LEARNING TO DETECT LANGUAGE MARKERS

By

Fengyi Tang

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science – Doctor of Philosophy

2021

ABSTRACT

LEARNING TO DETECT LANGUAGE MARKERS

By

Fengyi Tang

In the world of medical informatics, biomarkers play a pivotal role in determining the physical state of human beings, distinguishing the pathologic from the clinically normal. In recent years, behavioral markers, due to their availability and low cost, have attracted a lot of attention as a potential supplement to biomarkers. “Language markers” such as spoken words and lexical preference have been shown to be both cost-effective as well as predictive of complex diseases such as mild cognitive impairment (MCI).

However, language markers, although universal, do not possess many of the favorable properties that characterize traditional biomakers. For example, different people may exhibit similar use of language under certain conversational contexts (non-unique), and a person’s lexical preferences may change over time (non-stationary). As a result, it is unclear whether any set of language markers can be measured in a consistent manner. My thesis projects provide solutions to some of the limitations of language markers: (1) We formalize the problem of learning a dialog policy to measure language markers as an optimization problem which we call persona authentication. We provide a learning algorithm for finding such a dialog policy that can generalize to unseen personalities. (2) We apply our dialog policy framework on real-world data for MCI prediction and show that the proposed pipeline improves prediction against supervised learning baselines. (3) To address non-stationarity, we introduce an effective way to do temporally-dependent and non-i.i.d. feature selection through an adversarial learning framework which we call precision sensing. (4) Finally, on the prediction side, we propose a method for improving the sample efficiency of classifiers by retaining privileged information (auxiliary features available only at training time).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1

BACKGROUND

1.1 Biomakers and Their Limitations

In the world of medical informatics, biomarkers [1] play a pivotal role in determining the physical state of human beings, distinguishing the pathophysiologic from the clinically normal. In most chronic diseases such as congestive heart failure and Diabetes Mellitus, there exist biomarkers such as brain natriuritic peptide and HbA1c that sufficiently captures the presence and the stage of disease progression in almost all clinical populations [2, 3].

However, a key limitation of biomarkers is that they are very expensive to obtain. For example, the average cost to a basic metabolic panel in the inpatient setting ranges from \$300-\$10,000 dollars per patient per test [4]. Note that this is the cost of simply getting a baseline reading of metabolic enzymes; complete metabolic panels and blood tests can often multiply the cost. In general, the higher the granularity of the biomarker, the higher the cost. A computerized tomography (CT) scan can reveal numerous respiratory pathologies but can range between \$1,450 - 7,800 per person depending on availability. As a result, much of the *diagnostic decisions* in medicine reside on the physician's judgement regarding cost-vs.-effectiveness: which biomarkers should be obtained for which patients?

From the data mining perspective, the decision of "which biomarker to select for which patient" corresponds to a *feature selection* problem in which each data sample (patient state information) consists of only a sparse set of observed features due to cost constraints. However, this feature selection problem is often treated in an ad-hoc manner in the practice: a mixture of insurance policies, practitioner guidelines and human judgement dictate the manner in which we engage in feature selection in the clinics. Naturally, two questions deserve our attention:

(1) Are there cheaper (but effective) alternatives to biomarkers?

(2) Are there more effective ways to acquire clinical features?

1.2 Behavioral Markers as Alternatives

In recent years, behavioral markers have attracted a lot of attention as a potential supplement to biomarkers. For example, in dementia progression, there exists a disease state clinically defined as mild cognitive impairment (MCI) that precede noticeable memory loss and severe cognitive decline [5]. In traditional Alzheimer Disease (AD), brain imaging is done to confirm diagnosis, but intervention is futile at this stage because irreversible damage would have already occurred [6]. As a result, much research efforts have been channeled toward early detection of MCI, when irreversible changes have yet to occur. However, there is a tradeoff for using biomarkers in MCI detection: while MCI can be detected by classic biomarker approaches such as brain imaging and cerebral spinal fluid markers [7, 8, 9], the presence of such biomarkers correspond to the existence of irreversible changes having already occurred in the patient. Thus, other measures of cognitive functions may be needed for early detection of MCI.

Recently, there have been several works showing that behavioral markers [10, 11, 12] can provide useful diagnostic signals for early stage MCI. For example, gait monitoring has been shown to change in early MCI [13], enabling programs for at-home monitoring of behavioral changes [14]. However, in terms of cost of data acquisition, "language markers" such as spoken words [15] and linguistic features [11] have been shown to be both low-cost as well as predictive of early MCI states. Due to the ubiquity of language, the notion of developing high-performing language markers presents a promising direction for alternative diagnostic markers.

1.3 Limitations of Language Markers

However, language markers, although universal, do not possess many of the favorable properties that characterize biomakers. For example, there are several key properties allow a biomarker to be generalizable across populations [16]:

- Universality: the feature of interest can be measured in everyone in the disease group.

- Uniqueness: the feature of interest should differ between disease group and normal.
- Measurability: the feature can be measured consistently across everyone.
- Permanence: the feature remains invariant to time and algorithms used to measure it.

In the case of language markers, only universality is satisfied: one can potentially obtain linguistic and speech features from text and audio recordings for everyone, barring disabilities. However, language markers are certainly not unique: different people may exhibit similar use of language under certain contexts. Language markers may also be transient. For example, a person's word choices changes throughout one's lifetime. Depending on the environmental context (e.g., time of day, the presence of other people), one's choice of words and speech tone may differ greatly. As a result, it is unclear whether any set of linguistic features can be measured in a consistent manner.

In spite of these limitations, however, language markers do provide ample signal for each of these classic biometric categories. Even though speech is not unique, higher-level features *composed* of language preferences – e.g., personality traits, use of idioms, pausing patterns – are regularly used to distinguish one group of people from another. Beyond the individual identification, such traits are regularly used for market segmentation and sentiment reports. Although language, and perhaps even resulting higher-level features change over time, they often follow statistical patterns of change. For example, it is well-known that human infants follow very stable patterns of language acquisition: babbling starts at 6 months, first words start at 9 month to 1 year, vocabulary expansion (250-900 words) by age 2, and can recite numbers and addresses by age 5. It is perhaps reasonable to hypothesize that linguistic decline may follow predictable patterns in the elderly.

Finally, the issue of measurability provides a difficult challenge: how can we consistently measure a set of language-related features in a person when language use itself is highly context dependent? As a toy example, consider psychometric and cognitive tests, both of which rely heavily on questionnaires and surveys. These would not work well for surveying language markers, as certain features such as word choice, sentence structures and topics of interest are open observable in unstructured, open-domain conversations. However, a completely open-domain conversation setting presents another layer of problems. The performance of automatic extraction of speech markers depend on restricting the spoken words used to generate speech [17], as variations in lexicon and sentence structure can introduce noise in the extracted acoustic features. Thus, there seems to numerous points of potential conflict between the vast number of potential features that comprise language markers. Additionally, it is unclear whether what questions would elicit what set of linguistic or acoustic features. In order to tackle the problem of measurability, we have to formally define the relationship between language markers of interest and the question contexts used to elicit them.

1.4 Contributions

The goal of this thesis project is to provide solutions to the limitations of language markers. The main contributions are summarized as follows:

- We formalize the problem of learning a question policy to measure linguistic features as *persona authentication*. We provide an algorithm for finding such a question policy that can generalize to unseen personalities.

- We apply the question policy on real-world data for MCI prediction and show that the proposed pipeline improves prediction against supervised learning baselines.

- To address the challenges of permanence (i.e., temporally varying disease markers), we introduce an effective way to do temporally-dependent and non-i.i.d. feature selection in the clinical setting. Specifically, we propose "precision sensing", an adversarial learning framework that exploits the relationship between adversarial sample generation and classifier performance to produce feature selection strategies that consider cross-sample and temporal dependencies which make training examples non-i.i.d.

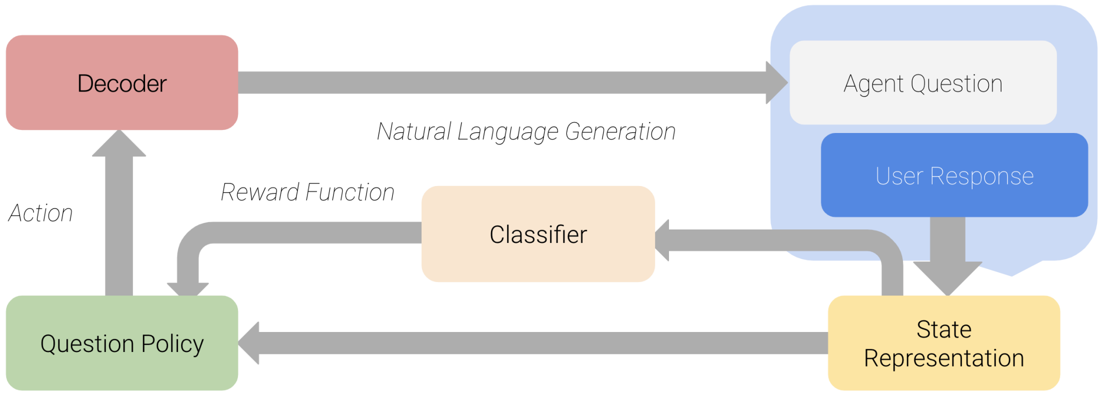
- Finally, on the prediction side, we propose a method for improving the sample efficiency of classifiers. Specifically, we propose an algorithm that leverages privileged information (linguistic features available only at training time) to supplement the lack of training labels.

1.5 Proposed Research

Figure 6.1 provides an overview of the main components of our proposed pipeline. As noted before, the goal is to learn a question policy to interact with users to obtain language markers. We will summarize the key components below and discuss them in separate sections:

- State Representation: the actual representation of linguistic markers as a *feature vector* for downstream tasks such as classification, feature selection, and question generation.
- Classifier: the prediction model trained by supervised learning. Labels of the prediction task are provided for each sample, e.g., MCI = 1 or normal = 0.
- Question Policy: a second model used to acquire new training samples from the user; in other words, its role is to do data acquisition. The output of the question policy can be discrete or continuous variables, depending on the decoder used for NLG.
- Decoder: the NLG unit used to decode policy actions into human understandable language. The decoder is an open-domain conversational model. The goal is to do conditional decoding based on both conversational history and policy actions.

Figure 1.1: Proposed learning to interview pipeline.



