FACTORIZATION-ORIENTED REPRESENTATION LEARNING FOR LINGUISTIC STRUCTURED PREDICTION

by

Jie Cao

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

School of Computing

The University of Utah

Dec 2021

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Jie Cao**
has been approved by the following supervisory committee members:

Vivek Srikumar ,	Chair(s)	**xx 2021** Date Approved
Ellen Riloff ,	Member	**xx 2021** Date Approved
Qingyao Ai ,	Member	**xx 2021** Date Approved
Zac Imel ,	Member	**xx 2021** Date Approved
Yi Zhang ,	Member	**xx 2021** Date Approved

by **xx**, Chair/Dean of
the Department/College/School of **Computing**
and by **xx**, Dean of The Graduate School.

ABSTRACT

Natural language, when described as a system of symbolic human communication, it is inherently studied via various discrete symbolic representations. Designing the artificial symbolic representation for natural language intuitively follows the principle of compositionality in natural language. Leveraging this as an inductive bias, we conduct systematic studies on factorization-oriented representation learning to predict those combinatorial and symbolic structures. We propose the corresponding neural representation learning approaches for three different kinds of factorization: *Independent Factorization*, *Label Representation*, *Auto-regressive Factorization*. We ground our studies on both broad-coverage linguistic representations (such as graph-based meaning representations, DM, PSD, AMR, and UCCA, etc.) and application-specific representations (such as MISC code for psychotherapy dialog and dialog state representations for task-oriented dialog). The experimental results show our proposed methods achieve competetive performance on each task, and they potentially help for other linguistic structured prediction tasks with similar factorizations.

For my parents, Alice and Bob.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOTATION AND SYMBOLS

α	fine-structure (dimensionless) constant, approximately $1/137$
α	radiation of doubly-ionized helium ions, He++
β	radiation of electrons
γ	radiation of very high frequency, beyond that of X rays
γ	Euler's constant, approximately $0.577\,215\ldots$
δ	stepsize in numerical integration
$\delta(x)$	Dirac's famous function
ϵ	a tiny number, usually in the context of a limit to zero
$\zeta(x)$	the famous Riemann zeta function
$\ldots$	$\ldots$
$\psi(x)$	logarithmic derivative of the gamma function
ω	frequency

CHAPTER 1

INTRODUCTION

Large models are taking over NLP.[*add examples*$_{\text{TODO}}$] Most of them benifit from raw text data with self-supervised learning. [*probing*$_{\text{TODO}}$] Recent work on post-hoc probing have shown that Large lagnauge model may already learn the syntax and some semantic structures in some extent. However, few shot learning based on pure language model on X is not robust. [*Such a prompt work for syntax and parsing*$_{\text{TODO}}$]. Large Seq2Seq model has been shown lack of generalizable compositionality. [*more recent paper about this, but in our model, we need to show that our inductive can resolve this.*$_{\text{TODO}}$]

1.1 Motivation

1.1.1 The Need for Inductive Bias

As the no-free-lunch theorem for machine learning said, Baxter (2000); Wolpert *et al.* (1995), inductive biases that influence hypothesis selection is necessary to obtain generalization. [*more formal defintion*$_{\text{TODO}}$] [*Some easy to follow examples for inductive bias*$_{\text{TODO}}$]

1.1.2 Where Do Model Get Inductive Biases

Inductive biases are widely used in the whole history, which can get from every component of machine learning, from training data, to optimization algorithm, feature design, even recent neural architures.

From Training Data. In traditional supervision, models simplely use training data as inductive bias, such error-based perceptron model simply update the weight according to the training data with simple linear models. [*history of perceptron, and its success*$_{\text{TODO}}$].

Feature Engineering and Kernel Methods. [*Inductve biases also can be found in feature engineering,* $_{\text{TODO}}$] However, not every training data are linear seperable, and can be learned by perceptron. One solution to this kernal-based SVM. The design of kernel is new inductive.

Optimization Method. Even in existed computing-heavy and data-heavy scenarios, inductive biases are widely used in well-known deep learning techniques. For example, smoothness assumption in optimization method, such as Stochostic gradient decent, which was proofed as another inductive bias to have better generalization. [*SGD generalization*$_{\text{TODO}}$]

Neural Artchitecture. What's more, inductive bias are widely used in modern deep learning architeuctures. For example, translation invariance for convolutional neural networks (CNN) and pooling, and recurrent assumption of recurrent neural networks (RNN), equivariance over permutation for neural graph networks (GNN), etc.

Towards flexible out-of-distribution and systematic generalization, the search for appropriate inductive biases are necessary for bias-variance trade-off when building an NLP system. In this thesis, I mainly focused on three kinds of inductive biases useful in deep-learning based NLP system: Structural/Compositional Inductive Bias, Natural Language supverision as inductive bias, Representation tuning as inductive bias.

1.1.3 Thesis Statement

[*rephrase*$_{\text{TODO}}$] The use of Compositional inductive bias, Natural language Supervision as inductive bias, representation tuning as inductive bias can get beyond what previous component can give.

1.2 Contribuition and Roadmap

Our study on linguistic structure prediction covers two sets of tasks, including broad-coverage meaning representation and application-specific representations. Our contributions and thesis outlines can be summarized as follows:

1. Structural Biases for Cross-framework Meaning Representation Parsing

   For the alignment distribution, we assume it is gumbel distribution, which can be differentiate sampled by gumble sinkhorn operators.

2. Structural Biases for Pschotherapy Dialog Understanding

3. Natural Language as Inductive Biases for Schema-guided Dialog State Tracking

4. Future Work and Other Applications

5.

1.3 Roadmap

To coherently present the thesis, we think it is helpful to discuss prior work related to an application in its own chapter, instead of putting them all in a single chapter. Therefore, we include a section of related work at the end of each application chapter.

CHAPTER 2

STRUCTURES IN NATURAL LANGUAGE

In the following two sections, first, we will briefly review the background on symbolic representation for natural language including broad-coverage meaning representations and application-specific representations. Then, we will give an overview of the linguistic structure prediction and recent advances on deep structured prediction.

2.1 Symbolic Representations for Natural Language

This is a chapter to indtroduce the background of meaning representations and related works on each of them.

2.1.1 Broad-coverage Semantic Representation

For linguistic analysis, structures have been studied from subword-level morphology **?**, word-level lexicon semantics **?**, to single-sentence syntax/semantic representations **???**, and multi-sentences discourse analysis **???**. This thesis covers broad-coverage graph-based semantic representations in different anchoring types. For lexical anchoring, we cover the DELPH-IN MRS Bi-lexical Dependencies (DM, **?**) and Prague Semantic Dependencies (PSD, **??**), Abstract Meaning Representation (AMR, **?**); While for phrasal-anchoring, we study Universal Conceptual Cognitive Annotation (UCCA, **?**). Detailed introductions of each meaning representation will make the proposal extremly long, thus more details will be added in the thesis chapters.

2.1.1.1 Bi-lexical Semantic Dependency Parsing

2.1.1.2 Abstract Meaning Representation

2.1.1.3 Universal Conceptual Cognitive Anotation

2.1.2 Application-specific Representation: Dialog

In this thesis, we ground the study on application-specific representation on dialogue. In utterance-level, dialogue acts are designed to represent the function of each utterance in

the dialogue. Similarly, Motivational Interview Skill Codes (MISC, **??**) are also proposed to represent the functions of each client and therapist utterance in the psychotherapy dialogue. Frame-based representation for dialogue is popular in modern dialogue systems, where the state of the dialogue is represented as the combinations of intent and slots. Recently, conversational parsing has attract much attention to represent the dialogue state in a more compositional way, such as the hierarchical tree structure in the Task-Oriented dialogue Parsing (TOP, **?**) and so on.

2.1.2.1 Dialog Act

2.1.2.2 Dialog State Tracking

2.1.2.3 Conversational Semantic Representation

2.2 Chapter Summary

Instead of studying the structured prediction for each of them separately, we categorize them according to different factorization patterns, and then we study the factorization-oriented representation learning methods for each of them.

CHAPTER 3

UNIFIED GRAPH-BASED PARSING FOR LEXICAL-ANCHORING

[*adding a summary for this chapter*$_{\texttt{TODO}}$] The design and implementation of broad-coverage and linguistically motivated meaning representation frameworks for natural language is attracting growing attention in recent years. With the advent of deep neural network-based machine learning techniques, we have made significant progress to automatically parse sentences intro structured meaning representation **????**. Moreover, the differences between various representation frameworks has a significant impact on the design and performance of the parsing systems.

Due to the abstract nature of semantics, there is a diverse set of meaning representation frameworks in the literature **?**. In some application scenario, tasks-specific formal representations such as database queries and arithmetic formula have also been proposed. However, primarily the study in computational semantics focuses on frameworks that are theoretically grounded on formal semantic theories, and sometimes also with assumptions on underlying syntactic structures.

Anchoring is crucial in graph-based meaning representation parsing. Training a statistical parser typically starts with a conjectured alignment between tokens/spans and the semantic graph nodes to help to factorize the supervision of graph structure into nodes and edges. In our paper, with evidence from previous research on AMR alignments **??????**, we propose a uniform handling of three meaning representations from `Flavor-0` (DM, PSD) and `Flavor-2` (AMR) into a new group referred to as the **lexical-anchoring** MRs. It supports both explicit and implicit anchoring of semantic concepts to tokens. The other two meaning representations from `Flavor-1` (EDS, UCCA) is referred to the group of **phrasal-anchoring** MRs where the semantic concepts are anchored to phrases as well. /todoadding more about the contributions

3.1 Anchor Analysis on DM, PSD and AMR

The 2019 Conference on Computational Language Learning (CoNLL) hosted a shared task on Cross-Framework Meaning Representation Parsing (MRP 2019, **?**), which encourage participants in building a parser for five different meaning representations in three distinct flavors. `Flavor-0` includes the DELPH-IN MRS Bi-lexical Dependencies (DM, **?**) and Prague Semantic Dependencies (PSD, **??**). Both frameworks under this representation have a syntactic backbone that is (either natively or by-proxy) based on bi-lexical dependency structures. As a result, the semantic concepts in these meaning representations can be anchored to the individual lexical units of the sentence. `Flavor-1` includes Elementary Dependency Structures (EDS, **?**) and Universal Conceptual Cognitive Annotation framework (UCCA, **?**), which shows an explicit, many-to-many anchoring of semantic concepts onto sub-strings of the underlying sentence. Finally, `Flavor-2` includes Abstract Meaning Representation (AMR, **?**), which is designed to abstract the meaning representation away from its surface token. But it leaves open the question of how these are derived. Previous studies have shown that the nodes in AMR graphs are predominantly aligned with the surface lexical units, although explicit anchoring is absent from the AMR representation. In this section, we review the related work supporting the claim of the implicit anchoring in AMR is actually lexical-anchoring, which can be merged into `Flavor-0` when we consider the parsing methods on it.

3.1.1 Explicit Alignments: DM, PSD

[*introduction bi-lexical dependency*$_{\texttt{TODO}}$]

3.1.2 Implicit Anchoring in AMR

AMR tries to abstract the meaning representation away from the surface token. The absense of explicit anchoring can present difficulties for parsing. In this section, by extensive analysis on previous work AMR alignments, we show that AMR nodes can be implicitly aligned to the leixical tokens in a sentence.

3.1.2.1 AMR-to-String Alignments

A straightforward solution to find the missing anchoring in an AMR Graph is to align it with a sentence; We denote it as AMR-to-String alignment.

ISI alignments **?** first linearizes the AMR graph into a sequence, and then use IBM word alignment model **?** to align the linearized sequence of concepts and relations with tokens in the sentence. According to the AMR annotation guidelines and error analysis of ISI aligner, some of the nodes or relations are evoked by subwords, e.g., the whole graph fragment `(p/possible-01 :polarity -)` is evoked by word "impossible", where the subword `"im-"` actually evoked the relation polarity and concept `"-"`; On the other side, sometimes concepts are evoked by multiple words, e.g., named entities, `(c/city :name (n/name :op1 "New":op2 "York"))`, which also happens in explict anchoring of DM and PSD. Hence, aligning and parsing with recategorized graph fragments are a natural solution in aligners and parsers. JAMR aligner **?** uses a set of rules to greedily align single tokens, special entities and a set of multiple word expression to AMR graph fragments, which is widely used in previous AMR parsers (*e.g.* **???????**).

Other AMR-to-String Alignments exists, such as the extended HMM-based aligner. To consider more structure info in the linearized AMR concepts, **?** proposed a Hidden Markov Model (HMM)-based alignment method with a novel graph distance. All of them report over 90% F-score on their own hand-aligned datasets, which shows that AMR-to-String alignments are almost token-level anchoring.

3.1.2.2 AMR-to-Dependency Alignments

see Figure 3.1 **?** first tries to align an AMR graph with a syntactic dependency tree. **?** conducted further analysis on dependency tree and AMR interface. It showed 97% of AMR edges can be evoked by words or the syntactic dependency edges between words. Those nodes in the dependency graph are anchored to each lexical token in the original sentence. Hence, this observation indirectly shows that AMR nodes can be aligned to the lexical tokens in the sentence.

