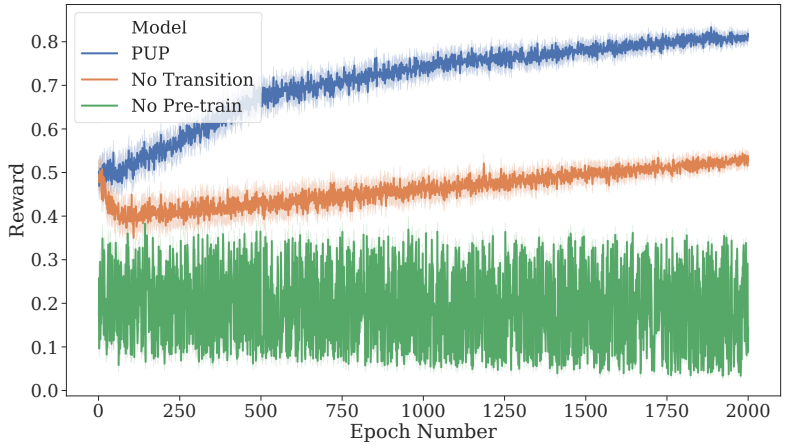


Figure 5.4: Evolution of the reward value for PUP variants over the course of the training.

5.6 Related Work

The task of automatic paraphrasing is one of the common NLP research problems, which has widespread applications. A wide range of approaches have been developed to solve this problem. Rule-based [93, 33, 107, 6] and data-driven approaches [92, 172] are some of the earliest techniques. Automatically constructed paraphrase detection datasets using SVM-based classifiers and other unsupervised approaches are introduced in [30, 31].

Recently, supervised deep learning approaches have also been used for paraphrase generation. Stacked residual LSTM networks [111] is one of the earliest efforts in the paraphrase generation utilizing deep networks. [76] makes use of deep reinforcement learning for paraphrase generation in a supervised fashion. Supervised paraphrase generation using LSTM-based VAE [49], transformer model [151], pointer-generator networks [123] have also shown promising results. Supervised paraphrase generation at different granularity levels (i.e., lexical, phrasal and sentential levels) [77] is achieved with template learning. Additionally

these models can also be adapted to new domains in an unsupervised fashion, utilizing the learned templates with the assumption that both domains share similar templates.

Unsupervised paraphrasing is a challenging and emerging NLP task, and the literature is relatively limited. The VAE [15] is trained in an unsupervised fashion (i.e., no parallel corpus is required), by maximizing the lower bounds for the log-likelihood. The VAE's decoder can sample sentences (i.e., paraphrases), which are less controllable [98], but serve as a good baseline for the unsupervised paraphrasing task. CGMH [98] proposes a constrained sentence generation using Metropolis-Hastings Sampling by adding constraints on the decoder at inference time, and hence does not require parallel corpora. UPSA [82] generates paraphrases by simulated annealing, and achieves state-of-art results on the task. It proposes a search objective function, which involves semantic similarity and fluency for performing diverse word replacement, insertion or deletion operations, thus generating paraphrases in an unsupervised fashion. In contrast, we formulate the task as a deep reinforcement learning problem and progressively train the policy to maximize the expected reward, which includes semantic adequacy, language fluency, and diversity in expression.

5.7 Conclusion

We have presented a progressive approach to train a DRL-based unsupervised paraphrasing model. Our method provides a warm-start to the DRL-based model with a pre-trained VAE (i.e., trained on non-parallel corpus). Then, our model progressively transitions from VAE's output to acting according to its policy. We also propose a reward function which incorporates all the attributes of a good paraphrase and does not require

parallel sentences. The paraphrases generated by our model outperform both state-of-the-art unsupervised paraphrasing and domain-adapted supervised models on automatic metrics. Specifically, our method achieves up to 90% and 34% performance gains for the BLEU and the i-BLEU metrics compared to state-of-the-art unsupervised methods, respectively. Moreover, the paraphrases generated by our method were rated the highest by human evaluators for all considered criteria: diversity of expression, fluency, and semantic similarity to input sentences. Since our technique is the first to successfully warm-start DRL with an unsupervised model, we plan on investigating the broader implications of our technique on other NLP problems with scarce labeled training data.