1.6 Question Policy

1.6.1 MCI Proof-of-Concept

Our first work introduces a preliminary pipeline for learning a question policy to ascertain linguistic markers for the purposes of MCI diagnosis. We use this as a proof-of-concept and illustrate that our "learning to interview" approach can indeed detect predictive signals that can be potentially more informative than unstructured conversations with human interviewers.

Figure 1.2: Preliminary dialog agent trained by deep Q-learning (DQN) [18] to interview patients for MCI screening.

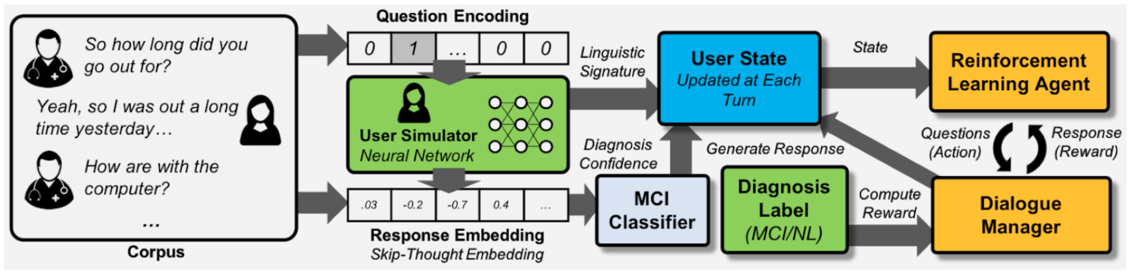


Figure 1.2 provides a summary figure for the first work. Here, the prediction task is binary classification of patient MCI status. The action space of the question policy is a discrete set of questions. We simplify the problem setting such that NLG issues such as dialog consistency, felicity of language generation, and multi-turn co-reference reasoning are not considered; the question policy simply outputs a deterministic question (out of a pool of 107 possible questions) and obtains a response generated by the skip-thought model [19] (user simulator in Figure 1.2). We use treat the skip-thought embedding as the state input and learn the question policy using straight-forward DQN over 107 actions. The reward function is hand-crafted: the agent receives a small negative reward per turn to penalize lengthy conversations and a large positive reward at the end of conversations if the skip-thought features of the dialog history results in correct prediction by the MCI classifier. We compare this proposed framework against classifier performance on supervised learning data from the original corpus. In addition to prediction accuracy, we also compare the length of conversations needed (conversational efficiency) to obtain a threshold level of performance.

1.6.2 Persona Authentication

In our second work, we formalize the problem of learning a question policy as *person authentication*. Specifically, we break down persona authentication into two parts: *persona identification*, which is inferring a set of persona features from a given dialog trajectory, and *persona verification*, the problem of finding a second conversational model – we call it a *question policy* – to elicit dialog trajectories for persona identification.

Problem 1. *Persona Verification. Given a space of persona information $\mathcal{P}$, persona verification is the optimization objective:*

$$\min_{\theta} \mathbb{E}_{P\sim\mathcal{P}}\big[\mathcal{L}(\tau_\theta, P)\big] \tag{1.1}$$

where $\mathcal{L}(\cdot,\cdot)$ is the authentication loss:

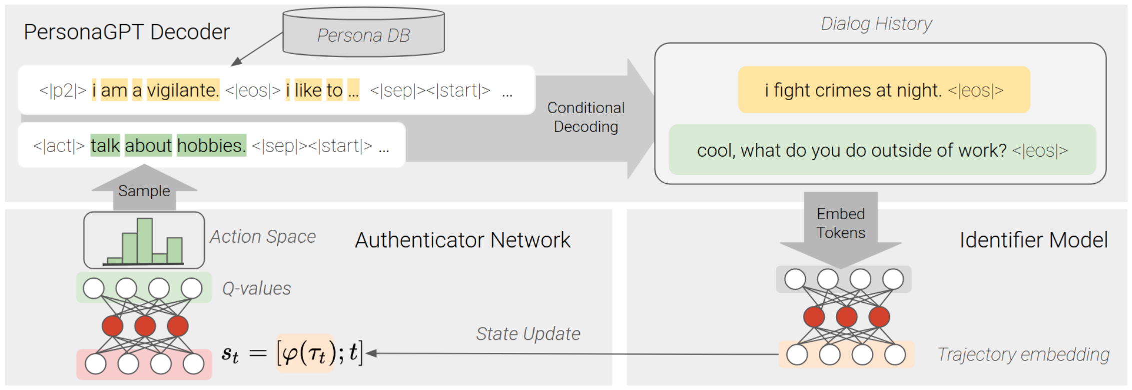
$$\mathcal{L}(\tau_\theta, P) = \max\{0, C + d(\tau_\theta, P^+) - d(\tau_\theta, P^-)\} - \log p(\tau_\theta)). \tag{1.2}$$

P^+ denotes persona facts that co-occur with trajectory τ, P^- the opposite. C specifies the desired margin of separation, and τ_θ is the dialog trajectory generated by the question policy (θ).

In the persona authentication chapter, we illustrate some promising theoretical guarantees of our proposed optimization problem. For example, we show that estimators trained under the authentication loss shown in Eqn. (6.5) maximizes the mutual information between the conversational trajectory τ generated by the policy and the persona information of the input agent (i.e., the user).

We then present a refined version of the original pipeline that is capable of handling open-domain conversations, all the while directing the conversation toward the end goal of classification. The overview figure is shown in Figure 1.3 which outlines the key components: verifier corresponds to the question policy of interest, identifier corresponds to the state representation, and PersonaGPT corresponds to the decoder. Details of conditional decoding, active learning of verifier action codes and policy learning details are revealed in the persona authentication chapter. Empirically, we illustrate that the learned policy outperforms human evaluators as well as unstructured conversation policies in identifying personality traits through conversation.

Figure 1.3: Persona Authentication pipeline.


PersonaGPT Decoder
Persona DB
Dialog History
<|p2|> i am a vigilante. <|eos|> i like to ... <|sep|><|start|> ...
<|act|> talk about hobbies. <|sep|><|start|> ...
Conditional Decoding
i fight crimes at night. <|eos|>
cool, what do you do outside of work? <|eos|>
Sample
Action Space
Authenticator Network
Embed Tokens
Identifier Model
Q-values
State Update
$s_t = [\varphi(\tau_t); t]$
Trajectory embedding

Our question policy addresses the problem of measurability of language markers. We cannot elicit the same set of linguistic features with the same set of questions, but we can obtain a question policy that maximizes the mutual information between generated dialog responses and the user's salient characteristics.

1.7 Classifier

1.7.1 Precision Sensing

On the classifier front, we present two works that address various technical issues in medical prediction tasks. First, language markers, like many other disease markers in medicine, suffer from the issue of permanence: some disease markers fluctuate through time and require repeated measurements to gain insight into temporal trends. While we cannot change the fact time-dependence of certain disease markers, we can deal with the permanence issue by learning to detect stable temporal patterns from data. Our third work introduces the *precision sensing* framework which addresses this issue by leveraging the interaction between adversarial examples [20] and classifier decision boundaries.

Definition 1. *(Precision Sensing) The problem of precision sensing seeks a sensor tensor A^* that minimizes the empirical risk (ERM) [21] according to:*

$$A^* \in \underset{A_i \in \mathcal{A}}{argmin}\, \mathbb{E}_{P(X,Y)}[L(h(A_i \odot X_i), y_i)] \approx \underset{A_i \in \mathcal{A}}{argmin}\, \frac{1}{m} \sum_{(X_i, y_i) \in \mathcal{D}} L(y_i, h(A_i \odot X_i)),$$

where h is a hypothesis class that maps $h : X \to Y$, and $L(.)$ is a risk function that evaluates the hypothesis mappings against the actual label.

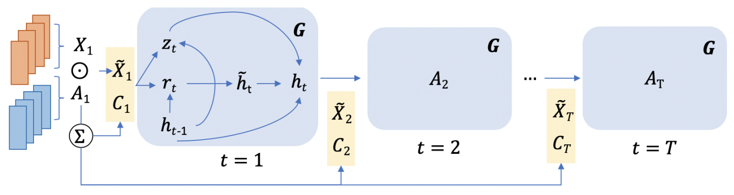
Here, $\mathcal{A}$ denotes the space of sensing matrices, and each matrix A_i can be thought of as a per-sample sensing matrix. From this perspective A_i describes the subset of sensed features across time for each sample. Unlike regular feature selection which chooses the same subset of features for sample, precision sensing builds the sensing tensor A incrementally: at each time step t, the sensing matrix $A^{(t)} \in \{0, 1\}^{m \times d}$ is computed across m samples for all d features. To learn A, we propose the following loss to simultaneous learn the hypothesis function the sensing tensor:

$$\begin{aligned} \min_{\theta_F} \max_{\theta_G} \ & \underbrace{\{F(X - X \odot G(X))_{\neg y} - F(X - X \odot G(X))_y\}}_{\text{Feature Sensing}}{}^{+} \\ & + \underbrace{CE_y(F(X \odot G(X)))}_{\text{Classifier Reconstruction}} + \underbrace{\beta ||X - X \odot G(X)||_1}_{\text{Budget Management}} . \end{aligned} \tag{1.3}$$

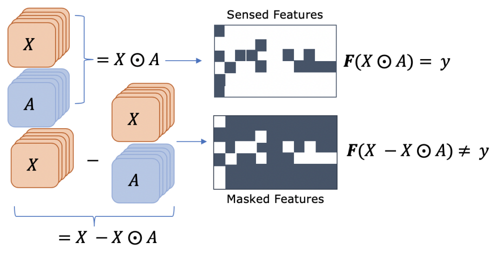
In Eqn. (1.3), F represents the classifier and G is a recurrent neural network (RNN) that outputs the elements of $A^{(t)}_{i,j} \in 0, 1$ for each sample i and each feature j at time t. We call this RNN the feature sensor that not only takes into account the temporal dependence of features within a sample but also the performance of F *across* samples. On the other hand, the classifier F learns to adapt its decision boundary based on the cross entropy loss CE calculated based on decision function mapping from the *sensed* features $X \odot G(X)$ rather than the full set of features X.