Both AMR-to-String and AMR-to-dependency alignments shows that AMR nodes, including recategorized AMR graph fragements, do have implicit lexical anchoring. Based on this, **?** propose to treat token-node alignments as discrete and exclusive alignment matrix and learn the latent alignment jointly with parsing. Recently, attention-based seq2graph model also achieved the state-of-the-art accuracy on AMR parsing **?**. However, whether the attention weights can be explained as AMR alignments needs more investigation in

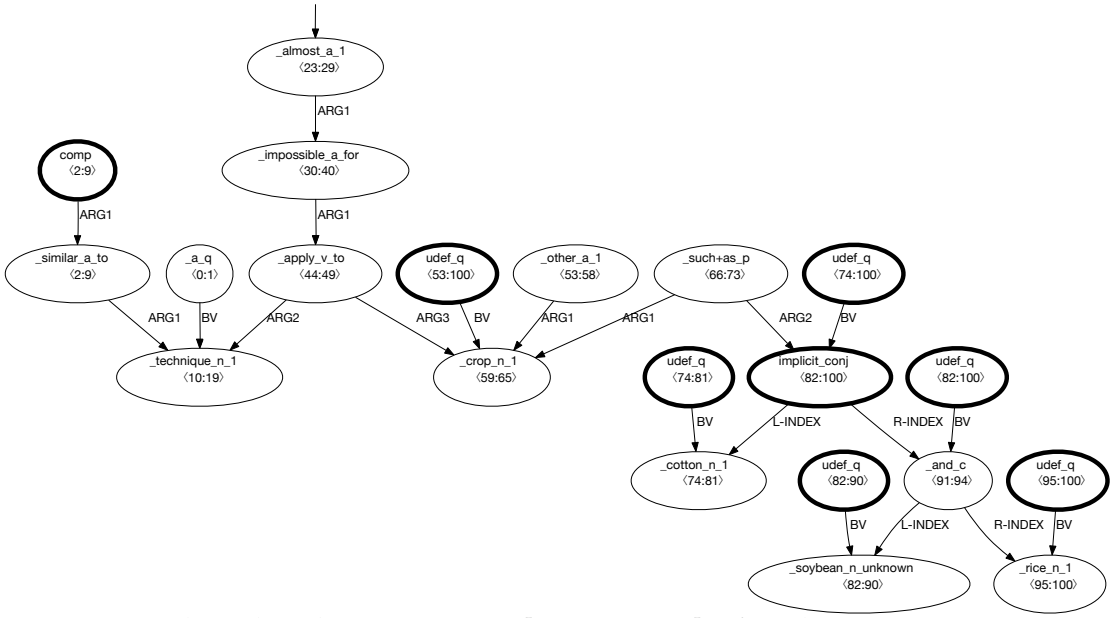


Figure 3.1: Phrasal-anchoring in EDS[wsj#0209013], for the sentence `"A similar technique is almost impossible to apply to other crops, such as cotton, soybeans and rice."`. Bold nodes are similar to the non-terminal nodes in UCCA, which are anchored multiple tokens, thus overlapping with the anchors of other nodes.

future.

3.1.3 Lexical-Anchoring

According to the bi-lexical dependency structures of DM and PSD, and implicit lexical token anchoring on AMR, the nodes/categorized graph fragments of DM, PSD, and AMR are anchored to surface lexical units in an explicit or implict way. Especially, those lexical units do not overlap with each other, and most of them are just single tokens, multiple word expression, or named entities. In other words, when parsing a sentence into DM, PSD, AMR graphs, tokens in the original sentence can be merged by looking up a lexicon dict when preprocessing and then may be considered as a single token for aligning or parsing.

3.2 Graph-based Parsing Framework with Latent Alignment

Before formulating the graph-based model into a probabilistic model as Equation 3.1, we denote some notations: C, R are sets of concepts (nodes) and relations (edges) in the graph, and w is a sequence of tokens. $a \in \mathbb{Z}^m$ as the alignment matrix, each a_i is the index

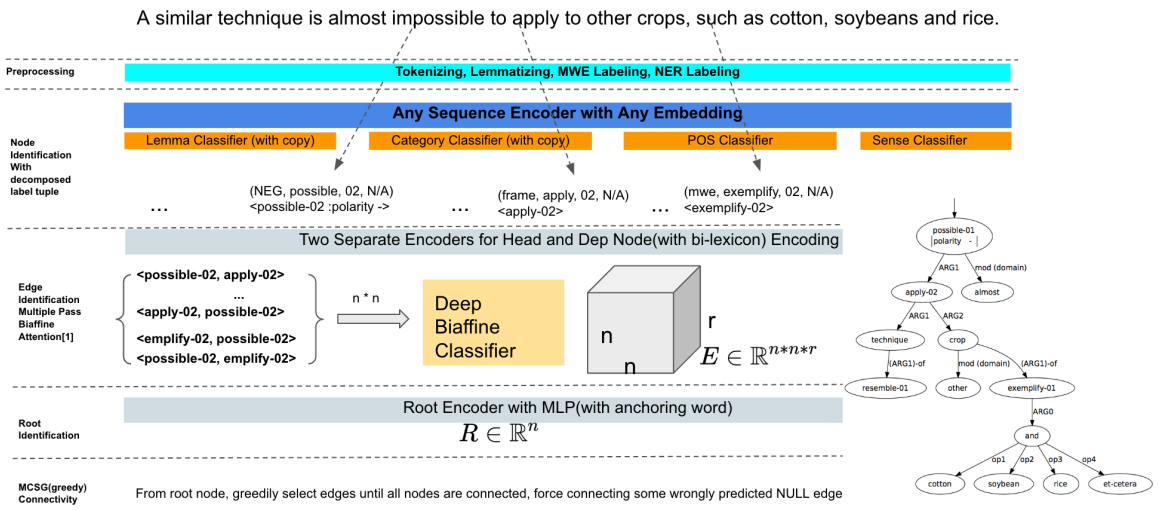


Figure 3.2: Architecture of graph-based model and inference, for running exmaple [wsj#0209013]

of aligned token where ith node aligned to. When modeling the negative log likelihood loss (NLL), with independence assumption between each node and edge, we decompose it into node- and edge-identification pipelines.

$$
\begin{aligned}
& NLL(P(C, R \mid w)) \\
&= -\log(P(C, R \mid w)) \\
&= -\log(\sum_{a} P(a) P(C, R \mid w, a)) \\
&= -\log\left(\sum_{a} P(a) P(R \mid w, a, c) P(c \mid w, a)\right) \\
&= -\log\left(\sum_{a} P(a) \prod_{i}^{m} P(c_i \mid h_{a_i}) \right. \\
&\qquad \left. \cdot \prod_{i,j=1}^{m} P(r_{ij} \mid h_{a_i}, c_i, h_{a_j}, c_j)\right)
\end{aligned}
\tag{3.1}
$$

In DM, PSD, and AMR, every token will only be aligned once. Hence, we train a joint model to maximize the above probability for both node identification $P(c_i \mid h_{a_i})$ and edge identification $P(r_{ij} \mid h_{a_i, c_i, h_{a_j}, c_j})$, and we need to marginalize out the discrete alignment variable a.

3.2.1 Alignment Model

The above model can support both explicit alignments for DM, PSD, and implicit alignments for AMR.

3.2.1.1 Explicit Alignments

For DM, PSD, with explicit alignments $a*$, we can use $P(a^*) = 1.0$ and other alignments $P(a|a \neq a^*) = 0.0$

3.2.1.2 Implicit Alignments

For AMR, without gold alignments, one requires to compute all the valid alignments and then condition the node- and edge-identification methods on the alignments.

$$\begin{aligned} &\log(P(C, R \mid w)) \geq \\ &E_Q[\log(P_\theta(c \mid w, a)P_\Phi(R \mid w, a, c))] \\ &- D_{KL}(Q_\Psi(a \mid c, R, w) \mid\mid P(a)) \end{aligned} \tag{3.2}$$

However, it is computationally intractable to enumerate all alignments. We estimate posterior alignments model Q as Equation 3.3, please refer to **?** for more details.

- Applying variational inference to reduce it into Evidence Lower Bound (ELBO, **?**)
- The denominator Z_Ψ in Q can be estimated by Perturb-and-Max(MAP) **?**

$$Q_\Psi(a \mid c, R, w) = \frac{\exp(\sum_{i=1}^{n} \phi(g_i, h_{a_i}))}{Z_\Psi(c, w)} \tag{3.3}$$

  Where $\phi(g_i, h_{a_i})$ score each alignment link between node i and the corresponding words, g_i is node encoding, and h_{a_i} is encoding for the aligned token.

- Discrete `argmax` of a permutation can be estimated by Gumbel-Softmax Sinkhorn Networks **??**

3.2.2 Node Identification

Node Identification predicts a concept c given a word. A concept can be either *NULL* (when there is no semantic node anchoring to that word, e.g., the word is dropped), or a node label (e.g., lemma, sense, POS, name value in AMR, frame value in PSD), or other node properties. One challenge in node identification is the data sparsity issue. Many of the labels are from open sets derived from the input token, e.g., its lemma. Moreover, some labels are constrained by a deterministic label set given the word. Hence, we designed a copy mechanism **?** in our neural network architecture to decide whether to copying deterministic label given a word or estimate a classification probability from a fixed label set.

3.2.3 Edge Identification

By assuming the independence of each edge, we model the edges probabilites independently. Given two nodes and their underlying tokens, we predict the edge label as the semantic relation between the two concepts with a bi-affine classifier **?**.

3.2.4 Inference

In our two-stage graph-based parsing, after nodes are identified, edge identification only output a probility distribution over all the relations between identified nodes. However, we need to an inference algorithm to search for the maximum spanning connected graph from all the relations. We use **?** to greedily select the most valuable edges from the identified nodes and their relations connecting them. As shown in Figure 3.2, an input sentence goes through preprocessing, node identification, edge identification, root identification, and MCSG to generate a final connected graph as structured output.

3.3 Latent Alignment Model

3.3.1 Continuous Relaxation for Discrete Alignments

3.3.2 VAE, Perturb-and-Map, Gumble Sinkhorn

3.4 Experiments and Results

3.4.1 Dataset and Evaluation

For DM, PSD, EDS, we split the training set by taking WSJ section (00-19) as training, and section 20 as dev set. For other datasets, when developing and parameter tuning, we use splits with a ratio of 25:1:1. In our submitted model, we did not use multitask learning for training. Following the unified MRP metrics in the shared tasks, we train our model based on the development set and finally evaluate on the private test set. For more details of the metrics, please refer to the summarization of the MRP 2019 task **?**,

3.4.2 Summary of Implementation

We summarize our implementation for five meaning representations as Table **??**. As we mentioned in the previous sections, we use latent-alignment graph-based parsing for lexical anchoring MRs (DM, PSD, AMR), and use CKY-based constituent parsing phrasal anchoring in MRs (UCCA, EDS). This section gives information about various decision for our models.

	Lexicon Anchoring		
	DM	PSD	AMR
Top	1	$\geq$ 1 (11.56%)	1
Node Label	Lemma	Lemma(*)	Lemma(*) + NeType(143+)
Node Properties	POS semi(160*)_args(25)	POS wordid_sense(25)	constant values polarity, Named entity
Edge Label	(45)	(91)	(94+)
Edge Properties	N/A	N/A	N/A
Connectivity	True	True	True
Training Data	35656	35656	57885
Test Data	3269	3269	1998

Table 3.1: Detailed classifiers in our model, round bracket means the number of ouput classes of our classify, * means copy mechanism is used in our classifier. At the end of shared task, EDS are not fully supported to get an official results, we leave it as our future work.

3.4.2.1 Top

The first row “Top" shows the numbers of root nodes in the graph. We can see that for PSD, 11.56% of graphs with more than 1 top nodes. In our system, we only predict one top node with a N (N is size of identified nodes) way classifier, and then fix this with a post-processing strategy. When our model predicts one node as the top node, and if we find additional coordination nodes with it, we add the coordination node also as the top node.

3.4.2.2 Node

Except for UCCA, all other four MRs have labeled nodes, the row “Node Label" shows the templates of a node label. For DM and PSD, the node label is usually the lemma of its underlying token. But the lemma is neither the same as one in the given companion data nor the predicted by Stanford Lemma Annotators. One common challenge for predicting the node labels is the open label set problem. Usually, the lemma is one of the morphology derivations of the original word. But the derivation rule is not easy to create manually. In our experiment, we found that handcrafted rules for lemma prediction only works worse than classification with copy mechanism, except for DM.

For AMR and EDS, there are other components in the node labels beyond the lemma. Especially, the node label for AMR also contains more than 143 fine-grained named entity types; for EDS, it uses the full SEM-I entry as its node label, which requires extra classifiers

for predicting the corresponding sense. In addition to the node label, the properties of the label also need to be predicted. Among them, node properties of DM are from the SEMI sense and arguments handler, while for PSD, senses are constrained the senses in the predefined the vallex lexicon.

3.4.2.3 Edge

Edge predication is another challenge in our task because of its large label set (from 45 to 94) as shown in row “Edge Label", the round bracket means the number of output classes of our classifiers. For Lexical anchoring MRs, edges are usually connected between two tokens, while phrasal anchoring needs extra effort to figure out the corresponding span with that node. For example, in UCCA parsing, To predict edge labels, we first predicted the node spans, and then node labels based that span, and finally we transform back the node label into edge label.

3.4.2.4 Connectivity

Beside the local label classification for nodes and edges, there are other global structure constraints for all five MRs: All the nodes and edges should eventually form a connected graph. For lexical anchoring, we use MSCG algorithm to find the maximum connected graph greedily; For phrasal anchoring, we use dynamic programming to decoding the constituent tree then deterministically transforming back to a connected UCCA Graph [1]

3.4.3 Model Setup

For lexical-anchoring model setup, our network mainly consists of node and edge prediction model. For AMR, DM, and PSD, they all use one layer Bi-directional LSTM for input sentence encoder, and two layers Bi-directional LSTM for head or dependent node encoder in the bi-affine classifier. For every sentence encoder, it takes a sequence of word embedding as input (We use 300 dimension Glove here), and then their output will pass a softmax layer to predicting output distribution. For the latent AMR model, to model the posterior alignment, we use another Bi-LSTM for node sequence encoding. For phrasal-anchoring model setup, we follow the original model set up in **?**, and we use

[1] Due to time constraint, we ignored all the discontinuous span and remote edges in UCCA

8-layers 8-headers transformer with position encoding to encode the input sentence.

For all sentence encoders, we also use the character-level CNN model as character-level embedding without any pre-trained deep contextualized embedding model. Equipping our model with Bert or multi-task learning is promising to get further improvement. We leave this as our future work.

Our models are trained with Adam **?**, using a batch size 64 for a graph-based model, and 250 for CKY-based model. Hyper-parameters were tuned on the development set, based on labeled F1 between two graphs. We exploit early-stopping to avoid over-fitting.

3.4.4 Results

At the time of official evaluation, we submitted three lexical anchoring parser, and then we submitted another phrasal-anchoring model for UCCA parsing during post-evaluation stage, and we leave EDS parsing as future work. The following sections are the official results and error breakdowns for lexical-anchoring and phrasal-anchoring respectively.

3.4.4.0.1 Official Results on Lexical Anchoring Table 3.2 shows the official results for our lexical-anchoring models on AMR, DM, PSD. By using our latent alignment based AMR parser, our system ranked top 1 in the AMR subtask, and outperformed the top 5 models in large margin. Our parser on PSD ranked 6, but only 0.02% worse then the top 5 model. However, official results on DM and PSD shows that there is still around 2.5 points performance gap between our model and the top 1 model.