Chapter 6

Conclusions

This dissertation demonstrates various reliable techniques for incorporating readily available, domain-independent knowledge into neural models for open-domain NLP. We showed that when commonsense knowledge is integrated into neural models properly, it significantly improves their discriminative power, particularly in the open-domain settings, and we showed that pre-trained NLP models offer yet another rich resource that, when combined with otherwise low-performing embeddings, greatly improves the expressive power of textual embeddings. Additionally, this dissertation demonstrates a progressive training approach for the deep reinforcement learning paradigm — the first instance that has successfully warmed-started a deep reinforcement learning agent with an unsupervised model. With the proposed framework, DRL agents save time and resources by avoiding expensive global exploration, which is particularly critical in open-domain settings. Combining the ideas presented in this dissertation facilitates the development of robust, reliable, and open-domain NLP models. Specifically, this dissertation has the following take-aways.

First, this dissertation proposes a robust and generalizable method for combining commonsense knowledge with embeddings from pre-trained language models. We proposed a framework for studying direct and latent relationships between pairs of phrases with the help of a commonsense knowledge graph. We have built a neural model that takes advantage of our framework, RIDE, which is able to accurately detect both seen and unseen intents in utterances (i.e., generalized zero-shot setting). Compared to state-of-the-art models (in terms of accuracy and F1 score), RIDE is better by up to 42.21% and 58.50%, respectively, for unseen intents on SNIPS, MultiWOZ, and SGD datasets. Additionally, our model consistently results in the highest F1 score on seen intents, which confirms the generalizability of our framework. It would be worthwhile to investigate whether commonsense knowledge can improve the performance of other zero-shot classification tasks, in particular for class labels with little to no training data.

Second, this dissertation proposes to integrate domain-independent, pre-trained NLP models with context-aware utterance-slot similarity features to facilitate sequence tagging tasks in contexts where training data is scarce. We applied our ideas to the critical problem of slot filling and showed that they significantly improve the performance of slot filling models in unseen domains. To the best of our knowledge, this is the first work to use pre-trained NLP models for zero-shot slot filling. Our proposed model, LEONA, outperforms 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. Moreover, 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 state-of-the-art model. It remains to be seen whether the same techniques can benefit other zero-shot NLP tasks.

Finally, this dissertation proposes a progressive approach to stabilize the training process of DRL agents by warm-starting the DRL model with predictions from an unsupervised model. We applied our approach in the context of unsupervised paraphrasing, and we showed that when this training framework is augmented with a novel reward function (that captures semantic adequacy, language fluency, and expression diversity in an unsupervised fashion), our resulting model, PUP, produces high-quality paraphrases. Comparing PUP to state-of-the-art unsupervised methods, we achieve 90% and 34% performance gains for the BLEU and i-BLEU metrics, respectively. PUP also outperforms domain-adapted methods for automatic metrics on `Quora` and `WikiAnswers` datasets. Furthermore, human evaluators ranked our paraphrases as the best in all criteria: fluency, diversity of expression, and semantic similarity to the input text. Since our technique is the first to successfully warm-start DRL with an unsupervised model, it would be interesting to explore the broader implications of our technique on other NLP tasks with scarce labeled training data.

Bibliography

[1] Quora question pairs — kaggle. `https://www.kaggle.com/c/quora-question-pairs`. (Accessed on 02/14/2020).

[2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[3] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[4] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.

[5] Ankur Bapna, Gokhan Tür, Dilek Hakkani-Tür, and Larry Heck. Towards zero-shot frame semantic parsing for domain scaling. In *Proc. Interspeech 2017*, pages 2476–2480, 2017.