Since the performance of G is coupled to F, we provide a co-training scheme for both models in the Classifier chapter of the thesis. Figure 1.4a-1.4b gives an overview of the proposed co-training scheme. Fig. 1.4a describes incremental generation of the *sensing tensor* A by the G. Fig. 1.4b describes the minimax game between the F and G. F (top) tries to minimize the classification error with sensed features $X \odot A$ while G (bottom) tries to allocate A such that unsensed features lead to misclassification. We prove that iterative gradient descent on Eqn. (1.3) converges to local Nash Equilibrium points which are guaranteed to exist. In experiments, we show that precision sensing outperforms SOTA baselines from active sensing on a benchmark mortality prediction task.

Figure 1.4: Overview of co-training scheme between the feature sensor G and classifier F.



(a) Incremental generation of the *sensing tensor*.



(b) Minimax game between F and G.

1.7.2 Multitask LUPI

In addition to feature selection, the sample complexity of learning is especially important in medical prediction tasks. For example, in electronic health records (EHR), we often find the distribution of labels is highly skewed: there are numerous diseases with very few labels but an abundance of text descriptions from literature. We propose a method for improving the sample efficiency of these learning tasks by leveraging clinical linguistic markers as privileged information. Specifically, we consider *physician notes* in the form of discharge summaries, linking standard medical terminologies (i.e., UMLS codes [22]) with diagnostic findings in the EHR. *UMLS codes* are a set of standardized medical concepts used by clinicians to describe physical findings of diseases and are used widely in both the EHR as well as medical research [22]. [23] alluded at the idea that medical datasets also contain vast amounts of privileged information in the *physician notes*, which serve to explain the qualities of diseases that can greatly aid decision rules. For experiments, we consider the following set of data for example features, PI, and labels:

- Example Features X: continuous time-series data (i.e. lab values, blood tests, imaging) and discrete static variables (i.e. demographics information) that describe a patient.
- Privileged Information X^*: physician notes containing descriptions in natural language and medical terms (UMLS concepts [22]) that summarize a particular visit for a patient.
- Target Task Y: prediction tasks of interest, such as mortality (binary classification), disease prediction (multi-task and transfer learning), ... etc.

Formally, we consider a similar problem setting where we are given $\mathcal{D}_{train} = \{(x_i, x_i^*, y_i)\}_{i=1}^m$. We assume that there exists a set of *vocabulary* with size d that define the privileged information (PI) space. $x_{ij}^* = 1$ if the j^{th} term in the PI vocabulary is contained in the x_i^* sample, $x_{ij}^* = 0$ otherwise. We also denote $x_i^* = \{w_1, ..., w_k\}$ as a decomposition of x_i^* into k individual components such that each $w_j \in \{0, 1\}^d$ is a one-hot vector, corresponding to a non-zero component in x_i^*, and $x_i^* = w_1 + w_2 +, ..., +w_k$. In the following sections, we denote $\{w_j\}_{j=1}^d$ as the set of "words" that compose the PI vocabulary. Thus, x_i^* gives the *co-occurrence* label of each word $w_j \in \{w_j\}_{j=1}^d$ with respect to the sample x_i.

We make no assumptions on the example features (X) with regards to data type. In practice, the example features correspond to temporal features, each corresponding to a biomarker (e.g, lab test value) measured at various time intervals. Finally, we define the multi-task learning objective as learning $y_i = \{0, 1\}^C$ to be a set of C binary classification tasks.

Our approach to retaining privileged information in a multitask setting is to align PI features (linguistic descriptions) with temporal features (biomarkers).

1. Build a dictionary of PI features and learn a distributed representation [24] over the PI vocabulary.

2. Find a joint representation space (Φ) between the PI and example features.

3. Jointly learn the decision functions $h^* : \Phi \longrightarrow Y$ by feature-matching in the joint representation space.

The first process uses unsupervised learning to embed the PI vocabulary into a vector space. The second process allows for some of the privileged information to be *retained* at inference time, despite not having direct access to the PI vectors. The third process allows for PI information for one task to be transferred for other C-1 tasks in the label space. In the Classifier chapter, we will examine how to achieve (1) - (3) in detail. We also provide analysis of how (3) can maintain the favorable LUPI sample efficiency. We validate our multitask LUPI framework on multilabel classification tasks using the MIMIC-III database [25] and compare with several prevalent transfer learning schemes.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Kyle Strimbu and Jorge A Tavel. What are biomarkers? *Current Opinion in HIV and AIDS*, 5(6):463, 2010.

[2] VO Puntmann. How-to guide on biomarkers: biomarker definitions, validation and applications with examples from cardiovascular disease. *Postgraduate medical journal*, 85(1008):538–545, 2009.

[3] Jeffrey K Aronson and Robin E Ferner. Biomarkers—a general review. *Current protocols in pharmacology*, 76(1):9–23, 2017.

[4] Renee Y Hsia, Yaa Akosa Antwi, and Julia P Nath. Variation in charges for 10 common blood tests in california hospitals: a cross-sectional analysis. *BMJ open*, 4(8), 2014.

[5] Serge Gauthier, Barry Reisberg, Michael Zaudig, Ronald C Petersen, Karen Ritchie, Karl Broich, Sylvie Belleville, Henry Brodaty, David Bennett, Howard Chertkow, et al. Mild cognitive impairment. *The lancet*, 367(9518):1262–1270, 2006.

[6] Jeffrey L Cummings, Rachelle Doody, and Christopher Clark. Disease-modifying therapies for alzheimer disease: challenges to early intervention. *Neurology*, 69(16):1622–1634, 2007.

[7] Keith A Johnson, Nick C Fox, Reisa A Sperling, and William E Klunk. Brain imaging in alzheimer disease. *Cold Spring Harbor perspectives in medicine*, 2(4):a006213, 2012.

[8] D Heister, James B Brewer, Sebastian Magda, Kaj Blennow, Linda K McEvoy, Alzheimer's Disease Neuroimaging Initiative, et al. Predicting mci outcome with clinically available mri and csf biomarkers. *Neurology*, 77(17):1619–1628, 2011.

[9] Liang Zhan, Yashu Liu, Yalin Wang, Jiayu Zhou, Neda Jahanshad, Jieping Ye, and Paul Matthew Thompson. Boosting brain connectome classification accuracy in alzheimer's disease using higher-order singular value decomposition. *Frontiers in neuroscience*, 9:257, 2015.

[10] Clifford R Jack Jr, David S Knopman, William J Jagust, Leslie M Shaw, Paul S Aisen, Michael W Weiner, Ronald C Petersen, and John Q Trojanowski. Hypothetical model of dynamic biomarkers of the alzheimer's pathological cascade. *The Lancet Neurology*, 9(1):119–128, 2010.

[11] Kathleen C Fraser, Jed A Meltzer, and Frank Rudzicz. Linguistic features identify alzheimer's disease in narrative speech. *Journal of Alzheimer's Disease*, 49(2):407–422, 2016.

[12] Meysam Asgari, Jeffrey Kaye, and Hiroko Dodge. Predicting mild cognitive impairment from spontaneous spoken utterances. *Alzheimer's & Dementia: Translational Research & Clinical*

Interventions, 3(2):219–228, 2017.

[13] Juan Manuel Fernandez Montenegro and Vasileios Argyriou. Cognitive evaluation for the diagnosis of alzheimer's disease based on turing test and virtual environments. *Physiology & behavior*, 173:42–51, 2017.

[14] Hiroko H Dodge, Jian Zhu, Nora C Mattek, Molly Bowman, Oscar Ybarra, Katherine V Wild, David A Loewenstein, and Jeffrey A Kaye. Web-enabled conversational interactions as a method to improve cognitive functions: Results of a 6-week randomized controlled trial. *Alzheimer's & dementia: translational research & clinical interventions*, 1(1):1–12, 2015.

[15] Tuka Alhanai, Rhoda Au, and James Glass. Spoken language biomarkers for detecting cognitive impairment. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 409–416. IEEE, 2017.

[16] Anil K Jain, Ruud Bolle, and Sharath Pankanti. *Biometrics: personal identification in networked society*, volume 479. Springer Science & Business Media, 2006.

[17] Brian Roark, Margaret Mitchell, John-Paul Hosom, Kristy Hollingshead, and Jeffrey Kaye. Spoken language derived measures for detecting mild cognitive impairment. *IEEE transactions on audio, speech, and language processing*, 19(7):2081–2090, 2011.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[19] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[20] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[21] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[22] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270, 2004.

[23] Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: similarity control and knowledge transfer. *Journal of machine learning research*, 16(2023-2049):2, 2015.

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*,

2013.

[25] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3, 2016.

CHAPTER 2

REVIEW OF CURRENT RESEARCH

2.1 Language Markers in Mild Cognitive Impairment

Fraser et al. [1] introduced "linguistic markers" such as word count, use frequencies of certain parts-of-speech, and sentiments as predictors of dementia in AD patients. Beyond summary statistics, Asgari et al. [2] introduce word-level language markers which utilize expert provided linguistic features such as socioeconomic context, presence of biological process, psychologic factors. These linguistic dimensions are standardized as Liguistic Inquiry and Word Count (LIWC) [3] features for over 5,000 English words. Beyond expert annotations, statistical properties of words have been examined by a wide range of machine learning approaches such as topic modeling [4], neural word2vec [5] and contextualized word embeddings [6]. More recently, sentence-level representations such as Skip-Thought vectors [7] have shown capacity in representing phrase and utterance level linguistic features such as semantic similarity, phrase relationships and user intentions. Currently, the state-of-the-art (SOTA) representation of linguistic features reside in the recently introduced transformer language models (LMs) [8]. For example, BERT encoders [9] have long been shown to produce state-of-the-art results in language comprehension tasks – otherwise known as natural language understanding (NLU). On the natural language generation (NLG) end, the GPT-2 decoder [10] has been shown to generate text that is difficult to distinguish from human text.

2.2 Dialog Systems

Central to our characterization of question contexts and language markers is the idea of a *dialog system*: a conversational agent (a machine learning model) that generates text (prompts / questions) to user input based on feedback signals from the user (responses). There are two main types of dialog systems: task-oriented and open-domain. They can synonymous be thought of as structured (task-oriented) and unstructured (open-domain) conversation. However, our problem is somewhere in the middle: our desired question policy should be able to adapt to a variety of conversation types (open-domain), yet still service the goal of MCI diagnosis (task-oriented). We introduce the crux of task-oriented vs. open-domain conversation before formalizing the semi-structured setting mathematically in proceeding chapters.