MR	Ours (P/R/F1)	Top 1/3/5 (F1)
AMR(1)	**75/71/73.38**	**73.38/71.97/71.72**
PSD(6)	89/89/**88.75**	90.76/89.91/**88.77**
DM(7)	93/92/92.14	94.76/94.32/93.74

Table 3.2: Official results overview on unified MRP metric, we selected the performance from top 1/3/5 system(s) for comparison

3.4.4.0.2 Official Results on Phrasal Anchoring Table 4.2 shows that our span-based CKY model for UCCA can achieve 74.00 F1 score on official test set, and ranked 5th. When adding ELMo **?** into our model, it can further improve almost 3 points on it.

MR	Ours (P/R/F1)	Top 1/3/5 (F1)
UCCA(5)	80.83/73.42/**76.94**	81.67/77.80/73.22
EDS	N/A	94.47/90.75/89.10

Table 3.3: Official results overview on unified MRP metric, we selected the performance from top 1/3/5 system(s) for comparison. It shows our UCCA model for post-evluation can rank 5th

3.4.5 Error Breakdown

Table 3.4, 3.5, 3.6 and 4.3 shows the detailed error breakdown of AMR, DM, PSD and UCCA respectively. Each column in the table shows the F1 score of each subcomponent in a graph: top nodes, node lables, node properties, node anchors, edge labels, and overall F1 score. No anchors for AMR, and no node label and propertis for UCCA. We show the results of MRP metric on two datasets. "all" denotes all the examples for that specific MR, while lpps are a set of 100 sentences from `The Little Prince`, and annotated in all five meaning representations. To better understand the performance, we also reported the official results from two baseline models TUPA **?** and ERG **?**.

	data	tops	labels	prop	edges	all
TUPA single	all	63.95	57.20	22.31	36.41	44.73
	lpps	71.96	55.52	26.42	36.38	47.04
TUPA multi	all	61.30	39.80	27.70	27.35	33.75
	lpps	72.63	50.11	20.25	33.12	43.38
Ours(1)	all	65.92	82.86	**77.26**	63.57	**73.38**
	lpps	72.00	78.71	58.93	63.96	71.11
Top 2	all	78.15	82.51	71.33	63.21	72.94
	lpps	83.00	76.24	51.79	60.43	69.03

Table 3.4: Our parser on AMR ranked 1st. This table shows the error breakdown when comparing to the baseline TUPA model and top 2 **?** in official results

	data	tops	labels	prop	anchors	edges	all
ERG	all	91.83	98.22	95.25	98.82	90.76	95.65
	lpps	95.00	97.32	97.75	99.46	92.71	97.03
Top 1	all	93.23	94.14	94.83	98.40	91.55	94.76
	lpps	96.48	91.85	94.36	99.04	93.28	94.64
Ours(7)	all	70.95	93.96	92.13	97.25	86.45	92.14
	lpps	84.00	90.55	91.91	97.96	87.24	91.82

Table 3.5: Our parser on DM ranked 7th. This table shows the error breakdown when comparing to the model ranked Top 1 **?** in official results

	data	tops	labels	prop	anchors	edges	all
Top 1	all	93.45	94.68	91.78	98.35	77.79	90.76
	lpps	93.33	91.73	84.37	98.40	77.63	88.34
Ours(6)	all	82.01	94.18	91.28	96.94	72.40	88.75
	lpps	85.85	90.48	82.63	95.97	73.60	85.83

Table 3.6: Our parser on PSD ranked 6th. This table shows the error breakdown when comparing to the model ranked top 1 **?** in official results

3.4.5.1 Error Analysis on Lexical-Anchoring

As shown in Table 3.4, our AMR parser is good at predicting node properties and consistently perform better than other models in all subcomponent, except for top prediction. Node properties in AMR are usually named entities, negation, and some other quantity entities. In our system, we recategorize the graph fragements into a single node, which helps for both alignments and structured inference for those special graph fragments. We see that all our 3 models perform almost as good as the top 1 model of each subtask on node label prediction, but they perform worse on top and edge prediction. It indicates that our bi-affine relation classifier are main bottleneck to improve. Moreover, we found the performance gap between node labels and node anchors are almost consistent, it indicates that improving our model on predicting `NULL` nodes may further improve node label prediction as well. Moreover, we believe that multi-task learning and pre-trained deep models such as BERT **?** may also boost the performance of our paser in future.

3.4.5.2 Error Analysis on Phrasal-Anchoring

According to Table 4.3, our model with ELMo works slightly better than the top 1 model on anchors prediction. It means our model is good at predicting the nodes in UCCA and we belive that it is also helpful for prediction phrasal anchoring nodes in EDS.

However, when predicting the edge and edge attributes, our model performs 7-8 points worse than the top 1 model. In UCCA, an edge label means the relation between a parent nodes and its children. In our UCCA transformation, we assign edge label as the node label of its child and then predict with only child span encoding. Thus it actually misses important information from the parent node. Hence, in future, more improvement can be done to use both child and parent span encoding for label prediction, or even using another span-based bi-affine classifier for edge prediction, or remote edge recovering.

	data	tops	anchors	edge	attr	all
TUPA single	all	78.73	69.17	16.96	15.18	27.56
	lpps	86.03	76.26	28.32	24.00	40.06
TUPA multi	all	84.92	65.74	12.99	9.07	23.65
	lpps	88.89	77.76	26.45	18.32	41.04
?	all	1.00	95.36	72.66	61.98	81.67
	lpps	1.00	96.99	73.08	48.37	82.61
Ours(*5)	all	98.85	94.92	60.17	0.00	**74.00**
	lpps	96.00	96.75	60.20	0.00	75.17
Ours + ELMo	all	99.38	95.70	64.88	0.00	**76.94**
	lpps	98.00	96.84	66.63	0.00	78.77

Table 3.7: Our UCCA parser in post-evaluation ranked 5th according to the original official evaluation results. This table shows the error breakdown when comparing to the model ranked top 1 **?** in official results. * denotes the ranking of post-evaluation results

3.5 Related Work

[*add related work*$_{\text{TODO}}$]

3.6 Chapter Summary

In summary, by analyzing the AMR alignments, we show that implicit AMR anchoring is actually lexical-anchoring based. Thus we propose to regroup five meaning representations as two groups: lexical-anchoring and phrasal-anchoring. For lexical anchoring, we suggest to parse DM, PSD, and AMR in a unified latent-alignment based parsing framework. Our submission ranked top 1 in AMR sub-task, ranked 6th and 7th in PSD and DM tasks. For phrasal anchoring, by reducing UCCA graph into a constituent tree-like structure, and then use the span-based CKY parsing to parse their tree structure, our method would rank 5th in the original official evaluation results.

CHAPTER 4

TREE-FACTORIZATION FOR PHRASAL-ANCHORING

4.1 Analysis on Phrasal Anchoring in UCCA and TOP

As a result, the semantic concepts in these meaning representations can be anchored to the individual lexical units of the sentence. `Flavor-1` includes Elementary Dependency Structures (EDS, **?**) and Universal Conceptual Cognitive Annotation framework (UCCA, **?**), which shows an explicit, many-to-many anchoring of semantic concepts onto sub-strings of the underlying sentence.

However, different from the lexical anchoring without overlapping, nodes in EDS and UCCA may align to larger overlapped word spans which involves syntactic or semantic pharsal structure. Nodes in UCCA do not have node labels or node properties, but all the nodes are anchored to the spans of the underlying sentence. Furthermore, the nodes in UCCA are linked into a hierarchical structure, with edges going between parent and child nodes. With certain exceptions (e.g. remote edges), the majority of the UCCA graphs are tree-like structures. According to the position as well as the anchoring style, nodes in UCCA can be classified into the following two types:

1. **Terminal nodes** are the leaf semantic concepts anchored to individual lexical units in the sentence

2. **Non-terminal nodes** are usually anchored to a span with more than one lexical units, thus usually overlapped with the anchoring of terminal nodes.

The similar classification of anchoring nodes also applies to the nodes in EDS, although they do not regularly form a recursive tree like UCCA. As the running example in Figure 3.1, most of the nodes belongs to terminal nodes, which can be explicitly anchored to a single token in the original sentence. However, those bold non-teriminal nodes are

anchored to a large span of words. For example, the node "undef_q" with span <53:100> is aligned to the whole substring starting from “other crops" to the end; The abstract node with label imp_conj are corresponding to the whole coordinate structure between soybeans and rice

In summary, by treating AMR as an implicitly lexically anchored MR, we propose a simplified taxonomy for parsing the five meaning representation in this shared task.

- Lexical-anchoring: DM, PSD, AMR

- Phrasal-anchoring: EDS, UCCA

4.2 Minimal Span-based CKY Parsing Framework

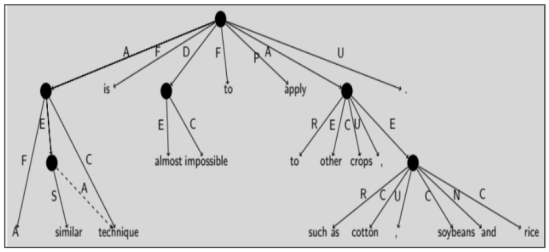
Let us now see our phrasal-anchoring parser for UCCA. We introduce the transformation we used to reduce UCCA parsing into a consituent parsing task, and finally introduce the detailed CKY model for the constituent parsing.

4.2.1 Graph-to-CT Transformation

We propose to transform a graph into a constituent tree structure for parsing, which is also used in recent work **?**. Figure 4.1 shows an example of transforming a UCCA graph into a constituent tree. The primary transformation assigns the original label of an edge to its child node. Then to make it compatible with parsers for standard PennTree Bank format, we add some auxiliary nodes such as special non-terminal nodes, TOP, HEAD, and special terminal nodes TOKEN and MWE. We remove all the “remote” annotation in UCCA since the constituent tree structure does not support reentrance. A fully compatible transformation should support both graph-to-tree and tree-to-graph transformation.

In our case, due to time constraints, we remove those remote edges and reentrance edges during training. Besides that, we also noticed that for multi-word expressions, the children of a parent node might not be in a continuous span (i.e., discontinuous constituent), which is also not supported by our constituent tree parser. Hence, when training the tree parser, by reattaching the discontinuous tokens to its nearest continuous parent nodes, we force every sub span are continuous in the transformed trees. We leave the postprocessing to recover those discontinuous as future work.

For inference, given an input sentence, we first use the trained constituent tree parsing model to parse it into a tree, and then we transform a tree back into a directed graph by assigning the edge label as its child's node label, and deleting those auxiliary labels, adding anchors to every remaining node.




```
(TOP
        (HEAD
                (:A
                        (:F   (TOK A))
                        (:E   (:S (TOK similar)))
                        (:C   (TOK technique)))
                (:F     (TOK is))
                (:D     (:E almost) (:C impossible))
                (:F     (TOK to))
                (:P     (TOK apply))
                (:A     (:R (TOK to))
                        (:E (TOK other))
                        (:C (TOK crops))
                        (:U (TOK ,))
                        (:C (TOK soybeans))
                        (:N (TOK and))
                        (:C (TOK rice))
                (:U     (TOK .))
        )
)
```

Figure 4.1: UCCA to Constituent Tree Transformation for [wsj#0209013]

4.2.2 CKY Parsing

After transforming the UCCA graph into a constituent tree, we reduce the UCCA parsing into a constituent tree parsing problem. Similar to the previous work on UCCA constituent tree parsing **?**, we use a minimal span-based CKY parser for constituent tree parsing. The intuition is to use dynamic programming to recursively split the span of a sentence recursively, as shown in Figure 4.1. The entire sentence can be splitted from top to bottom until each span is a single unsplittable tokens. For each node, we also need to assign a label. Two simplified assumptions are made when predicting the hole tree given a sentence. However, different with previous work, we use 8-layers with 8 heads transformer encoder, which shows better performance than LSTM in **?**.

4.2.2.0.1 Tree Factorization In the graph-to-tree transformation, we move the edge label to its child node. By assuming the labels for each node are independent, we factorize the tree structure prediction as independent span-label prediction as Equation 4.1. However, this assumption does not hold for UCCA. Please see more error analysis in §4.4.1.1

$$\begin{aligned} T^* &= \arg\max_T s(T) \\ s(T) &= \sum_{(i,j,l)\in T} s(i,j,l) \end{aligned} \tag{4.1}$$

4.2.2.0.2 CKY Parsing By assuming the label prediction is independent of the splitting point, we can further factorize the whole tree as the following dynamic programming in Equation 4.2.

$$\begin{aligned} s_{\text{best}}(i,i+1) &= \max_l s(i,i+1,l) \\ s_{\text{best}}(i,j) &= \max_l s(i,j,l) \\ &\quad + \max_k [s_{\text{best}}(i,k) + s_{\text{best}}(k,j)] \end{aligned} \tag{4.2}$$

4.2.3 Span Encoding

For each span (i,j), we represent the span encoding vector $v_{(i,j)} = [\vec{y}_j - \vec{y}_i] \oplus [\overleftarrow{y}_{j+1} - \overleftarrow{y}_{i+1}]$. $\oplus$ denotes vector concatenation. Assuming a bidirectional sentence encoder, we use the forward and backward encodings $\vec{y}_i$ and $\overleftarrow{y}_i$ of i_{th} word. Following the previous work, and we also use the loss augmented inference training. More details about the network architecture are in the Section 3.4.3

4.3 Experiemnts

4.4 Results

At the time of official evaluation, we submitted three lexical anchoring parser, and then we submitted another phrasal-anchoring model for UCCA parsing during post-evaluation stage, and we leave EDS parsing as future work. The following sections are the official results and error breakdowns for phrasal-anchoring .

4.4.1 Results on UCCA

Table 4.2 shows that our span-based CKY model for UCCA can achieve 74.00 F1 score on official test set, and ranked 5th. When adding ELMo **?** into our model, it can further

	Phrase Anchoring	
	EDS	UCCA
Top	1	1
Node Label	_lemma(*)_semi_sense	N/A
Node Properties	 carg: constant value	N/A N/A
Edge Label	(45)	(15)
Edge Properties	N/A	“remote"
Connectivity	True	True
Training Data	35656	6485
Test Data	3269	1131

Table 4.1: Detailed classifiers in our model, round bracket means the number of ouput classes of our classify, * means copy mechanism is used in our classifier. At the end of shared task, EDS are not fully supported to get an official results, we leave it as our future work.

improve almost 3 points on it.