[6] Regina Barzilay and Lillian Lee. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 16–23. Association for Computational Linguistics, 2003.

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

[8] Yoshua Bengio. Neural net language models. *Scholarpedia*, 3(1):3881, 2008.

[9] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155, 2003.

[10] Hemanthage S Bhathiya and Uthayasanker Thayasivam. Meta learning for few-shot joint intent detection and slot-filling. In *Proceedings of the 2020 5th International Conference on Machine Learning Technologies*, pages 86–92, 2020.

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

[12] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA, 2008. Association for Computing Machinery.

[13] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

[14] Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, 2015.

[15] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[16] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[17] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

[18] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[19] Wei-Lun Chao, Soravit Changpinyo, Boqing Gong, and Fei Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In *European conference on computer vision*, pages 52–68. Springer, 2016.

[20] David L Chen and William B Dolan. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 190–200. Association for Computational Linguistics, 2011.

[21] Yun-Nung Chen, Dilek Hakkani-Tür, and Xiaodong He. Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6045–6049. IEEE, 2016.

[22] Zhiyuan Chen, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. Identifying intention posts in discussion forums. In *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1041–1050, 2013.

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

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

[25] Hal Daumé, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

[26] Yann N Dauphin, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. Zero-shot learning for semantic utterance classification. *arXiv preprint arXiv:1401.0509*, 2013.

[27] Peter Dayan and Yael Niv. Reinforcement learning: the good, the bad and the ugly. *Current opinion in neurobiology*, 18(2):185–196, 2008.

[28] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

[30] Bill Dolan, Chris Quirk, and Chris Brockett. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[31] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[32] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220, 2008.

[33] Michael Ellsworth and Adam Janin. Mutaphrase: Paraphrasing with framenet. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 143–150. Association for Computational Linguistics, 2007.

[34] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1608–1618, 2013.

[35] Umar Farooq, A.B. Siddique, Fuad Jamour, Zhijia Zhao, and Vagelis Hristidis. App-aware response synthesis for user reviews. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 699–708, 2020.

[36] Rafael Felix, Vijay BG Kumar, Ian Reid, and Gustavo Carneiro. Multi-modal cycle-consistent generalized zero-shot learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 21–37, 2018.

[37] Emmanuel Ferreira, Bassam Jabaian, and Fabrice Lefevre. Online adaptative zero-shot learning spoken language understanding using word-embedding. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5321–5325. IEEE, 2015.

[38] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

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

[40] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*, pages 2121–2129, 2013.

[41] Varun Gangal, Abhinav Arora, Arash Einolghozati, and Sonal Gupta. Likelihood ratios and generative classifiers for unsupervised out-of-domain detection in task oriented dialog. *arXiv preprint arXiv:1912.12800*, 2019.

[42] Tianyu Gao, Xu Han, Ruobing Xie, Zhiyuan Liu, Fen Lin, Leyu Lin, and Maosong Sun. Neural snowball for few-shot relation learning. In *AAAI*, pages 7772–7779, 2020.

[43] Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. Induction 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)*, pages 3895–3904, 2019.

[44] Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. Slot-gated modeling for joint slot filling and intent

prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757, 2018.

[45] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

[46] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

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

[48] Jiang Guo. Backpropagation through time. *Unpubl. ms., Harbin Institute of Technology*, 2013.

[49] Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. A deep generative framework for paraphrase generation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[50] Arshit Gupta, John Hewitt, and Katrin Kirchhoff. Simple, fast, accurate intent classification and slot labeling for goal-oriented dialogue systems. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 46–55, 2019.

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

[52] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1923–1933, 2017.

[53] Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 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)*, pages 473–483, 2017.

[54] Kenneth Heafield. Kenlm: Faster and smaller language model queries. In *Proceedings of the sixth workshop on statistical machine translation*, pages 187–197. Association for Computational Linguistics, 2011.