2.3 Task-Oriented System

Task-oriented dialogue systems are typically designed for retrievaltasks in which users provide queries and the chat-bot provides appropriate responses based on an external knowledge base [11, 12, 13], or identifying correct answers by looking into vast amounts of documents [14, 15]. Such dialogue systems are typically designed to be a pipeline containing a set of components including a language understanding unit that parses the intention and semantics from the input from humans, a dialogue manager that manages dialogue state tracking and policy learning, and a language generation unit that generates response [11, 16, 17].

Similar to task-oriented dialog, our problem setting is goal-oriented: we want to find a dialog policy that can lead to the correct classification of diseases. However, our problem setting differs from task-oriented dialog in that (1) the question generation do not have fixed frames, and (2) the user responses are certainly not constrained to specific tasks (e.g., hotel booking, movie recommendations). In this regard, the actual NLG part of the task is an open-domain dialog problem.

2.4 Open-Domain Systems

Open-domain system refers to dyadic conversations without any constraints on the dialog structure between the conversational agents. Open-domain dialogue generation has been formulated in two ways: (1) as a statistical machine translation (SMT) [18, 19] task in which the goal is to output a sequence of tokens $\mathbf{y}_{1:K}^{(t)}$ in response to a sequence of tokens $\mathbf{x}_{1:K}^{(t)}$, and (2) as a ranking problem where a response Y_t is selected among a list of candidate responses that best "matches" the query X_t [11]. Both approaches optimize over the maximum likelihood (MLE) objective:

$$\underset{Y_t}{\text{maximize}} \;\; \log p(Y_t|X_t).$$

The main difference between the two approaches is that SMT is generative, because the response is generated token-by-token, in nature while the ranking approach is discriminative due to requiring a pool of candidates. But neither takes into account the history of the conversation, and as a result both fail to capture dialogue transitions.

More recently, generative models such as HRED, VHRED and CVAE [20, 21, 22] try to capture dialogue history to provide more contextualized decoding. These approaches apply MLE learning to the objective:

$$\underset{Y_t}{\text{maximize}} \ \ \log p(Y_t | X_t, \tau_{1:t-1}). \tag{2.1}$$

Although this new objective encourages more response diversity, the dialogue agent still does not have any mechanism of encouraging a consistent personality across multiple turns [23].

2.4.1 Personalized Response Generation

An emerging direction in open-domain dialogue systems is the rise of *persona models*, which focus on generating responses that preserve a consistent set of personalities over the course of a dialogue. For example, Li et al. [19] learns a *speaker vector* from which to condition the generation process. In the PersonaChat dataset [23], each dialogue speaker is equipped with text descriptions that characterize their persona. The standard objective of persona models can be expressed as:

$$\underset{Y_t}{\text{maximize}} \ \ \log p(Y_t | X_t, \tau_{1:t-1}, P_Y) \tag{2.2}$$

where P_Y is the set of persona descriptions for the dialogue agent. Zhang et al. [23] and the ConvAI2 challenges [24] provided numerous approaches to incorporate the persona information into the dialogue generation process. Our problem setting is similar to open-domain dialog in that the user (patient) is allowed to generate arbitrary responses. Thus, our dialog agent must be able to handle open-domain conversation and exhibit common-sense reasoning, multi-turn consistency and response diversity – challenges that are featured in open-domain dialog.

2.4.2 MCI Prediction via Utterance Data

[2] used a classical supervised learning framework to formulate MCI prediction as binary classification problem. For each interview, a corpus was constructed using only the participant responses to interviewer questions. For each participant, the response corpus over several interviews was preprocessed into feature vectors using the Linguistic Inquiry & Word Count (LIWC) dictionary [3]. The LIWC dictionary transforms each word in a given corpus to a 69-dimensional feature vector with latent dimensions representing grammatical and semantic properties of each word. A final 69-dimensional feature vector is then constructed at the end of the corpus by aggregation of all previous LIWC vectors. The resulting feature representation is a $m \times 69$ matrix. The best performing classifier in this benchmark study uses linear support vector machines (SVM) with ℓ_1-norm regularization [2]. The resulting performance is 72.5% AUC over 5-fold validation.

2.4.3 Healthcare Applications of Dialog Systems

Dialogue systems have been widely adopted in the healthcare domain for various applications. For example, chat-bots are available to assist the patient intake process [25], retrieve restaurant accommodation information for young adults with food allergies [26], and perform dialogue analysis and generation conversation to perform mental health treatment [27]. In the context of Alzheimer's disease research, [28] designed a virtual reality based chat-bot to evaluate memory loss using predefined questions and answers. [29] discussed applications of chat-bots as caregiviers for Alzheimer's patients, providing safety, personal assistance, entertainment, and stimulation. More recently, [30] introduced a computer avatar to ask a list of pre-defined questions from neuropsychological tests to diagnose dementia. This work is closely related to our system as it utilizes dialogue to glean disease-related information. However, one major issue in this approach is that the questions were obtained from mini-mental state examination (MMSE) [31], which is a confirmatory measure used to *define* clinical dementia (such as MCI) rather than a diagnostic tool to predict it. It is more clinical meaningful to identify diagnostic markers associated with the pathological pathways, such as lexical distribution associated with the cognitive changes for the purpose of diagnostic screening.

2.5 Beyond Linguistic Features

2.5.1 Acoustic Features

Beyond linguistic features, acoustic features such as mel-frequency cepstrum coefficients (MFCC) have been shown to effectively predict MCI status [32, 33], especially when combined with linguistic features. However, these studies are thus far limited to responses to structured conversations (e.g., responses to standardized cognitive tests and fixed questionnaires). It remains an open question whether acoustic markers can be used in semi-structured conversations and combining both linguistic and acoustic markers can improve the detection of MCI. Most predictive studies using acoustic markers rely on either fixed prompts [33] or pronunciation tasks [32] to control for the differences in linguistic contexts used in conversations. This is because variations in utterance lengths, word choices and sentence structures can introduce variance in acoustic features independent of vocal differences. In fact, Roark et al. points out that “narrow and topic-focused use of language” is important for “more accurate” acoustic marker extraction [32]. However, while these studies suggest that acoustic markers can have very high predictive value, the highly structured conversational settings may restrict the effectiveness of linguistic markers. Traditionally, semi-structured conversational settings have been used in linguistic marker studies because they reflect participant linguistic preferences in open conversations (without fixed sentences). Therefore, we are interested in combining acoustic and linguistic features in a synergistic way, despite the fact that the semi-structured setting introduces interesting trade-offs between the stability of acoustic features and the expressiveness of linguistic ones.

2.5.2 Personality Modeling

Persona modeling refers to non goal-oriented dialog, otherwise known as chit-chat dialogue systems. Vinyals et al. [18] introduced a statistical machine translation (SMT) framework for sequence-to-sequence modeling of dialogue turns, with neither dialogue-state tracking nor modeling long-term dependence between turns. As a result, vanilla sequence-to-sequence models often fail to capture long-term dependencies of dialogue responses and do not conform to a coherent "personality" in conversations [23, 24]. Li et al.[19] introduced *persona-based* modeling which incorporates persona information about a user as *word embeddings*. Recently, [23] extended this idea by building the PersonaChat dataset to permit the modeling of persona-dependent dialogue trajectories that capture both turn-based transitions as well as conditional response generation based on personalized text profiles. Such an approach generated great interest in the research community. The main focus of persona modeling has been how to incorporate persona information into the dialogue generation process. For example, Song et al. [34] introduced a method for incorporating persona information through variational autoencoders (VAE) with external memory, improving performance on automatic evaluations. The ConvAI2 competition [24] provided a diverse sample of state-of-the-art model architectures for persona modeling. However, as pointed out in the analysis of results from the ConvAI2 (NeurIPS '18) competition, a bottleneck of persona models remains the disparity between automatic and human evaluation.

2.5.3 Automatic Evaluation of Persona Models

The NeurIPS '18 challenge [24] introduced several limitations for *automatic evaluation* of dialogue quality. For example, that F1 and perplexity (PPL) scores do not consider temporal dependence of text generation, nor the underlying semantics of generated texts. This was reflected in the fact that performance of persona models on automatic evaluation was not predictive of performance against human evaluation – the eventual competition winner scored relatively poorly compared to other top models in the automatic evaluation category. Additionally, it was found that classical metrics such as F1 score and hits@1 automatically favor ranking models over generative models. PPL, F1 and Hits@1 also do not provide any information about the style of responses, for instance tendencies to ask questions to human users or bias toward certain conversational topics. Previously, Xing et al. [35] introduced a method of evaluating dialogue quality beyond classical metrics by using a classifier to predict meta-data about the underlying persona (e.g., extroversion and personality traits). However, such an approach requires strong supervision for each persona, which is not readily available for large dialogue corpus. Hu et al. [36] introduced the idea of using a discriminator as an automatic evaluator of generated dialogues. Zhang et al. [37] extended the use of discriminators to include language models for constraining the possible response outputs and policy gradients to deal with discrete generator inputs. However, discriminator-based approaches focus mostly on response-level coherence rather than coherence of the dialogue trajectory over multiple turns. In this work, we introduce an automatic evaluation scheme that systematically addresses each of these issues.

2.6 Feature Selection in Medical Informatics

One prominent problem in medical informatics is the problem of feature selection over disease markers. Aside from the curse of dimensionality, biomarkers are expensive and cannot be obtained for each patient at every timestep. A prime example of this phenomenon exists in the realm of medical informatics, specifically electronic health records (EHR) dominated by time-series data. Yet this hasn't stopped EHR from sparking numerous research interests in recent years [38, 39, 40, 41, 42]. Time-series data in the EHR consists of features that are sampled at different levels of temporal granularity (e.g., lab tests are sampled at longer time-scales than vital signs). Most of the time, only a small subset of features are observed at any time-step, as it is prohibitively expensive for the physician to obtain the full set of features for every patient for every time interval. In practice, a form of active sensing [43] is implicitly done in an ad-hoc manner by physicians based on a combination of inpatient work-flow, resource constraints, and domain expertise. As a result, the key difficulty with EHR time-series data is that they almost always require re-sampling and imputation due to large amounts of missing values. For example, consider the EHR setting, where at each time-step, the physician has to decide how to allocate a constrained amount of tests/resources among patients for some set of clinical tasks. In other words, the precision sensing problem for time-series is in fact a *resource distribution* problem at each time-step, with a fixed observation budget across a given set of samples. Since the querying process is applied across time rather than samples, any feature selection formulation must reflect both the temporal-dependency and the budgetary-constraint of the querying process. Therefore it is no surprise that several recent works have shown that temporal models trained on EHR data have fragile decision boundaries that are susceptible to small perturbations [44, 45].