MR	Ours (P/R/F1)	Top 1/3/5 (F1)
UCCA(5)	80.83/73.42/**76.94**	81.67/77.80/73.22
EDS	N/A	94.47/90.75/89.10

Table 4.2: Official results overview on unified MRP metric, we selected the performance from top 1/3/5 system(s) for comparison. It shows our UCCA model for post-evluation can rank 5th

4.4.1.1 Error Breakdown on UCCA

Table 3.4, 3.5, 3.6 and 4.3 shows the detailed error breakdown of AMR, DM, PSD and UCCA respectively. Each column in the table shows the F1 score of each subcomponent in a graph: top nodes, node lables, node properties, node anchors, edge labels, and overall F1 score. No anchors for AMR, and no node label and propertis for UCCA. We show the results of MRP metric on two datasets. “all" denotes all the examples for that specific MR, while lpps are a set of 100 sentences from `The Little Prince`, and annotated in all five meaning representations. To better understand the performance, we also reported the official results from two baseline models TUPA **?** and ERG **?**.

According to Table 4.3, our model with ELMo works slightly better than the top 1 model on anchors prediction. It means our model is good at predicting the nodes in UCCA and we belive that it is also helpful for prediction phrasal anchoring nodes in EDS.

	data	tops	anchors	edge	attr	all
TUPA single	all	78.73	69.17	16.96	15.18	27.56
	lpps	86.03	76.26	28.32	24.00	40.06
TUPA multi	all	84.92	65.74	12.99	9.07	23.65
	lpps	88.89	77.76	26.45	18.32	41.04
?	all	1.00	95.36	72.66	61.98	81.67
	lpps	1.00	96.99	73.08	48.37	82.61
Ours(*5)	all	98.85	94.92	60.17	0.00	**74.00**
	lpps	96.00	96.75	60.20	0.00	75.17
Ours + ELMo	all	99.38	95.70	64.88	0.00	**76.94**
	lpps	98.00	96.84	66.63	0.00	78.77

Table 4.3: Our UCCA parser in post-evaluation ranked 5th according to the original official evaluation results. This table shows the error breakdown when comparing to the model ranked top 1 **?** in official results. * denotes the ranking of post-evaluation results

However, when predicting the edge and edge attributes, our model performs 7-8 points worse than the top 1 model. In UCCA, an edge label means the relation between a parent nodes and its children. In our UCCA transformation, we assign edge label as the node label of its child and then predict with only child span encoding. Thus it actually misses important information from the parent node. Hence, in future, more improvement can be done to use both child and parent span encoding for label prediction, or even using another span-based bi-affine classifier for edge prediction, or remote edge recovering.

4.5 Related Work

4.6 Chapter Summary

In summary, by analyzing the AMR alignments, we show that implicit AMR anchoring is actually lexical-anchoring based. Thus we propose to regroup five meaning representations as two groups: lexical-anchoring and phrasal-anchoring. For lexical anchoring, we suggest to parse DM, PSD, and AMR in a unified latent-alignment based parsing framework. Our submission ranked top 1 in AMR sub-task, ranked 6th and 7th in PSD and DM tasks. For phrasal anchoring, by reducing UCCA graph into a constituent tree-like structure, and then use the span-based CKY parsing to parse their tree structure, our method would rank 5th in the original official evaluation results.

CHAPTER 5

MODELING ON SENTENCE-ANCHORING FOR DIALOG IN THERAPY

5.1 Background and Motivation

Motivational Interviewing (MI) is a style of psychotherapy that seeks to resolve a client's ambivalence towards their problems, thereby motivating behavior change. Several meta-analyses and empirical studies have shown the high efficacy and success of MI in psychotherapy **???**. However, MI skills take practice to master and require ongoing coaching and feedback to sustain **?**. Given the emphasis on using specific types of linguistic behaviors in MI (*e.g.*, open questions and reflections), fine-grained behavioral coding plays an important role in MI theory and training.

Motivational Interviewing Skill Codes (MISC, table **??**) is a framework for coding MI sessions. It facilitates evaluating therapy sessions via utterance-level labels that are akin to dialogue acts **??**, and are designed to examine therapist and client behavior in a therapy session.[1]

As Table **??** shows, client labels mark utterances as discussing changing or sustaining problematic behavior (CT and ST, respectively) or being neutral (FN). Therapist utterances are grouped into eight labels, some of which (RES, REC) correlate with improved outcomes, while MI non-adherent (MIN) utterances are to be avoided. MISC labeling was originally done by trained annotators performing multiple passes over a session recording or a transcript. Recent NLP work speeds up this process by automatically annotating a completed MI session (*e.g.*, **???**).

Instead of providing feedback to a therapist after the completion of a session, can a dialogue

[1]The original MISC description of **?** included 28 labels (9 client, 19 therapist). Due to data scarcity and label confusion, various strategies are proposed to merge the labels into a coarser set. We adopt the grouping proposed by **?**; the appendix gives more details.

i	s_i	u_i	l_i
1	T:	Have you used drugs recently?	QUC
2	C:	I stopped for a year, but relapsed.	FN
3	T:	You will suffer if you keep using.	MIN
4	C:	Sorry, I just want to quit.	CT
...		...	...

Table 5.1: An example of ongoing therapy session

observer provide online feedback? While past work has shown the helpfulness of post hoc evaluations of a session, prompt feedback would be more helpful, especially for MI non-adherent responses. Such feedback opens up the possibility of the dialogue observer influencing the therapy session. It could serve as an assistant that offers suggestions to a therapist (novice or veteran) about how to respond to a client utterance. Moreover, it could help alert the therapist to potentially important cues from the client (specifically, CT or ST).

5.2 Task Definitions

In this section, we will formally define the two NLP tasks corresponding to the vision in §**??** using the conversation in table 5.1 as a running example.

Suppose we have an ongoing MI session with utterances $u_1, u_2, \cdots, u_n$: together, the dialogue history H_n. Each utterance u_i is associated with its speaker s_i, either C (client) or T (therapist). Each utterance is also associated with the MISC label l_i, which is the object of study. We will refer to the last utterance u_n as the *anchor*.

We will define two classification tasks over a fixed dialogue history with n elements — *categorization* and *forecasting*. As the conversation progresses, the history will be updated with a sliding window. Since the therapist and client codes share no overlap, we will design separate models for the two speakers, giving us four settings in all.

Task 1: Categorization. The goal of this task is to provide real-time feedback to a therapist during an ongoing MI session. In the running example, the therapist's confrontational response in the third utterance is not MI adherent (MIN); an observer should flag it as such to bring the therapist back on track. The client's response, however, shows an inclination to change their behavior (CT). Alerting a therapist (especially a novice) can help guide the conversation in a direction that encourages it.

In essence, we have the following real-time classification task: *Given the dialogue history H_n which includes the speaker information, predict the MISC label l_n for the last utterance u_n.*

The key difference from previous work in predicting MISC labels is that we are restricting the input to the real-time setting. As a result, models can only use the dialogue history to predict the label, and in particular, we can not use models such as a conditional random field or a bi-directional LSTM that need both past and future inputs.

Task 2: Forecasting. A real-time therapy observer may be thought of as an expert therapist who guides a session with suggestions to the therapist. For example, after a client discloses their recent drug use relapse, a novice therapist may respond in a confrontational manner (which is not recommended, and hence coded MIN). On the other hand, a seasoned therapist may respond with a complex reflection (REC) such as *"Sounds like you really wanted to give up and you're unhappy about the relapse."* Such an expert may also anticipate important cues from the client.

The forecasting task seeks to mimic the intent of such a seasoned therapist: *Given a dialogue history H_n and the next speaker's identity s_{n+1}, predict the MISC code l_{n+1} of the yet unknown next utterance u_{n+1}.*

The MISC forecasting task is a previously unstudied problem. We argue that forecasting the type of the next utterance, rather than selecting or generating its text as has been the focus of several recent lines of work (*e.g.*, **???**), allows the human in the loop (the therapist) the freedom to creatively participate in the conversation within the parameters defined by the seasoned observer, and perhaps even rejecting suggestions. Such an observer could be especially helpful for training therapists **?**. The forecasting task is also related to recent work on detecting antisocial comments in online conversations **?** whose goal is to provide an early warning for such events.

5.3 Models for MISC Prediction

Modeling the two tasks defined in §**??** requires addressing four questions: (1) How do we encode a dialogue and its utterances? (2) Can we discover discriminative words in each utterance? (3) Can we discover which of the previous utterances are relevant? (4) How do we handle label imbalance in our data? Many recent advances in neural networks can be seen as plug-and-play components. To facilitate the comparative study of models, we

will describe components that address the above questions. In the rest of the paper, we will use **boldfaced** terms to denote vectors and matrices and SMALL CAPS to denote component names.

5.3.1 Encoding Dialogue

Since both our tasks are classification tasks over a dialogue history, our goal is to convert the sequence of utterences into a single vector that serves as input to the final classifier.

We will use a hierarchical recurrent encoder (**???**, and others) to encode dialogues, specifically a hierarchical gated recurrent unit (HGRU) with an utterance and a dialogue encoder. We use a bidirectional GRU over word embeddings to encode utterances. As is standard, we represent an utterance u_i by concatenating the final forward and reverse hidden states. We will refer to this utterance vector as $\boldsymbol{v}_i$. Also, we will use the hidden states of each word as inputs to the attention components in §5.3.2. We will refer to such contextual word encoding of the j^{th} word as $\boldsymbol{v}_{ij}$. The dialogue encoder is a unidirectional GRU that operates on a concatenation of utterance vectors $\boldsymbol{v}_i$ and a trainable vector representing the speaker s_i.[2] The final state of the GRU aggregates the entire dialogue history into a vector $\boldsymbol{H}_n$.

The HGRU skeleton can be optionally augmented with the word and dialogue attention described next. All the models we will study are two-layer MLPs over the vector $\boldsymbol{H}_n$ that use a ReLU hidden layer and a softmax layer for the outputs.

5.3.2 Word-level Attention

Certain words in the utterance history are important to categorize or forecast MISC labels. The identification of these words may depend on the utterances in the dialogue. For example, to identify that an utterance is a simple reflection (RES) we may need to discover that the therapist is mirroring a recent client utterance; the example in table **??** illustrates this. Word attention offers a natural mechanism for discovering such patterns.

We can unify a broad collection of attention mechanisms in NLP under a single high

[2]For the dialogue encoder, we use a unidirectional GRU because the dialogue is incomplete. For words, since the utterances are completed, we can use a BiGRU.

level architecture **?**. We seek to define attention over the word encodings $\boldsymbol{v}_{ij}$ in the history (called queries), guided by the word encodings in the anchor $\boldsymbol{v}_{nk}$ (called keys). The output is a sequence of attention-weighted vectors, one for each word in the i^{th} utterance. The j^{th} output vector $\boldsymbol{a}_j$ is computed as a weighted sum of the keys:

$$\boldsymbol{a}_{ij} = \sum_k \alpha_j^k \boldsymbol{v}_{nk} \tag{5.1}$$

The weighting factor α_j^k is the attention weight between the j^{th} query and the k^{th} key, computed as

$$\alpha_j^k = \frac{\exp\left(f_m(\boldsymbol{v}_{nk}, \boldsymbol{v}_{ij})\right)}{\sum_{j'} \exp\left(f_m(\boldsymbol{v}_{nk}, \boldsymbol{v}_{ij'})\right)} \tag{5.2}$$

Here, f_m is a match scoring function between the corresponding words, and different choices give us different attention mechanisms.

Finally, a combining function f_c combines the original word encoding $\boldsymbol{v}_{ij}$ and the above attention-weighted word vector $\boldsymbol{a}_{ij}$ into a new vector representation $\boldsymbol{z}_{ij}$ as the final representation of the query word encoding:

$$\boldsymbol{z}_{ij} = f_c(\boldsymbol{v}_{ij}, \boldsymbol{a}_{ij}) \tag{5.3}$$

The attention module, identified by the choice of the functions f_m and f_c, converts word encodings in each utterance $\boldsymbol{v}_{ij}$ into attended word encodings $\boldsymbol{z}_{ij}$. To use them in the HGRU skeleton, we will encode them a second time using a BiGRU to produce attention-enhanced utterance vectors. For brevity, we will refer to these vectors as $\boldsymbol{v}_i$ for the utterance u_i. If word attention is used, these attended vectors will be treated as word encodings.

To complete this discussion, we need to instantiate the two functions. We use two commonly used attention mechanisms: BiDAF **?** and gated matchLSTM **?**. For simplicity, we replace the sequence encoder in the latter with a BiGRU and refer to it as GMGRU. Table 5.2 shows the corresponding definitions of f_c and f_m. We refer the reader to the original papers for further details. In subsequent sections, we will refer to the two attended versions of the HGRU as BIDAF^H and GMGRU^H.

5.3.3 Utterance-level Attention

While we assume that the history of utterances is available for both our tasks, not every utterance is relevant to decide a MISC label. For categorization, the relevance of

Method	f_m	f_c
BiDAF	$\boldsymbol{v}_{nk}\boldsymbol{v}_{ij}^T$	$[\boldsymbol{v}_{ij};\ \boldsymbol{a}_{ij};$ $\boldsymbol{v}_{ij} \odot \boldsymbol{a}_{ij};\ \boldsymbol{v}_{ij} \odot \boldsymbol{a}']$
GMGRU	$\boldsymbol{w}^e \tanh(\boldsymbol{W}^k\boldsymbol{v}_{nk}$ $+ \boldsymbol{W}^q[\boldsymbol{v}_{ij}; \boldsymbol{h}_{j-1}])$	$[\boldsymbol{v}_{ij}; \boldsymbol{a}_{ij}]$

Table 5.2: Summary of word attention mechanisms. We simplify BiDAF with multiplicative attention between word pairs for f_m, while GMGRU uses additive attention influenced by the GRU hidden state. The vector $\boldsymbol{w}_e \in \mathbb{R}^d$, and matrices $\boldsymbol{W}^k \in \mathbb{R}^{d\times d}$ and $\boldsymbol{W}^q \in \mathbb{R}^{2d\times 2d}$ are parameters of the BiGRU. The vector $\boldsymbol{h}_{j-1}$ is the hidden state from the BiGRU in GMGRU at previous position $j-1$. For combination function, BiDAF concatenates bidirectional attention information from both the key-aware query vector $\boldsymbol{a}_{ij}$ and a similarly defined query-aware key vector $\boldsymbol{a}'$. GMGRU uses simple concatenation for f_c.

an utterance to the anchor may be important. For example, a complex reflection (REC) may depend on the relationship of the current therapist utterance to one or more of the previous client utterances. For forecasting, since we do not have an utterance to label, several previous utterances may be relevant. For example, in the conversation in Table 5.1, both u_2 and u_4 may be used to forecast a complex reflection.