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

[56] Sepp Hochreiter and Jürgen Schmidhuber. Lstm can solve hard long time lag problems. In *Advances in neural information processing systems*, pages 473–479, 1997.

[57] Yutai Hou, Wanxiang Che, Yongkui Lai, Zhihan Zhou, Yijia Liu, Han Liu, and Ting Liu. Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1381–1393, 2020.

[58] Eduard H Hovy, Chin-Yew Lin, Liang Zhou, and Junichi Fukumoto. Automated summarization evaluation with basic elements. In *LREC*, volume 6, pages 899–902. Citeseer, 2006.

[59] Matthew B Hoy. Alexa, siri, cortana, and more: an introduction to voice assistants. *Medical reference services quarterly*, 37(1):81–88, 2018.

[60] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[61] Rahul Jha, Alex Marin, Suvamsh Shivaprasad, and Imed Zitouni. 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)*, pages 153–161, 2018.

[62] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

[63] Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In *Advances in neural information processing systems*, pages 4284–4295, 2018.

[64] Joo-Kyung Kim, Gokhan Tur, Asli Celikyilmaz, Bin Cao, and Ye-Yi Wang. Intent detection using semantically enriched word embeddings. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 414–419. IEEE, 2016.

[65] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[66] Emily Kissner. Summarizing, paraphrasing and retelling. *Portsmouth, NH: Heinernann*, 2006.

[67] Kevin Knight and Daniel Marcu. Statistics-based summarization-step one: Sentence compression. *AAAI/IAAI*, 2000:703–710, 2000.

[68] Jason Krone, AI Amazon, Yi Zhang, and Mona Diab. Learning to classify intents and slot labels given a handful of examples. *ACL 2020*, page 96, 2020.

[69] Anjishnu Kumar, Pavankumar Reddy Muddireddy, Markus Dreyer, and Björn Hoffmeister. Zero-shot learning across heterogeneous overlapping domains. In *INTERSPEECH*, pages 2914–2918, 2017.

[70] Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo 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*, pages 2077–2083, Austin, Texas, November 2016. Association for Computational Linguistics.

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

[72] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270, 2016.

[73] Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. A continuously growing dataset of sentential paraphrases. *arXiv preprint arXiv:1708.00391*, 2017.

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

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

[76] Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. Paraphrase generation with deep reinforcement learning. *arXiv preprint arXiv:1711.00279*, 2017.

[77] Zichao Li, Xin Jiang, Lifeng Shang, and Qun Liu. Decomposable neural paraphrase generation. *arXiv preprint arXiv:1906.09741*, 2019.

[78] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

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

[80] Han Liu, Xiaotong Zhang, Lu Fan, Xuandi Fu, Qimai Li, Xiao-Ming Wu, and Albert YS Lam. Reconstructing capsule networks for zero-shot intent 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)*, pages 4801–4811, 2019.

[81] Kun Liu, Wu Liu, Huadong Ma, Wenbing Huang, and Xiongxiong Dong. Generalized zero-shot learning for action recognition with web-scale video data. *World Wide Web*, 22(2):807–824, 2019.

[82] Xianggen Liu, Lili Mou, Fandong Meng, Hao Zhou, Jie Zhou, and Sen Song. Unsupervised paraphrasing by simulated annealing. *arXiv preprint arXiv:1909.03588*, 2019.

[83] Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. Benchmarking natural language understanding services for building conversational agents. April 2019. 10th International Workshop on Spoken Dialogue Systems Technology 2019, IWSDS 2019 ; Conference date: 24-04-2019 Through 26-04-2019.

[84] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[85] Zihan Liu, Genta Indra Winata, Peng Xu, and Pascale 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*, pages 19–25, Online, July 2020. Association for Computational Linguistics.

[86] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170, 2011.

[87] Bingfeng Luo, Yansong Feng, Zheng Wang, Songfang Huang, Rui Yan, and Dongyan Zhao. Marrying up regular expressions with neural networks: A case study for spoken language understanding. In *56th Annual Meeting of the Association for Computational Linguistics*, pages 2083–2093, 2018.