In later chapters, we present a framework called *precision sensing* to overcome the temporal dependence and non-i.i.d. property of medical feature selection. We first introduce some related works in active sensing below.

2.6.1 Active Sensing

Precision sensing typically falls under the suite of active data acquisition problems such as active learning [46] (optimal experiment design) and active sensing [43], both of which attracted significant research interest from the machine learning community. Techniques such as incremental feature acquisition [47] and active sensing [43, 48] focus on selecting a subset of features by querying the original data in an online manner, with the goal of reconstructing (sample, view) pairs that maximize the mutual information of predictive classifiers. Note that the original active sensing problem proposed by Yu *et al.* [43] operates on *static* feature sets rather than time-series data. More recently, active sensing has been extended to time-series data by [48] and [49], but in a problem setting very different from this paper. This is because both [48] and [49] manage not only a feature selection budget *across features*, i.e., "data-streams", but also *across time*. However, a separate budget constraint is considered for each individual. Precision sensing presents an additional layer of complexity—the sensor model has to consider budget constraints *across samples*. That is, *we do not make the i.i.d. assumption across data samples*. In reality, a sensing strategy has to take into account an accumulating budget across all samples in a data batch. Using the toy example from [49], a physician may need to conserve the observation budget for one group of patients in order to decrease uncertainty of predictions for another batch at future time points. Thus, our goal is to dynamically adjust the allocation of a fixed observation budget across a time window and a set of samples. In experiments, we compare our method against state-of-the-art (SOTA) active sensing methods.

2.6.2 Temporal Feature Selection

Classic statistical methods [50, 51], perturbation models [44] and attention-based techniques [52] have also been applied to distill model-preferences over temporal features across time. Schulam [53] and Saria et al. [54] proposed modeling temporal relevance of features by abstracting hierarchical representations across time (e.g., population-level and subpopulation-level latent features). However, all these strategies lack an active component because their sensing strategies neither adapt to shifts in future data-streams nor do they incrementally build the future sensing strategy. While Clertant [55], like Chang et al. [49], formulated the sensing problem in a Markov Decision Process (MDP) framework, their proposed algorithm deals with the sequential selection of *static features*, rather than temporal data. Moreover, perturbation models and temporal feature selection [56] are specific to the hypothesis model that they are trained on. But ideally a sensing strategy should be generalizable to different hypothesis classes and decision boundaries. We experimentally show how our method maintains performance for different model types, including non-temporal models (e.g., feed-forward networks) as well as non-deep models (e.g., logistic regression).

2.7 Sample Efficiency and Privileged Information

Finally, we explore the issue of improving sample-efficiency in using disease markers for clinical prediction. In classical supervised learning, the learner is presented with the training tuple $\{(x_i, y_i)\}_{i=1}^{m}$ and performs an optimization task of finding the best model in a hypothesis space $h : X \mapsto Y$ to approximate some true $f : X \mapsto Y$ which explains the data. Given a new task, knowledge transfer [57] is often applied to accelerate the learning process by distilling and transferring relevant knowledge from previous tasks to the unseen one. Under classical formulations, the learner incorporates prior information in one of several ways:

1. Direct transfer of parameters from old hypothesis models to the new task and *fine-tuning* [57] the parameters.

2. Learning multiple tasks (online or batched) related to the current task [58, 42].

3. Using the prior knowledge (i.e. a knowledge graph) to constrain the hypothesis space by regularization [59].

4. Using representations (i.e. *embeddings*) of X and / or Y from previous tasks for new tasks [5, 60].

5. Accelerate learning rate and model compression by Distillation as typically seen in Teacher-Student models [61].

In each of these settings, knowledge transfer operates directly within the X, Y and $\mathcal{H}$ spaces to improve generalization of information from old models to the new task.

Recently, *Learning Using Privileged Information* (LUPI) [62] has provided a new paradigm for knowledge transfer. Under LUPI, the learner now interacts with a Teacher that provides *privileged information* (PI) and is only available at training time. From the learner's perspective, the training set is now extended to the tuple $\{(x_i, x_i^*, y_i)\}_{i=1}^n$, and the testing set stays the same. Some examples of PI include: 1) Future information that relates X and Y. For example, using future stock prices beyond the prediction window during training. 2) Auxiliary information describing the label space that is available only to a subset of samples. For example, physician notes that accompany diagnostic predictions which is only available after the diagnosis is made.

At a high-level, PI provides some similarity information between training samples from the original feature space, and the Teacher hypotehsis serves as additional "explanations" of the hypothesis space [63, 62]. As a result, [63] showed that the LUPI Teacher provides a principled way to improve the *generalization error* of Student learners using agnostic PAC models, providing some theoretical improvements in the number of samples required to achieve generalizability the new task (i.e. improves sampling efficiency)

However, under the current state-of-art LUPI formulations such as [64, 65], PI is incorporated by means of support vectors and dropout schemes, both of which fail to explore the underlying similarity structure between samples in the PI space X^*. For example the mode distribution and pairwise similarity between points in the X^* space is largely unused. The PI contributes as auxiliary training features and kernel information, but all LUPI information is lost at inference time and beyond. A significant question remains: **can privileged information be *retained* for future tasks?**

Ideally, we want the LUPI Teacher to incorporate PI in a way that is specific enough to inform similarity between training samples yet general enough to be retained across future tasks. In the following subsections, we highlight some of these differences as well as improvements over related LUPI works.

2.7.1 Transfer Learning Review

Consider the *transfer learning* setting: Assume we are given a source domain $\mathcal{D}_S = \{\mathcal{X}_S, P_S(X)\}$, a source task $\mathcal{T}_S = \{\mathcal{Y}_S, \mathcal{H}_S\}$, a target domain $\mathcal{D}_T = \{\mathcal{X}_T, P_T(X)\}$, and a target task $\mathcal{T}_T = \{\mathcal{Y}_T, \mathcal{H}_T\}$, where $\mathcal{X}$ defines a feature space, $P(X)$ defines its marginal distribution, $\mathcal{Y}$ defines a label space, and $\mathcal{H}$ defines a space of hypothesis which best approximates an underlying function $f : X \mapsto Y$ which explains the data. Transfer learning leverage knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$ to improve the learning of $h^*_T \in \mathcal{H}_T$, where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$. Improving the learning of h^*_T can come in two ways: (1) improving the sample complexity of $\mathcal{H}_T$, or (2) improving the rate of convergence for finding $h^*_T \in \mathcal{H}_T$.

Transductive learning describes a situation where $\mathcal{D}_S \neq \mathcal{D}_T$ but $\mathcal{T}_S = \mathcal{T}_T$, which can occur either due to mismatch in feature space, i.e. $\mathcal{X}_S \neq \mathcal{X}_T$, or due to co-variate shift, i.e. $P_S(X) \neq P_T(X)$[57, 66]. On the other hand, *inductive learning* describes a situation where $\mathcal{D}_S = \mathcal{D}_T$ but $\mathcal{T}_S \neq \mathcal{T}_T$, i.e. the same domain but different tasks. Briefly, the various transfer learning approaches are summarized in table 2.1.

2.7.2 Knowledge Transfer by Inductive Learning

A popular approach for solving the inductive learning problem is *parameter sharing*, whereby some or all of the parameters of $h^*_S \in \mathcal{H}_S$ is recycled for the target model, and *fine-tuning* is applied to accelerate the search for $h^*_T \in \mathcal{H}_T$ [57, 71]. For example, consider multi-task learning (MTL), where

Table 2.1: Summary Table of Transfer Learning Approaches

Setting / Method	Domain	Targets	Examples
Inductive learning	$\mathcal{D}_S = \mathcal{D}_T$	$\mathcal{T}_S \neq \mathcal{T}_T$	
Multi-task Learning	Same Features	$\mathcal{Y}_S \neq \mathcal{Y}_T$	[67]
Parameter Transfer	Same Features	$\mathcal{T}_S \neq \mathcal{T}_T$	
Relational Knowledge	Relational Graph	$\mathcal{Y}_S \neq \mathcal{Y}_T$	[59]
Transductive Learning	$\mathcal{D}_S \neq \mathcal{D}_T$	$\mathcal{T}_S = \mathcal{T}_T$	
Multi-View Learning	$\mathcal{X}_S \neq \mathcal{X}_T$	Same Tasks	[60, 68]
Sample Selection Bias	$P_S(Y\|X) \neq P_T(Y\|X)$	Same Tasks	[69]
Co-Variate Shift	$P_S(X) \neq P_T(X)$	Same Task	
Domain-Adaptation	$P_S(X,Y) \neq P_T(X,Y)$	$\mathcal{Y}_S = \mathcal{Y}_T$	
Unsupervised Learning	$\mathcal{D}_S \neq \mathcal{D}_T$	$\mathcal{T}_S \neq \mathcal{T}_T$	
Feature Representation	Transfer of $\phi(X_S) \longrightarrow \phi(X_T)$	New or Old Tasks	[5, 70]

multiple hypotheses are learned jointly for a set of related tasks [57]. Under this setting, the set of tasks $\mathcal{T}_1, ...\mathcal{T}_T$ are presumed to come from the same underlying distribution. Using the same feature space X, a subset of the joint hypotheses space $\bigcup_{k=1} \mathcal{H}_k \subset \{\mathcal{H}_1, ..\mathcal{H}_{T-1}\}$ should presumably accelerate the learning of the optimal hypothesis for the target task. Thus, by transferring parameters from $\{\mathcal{H}_1, ..\mathcal{H}_{T-1}\}$ to $\mathcal{H}_T$, convergence rate to the optimal h_T^* can be improved.

A disadvantage of MTL is the constraint on relatedness of tasks. For example, it is hard to know beforehand which subset of tasks will contribute positively to the target, and poor selection of parameter transfer can actually lead to *negative transfer*, leading to poor performance on the target task [57]. By contrast, LUPI uses PI that is by definition *specific to the current task*. However, unlike MTL, LUPI does not use a shared feature space between the PI and original data, i.e. $X \neq X^*$ and cannot directly incorporate the Teacher hypothesis via direct parameter sharing.