To model such utterance-level attention, we will employ the multi-head, multi-hop attention mechanism used in Transformer networks **?**. As before, due to space constraints, we refer the reader to the original work for details. We will use the $(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$ notation from the original paper here. These matrices represent a query, key and value respectively. The multi-head attention is defined as:

$$\text{Multihead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = [\text{head}_1; \cdots ; \text{head}_h]\boldsymbol{W}^O \tag{5.4}$$

$$\text{head}_i = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{W}_i^Q \left(\boldsymbol{K}\boldsymbol{W}_i^K\right)^T}{\sqrt{d_k}}\right)\boldsymbol{V}\boldsymbol{W}_i^V$$

The $\boldsymbol{W}_i$'s refer to projection matrices for the three inputs, and the final $\boldsymbol{W}^o$ projects the concatenated heads into a single vector.

The choices of the query, key and value defines the attention mechanism. In our work, we compare two variants: *anchor-based attention*, and *self-attention*. The anchor-based attention is defined by $Q = [v_n]$ and $K = V = [v_1 \cdots v_n]$. Self-attention is defined by setting all three matrices to $[v_1 \cdots v_n]$. For both settings, we use four heads and stacking them for two hops, and refer to them as SELF_{42} and ANCHOR_{42}.

Skeleton	Annotating	Forecasting
CON	$\boldsymbol{v}_n^{seg}$,$\boldsymbol{v}^{wordatt_n}$, $\boldsymbol{v}_n$	$\boldsymbol{C}_n$
HGRU	$\boldsymbol{H}_n$,$\boldsymbol{v}_n^{selfatt}$,$\boldsymbol{v}_n$	$\boldsymbol{H}_n$

Table 5.3: Input options for annotating and forecasting tasks based on CON and HGRU skeletons.

5.3.4 Predicting and Training

Predicting. From up to the bottom, every component will produce some of useful representation for inferences in our tasks. The dialogue encoding vector $\boldsymbol{H}_n$, as the final of the unidirectional GRU, it is also the contextual utterance encoding $\boldsymbol{h}_{u_n}$ of u_n. Hence H_n can be directly used as a representaion of u_n for classification in annotating tasks, also can be used as a represention of whole dialogue for forecasting task. Hence in HGRU setting, we always use $\boldsymbol{H}_n$ as the base option as input for inference.

However, for CON skeleton, the final state C_n doest not exactly represent the segement u_n in the whole concatnated dalogue. Hence, we concatenate the hidden state of the start position (0) and end position (T) of u_n into $\boldsymbol{v}_n^{seg} = [h_{u_n^0}; h_{u_n^T}]$, which is contextual utterance encoding in CON mode.

Beside the above $\boldsymbol{H}_n$ and $\boldsymbol{C}_n$ contextual utterance encoding in dialogue level, our components also produced the original utterance encoding $\boldsymbol{v}_n$ from utterance encoder. Whats'more, in **CON** mode, we can use history-aware utterance encoding $\boldsymbol{v}_n^{wordatt}$ While in HGRU, it produced a self-attentive utterance encoding. We denote it as $\boldsymbol{v}_n^{selfatt}$.

We summarize the option input encodings for inference in Table **??**. There are two ways to scoring withthese inputs, one is to score the concatenated those vectors together, denoted as $\text{concat}(A, B) = \text{MLP}([A; B)$; The other one is scoring each of them first and then add the scores up as the final scores, such as $\text{add}(A, B) = \text{MLP}(A) + \text{MLP}(B)$.

5.3.5 Addressing Label Imbalance

From Table **??**, we see that both client and therapist labels are imbalanced. Moreover, rarer labels are more important in both tasks. For example, it is important to identify CT and ST utterances. For therapists, it is crucial to flag MI non-adherent (MIN) utterances; seasoned therapists are trained to avoid them because they correlate negatively with pa-

tient improvements. If not explicitly addressed, the frequent but less useful labels can dominate predictions.

To address this, we extend the focal loss (FL **?**) to the multiclass case. For a label l with probability produced by a model p_t, the loss is defined as

$$\mathrm{FL}(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t) \tag{5.5}$$

In addition to using a label-specific balance weight α_t, the loss also includes a modulating factor $(1-p_t)^{\gamma}$ to dynamically downweight well-classified examples with $p_t \gg 0.5$. Here, the α_t's and the γ are hyperparameters. We use FL as the default loss function for all our models.

5.4 Experiments

The original psychotherapy sessions were collected for both clinical trials and Motivational Interviewing dissemination studies including hospital settings **?**, outpatient clinics **?**, college alcohol interventions **????**. All sessions were annotated with the Motivational Interviewing Skills Codes (MISC) **?**. We use the train/test split of **??** to give 243 training MI sessions and 110 testing sessions. We used 24 training sessions for development. As mentioned in §**??**, all our experiments are based on the MISC codes grouped by **?**.

5.4.1 Preprocessing and Model Setup

An MI session contains about 500 utterances on average. We use a sliding window of size $N = 8$ utterances with padding for the initial ones. We assume that we always know the identity of the speaker for all utterances. Based on this, we split the sliding windows into a client and therapist windows to train separate models. We tokenized and lower-cased utterances using spaCy **?**. To embed words, we concatenated 300-dimensional Glove embeddings **?** with ELMo vectors **?**. The appendix details the model setup and hyperparameter choices.

5.4.2 Results

5.4.3 Best Models

Our goal is to discover the best client and therapist models for the two tasks. We identified the following best configurations using F_1 score on the development set:

1. **Categorization**: For client, the best model does not need any word or utterance attention. For the therapist, it uses GMGRU^H for word attention and ANCHOR$_{42}$ for utterance attention. We refer to these models as $\mathcal{C}_C$ and $\mathcal{C}_T$ respectively
2. **Forecasting**: For both client and therapist, the best model uses no word attention, and uses SELF$_{42}$ utterance attention. We refer to these models as $\mathcal{F}_C$ and $\mathcal{F}_T$ respectively.

Here, we show the performance of these models against various baselines. The appendix gives label-wise precision, recall and F_1 scores.

Results on Categorization. Tables 5.4 and 5.5 show the performance of the $\mathcal{C}_C$ and $\mathcal{C}_T$ models and the baselines. For both therapist and client categorization, we compare the best models against the same set of baselines. The majority baseline illustrates the severity of the label imbalance problem. **?**, BiGRU$_{\text{generic}}$, **?** and **?** are the previous published baselines. The best results of previous published baselines are underlined. The last row Δ in each table lists the changes of our best model from them. BiGRU$_{\text{ELMo}}$, CONCAT^C, GMGRU^H and BiDAF^H are new baselines we define below.

Method	macro	FN	CT	ST
Majority	30.6	**91.7**	0.0	0.0
?	50.0	87.9	32.8	29.3
BiGRU$_{\text{generic}}$	50.2	87.0	35.2	28.4
BiGRU$_{\text{ELMo}}$	52.9	87.6	**39.2**	32.0
?	44.0	91.0	20.0	21.0
?	48.3	89.0	29.0	27.0
CONCAT^C	51.8	86.5	38.8	30.2
GMGRU^H	52.6	89.5	37.1	31.1
BiDAF^H	50.4	87.6	36.5	27.1
$\mathcal{C}_C$	**53.9**	89.6	39.1	**33.1**
$\Delta = \mathcal{C}_C - \underline{\text{score}}$	+3.5	-2.1	+3.9	+3.8

Table 5.4: Main results on categorizing client codes, in terms of macro F_1, and F_1 for each client code. Our model $\mathcal{C}_C$ uses final dialogue vector H_n and current utterance vector v_n as input of MLP for final prediction. We found that predicting using $\text{MLP}(H_n) + \text{MLP}(v_n)$ performs better than just $\text{MLP}(H_n)$.

The first set of baselines (above the line) do not encode dialogue history and use only the current utterance encoded with a BiGRU. The work of **?** falls in this category, and uses a 100-dimensional domain-specific embedding with weighted cross-entropy loss. Previ-

Method	macro	FA	RES	REC	GI	QUC	QUO	MIA	MIN
Majority	5.87	47.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
?	59.3	94.7	50.2	48.3	71.9	68.7	80.1	54.0	6.5
$\text{BiGRU}_{\text{generic}}$	60.2	94.5	50.5	49.3	72.0	70.7	80.1	54.0	10.8
$\text{BiGRU}_{\text{ELMo}}$	62.6	94.5	51.6	49.4	70.7	72.1	80.8	57.2	24.2
?	-	94.0	49.0	45.0	74.0	72.0	81.0	-	-
?	-	94.0	48.0	39.0	69.0	68.0	77.0	-	-
CONCAT^C	61.0	94.5	54.6	34.3	73.3	73.6	81.4	54.6	22.0
GMGRU^H	64.9	94.9	**56.0**	54.4	**75.5**	**75.7**	**83.0**	**58.2**	21.8
BiDAF^H	63.8	94.7	55.9	49.7	75.4	73.8	80.7	56.2	24.0
$\mathcal{C}_T$	**65.4**	**95.0**	55.7	**54.9**	74.2	74.8	82.6	56.6	**29.7**
$\Delta = \mathcal{C}_T - \underline{\text{score}}$	+5.2	+0.3	+3.9	+3.8	+0.2	+2.8	+1.6	+2.6	+18.9

Table 5.5: Main results on categorizing therapist codes, in terms of macro F_1, and F_1 for each therapist code. Models are the same as Table 5.4, but tuned for therapist codes. For the two grouped MISC set MIA and MIN, their results are not reported in the original work due to different setting.

ously, it was the best model in this class. We also re-implemented this model to use either ELMo or Glove vectors with focal loss.[3]

The second set of baselines (below the line) are models that use dialogue context. Both **?** and **?** use well-studied linguistic features and then tagging the current utterance with both past and future utterance with CRF and MEMM, respectively. To study the usefulness of the hierarchical encoder, we implemented a model that uses a bidirectional GRU over a long sequence of flattened utterance. We refer to this as CONCAT^C. This model is representative of the work of **?**, but was reimplemented to take advantage of ELMo.

For categorizing client codes, $\text{BiGRU}_{\text{ELMo}}$ is a simple but robust baseline model. It outperforms the previous best no-context model by more than 2 points on macro F_1. Using the dialogue history, the more sophisticated model $\mathcal{C}_C$ further gets 1 point improvement. Especially important is its improvement on the infrequent, yet crucial labels CT and ST. It shows a drop in the F_1 on the FN label, which is essentially considered to be an unimportant, background class from the point of view of assessing patient progress. For therapist codes, as the highlighted numbers in Table 5.5 show, only incorporating GMGRU-based word-level attention, GMGRU^H has already outperformed many baselines, our proposed model $\mathcal{F}_T$ which uses both GMGRU-based word-level attention and anchor-based multi-

[3]Other related work in no context exists (e.g., **??**), but they either do not outperform **?** or use different data.

head multihop sentence-level attention can further achieve the best overall performance. Also, note that our models outperform approaches that take advantage of future utterances.

For both client and therapist codes, concatenating dialogue history with CONCAT^C always performs worse than the hierarchical method and even the simpler BiGRU$_{\text{ELMo}}$.

Method	Dev		Test			
	CT	ST	macro	FN	CT	ST
CONCAT^F	20.4	30.2	43.6	84.4	23.0	**23.5**
HGRU	19.9	31.2	**44.4**	85.7	**24.9**	22.5
GMGRU^H	19.4	30.5	44.3	87.1	23.3	22.4
$\mathcal{F}_C$	**21.1**	**31.3**	44.3	85.2	24.7	22.7

Table 5.6: Main results on forecasting client codes, in terms of F_1 for ST, CT on dev set, and macro F_1, and F_1 for each client code on the test set.

Method	Recall	F_1								
	R@3	macro	FA	RES	REC	GI	QUC	QUO	MIA	MIN
CONCAT^F	72.5	23.5	63.5	0.6	0.0	53.7	27.0	15.0	18.2	9.0
HGRU	76.0	28.6	71.4	12.7	**24.9**	58.3	28.8	5.9	**17.4**	9.7
GMGRU^H	76.6	26.6	**72.6**	10.2	20.6	58.8	27.4	6.0	8.9	7.9
$\mathcal{F}_T$	**77.0**	**31.1**	71.9	**19.5**	24.7	**59.2**	**29.1**	**16.4**	15.2	**12.8**

Table 5.7: Main results on forecasting therapist codes, in terms of Recall@3, macro F_1, and F_1 for each label on test set

Results on Forecasting. Since the forecasting task is new, there are no published baselines to compare against. Our baseline systems essentially differ in their representation of dialogue history. The model CONCAT^F uses the same architecture as the model CONCAT^C from the categorizing task. We also show comparisons to the simple HGRU model and the GMGRU^H model that uses a gated matchGRU for word attention.[4]

Tables **??** (a,b) show our forecasting results for client and therapist respectively. For client codes, we also report the CT and ST performance on the development set because

[4]The forecasting task bears similarity to the next utterance selection task in dialogue state tracking work **?**. In preliminary experiments, we found that the Dual-Encoder approach used for that task consistently underperformed the other baselines described here.

of their importance. For the therapist codes, we also report the recall@3 to show the performance of a suggestion system that displayed three labels instead of one. The results show that even without an utterance, the dialogue history conveys signal about the next MISC label. Indeed, the performance for some labels is even better than some categorization baseline systems. Surprisingly, word attention (GMGRU^H) in Table **??** did not help in forecasting setting, and a model with the SELF$_{42}$ utterance attention is sufficient. For the therapist labels, if we always predicted the three most frequent labels (FA, GI, and RES), the recall@3 is only 67.7, suggesting that our models are informative if used in this suggestion-mode.

5.5 Analysis and Ablations

This section reports error analysis and an ablation study of our models on the development set. The appendix shows a comparison of pretrained domain-specific ELMo/glove with generic ones and the impact of the focal loss compared to simple or weighted cross-entropy.

5.5.1 Label Confusion and Error Breakdown

Figure 5.1 shows the confusion matrix for the client categorization task. The confusion between FN and CT/ST is largely caused by label imbalance. There are 414 CT examples that are predicted as ST and 391 examples vice versa. To further understand their confusion, we selected 100 of each for manual analysis. We found four broad categories of confusion, shown in Table 5.8.

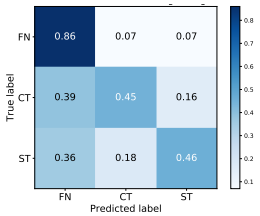


Figure 5.1: Confusion matrix for categorizing client codes, normalized by row.