[88] Xin Lv, Yuxian Gu, Xu Han, Lei Hou, Juanzi Li, and Zhiyuan Liu. Adapting meta knowledge graph information for multi-hop reasoning over few-shot relations. In *EMNLP/IJCNLP (1)*, 2019.

[89] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, 2016.

[90] Yue Ma, Zengfeng Zeng, Dawei Zhu, Xuan Li, Yiying Yang, Xiaoyuan Yao, Kaijie Zhou, and Jianping Shen. An end-to-end dialogue state tracking system with machine reading comprehension and wide & deep classification. *arXiv preprint arXiv:1912.09297*, 2019.

[91] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[92] Nitin Madnani and Bonnie J Dorr. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387, 2010.

[93] Kathleen R McKeown. Paraphrasing questions using given and new information. *Computational Linguistics*, 9(1):1–10, 1983.

[94] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 437–452. Springer, 2011.

[95] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.

[96] Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539, 2014.

[97] Donald Metzler, Eduard Hovy, and Chunliang Zhang. An empirical evaluation of data-driven paraphrase generation techniques. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 546–551. Association for Computational Linguistics, 2011.

[98] Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6834–6842, 2019.

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

[100] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[101] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[102] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[103] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[104] Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2(3):4, 2018.

[105] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[106] Adam Pauls and Dan Klein. Faster and smaller n-gram language models. In *Proceedings of the 49th annual meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 258–267, 2011.

[107] Ellie Pavlick, Travis Wolfe, Pushpendre Rastogi, Chris Callison-Burch, Mark Dredze, and Benjamin Van Durme. Framenet+: Fast paraphrastic tripling of framenet. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 408–413, 2015.

[108] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[109] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

[110] Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40, 2012.

[111] Aaditya Prakash, Sadid A Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. Neural paraphrase generation with stacked residual lstm networks. *arXiv preprint arXiv:1610.03098*, 2016.

[112] PyTorch. torch.optim — pytorch 1.3.0 documentation. `https://pytorch.org/docs/stable/optim.html#torch.optim.lr_scheduler.ReduceLROnPlateau`, 2020. (Accessed on 11/12/2020).

[113] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016.

[114] Lance Ramshaw and Mitch Marcus. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*, 1995.

[115] Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696, 2020.

[116] Suman Ravuri and Andreas Stolcke. Recurrent neural network and lstm models for lexical utterance classification. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[117] Scott Reed, Yutian Chen, Thomas Paine, Aäron van den Oord, SM Eslami, Danilo Rezende, Oriol Vinyals, and Nando de Freitas. Few-shot autoregressive density estimation: Towards learning to learn distributions. *arXiv preprint arXiv:1710.10304*, 2017.

[118] Nils Reimers and Iryna Gurevych. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *arXiv preprint arXiv:1707.06799*, 2017.

[119] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[120] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[121] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866, 2017.

[122] Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1757–1772, 2012.

[123] Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*, 2017.

[124] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[125] Burr Settles. 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)*, pages 107–110, 2004.

[126] Fei Sha and Fernando Pereira. 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*, pages 213–220, 2003.

[127] Darsh Shah, Raghav Gupta, Amir Fayazi, and Dilek Hakkani-Tur. Robust zero-shot cross-domain slot filling with example values. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5484–5490, Florence, Italy, July 2019. Association for Computational Linguistics.

[128] Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871*, 2018.

[129] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[130] A.B. Siddique, Fuad Jamour, and Vagelis Hristidis. Linguistically-enriched and context-aware zero-shot slot filling. In *Proceedings of the Web Conference 2021*, New York, NY, USA, 2021. Association for Computing Machinery.

[131] A.B. Siddique, Fuad Jamour, Luxun Xu, and Vagelis Hristidis. Generalized zero-shot intent detection via commonsense knowledge. *arXiv preprint arXiv:2102.02925*, 2021.