In recent years, *relational-knowledge* provides an alternative approach to inductive learning by incorporating domain-specific knowledge in the form of regularized priors for target tasks [59, 72]. In contrast to parameter-sharing, relational-knowledge is agnostic to the collection of source tasks and should apply universally to all learning tasks given the same feature space X. This formulation overcomes the limitation of task similarity in the former case, but it is also very expensive to construct reliable relational-knowledge, such as knowledge graphs. By constraining the hypothesis space of the target task with relational-knowledge, the speed of convergence can be improved.

However, inductive learning techniques generally focus on improving either accuracy of prediction or convergence rate rather than sample complexity. The main focus of LUPI, on the other hand, is to provide some guarantees on improving the sample efficiency of the Student learner.

2.7.3 Knowledge Transfer by Transductive Learning

In contrast to inductive learning, transductive learning considers two sub-problems: $\mathcal{X}_S \neq \mathcal{X}_T$ and $P_S(Y|X) \neq P_T(Y|X)$. *Multi-view Learning* (MVL) [60, 73] and *Multi-modal Learning* (MML) [74, 75] are main methods which deal with the $\mathcal{X}_S \neq \mathcal{X}_T$ case, where the feature spaces differ between the source and target domains. MVL is often involved in processing different subsets of features describing the same set of samples, for example different channels of EEG signals for neurological diagnostics [76]. MML deals with different modalities of data, for example picture and text descriptions of a disease process [77]. Often times, both MVL and MML utilize some form of *data fusion*, whereby some shared representation of the multiple source domains is used together to predict the same target task [67]. One drawback of these approaches is that modalities such as PI is *unavailable at test time*, leading to poor generalization of the hypothesis model since it is conditioned on both X and X^*.

For example, suppose we have some data fusion model $g : X \times X^* \mapsto Z$, and a hypothesis function $h : Z \mapsto Y$. At training time, both the PI and the original features are utilized to train $h(g(x, x^*)) = y$. At test time, however, since only X is available, $g(x, 0)$ may actually map to a completely different set of features in Z, leading to a biased $h(z) = y$. In other words, if X is under-utilized during training, $h(z)$ will likely lead to poor generalization at test time. Unfortunately, since PI is by definition a more task-specific descriptor of Y, this is the most likely case and presents a limitation for data fusion methods for incorporating PI.

On the other hand, methods such as biased selection sampling [78] and Optimal Transport [79] are used to deal with the same set of features, but their marginal distributions disagree, i.e. $P_S(X) \neq P_T(X)$. This is otherwise known as *co-variate shift* [66], and recently generative models such as [80, 81] and [82] have dominated the state-of-art. For example, VRADA and

RadialGAN both attempt to learn a *domain-invariant* latent distribution s.t. $P(Z|X_S) = P(Z|X_T)$, and $P_S(Y|Z) = P_T(Y|Z)$[82, 80]. Methods such as [83] and [81] try to directly learn a mapping between $P(X_T|X_S)$ so that samples from X_S can be used to augment the training of $h(x) = y$. These generative models present a possible way of incorporating PI into the modeling process by learning a domain-invariant latent representation or transformation function between X and X^*. However, these models improve the hypothesis function by means of *data augmentation* (i.e. increasing the samples available) rather than decreasing the sample complexity required to train an accurate model. In this regard, LUPI is advantageous in that it provides a framework for decreasing the samples necessary by leveraging the Teacher's hypothesis function f^*. This subtle different becomes important when "big data" is not available for complex data problems, for example modeling rare diseases in healthcare records.

2.7.4 Knowledge Transfer with Distributed Representations

Transfer of feature representations is another paradigm of transfer learning that extends beyond inductive transfer [57]. Under this setting, no label information is available for either the source or target domains. However, the underlying structure of the source domain X_S can first be extracted using *unsupervised learning* before applying to downstream tasks. A prime example of this can be seen in the learning of *distributed representations* for words [5], which is widely used in NLP applications by converting the feature space of words (which have no intrinsic distance or similarity properties) into an embedding vector space where distance and similarity can be computed based on co-occurrence frequencies. Several related methods have been applied in the context of medicine by learning distributed representations for medical concepts obtained from large corpora of journal publications [70], EHR notes [41], and medical claims [72]. This framework provides a tool for understanding the structural properties of PI, which can then be transfered to improve the actual learning task of estimating the optimal Student hypothesis function. However, tactful incorporation of these embeddings may not be as simple, since the learning regime must be able to incorporate the embedded information in the hypothesis somehow. Our methodology introduces a way to achieve this and incorporates both structural information about the PI as well as the Teacher hypothesis function into the Student learning procedure.

2.7.5 Knowledge Transfer by Knowledge Distillation

A closely related concept to LUPI is *knowledge distillation* [61] in which a Teacher model outputs are used as "soft-labels" to accelerate Student learning of the target task. Similar to LUPI, the Teacher learns a more accurate model of the task, $f^* : X \mapsto Y$, and the Student tries to learn a "distilled" representation of the hypothesis space, where the VC-dimension of the Student hypothesis space is less than that of the teacher [84]. By decreasing its VC-dimension, the Student learner can improves its sampling efficiency at least by a constant factor. However, the Teacher model does not incorporate PI, which provides a better prediction of the label than the original feature space. Additionally, the original LUPI provides a more favorable generalization bound compared to model distillation without the help of PI [84, 63].

2.7.6 Knowledge Transfer by LUPI

Finally, LUPI provides some performance guarantees with regard to the sample efficiency of the Student learner, so long as the PI and the Teacher model satisfies some conditions [62, 63]. However, the main drawback of current formulations of LUPI is that the PI used is highly specific to the task at hand – no information is retained for *related tasks*. Our work applies elements of *unsupervised learning* and *transductive learning* to alleviate this limitation of LUPI. Although recent works such as [64] has generalized the LUPI framework to deep learning settings, our work extends LUPI to allow for multi-task and transfer learning, enabling the generalization of a PI source to accelerate the sample efficiency of many tasks.

2.7.7 LUPI Preliminaries

Traditionally, LUPI is applied to training data of the form:

$$(x_1, x_1^*, y_1), (x_2, x_2^*, y_2)...(x_m, x_m^*, y_m) \in \mathcal{D}_{\text{train}},$$

where $x_i \in \mathbb{R}^n$ denotes example feature (EF) vectors from the original feature space, and $x_i^* \in \mathbb{R}^{n*}$ denotes privileged information (PI) vectors from the privileged information space. $\mathcal{D}_{train}$ indicates that the PI inputs are only available during training. $y_i \in \{-1, +1\}$ denotes the ground truth labels for inputs (x_i, x_i^*). LUPI then considers two pattern recognition problems:

- Using $\{(x_i, y_i)\}_{i=1}^m$, find rule $h(x_i) = y_i$.
- Using $\{(x_i^*, y_i)\}_{i=1}^m$, find rule $f^*(x_i^*) = y_i$

Suppose that the $f^*(x_i^*) = y$ can produce low generalization error, the LUPI task is to transfer knowledge of rules in the X^* space to improve learning in the X space. The original LUPI formulation in [62] considers only SVM models, whereby the privileged information is incorporated into the SVM objective as follows:

$$\begin{aligned} \underset{w,w^*}{\text{minimize}} \ & \frac{1}{2}(w^T w + \lambda w^{*T} w^*) + C \sum\nolimits_i^m (w^{*T} \psi(x_i^*) + b^*) \\ s.t. \ & y_i(w^T \phi(x) + b) \geq 1 - (w^{*T} \psi(w^*) + b^*), \\ & w^{*T} \psi(x^*) + b^* \geq 0 \end{aligned}$$

Here, w and w^* are the parameter vectors, and b and b^* are biases of the decision functions. $\psi(.)$ and $\phi(.)$ are feature mapping kernels applied on PI and EF vectors, respectively. $C > 0$ and $\lambda > 0$ are hyperparameters which control the contribution of privileged information to the overall cost function. C controls the contribution of the Teacher loss $f^* : X^* \mapsto Y$, while λ controls correction of the parameter space by $w^{*T} w^*$.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Kathleen C Fraser, Jed A Meltzer, and Frank Rudzicz. Linguistic features identify alzheimer's disease in narrative speech. *Journal of Alzheimer's Disease*, 49(2):407–422, 2016.

[2] Meysam Asgari, Jeffrey Kaye, and Hiroko Dodge. Predicting mild cognitive impairment from spontaneous spoken utterances. *Alzheimer's & Dementia: Translational Research & Clinical Interventions*, 3(2):219–228, 2017.

[3] James W Pennebaker, Martha E Francis, and Roger J Booth. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001):2001, 2001.

[4] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.

[6] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[7] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

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

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

[10] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

[11] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter*, 19(2):25–35, 2017.

[12] Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and

Li Deng. End-to-end reinforcement learning of dialogue agents for information access. *arXiv preprint arXiv:1609.00777*, 2016.

[13] Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*, 2016.

[14] Wei He, Kai Liu, Jing Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, et al. Dureader: a chinese machine reading comprehension dataset from real-world applications. *arXiv preprint arXiv:1711.05073*, 2017.

[15] Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. Reinforced mnemonic reader for machine reading comprehension. *arXiv preprint arXiv:1705.02798*, 2017.

[16] Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *Knowledge Engineering Review*, 21(2):97–126, 2006.

[17] Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, 16:105–133, 2002.

[18] Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

[19] Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*, 2016.

[20] Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[21] Yookoon Park, Jaemin Cho, and Gunhee Kim. A hierarchical latent structure for variational conversation modeling. *arXiv preprint arXiv:1804.03424*, 2018.

[22] Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *arXiv preprint arXiv:1703.10960*, 2017.

[23] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*, 2018.

[24] Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack

Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, et al. The second conversational intelligence challenge (convai2). In *The NeurIPS'18 Competition*, pages 187–208. Springer, 2020.

[25] Lin Ni, Chenhao Lu, Niu Liu, and Jiamou Liu. Mandy: Towards a smart primary care chatbot application. In *International Symposium on Knowledge and Systems Sciences*, pages 38–52. Springer, 2017.

[26] Paris Hsu, Jingshu Zhao, Kehan Liao, Tianyi Liu, and Chen Wang. Allergybot: A chatbot technology intervention for young adults with food allergies dining out. In *CHI*, pages 74–79. ACM, 2017.

[27] Kyo-Joong Oh, Dongkun Lee, Byungsoo Ko, and Ho-Jin Choi. A chatbot for psychiatric counseling in mental healthcare service based on emotional dialogue analysis and sentence generation. In *MDM*, pages 371–375. IEEE, 2017.