The first category requires more complex reasoning than just surface form matching. For example, the phrase *seven out of ten* indicates that the client is very confident about changing behavior; the phrase *wind down after work* indicates, in this context, that the

Category and Explaination	Client Examples (Gold MISC)
Reasoning is required to understand whether a client wants to change behavior, even with full context (50,42)	T: On a scale of zero to ten how confident are you that you can implement this change ? C: I don't know, seven maybe (CT); I have to wind down after work (ST)
Concise utterances which are easy for humans to understand, but missing information such as coreference, zero pronouns (22,31)	I mean I could try it (CT) Not a negative consequence for me (ST) I want to get every single second and minute out of it(CT)
Extremely short ($\leq$ 5) or long sentence ($\geq$ 40), caused by incorrect turn segementation. (21,23)	It is a good thing (ST) Painful (CT)
Ambivalent speech, very hard to understand even for human. (7,4)	What if it does n't work I mean what if I can't do it (ST) But I can stop whenever I want(ST)

Table 5.8: Categorization of CT/ST confusions.The two numbers in the brackets are the count of errors for predicting CT as ST and vice versa. We exampled 100 examples for each case.

client drinks or smokes after work. We also found that the another frequent source of error is incomplete information. In a face-to-face therapy session, people may use concise and effient verbal communication, with guestures and other body language conveying information without explaining details about, for example, coreference. With only textual context, it is difficult to infer the missing information. The third category of errors is introduced when speech is transcribed into text. The last category is about ambivalent speech. Discovering the real attitude towards behavior change behind such utterances could be difficult, even for an expert therapist.

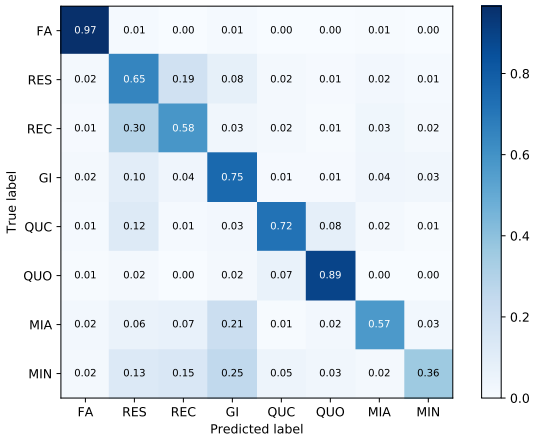


Figure 5.2: Confusion matrix for categorizing therapist codes, normalized by row.

Figures 5.1 and 5.2 show the label confusion matrices for the best categorization mod-

els. We will examine confusions that are not caused purely by a label being frequent. We observe a common confusion between the two reflection labels, REC and RES. Compared to the confusion matrix from **?**, we see that our models show much-decreased confusion here. There are two reason for this confusion persisting. First, the reflections may require a much longer information horizon. We found that by increasing the window size to 16, the overall reflection results improved. Second, we need to capture richer meaning beyond surface word overlap for RES. We found that complex reflections usually add meaning or emphasis to previous client statements using devices such as analogies, metaphors, or similes rather than simply restating them.

Closed questions (QUC) and simple reflections (RES) are known to be a confusing set of labels. For example, an utterance like *Sounds like you're suffering?* may be both. Giving information (GI) is easily confused with many labels because they relate to providing information to clients, but with different attitudes. The MI adherent (MIA) and non-adherent (MIN) labels may also provide information, but with supportive or critical attitude that may be difficult to disentangle, given the limited number of examples.

5.5.2 How Context and Attention Help?

We evaluated various ablations of our best models to see how changing various design choices changes performance. We focused on the context window size and impact of different word level and sentence level attention mechanisms. Tables 5.9 and 5.10 summarize our results.

History Size. Increasing the history window size generally helps. The biggest improvements are for categorizing therapist codes (Table 5.10), especially for the RES and REC. However, increasing the window size beyond 8 does not help to categorize client codes (Table 5.9) or forecasting (in appendix).

Word-level Attention. Only the model $\mathcal{C}_T$ uses word-level attention. As shown in Table 5.10, when we remove the word-level attention from it, the overall performance drops by 3.4 points, while performances of RES and REC drop by 3.3 and 5 points respectively. Changing the attention to BiDAF decreases performance by about 2 points (still higher than the model without attention).

Ablation	Options	macro	FN	CT	ST
history window size	0	51.6	87.6	39.2	32.0
	4	52.6	88.5	37.8	31.5
	8*	53.9	89.6	39.1	33.1
	16	52.0	89.6	39.1	33.1
word attention	+ GMGRU	52.6	89.5	37.1	31.1
	+ BiDAF	50.4	87.6	36.5	27.1
sentence attention	+ SELF_{42}	53.9	89.2	39.1	33.2
	+ ANCHOR_{42}	53.0	88.2	38.9	32.0

Table 5.9: Ablation study on categorizing client code. $*$ is our best model $\mathcal{C}_C$. All ablation is based on it. The symbol $+$ means adding a component to it. The default window size is 8 for our ablation models in the word attention and sentence attention parts.

Ablation	Options	macro	RES	REC	MIN
history window size	0	62.6	51.6	49.4	24.2
	4	64.4	54.3	53.2	23.7
	8*	65.4	55.7	54.9	29.7
	16	**65.6**	55.4	**56.7**	26.7
word attention	- GMGRU	62.0	51.9	51.7	16.0
	\ BiDAF	63.5	54.2	51.3	22.6
sentence attention	- ANCHOR_{42}	64.9	56.0	54.4	21.8
	\ SELF_{42}	63.4	55.5	48.2	21.1

Table 5.10: Ablation study on categorizing therapist codes, $*$ is our proposed model $\mathcal{C}_T$. \ means substituting and $-$ means removing that component. Here, we only report the important REC, RES labels for guiding, and the MIN label for warning a therapist.

Sentence-level Attention. Removing sentence attention from the best models that have it decreases performance for the models $\mathcal{C}_T$ and $\mathcal{F}_T$ (in appendix). It makes little impact on the $\mathcal{F}_C$, however. Table 5.9 shows that neither attention helps categorizing clients codes.

5.5.3 Can We Suggest Empathetic Responses?

Our forecasting models are trained on regular MI sessions, according to the label distribution on Table **??**, there are both MI adherent or non-adherent data. Hence, our models are trained to show how the therapist usually respond to a given statement.

To show whether our model can mimic *good* MI policies, we selected 35 MI sessions from our test set which were rated 5 or higher on a 7-point scale empathy or spirit. On these sessions, we still achieve a recall@3 of 76.9, suggesting that we can learn good MI

policies by training on all therapy sessions. These results suggest that our models can help train new therapists who may be uncertain about how to respond to a client.

5.6 Conclusion

We addressed the question of providing real-time assistance to therapists and proposed the tasks of categorizing and forecasting MISC labels for an ongoing therapy session. By developing a modular family of neural networks for these tasks, we show that our models outperform several baselines by a large margin. Extensive analysis shows that our model can decrease the label confusion compared to previous work, especially for reflections and rare labels, but also highlights directions for future work.

CHAPTER 6

REPRESENTING INTENT/SLOT CONCEPT WITH NATURAL LANGUAGE DESCRIPTION

6.1 Introduction

From early frame-driven dialog system GUS **?** to virtual assistants (Alexa, Siri, and Google Assistant *et al.*), frame-based dialog state tracking has long been studied to meet various challenges. In particular, how to support an ever-increasing number of services and APIs spanning multiple domains has been a focal point in recent years, evidenced by multi-domain dialog modeling **???** and transferable dialog state tracking to unseen intent/slots **???**.

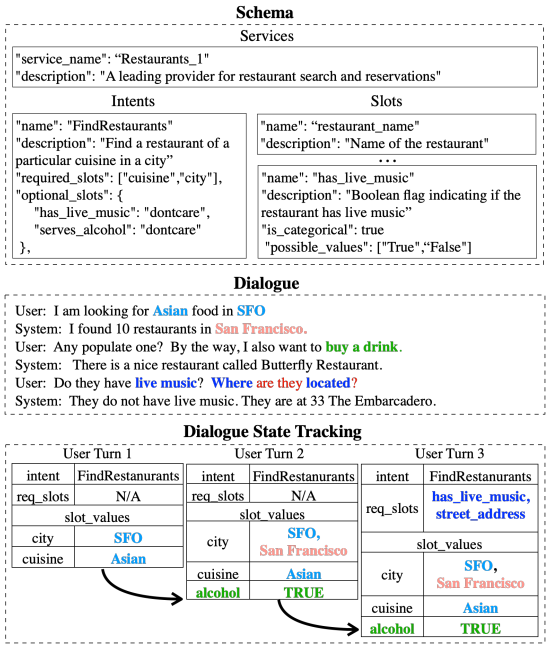


Figure 6.1: An example dialog from Restaurant_1 service, along with its service/intent/slot descriptions and dialog state representation.

Recently, **?** proposed a new paradigm called schema-guided dialog for transferable

dialog state tracking by using natural language description to define a dynamic set of service schemata. As shown in Figure 6.1, the primary motivation is that these descriptions can offer effective knowledge sharing across different services, e.g., connecting semantically similar concepts across heterogeneous APIs, thus allowing a unified model to handle unseen services and APIs. With the publicly available schema-guided dialog dataset (SG-DST henceforward) as a testbed, they organized a state tracking shared task composed of four subtasks: intent classfication (INTENT), requested slot identification (REQ), categorical slot labeling (CAT), and noncategorical slot labeling (NONCAT) **?**. Many participants achieved promising performance by exploiting the schema description for dialog modeling, especially on unseen services.

Despite the novel approach and promising results, current schema-guided dialog state tracking task only evaluates on a single dataset with limited variation in schema definition. It is unknown how this paradigm generalizes to other datasets and other different styles of descriptions. In this paper, we focus our investigation on the study of three aspects in schema-guided dialog state tracking: (1) schema encoding model architectures (2) supplementary training on intermediate tasks (3) various styles for schema description. To make a more general discussion on the schema-guided dialog state tracking, we perform extensive empirical studies on both SG-DST and MULTIWOZ 2.2 datasets. In summary, our contributions include:

- A comparative study on schema encoding architectures, suggesting a partial-attention encoder for good balance between inference speed and accuracy.

- An experimental study of supplementary training on schema-guided dialog state tracking, via intermediate tasks including natural language inference and question answering.

- An in-depth analysis of different schema description styles on a new suite of benchmarking datasets with variations in schema description for both SG-DST and MULTIWOZ 2.2.

6.2 Schema-Guided Dialog State Tracking

A classic dialog state tracker predicts a dialog state frame at each user turn given the dialog history and predefined domain ontology. As shown in Figure 6.1, the key difference between schema-guided dialog state tracking and the classic paradigm is the newly added natural language descriptions. In this section, we first introduce the four subtasks and schema components in schema-guided dialog state tracking, then we outline the research questions in our paper.

Subtasks. As shown in Figure 6.1, the dialog state for each service consists of 3 parts: *active intent*, *requested slots*, *user goals (slot values)*. Without loss of generality, for both SG-DST and MULTIWOZ 2.2 datasets, we divide their slots into categorical and non-categorical slots by following previous study on dual-strategies **?**. Thus to fill the dialog state frame for each user turn, we solve four subtasks: intent classification (INTENT), requested slot identification (REQ), categorical slot labeling (CAT), and non-categorical slot labeling (NONCAT). All subtasks require matching the current dialog history with candidate schema descriptions for multiple times.

Schema Components. Figure 6.1 shows three main schema components: service, intent, slot. For each intent, the schema also describes *optional* or *required* slots for it. For each slot, there are flags indicating whether it is categorical or not. *Categorical* means there is a set of predefined candidate values (Boolean, numeric or text). For instance, *has_live_music* in Figure 6.1 is a categorical slot with Boolean values. *Non-categorical*, on the other hand, means the slot values are filled from the string spans in the dialog history.

New Questions. These added schema descriptions pose the following three new questions. We discuss each of them in the following sections.

Q1. How should dialogue and schema be encoded? §**??**
Q2. How do different supplementary trainings impact each subtask? §**??**
Q3. How do different description styles impact the state tracking performance? §6.6

6.3 Datasets

To the best of our knowledge, at the time of our study, SG-DST and MULTIWOZ 2.2 are the only two publicly available corpus for schema-guided dialog study. We choose

Datasets	Splits	Dialog	Domains	Services	Zero-shot Domains	Zero-shot Services	Function Overlapp	Collecting Method
SG-DST	Train	16142	16	26	-	-	Across-domain	M2M
	Dev	2482	16	17	1	8		
	Test	4201	18	21	3	11	Within-domain	
MULTIWOZ 2.2	Train	9617	3	3	-	-		H2H
	Dev	2455	5	5	2	2	Across-domain	
	Test	2969	8	8	5	5		

Table 6.1: Summary of characteristics of SG-DST MULTIWOZ 2.2 datasets, in domain diversity, function overlap, data collecting methods

both of them for our study. In this section, we first introduce these two representative datasets, then we discuss the generalizibility in domain diversity, function overlapping, data collecting methods.

Schema-Guided Dialog Dataset. SG-DST dataset [1] is especially designed as a test-bed for schema-guided dialog, which contains well-designed heterogeneous APIs with overlapping functionalities between services **?**. In DSTC8 **?**, SG-DST was introduced as the standard benchmark dataset for schema-guided dialog research. SG-DST covers 20 domains, 88 intents, 365 slots.[2] However, previous research are mainly conducted based on this single dataset and the provided single description style. In this paper, we further extended this dataset with other benchmarking description styles as shown in §6.6, and then we perform both homogenous and hetergenous evalution on it.

Remixed MultiWOZ 2.2 Dataset. To eliminate potential bias from the above single SG-DST dataset, we further add MULTIWOZ 2.2 **?** to our study. Among various extended versions for MultiWOZ dataset (2.0-2.3, **????**) , besides rectifying the annotation errors, MULTIWOZ 2.2 also introduced the schema-guided annotations, which covers 8 domains, 19 intents, 36 slots. To evaluate performance on seen/unseen services with MultiWOZ, we remix the MULTIWOZ 2.2 dataset to include as seen services dialogs related to *restaurant*, *attraction* and *train* during training, and eliminate slots from other domains/services from training split. For dev, we add two new domains *hotel* and *taxi* as unseen services. For test, we add all remaining domains as unseen, including those that have minimum overlap

[1] `https://github.com/google-research-datasets/dstc8-schema-guided-dialogue`

[2] Please refer to the original paper for more details.

with seen services, such as *hospital*, *police*, *bus*. The statistics of data splits are shown in Appendix **??**. Note that this data split is different from the previous work on zero-shot MultiWOZ DST which takes a leave-one-out approach in **?**. By remixing the data in the way described above, we can evaluate the zero-shot performance on MultiWOZ in a way largely compatible with SG-DST.