[132] A.B. Siddique, Samet Oymak, and Vagelis Hristidis. Unsupervised paraphrasing via deep reinforcement learning. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 1800–1809, New York, NY, USA, 2020. Association for Computing Machinery.

[133] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[134] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017.

[135] Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200, 2006.

[136] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943, 2013.

[137] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[138] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. *arXiv preprint arXiv:1612.03975*, 2016.

[139] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge, 2017.

[140] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[141] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706, 2007.

[142] Hong Sun and Ming Zhou. Joint learning of a dual smt system for paraphrase generation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 38–42. Association for Computational Linguistics, 2012.

[143] Shengli Sun, Qingfeng Sun, Kevin Zhou, and Tengchao Lv. 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)*, pages 476–485, 2019.

[144] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[145] Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning*, 2:93–128, 2006.

[146] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[147] Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. Deep semantic role labeling with self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[148] Ben Taskar, Dan Klein, Mike Collins, Daphne Koller, and Christopher D Manning. Max-margin parsing. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 1–8, 2004.

[149] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003.

[150] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. International Conference on Machine Learning (ICML), 2016.

[151] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[152] Nikhita Vedula, Nedim Lipka, Pranav Maneriker, and Srinivasan Parthasarathy. Open intent extraction from natural language interactions. In *Proceedings of The Web Conference 2020*, pages 2009–2020, 2020.

[153] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[154] Jinpeng Wang, Gao Cong, Wayne Xin Zhao, and Xiaoming Li. Mining user intents in twitter: A semi-supervised approach to inferring intent categories for tweets. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 318–324. AAAI Press, 2015.

[155] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1):396–413, 2020.

[156] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

[157] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[158] Kyle Williams. Zero shot intent classification using long-short term memory networks. *Proc. Interspeech 2019*, pages 844–848, 2019.

[159] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[160] Sam Wiseman and Alexander M Rush. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960*, 2016.

[161] Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip S Yu. Zero-shot user intent detection via capsule neural networks. *arXiv preprint arXiv:1809.00385*, 2018.

[162] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2251–2265, 2018.

[163] Yongqin Xian, Bernt Schiele, and Zeynep Akata. Zero-shot learning-the good, the bad and the ugly. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4582–4591, 2017.

[164] Puyang Xu and Ruhi Sarikaya. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *2013 ieee workshop on automatic speech recognition and understanding*, pages 78–83. IEEE, 2013.

[165] Guangfeng Yan, Lu Fan, Qimai Li, Han Liu, Xiaotong Zhang, Xiao-Ming Wu, and Albert YS Lam. Unknown intent detection using gaussian mixture model with an application to zero-shot intent classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1050–1060, 2020.

[166] Leiming Yan, Yuhui Zheng, and Jie Cao. Few-shot learning for short text classification. *Multimedia Tools and Applications*, 77(22):29799–29810, 2018.

[167] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

[168] Majid Yazdani and James Henderson. A model of zero-shot learning of spoken language understanding. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 244–249, 2015.

[169] Steve Young. Talking to machines (statistically speaking). In *Seventh International Conference on Spoken Language Processing*, 2002.

[170] Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. 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*, pages 109–117, Online, July 2020. Association for Computational Linguistics.

[171] Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip Yu. Joint slot filling and intent detection via capsule neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5259–5267, Florence, Italy, July 2019. Association for Computational Linguistics.

[172] Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. Application-driven statistical paraphrase generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 834–842. Association for Computational Linguistics, 2009.

[173] Shiqi Zhao and Haifeng Wang. Paraphrases and applications. In *Coling 2010: Paraphrases and Applications–Tutorial notes*, pages 1–87, 2010.

[174] Shiqi Zhao, Haifeng Wang, Xiang Lan, and Ting Liu. Leveraging multiple mt engines for paraphrase generation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1326–1334. Association for Computational Linguistics, 2010.