[28] Juan Manuel Fernandez Montenegro and Vasileios Argyriou. Cognitive evaluation for the diagnosis of alzheimer's disease based on turing test and virtual environments. *Physiology & behavior*, 173:42–51, 2017.

[29] Miguel A Salichs, Irene P Encinar, Esther Salichs, Álvaro Castro-González, and María Malfaz. Study of scenarios and technical requirements of a social assistive robot for alzheimer's disease patients and their caregivers. *International Journal of Social Robotics*, 8(1):85–102, 2016.

[30] Hiroki Tanaka, Hiroyoshi Adachi, Norimichi Ukita, Manabu Ikeda, Hiroaki Kazui, Takashi Kudo, and Satoshi Nakamura. Detecting dementia through interactive computer avatars. *IEEE journal of translational engineering in health and medicine*, 5:1–11, 2017.

[31] Tom N Tombaugh and Nancy J McIntyre. The mini-mental state examination: a comprehensive review. *J. of the Ame. Geriatrics Soc.*, 40(9):922–935, 1992.

[32] Brian Roark, Margaret Mitchell, John-Paul Hosom, Kristy Hollingshead, and Jeffrey Kaye. Spoken language derived measures for detecting mild cognitive impairment. *IEEE transactions on audio, speech, and language processing*, 19(7):2081–2090, 2011.

[33] Tuka Alhanai, Rhoda Au, and James Glass. Spoken language biomarkers for detecting cognitive impairment. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 409–416. IEEE, 2017.

[34] Haoyu Song, Wei-Nan Zhang, Yiming Cui, Dong Wang, and Ting Liu. Exploiting persona information for diverse generation of conversational responses. *arXiv preprint arXiv:1905.12188*, 2019.

[35] Yujie Xing and Raquel Fernández. Automatic evaluation of neural personality-based chatbots. *arXiv preprint arXiv:1810.00472*, 2018.

[36] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1587–1596. JMLR. org, 2017.

[37] Zhirui Zhang, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. Bidirectional generative adversarial networks for neural machine translation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 190–199, 2018.

[38] Yu Cheng, Fei Wang, Ping Zhang, and Jianying Hu. Risk prediction with electronic health records: A deep learning approach. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 432–440. SIAM, 2016.

[39] Fengyi Tang, Cao Xiao, Fei Wang, and Jiayu Zhou. Predictive modeling in urgent care: a comparative study of machine learning approaches. *JAMIA Open*, 2018.

[40] Abhyuday N Jagannatha and Hong Yu. Bidirectional rnn for medical event detection in electronic health records. In *Proceedings of the conference. Association for Computational Linguistics. North American Chapter. Meeting*, volume 2016, page 473. NIH Public Access, 2016.

[41] Edward Choi, Mohammad Taha Bahadori, Elizabeth Searles, Catherine Coffey, Michael Thompson, James Bost, Javier Tejedor-Sojo, and Jimeng Sun. Multi-layer representation learning for medical concepts. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1495–1504. ACM, 2016.

[42] Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *arXiv preprint arXiv:1703.07771*, 2017.

[43] Shipeng Yu, Balaji Krishnapuram, Romer Rosales, and R Bharat Rao. Active sensing. In *Artificial Intelligence and Statistics*, pages 639–646, 2009.

[44] Mengying Sun, Fengyi Tang, Jinfeng Yi, Fei Wang, and Jiayu Zhou. Identify susceptible locations in medical records via adversarial attacks on deep predictive models. *arXiv preprint arXiv:1802.04822*, 2018.

[45] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[46] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[47] Prem Melville, Maytal Saar-Tsechansky, Foster Provost, and Raymond Mooney. Active feature-value acquisition for classifier induction. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 483–486. IEEE, 2004.

[48] Jinsung Yoon, William R Zame, and Mihaela van der Schaar. Deep sensing: Active sensing using multi-directional recurrent neural networks. 2018.

[49] Chun-Hao Chang, Mingjie Mai, and Anna Goldenberg. Dynamic measurement scheduling for event forecasting using deep rl. *arXiv preprint arXiv:1901.09699*, 2019.

[50] Michail Tsagris, Vincenzo Lagani, and Ioannis Tsamardinos. Feature selection for high-dimensional temporal data. *BMC bioinformatics*, 19(1):17, 2018.

[51] Liying Fang, Han Zhao, Pu Wang, Mingwei Yu, Jianzhuo Yan, Wenshuai Cheng, and Peiyu Chen. Feature selection method based on mutual information and class separability for dimension reduction in multidimensional time series for clinical data. *Biomedical Signal Processing and Control*, 21:82–89, 2015.

[52] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems*, pages 3504–3512, 2016.

[53] Peter Schulam and Suchi Saria. A framework for individualizing predictions of disease trajectories by exploiting multi-resolution structure. In *Advances in Neural Information Processing Systems*, pages 748–756, 2015.

[54] Suchi Saria, Andrew Duchi, and Daphne Koller. Discovering deformable motifs in continuous time series data. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[55] Matthieu Clertant, Nataliya Sokolovska, Yann Chevaleyre, and Blaise Hanczar. Interpretable cascade classifiers with abstention. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2312–2320, 2019.

[56] Rohan A Baxter, Graham J Williams, and Hongxing He. Feature selection for temporal health records. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 198–209. Springer, 2001.

[57] Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[58] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[59] Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F Stewart, and Jimeng Sun. GRAM: graph-based attention model for healthcare representation learning. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 787–795. ACM, 2017.

[60] Jing Zhao, Xijiong Xie, Xin Xu, and Shiliang Sun. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54, 2017.

[61] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[62] Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural networks*, 22(5-6):544–557, 2009.

[63] Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: similarity control and knowledge transfer. *Journal of machine learning research*, 16(2023-2049):2, 2015.

[64] John Lambert, Ozan Sener, and Silvio Savarese. Deep learning under privileged information using heteroscedastic dropout. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8886–8895, 2018.

[65] Xue Li, Bo Du, Chang Xu, Yipeng Zhang, Lefei Zhang, and Dacheng Tao. R-svm+: Robust learning with privileged information. In *IJCAI*, pages 2411–2417, 2018.

[66] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[67] Dhanesh Ramachandram and Graham W Taylor. Deep multimodal learning: A survey on recent advances and trends. *IEEE Signal Processing Magazine*, 34(6):96–108, 2017.

[68] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, and Jieping Ye. Deep multi-view spatial-temporal network for taxi demand prediction. *arXiv preprint arXiv:1802.08714*, 2018.

[69] Yi Luo, Guojie Song, Pengyu Li, and Zhongang Qi. Multi-task medical concept normalization using multi-view convolutional neural network. 2018.

[70] Youngduck Choi, Chill Yi-I Chiu, and David Sontag. Learning low-dimensional representations of medical concepts. *AMIA Summits on Translational Science Proceedings*, 2016:41, 2016.

[71] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[72] Samuel G Finlayson, Paea LePendu, and Nigam H Shah. Building the graph of medicine from millions of clinical narratives. *Scientific data*, 1:140032, 2014.

[73] Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*, 2013.

[74] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[75] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011.

[76] Ye Yuan, Guangxu Xun, Kebin Jia, and Aidong Zhang. A multi-view deep learning method for epileptic seizure detection using short-time fourier transform. In *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 213–222. ACM, 2017.

[77] C. Nagpal. Deep multimodal fusion of health records and notes for multitask clinical event prediction. In *NIPS ML4H Workshop*, 2017.

[78] Bianca Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, page 114. ACM, 2004.

[79] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[80] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. Radialgan: Leveraging multiple datasets to improve target-specific predictive models using generative adversarial networks. *arXiv preprint arXiv:1802.06403*, 2018.

[81] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. *arXiv preprint*, 1711, 2017.

[82] Sanjay Purushotham, Wilka Carvalho, Tanachat Nilanon, and Yan Liu. Variational recurrent adversarial deep domain adaptation. 2016.

[83] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint*, 2017.

[84] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*, 2015.

CHAPTER 3

SCALABLE DIAGNOSTIC SCREENING OF MCI USING DIALOG POLICY

3.1 Introduction

The progression of Alzheimer Disease (AD) has consistently been a heavy area of research in clinical medicine because while the disease itself is incurable, early intervention at the prodromal phases of the disease has proven to delay the onset of AD-related mental degeneration and systemic issues for months to years [1, 2]. Consequently, much of the recent clinical research efforts have focused on detecting early stages of mild cognitive impairment (MCI), which is a prodromal phase in AD progression occurring months to years before visible mental decline begins [3]. If successfully detected at this stage, intervention methods may confer numerous benefits in the longevity of cognitive and physiological health of AD patients [1, 2].

Brain imaging, such as the structural magnetic resonance imaging (MRI), was shown to contain prime markers of AD, capturing the physiologic changes in the AD pathological process [4, 5]. However, the identification of MCI from normal aging (NL) is particularly challenging due to the fact that structural changes in the brain in this phase are minor and hard to detect through structural MRI [6], even though decline in mental status and cognitive have already begun in most cases. Recently, the structural connections among brain regions inferred from diffusion MRI have provided promising predictive performance of MCI detection [7, 8], yet sketching brain networks via imaging still remains rather prohibitively expensive and difficult to scale. Moreover, the high dimensionality of brain imaging combined with small sample size usually imposes significant challenges in learning algorithms and leads to unstable generalization performance.

On the other hand, behavior and social markers could offer a much more cost- effective option for MCI detection [9, 10, 11, 12]. A recent clinical trial has studied differentiating early stage MCI from NL cohort groups using transcripts of extensive conversations between patients and trained interviewers [11]. In a recent preliminary study [12], the authors trained supervised learning models from the *lexical distribution* of the conversation, and showed that conversational responses of MCI and NL patients take on different distribution under various conversational topics. The success of [12] in predicting MCI using human dialogue introduced an alternative natural language processing (NLP) approach to a classically clinically expensive problem. However, the use of human interviewers still requires substantial amounts of interaction between trained staff which incur significant expense in its current form. Thus, the bottleneck questions remain: *(1) can we cut down on the amount of conversations needed to achieve accurate prediction*, *(2) can we improve upon baseline performance given limited cohort-specific data*?