Discussion. First, the two datasets cover diverse domains. MULTIWOZ 2.2 covers various possible dialogue scenarios ranging from requesting basic information about attractions through booking a hotel room or travelling between cities. While SG-DST covers more domains, such as 'Payments', 'Calender', 'DoctorServices' and so on.

Second, they include different levels of overlapping functionalities. SG-DST allows frequent function overlapping between multiple services, within the same domain (e.g. BookOneWayTicket v.s. BookRoundTripTicket), or across different domains (BusTicket v.s. TrainTicket). However, the overlapping in MULTIWOZ 2.2 only exists across different domains, e.g., 'destination', 'leaveat' slots for Taxi and Bus services, 'pricerange', 'bookday' for Restaurant and Hotel services.

Third, they are collected by two different approaches which are commonly used in dialog collecting. SG-DST is firstly collected by machine-to-machine self-play (M2M, **?**) with dialog flows as seeds, then paraphrased by crowd-workers. While MULTIWOZ 2.2 are human-to-human dialogs (H2H, **?**), which are collected with the Wizard-of-Oz approach.

We summarize the above discussion in Table 6.1. We believe that results derived from these two representative datasets can guide future research in schema guided dialog.

6.4 Dialog & Schema Representation and Inference (Q1)

In this section, we focus on the model architecture for matching dialog history with schema descriptions using pretrained BERT **?** [3]. To support four subtasks, we first extend *Dual-Encoder* and *Cross-Encoder* to support both sentence-level matching and token-level prediction. Then we propose an additional *Fusion-Encoder* strategy to get faster inference without sacrificing much accuracy. We summarize different architectures in Figure 6.2.

[3]We use BERT-base-cased for all main experiments. Other pretrained language models can be easily adapted to our study

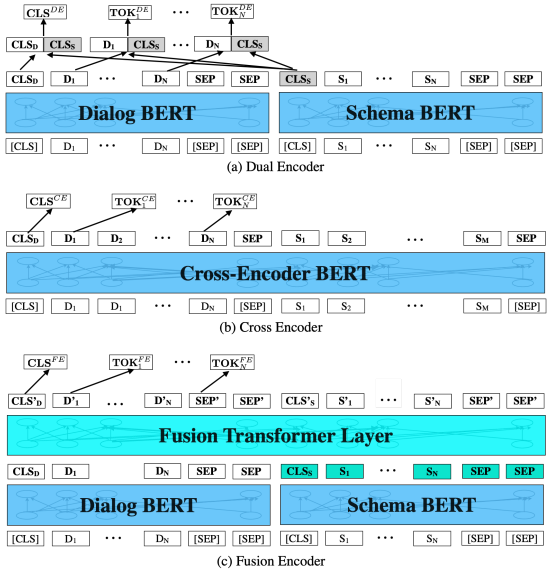


Figure 6.2: Dual-Encoder, Cross-Encoder and Fusion Encoder, shaded block will be cached during training

Then we show the classification head and results for each subtask.

6.4.1 Encoder Architectures

Dual-Encoder. It consists of two separate BERTs to encode dialog history and schema description respectively, as Figure 6.2 (a). We follow the setting in the official baseline provided by DSTC8 Track4 **?**. We first use a fixed BERT to encode the schema description once and cached the encoded schema $\mathbf{CLS}_S$. Then for sentence-level representation, we concatenate dialog history representation $\mathbf{CLS}_D$ and candidate schema representation $\mathbf{CLS}_S$ as the whole sentence-level representation for the pair, denoted as $\mathbf{CLS}^{DE}$. For token-level representation, we concatenate the candidate schema $\mathbf{CLS}_S$ with each token embedding in the dialog history, denoted as $\mathbf{TOK}^{DE}$.[4] Because the candidate schema embeddings are encoded independently from the dialog context, they can be pre-computed once and cached for fast inference.

Cross-Encoder. Another popular architecture as Figure 6.2 (b) is *Cross-Encoder*, which concatenates the dialog and schema as a single input, and encodes jointly with a single self-attentive encoder spanning over the two segments. When using BERT to encode

[4] A schema-aware dialog token embedding can also be computed by attention or other method for span-based detection tasks **??**

the concatenated sentence pair, it performs full (cross) self-attention in every transformer layers, thus offer rich interaction between the dialog and schema. BERT naturally produces a summarized representation with [CLS] embedding $\mathbf{CLS}^{CE}$ and each schema-attended dialog token embeddings $\mathbf{TOK}^{CE}$. Since the dialog and schema encoding always depend on each other, it requires recomputing dialog and schema encoding for multiple times, thus much slower in inference.

Fusion-Encoder. In Figure 6.2 (c), similar to *Dual-Encoder*, *Fusion-Encoder* also encodes the schema independently with a fixed BERT and finetuning another BERT for dialog encoding. However, instead of caching a single [CLS] vector for schema representation, it caches **all token representation** for the schema including the [CLS] token. What's more, to integrate the sequences dialog token representation with schema token representation, an extra stack of transformer layers are added on top to allow token-level fusion via self-attention, similar to *Cross-Encoder*. The top transformer layers will produce embeddings for each token $\mathbf{TOK}^{FE}$ including a schema-attended $\mathbf{CLS}^{FE}$ of the input [CLS] from the dialog history. With cached schema token-level representations, it can efficiently produce schema-aware sentence- and token-level representation for each dialog-schema pairs.

6.4.2 Model Overview

All the above 3 encoders will produce both sentence- and token-level representations for a given sentence pair. In this section, we abstract them as two representations **CLS** and **TOK**, and present the universal classification heads to make decisions for each subtask.

Active Intent. To decide the intent for current dialog turn, we match current dialog history D with each intent descriptions $I_0 ... I_k$. For each dialog-intent pair (D, I_k), we project the final sentence-level **CLS** representation to a single number $P_{I_k}^{active}$ with a linear layer follows a sigmoid function. We predict *"NONE"* if the $P_{I_k}^{active}$ of all intents are less than a threshold 0.5, which means no intent is active. Otherwise, we predict the intent with largest $P_{I_k}^{active}$. We predict the intent for each turn independently without considering the prediction on previous turns.

Requested Slot. As in Figure 6.1, mulitple requested slots can exist in a single turn. We use the same strategy as in active intent prediction to predict a number P_{req}^{active}. However,

to support the multiple requested slots prediction. We predict all the requested slots with $P_{req}^{active} > 0.5$.

Categorical Slot. Categorical slots have a set of candidate values. We cannot predict unseen values via n-way classification. Instead, we do binary classification on each candidate value. Besides, rather than directly matching with values, we also need to check that whether the corresponding slot has been activated. For *Cross-Encoder* and *Fusion-Encoder*, we use typical two-stage state tracking to incrementally build the state: **Step** 1. Using **CLS** to predict the slot status as *none*, *dontcare* or *active*. When the status is *active*, we use the predicted slot value; Otherwise, it will be assigned to *dontcare* meaning no user preference for this slot, or *none* meaning no value update for the slot in current turn; **Step** 2. If Step 1 is *active*, we match the dialog history with each value and select the most related value by ranking. We train on cross entropy loss. Two-stage strategy is efficient for *Dual-Encoder* and *Fusion-Encoder*, where cached schema can be reused, and get efficiently ranked globally in a single batch. However, it is not scalable for *Cross-Encoder*, especially for large number of candidate values in MultiWOZ dataset. Hence, during training, we only use a binary cross-entropy for each single value and postpone the ranking only to the inference time.

Noncategorical Slot. The slot status prediction for noncategorical slot use the same two-stage strategy. Besides that, we use the token representation of dialog history **TOK** to compute two softmax scores f_{start}^{i} and f_{end}^{i} for each token i, to represent the score of predicting the token as start and end position respectively. Finally, we find the valid span with maximum sum of the start and end scores.

6.4.3 Experiments on Encoder Comparison

To fairly compare all three models, we follow the same schema input setting as in Table 6.2. We trained separate models for SG-DST and the remixed MultiWOZ datasets for all the experiments in our papers[5]. Because there are very few intent and requested slots in MULTIWOZ 2.2 dataset, we ignore the intent and requested slots tasks for MULTIWOZ 2.2 in our paper.

Results. As shown in Table 6.3, *Cross-Encoder* performs the best over all subtasks. Our

[5]Appendix **??** shows the detailed experiment setup

Intent	service description, intent description
Req	service description, slot description
Cat	slot description, cat value
NonCat	service description, slot description

Table 6.2: Schema description input used for different tasks to compare *Dual-Encoder*, *Cross-Encoder*, and *Fusion-Encoder*. In the appendix **??**, we also studies other compositions of description input. We found that service description will not help for INTENT, REQ and CAT tasks, while the impact on NONCAT task also varies from SG-DST and MULTIWOZ 2.2 dataset.

	SG-DST					MULTIWOZ 2.2		
Method/Task	Acc	F1	Joint Acc			Joint Acc		
	Intent	Req	Cat	NonCat	All	Cat	NonCat	All
Seen Services								
Dual-Encoder	94.51	99.62	87.92	47.77	43.20	79.20	79.34	65.64
Fusion-Encoder	94.90	**99.69**	88.94	48.78	58.52	81.37	80.58	67.43
Cross-Encoder	**95.55**	99.59	**93.68**	**91.85**	**87.58**	**85.99**	**81.02**	**71.93**
Unseen Services								
Dual-Encoder	89.73	95.20	42.44	31.62	19.51	56.92	50.82	31.83
Fusion-Encoder	90.47	95.95	48.79	35.91	22.85	57.01	52.23	33.64
Cross-Encoder	**93.84**	**98.26**	**71.55**	**74.13**	**54.54**	**59.85**	**59.62**	**38.46**

Table 6.3: Test set results on SG-DST and MULTIWOZ 2.2. The *Dual-Encoder* model is a re-implementation of official DSTC8 baseline from **?**. Other models are trained with the architecture described in our paper.

Fusion-Encoder with partial attention outperforms the *Dual-Encoder* by a large margin, epsecially on categorical and noncategorical slots predictions. Additionally, on seen services, we found that *Dual-Encoder* and *Fusion-Encoder* can perform as good as *Cross-Encoder* on INTENT and REQ tasks. However, they cannot generalize well on unseen services as *Cross-Encoder*.

Inference Speed. To test the inference speed, we conduct all the experiments with a maximum affordable batch size to fully exploit 2 V100 GPUs (with 16GB GPU RAM each). During training, we log the inference time of each evaluation on dev set. Both *Dual-Encoder* and *Fusion-Encoder* can do joint inference across 4 subtasks to obtain an integral dialog state for a dialog turn example. *Dual-Encoder* achieves the highest inference speed of **603.35** examples per GPU second, because the encoding for dialog and schema are fully separated. A dialog only needed to be encoded for once during the inference of a dialog state example while the schema are precomputed once. However, for *Cross-Encoder*, to predict a dialog state for a single turn, it need to encode more than 300 sentence pairs

	SG-DST												MULTIWOZ 2.2					
	intent			req			cat			noncat			cat			noncat		
	all	seen	unseen	all	seen	unseen	all	seen	unseen	all	seen	unseen	all	seen	unseen	all	seen	unseen
Δ_{SNLI}	**+0.51**	+0.02	**+0.68**	-0.19	+0.38	-0.38	-1.63	-2.87	-1.23	-4.7	-0.1	-6.25	+2.05	+0.6	–0.7	**+3.64**	+1.05	**+4.84**
Δ_{SQuAD}	-1.81	-0.17	-1.32	-0.25	-0.01	-0.33	-2.87	-3.02	-5.17	**+1.99**	-1.79	**+3.25**	+0.04	-0.71	+0.41	**+1.93**	-2.21	**+4.27**

Table 6.4: Relative performance improvement of different supplementary training on SG-DST and MULTIWOZ 2.2 dataset

	SG-DST							
	Intent		Req		Cat		NonCat	
	seen	unseen	seen	unseen	seen	unseen	seen	unseen
Δ_{SNLI}	+0.02	**+0.68**	+0.38	-0.38	-2.87	-1.23	-0.1	-6.25
Δ_{SQuAD}	-0.17	-1.32	-0.01	-0.33	-3.02	-5.17	-1.79	**+3.25**

Table 6.5: Relative performance improvement of different supplementary training on SG-DST and MULTIWOZ 2.2 dataset

in a batch, thus only processes **4.75** examples per GPU second. *Fusion-Encoder* performs one time encoding on dialog history, but it needs to jointly encode the same amount of dialog-schema pair ws *Cross-Encoder*, instead, however, with a two-layer transformer encoder. Overall it achieves **10.54** examples per GPU second, which is **2.2x** faster than *Cross-Encoder*. With regarding to the accuracy in Table 6.3, *Fusion-Encoder* performs much better than *Dual-Encoder*, especially on unseen services.

6.5 Supplementary Training (Q2)

Besides the pretrain-fintune framework used in §**??**, **?** propose to add a supplementary training phase on an intermediate task after the pretraining, but before finetuning on target task. It shows significant improvement on the target tasks. Moreover, large amount pretrained and finetuned transformer-based models are publicly accessible, and well-organized in model hubs for sharing, training and testing[6]. Given the new task of schema-guided dialog state tracking, in this section, we study our four subtasks with different intermediate tasks for supplementary training.

[6]e.g., Huggingface(`https://huggingface.co/models`) and ParlAL(`https://parl.ai/docs/zoo.html`), etc.

6.5.1 Intermediate Tasks

As described in § 6.4.2, all our 4 subtasks take a pair of dialog and schema description as input, and predict with the summerized sentence-pair **CLS** representation. While NON-CAT also requires span-based detection such as question answering. Hence, they share the similar problem structure with the following sentence-pair encoding tasks.

Natural Language Inference. Given a hypothesis/premise sentence pair, natural language inference is a task to determine whether a hypothesis is entailed, contradicted or neutral given that premise. **Question Answering.** Given a passage/question pairs, the task is to extract the span-based answer in the passage.

Hence, when finetuning BERT on our subtaks, instead of directly using the originally pretrained BERT, we use the BERT finetuned on the above two tasks for further finetuning. Due to better peformance of *Cross-Encoder* in §**??**, we directly use the finetuned *Cross-Encoder* version of BERT models on SNLI and SQuAD2.0 dataset from Huggingface model hub. We add extra speaker tokens [user:] and [system:] into the vocabulary for encoding the multi-turn dialog histories.

6.5.2 Results on Supplementary Training

Table 6.4 shows the performances gain when finetuning 4 subtasks based on models with the above SNLI and SQuAD2.0 supplementary training.

We mainly find that SNLI helps on INTENT task, SQuAD2 mainly helps on NON-CAT task, while neither of them helps much on CAT task. Recently, **?** also found that when modeling dialog understanding as question answering task, it can benefit from a supplementary training on SQuAD2 dataset, especially on few-shot scenarios, which is a similar findings as our NONCAT task. Result difference on REQ task is minor, because it is a relatively easy task, adding any supplementary training did n't help much. Moreover, for CAT task, the sequence 2 of the input pair is the slot description with a categorical slot value, thus the meaning overlapping between the full dialog history and the slot/value is much smaller than SNLI tasks. On the other side, **CLS** token in SQuAD BERT is finetuned for null predictions via start and end token classifers, which is different from the the single CLS classifer in CAT task.

6.6 Impact of Description Styles (Q3)

Previous work on schema-guided dialog **?** are only based on the provided descriptions in SG-DST dataset. Recent work on modeling dialog state tracking as reading comprehension **?** only formulate the descriptions as simple question format with existing intent/slot names, it is unknown how it performs when compared to other description styles. Moreover, they only conduct homogeneous evaluation where training and test data share the same description style. In this section, We also investigate how a model trained on one description style will perform on other different styles, especially in a scenario where chat-bot developers may design their own descriptions. We first introduce different styles of descriptions in our study, and then we train models on each description style and evaluate on tests with corresponding homogeneous and heterogeneous styles of descriptions. Given the best performance of *Cross-Encoder* shown in the previous section and its popularity in DSTC8 challenges, we adopt it as our model architecture in this section.

6.6.1 Benchmarking Styles

For each intent/slot, we describe their functionalities by the following different descriptions styles:

IDENTIFER. This is the least informative case of name-based description: we only use meaningless intent/slot identifiers, e.g. Intent_1, Slot_2. It means we don't use description from any schema component. We want to investigate how a simple identifier-based description performs in schema-guided dialog modeling, and the performance lower-bound on transferring to unseen services.

NAMEONLY. Using the original intent/slot names in SG-DST and MULTIWOZ 2.2 dataset as descriptions, to show whether name is enough for schema-guided dialog modeling.

Q-NAME. This is corresponding to previous work by **?**. For each intent/slot, it generate a question to inquiry about the intent and slot value of the dialog. For each slot, it simply follows the template '*What is the value for* **slot i***?*'. Besides that, our work also extend the intent description by following the template "*Is the user intending to* **intent j** ".

ORIG. The original descriptions in SG-DST and MULTIWOZ 2.2 dataset.

Q-ORIG. Different from the Q-NAME, firstly it is based on the original descriptions; secondly, rather than always use the "what is" template to inquiry the intent/slot value, We add "what", "which", "how many" or "when" depending on the entity type required for the slot. Same as Q-NAME, we just add prefixes as "Is the user intending to..." in front of the original description. In a sum, this description is just adding question format to original description. The motivation of this description is to see whether the question format is helpful or not for schema-guided dialog modeling.

To test the model robustness, we also create two paraphrased versions NAME-PARA and ORIG-PARA for NAMEONLY and ORIG respectively. We first use nematus **?** to automatically paraphrase the description with back translation, from English to Chinese and then translate back, then we manually check the paraphrase to retain the main meaning. Appendix **??** shows examples for different styles of schema descriptions.

6.6.2 Results on Description Styles

Unlike the composition used in Table 6.2, we don't use the service description to avoid its impact. For each style, we train separate models on 4 subtasks, then we evaluate them on different target styles. First, Table 6.6 summarizes the performance for homogeneous evaluation, while Table 6.7 shows how the question style description can benefit from SQuAD2 finetuning. Then we also conduct heterogeneous evaluation on the other styles[7] as shown in Table 6.8.

Style\Task	SG-DST				MULTIWOZ 2.2	
	Intent	Req	Cat	NonCat	Cat	NonCat
IDENTIFER	61.16	91.48	62.47	30.19	34.25	52.28
NAMEONLY	**94.24**	98.84	74.01	75.63	53.72	56.18
Q-NAME	93.31	**98.86**	74.36	74.86	**54.19**	56.17
ORIG	93.01	98.55	74.51	75.76	52.19	57.20
Q-ORIG	93.42	98.51	**76.64**	**76.60**	53.61	**57.80**

Table 6.6: Homogeneous evaluation results of different description style on SG-DST dataset and MULTIWOZ 2.2 datasets. The middle horizontal line separate the two name-based descriptions and two rich descriptions in our settings. All numbers in the table are mixed performance including both seen and unseen services.

[7] We don't consider the meaningless IDENTIFER style due to its bad performance

6.6.2.1 Homogeneous Evaluation

Is name-based description enough? As shown in Table 6.6, IDENTIFER is the worst case of using name description, its extremely bad performance indicates name-based description can be very unstable. However, we found that simple meaningful name-based description actually can perform the best in INTENT and REQ task, and they perform worse on CAT and NONCAT tasks comparing to the bottom two *rich* descriptions. [8] After careful analysis on the intents in SG-DST datasets, we found that most services only contains two kinds of intents, an information retrieval intent with a name prefix "Find-", "Get-", "Search-"; another transaction intent like "Add-", "Reserve-" or "Buy-". Interestingly, we found that all the intent names in the original schema-guided dataset strictly follows an action-object template with a composition of words without abbreviation, such as "FindEvents", "BuyEventTickets". This simple name template is good enough to describe the core functionality of an intent in SG-DST dataset. [9] Additionally, REQ is a relaitively simper task, requesting information are related to specifial attributes, such as "has_live_music", "has_wifi", where keywords co-occured in the slot name and in the user utterance, hence rich explanation cannot help further. On the other side, *rich* descriptions are more necessary for CAT and NONCAT task. Because in many cases, slot names are too simple to represent the functionalities behind it, for example, slot name "passengers" cannot fully represent the meaning "number of passengers in the ticket booking".

Does question format help? As shown in Table 6.6, when comparing row Q-ORIG v.s. ORIG, we found extra question format can improve the performance on CAT and NONCAT task on both SG-DST and MULTIWOZ 2.2 datasets, but not for INTENT and REQ tasks. We believe that question format helps the model to focus more on specific entities in the dialog history. However, when adding a simple question pattern to NAMEONLY, comparing row Q-NAME and NAMEONLY, there is no consistent improvement on both of the two datasets. Further more, we are curious about whether BERT finetuned on SQuAD2 (SQuAD2-BERT)

---

[8] Only exception happens in CAT on MULTIWOZ 2.2. When creating MULTIWOZ 2.2 **?**, the slots with less than 50 different slot values are classified as categorical slots, which leads to inconsistencies. We put detailed discuss about MULTIWOZ 2.2 in the supplementary material

[9] This action-object template has also been found efficient for open domain intent induction task(e.g., **?**, OPINE).

Style/Dataset	SG-DST			MULTIWOZ 2.2		
	all	seen	unseen	all	seen	unseen
ORIG	+1.99	-1.79	**+3.25**	+1.93	-2.21	**+4.27**
Q-ORIG	+6.13	-2.01	**+8.84**	+1.06	-1.28	**+3.06**
NAMEONLY	-0.45	-1.49	-0.11	+1.75	+0.58	**+1.77**
Q-NAME	+0.05	-2.98	**+1.04**	-0.04	-0.32	**+1.25**

Table 6.7: Performance changes when using BERT finetuned on SQuAD2 dataset to further finetuning on our NONCAT task.

can further help on the question format. Because NONCAT are similar with span-based question answering, we focus on NONCAT here. Table 6.7 shows that, after applying the supplementary training on SQuAD2 (§**??**), almost all models get improved on unseen splits however slightly dropped on seen services. Moreover, comparing to Q-NAME, Q-ORIG is more similar to the natural questions in the SQuAD2, we obverse that Q-ORIG gains more than Q-NAME from pretrained model on SQuAD2.

6.6.2.2 Heterogeneous

In this subsection, we first simulate a scenario when there is no recommended description style for the future unseen services. Hence, unseen services can follow any description style in our case. We average the evaluation performance on three other descriptions and summarized in Table 6.8. The Δ column shows the performance change compared to the homogeneous performance. It is not surprising that almost all models perform worse on heterogeneous styles than on homogeneous styles due to different distribution between training and evaluation. The bold number shows the best average performance on heterogeneous evaluation for each subtask. The trends are similar with the analysis in homogeneous evaluation 6.6.2.1, the name-based descriptions perform better than other rich descriptions on intent classification tasks. While on other tasks, the ORIG description performs more robust, especially on NONCAT task.

Furthermore, we consider another scenario where fixed description convention such as NAMEONLY and ORIG are suggested to developers, they must obey the basic style convention but still can freely use their own words, such as abbreviation, synonyms, adding extra modifiers. We train each model on NAMEONLY and ORIG, then evaluate on the corresponding paraphrased version respectively. In the last two rows of Table

6.8, the column 'para' shows performance on paraphrased schema, while Δ shows the performance change compared to the homogeneous evaluation. ORIG still performs more robust than NAMEONLY when schema descriptions get paraphrased on unseen services.

Style\Task	SG-DST							
	Intent(Acc)		Req(F1)		Cat(Joint Acc)		NonCat(Joint Acc)	
	mean	Δ	mean	Δ	mean	Δ	mean	Δ
NAMEONLY	82.47	-11.47	96.92	-1.64	61.37	-5.54	56.53	-14.68
Q-NAME	**93.27**	+0.58	**97.88**	-0.76	68.55	+2.63	62.92	-6.30
ORIG	79.47	-12.70	97.42	-0.74	**68.58**	-0.3	**66.72**	-3.11
Q-ORIG	84.57	-8.24	96.70	-1.45	68.40	-2.89	56.17	-15.00
	para	Δ	para	Δ	para	Δ	para	Δ
NAMEONLY	**92.22**	-1.74	97.69	-0.87	67.39	-0.7	67.17	-4.04
ORIG	91.54	**-0.63**	**98.42**	**+0.26**	**71.74**	**+2.86**	**67.68**	**-2.16**

Table 6.8: Results on unseen service with heterogeneous description styles on SG-DST dataset. More results and qualitative analysis are in the appendix **??**

6.7 Related Work

Our work is related to three lines of research: multi-sentence encoding, multi-domain and transferable dialog state tracking. However, our focus is on the comparative study of different encoder architectures, supplementary training, and schema description style variation. Thus we adopt existing strategies from multi-domain dialog state tracking.

Multi-Sentence Encoder Strategies. Similar to the recent study on encoders for response selection and article search tasks **?**, we also conduct our comparative study on the two typical architectures *Cross-Encoder* **??** and *Dual-Encoder* **??**. However, they only focus on sentence-level matching tasks. All subtasks in our case require sentence-level matching between dialog context and each schema, while the non-categorical slot filling task also needs to produce a sequence of token-level representation for span detection. Hence, we study multi-sentence encoding for both sentence-level and token-level tasks. Moreover, to share the schema encoding across subtasks and turns, we also introduce a simple *Fusion-Encoder* by caching schema token embeddings in §6.4.1, which improves efficiency without sacrificing much accuracy.

Multi-domain Dialog State Tracking. Recent research on multi-domain dialog system have been largely driven by the release of large-scale multi-domain dialog datasets, such as MultiWOZ **?**, M2M **?**, accompanied by studies on key issues such as in/cross-domain

carry-over **?**. In this paper, our goal is to understanding the design choice for schema descriptions in dialog state tracking. Thus we simply follow the in-domain cross-over strategies used in **TRADE ?**. Additionally, explicit cross-domain carryover **?** is difficult to generalize to new services and unknown carryover links. We use longer dialog history to inform the model on the dialog in the previous service. This simplified strategy does impact our model performance negatively in comparison to a well-designed dialog state tracking model on seen domains. However, it helps reduce the complexity of matching extra slot descriptions for cross-service carryover. We leave the further discussion for future work.

Transferable Dialog State Tracking. Another line of research focuses on how to build a transferable dialog system that is easily scalable to newly added intents and slots. This covers diverse topics including e.g., resolving lexical/morphological variabilities by symbolic de-lexicalization-based methods **??**, neural belief tracking **?**, generative dialog state tracking **??**, modeling DST as a question answering task **????**. Our work is similar with the last class. However, we further investigate whether the DST can benefit from NLP tasks other than question answering. Furthermore, without rich description for the service/intent/slot in the schema, previous works mainly focus on simple format on question answering scenarios, such as domain-slot-type compounded names (e.g., "*restaurant-food*"), or simple question template "*What is the value for* **slot i***?*". We incorporate different description styles into a comparative discussion on §6.6.1.

6.8 Conclusion

In this paper, we studied three questions on schema-guided dialog state tracking: encoder architectures, impact of supplementary training, and effective schema description styles. The main findings are as follows:

By caching the token embedding instead of the single CLS embedding, a simple partial-attention *Fusion-Encoder* can achieve much better performance than *Dual-Encoder*, while still infers two times faster than *Cross-Encoder*. We quantified the gain via supplementary training on two intermediate tasks. By carefully choosing representative description styles according to recent works, we are the first of doing both homogeneous/heterogeneous evaluations for different description style in schema-guided dialog. The results show that

simple name-based description performs well on INTENT and REQ tasks, while NON-CAT tasks benefits from richer styles of descriptions. All tasks suffer from inconsistencies in description style between training and test, though to varying degrees.

Our study are mainly conducted on two datasets: SG-DST and MULTIWOZ 2.2, while the speed-accuracy balance of encoder architectures and the findings in supplementary training are expected to be dataset-agnostic, because they depend more on the nature of the subtasks than the datasets. Based on our proposed benchmarking descriptions suite, the homogeneous and heterogeneous evaluation has shed the light on the robustness of cross-style schema-guided dialog modeling, we believe our study will provide useful insights for future research.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

APPENDIX A

THE FIRST

This is an appendix. Notice that the LaTeX markup for an appendix is, surprisingly, `\chapter`. The `\appendix` command does not produce a heading; instead, it just changes the numbering style from numeric to alphabetic, and it changes the heading prefix from **CHAPTER** to **APPENDIX**.

Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah.

APPENDIX B

THE SECOND

This is an appendix.

Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah.

APPENDIX C

THE THIRD

This is an appendix.

REFERENCES

Baxter, Jonathan. 2000. A model of inductive bias learning. *Journal of artificial intelligence research*, **12**, 149–198.

Wolpert, David H, Macready, William G, *et al.* . 1995. *No free lunch theorems for search*. Tech. rept. Technical Report SFI-TR-95-02-010, Santa Fe Institute.

BINOMIAL NOMENCLATURE INDEX

TOPIC INDEX