To address the aforementioned questions above, in this chapter we propose a novel reinforcement learning (RL) framework, which learns a MCI diagnosis agent using only very limited amount of offline human dialogue transcripts. The learned diagnosis agent can conduct subject-specific conversation with humans, asking questions based on existing conversations to efficiently sketch the lexical distribution and give high-performance MCI prediction. In order to facilitate RL using offline transcripts, we introduce a *dialogue simulator pipeline* which generates new conversational episodes that are less noisy and out-perform the original corpus for MCI prediction.

Our dialogue pipeline provides a self-contained framework for directing dialogue generation for diagnostic screening which can potentially replace the need for human-expert interviews. Our RL-agent learns optimal dialogue strategies that are adaptive to unseen users, enabling medically-relevant NLP data to be generated on a large scale if deployed in a realistic setting. Furthermore, data generated from our dialogue simulations may be used for data augmentation or to perhaps guide the medical data collection process in the future. Ultimately, by greatly decreasing the cost of data collection and the amount needed for high-level performance, we introduce a clinical direction that is much more cost-effective and scalable to large-scale diagnostic screening and data collection. The combination of NLP features with our reinforcement learning framework may extend the process of diagnostic screenings to well beyond the confines of hospitals and primary care facilities.

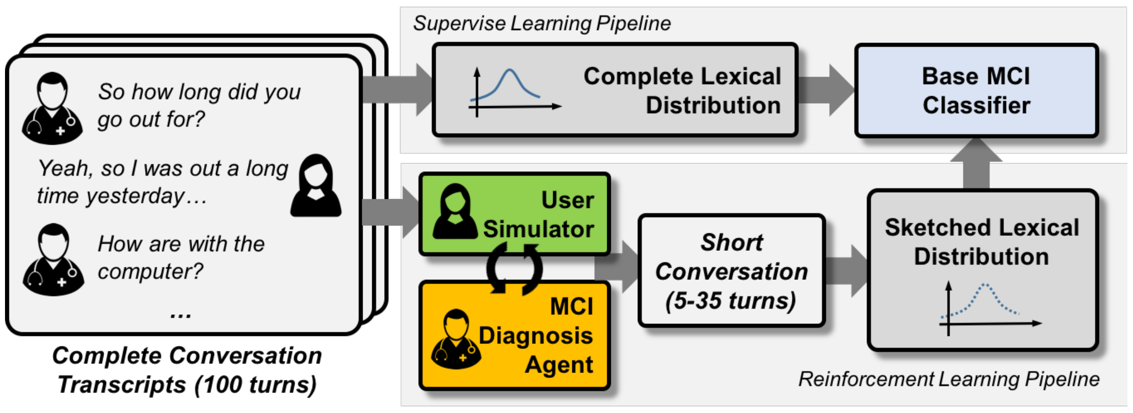
3.2 Methodology

3.2.1 Dialog Systems

Our approach is inspired by recent advancements in dialog systems. Dialog systems provide a natural human-computer interface and have been an active research field for decades. Task-oriented dialogue systems are typically designed for retrieval- tasks in which users provide queries and the chat-bot provides appropriate responses based on an external knowledge base [13, 14, 15], or identifying correct answers by looking into vast amounts of documents [16, 17]. Such dialogue systems are typically designed to be a pipeline containing a set of components including a language understanding unit that parses the intention and semantics from the input from humans, a dialogue manager that manages dialogue state tracking and policy learning, and a language generation unit that generates response [15, 18, 19]. While each of the components can be handcrafted or trained individually, recent advances of deep learning allows end-to-end training [13, 14, 20, 20] and significantly improves the performance and the capability to adapt to new domains [21]. The end-to-end systems can be trained using supervised learning [13, 22] or reinforcement learning (RL), by leveraging a user simulator [14, 20]. The main advantage of RL is that less training samples are needed to learn the high-degree-of-freedom deep models. In our work, we design a simulator to enable RL due to the limited amount of clinical data available for supervised training. We note that even though our dialogue system also tries to achieve a task (identifying MCI patients), the nature of our system is radically different from existing task-oriented dialogue systems: its goal is to efficiently sketch a disease specific lexical distribution through asking subject-specific questions and give classification results.

The framework we propose in this chapter involves the use of reinforcement learning to learn the optimal set of questions π^* to ask participants for the purposes of distinguishing MCI. On test set, we generate new episodes from these questions for prediction rather than the original corpus. To actualize the RL + dialogue simulation framework, we proposed a multi-step approach for implementation which capitalizes on the vast existing knowledge of NLP research. In the following

Figure 3.1: Overview of the proposed methodology.

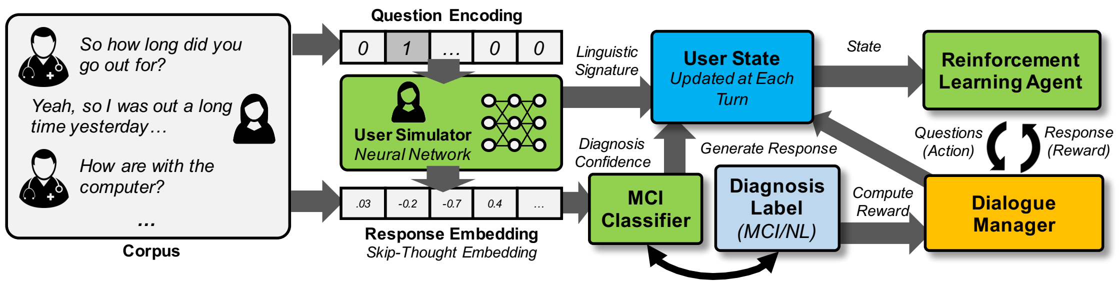

Supervise Learning Pipeline
So how long did you go out for?
Yeah, so I was out a long time yesterday…
How are with the computer?
…
Complete Conversation Transcripts (100 turns)
Complete Lexical Distribution
Base MCI Classifier
User Simulator
MCI Diagnosis Agent
Short Conversation (5-35 turns)
Sketched Lexical Distribution
Reinforcement Learning Pipeline

section, we present the details of each component of the dialogue system. Figure 3.1 shows an overview of the components of our experimental pipeline. Complete conversation from participants are used to build user simulators. The simulators are then used to train an MCI diagnosis agent (chat-bot), which conducts minimal turns of conversation with participants to sketches the lexical distribution that is then used to perform MCI classification.

3.2.2 Overview of Pipeline

Our proposed framework contains three key learning modules: the *user simulator*, the *MCI classifier* and the *RL-agent*. The proposed pipeline is illustrated in Figure 3.2. First, the user simulator is trained by unsupervised learning, which simulates the distributed representation of user responses given feasible question inputs. Next, the MCI classifier predicts the patient label based on the averaged distributed representation of its corpus responses. The above two components and dialogue manager comprise the training environment for the RL-agent. The dialogue manager utilizes the user simulator and MCI classifier to handle the state transitions and also computes of the reward based on the ground-truth labels from the training set and MCI classifier prediction. After training in this environment, the RL-agent is able to deliver the optimal sequences of questions for training-set users at various stages of conversations. During testing, the RL-agent produces query inputs to the test-set user simulators, which represent the unseen users. Using these new queries, the user simulators generate the corresponding distributed representation of test-set user responses for MCI

Figure 3.2: Illustration of reinforcement learning components in our proposed approach.



prediction. In the following subsections, we will present each component of the pipeline in detail and demonstrate the effectiveness of the RL framework in improving prediction accuracy while reducing conversational turns.

3.2.3 Construction of Turn-Based Dialogue

Since utterance data was collected in the form of conversational transcripts for each participant, we must reconstructed turn-based dialogue from participant-responses. The participant responses were unstructured while interviewer questions ranged over preset question topics, as illustrated below.

Interviewer: *so what did you do yesterday?*

Participant: *i had yesterday morning i yesterday was a busy day for me. i im forgetting i went to where did i go in the morning. well i went to albertsons yesterday...*

Interviewer: *what do you see in this picture?*

Participant: *we got a picture gosh. it looks like my uncle lou. but he never ...*

Interviewer: *when do you think this picture was taken?*

Participant: *this picture was probably eighteen seventy or something or nineteen twenty. so he looks too old for war he must have been ...*

In total there were well over 150 possible queries from the interviewers. However, for the purposes of this study, we re-compiled the question list into 107 general questions which were ubiquitous across all conversations. A snapshot of questions are in Table 3.1.

Table 3.1: Examples of questions from conversations

Category	**Question**
Activity	Did you go outside lately? So what did you do yesterday?
Social	Did you run into any familiar faces lately? Where did you have dinner?
Picture	What do you see in this picture? Where do you think this picture was taken?
Tech	How are you with the computer? Did you use your computer lately?
Unspecified	<unspecified scheduling comment> <unspecified picture comment>

We created a total of 16 question categories, including: *greetings, activity check, living situation, travel, entertainment, social, picture-related, tech, occupation, hobbies, family, pets, confirmation, clarification, goodbye* and *unspecified* comments. For some of these comments, we delexicalised certain topic words such as "<*activity*>", "<*social topic*>" in order to (1) control for domain expansion [23] and (2) reduce model complexity of our user simulators. In the past, [23] and [22] have shown the effectiveness of delexicalisation in controlling for domain expansion in user simulators without sacrificing the contextual meaning of sentence queries. Additionally, we also created *unspecified comments* category, which included comments that deviated from general question prompts. These comments often result from interviewer follow-up on specific topics mentioned by the user. We consolidated these comments into a single category to distinguish the context-specific from general questions based on the corpus. However, we do demarcate the type of *unspecified comment* used by the interviewer. For example, a follow-up comment to an occupational story is tagged <*unspecified occupational comment*> whereas a follow-up comment about a health concern is tagged <*unspecified health comment*>. The role of these comments serve to build rapport and improve flow of conversation. In future studies we may look to generate user-specific grounding statements for these slots [24]. Implemented in this way, the corpus is tokenized into turn-based responses to questions for each user.

3.2.4 Unsupervised Learning for User Simulator

To effectively capture contextual representation of user conversation style, we utilize vector embedding of user corpus at the sentence-level representation [25, 26]. Given that we want to capture the flow of the conversation from one response to the next, we implement skip-thought embedding, which has shown effectiveness over large corporal datasets by capturing contextual information of sentences given neighboring ones [25]. For encoding sentences, we use a model that was pretrained on the BookCorpus dataset, which contains turn-based conversations from various English novels [25]. For the decoder, we train skip-thought vectors to recover the original response of the user during NLG portion of the pipeline.

Since each user has individual response styles to questions, we train a personalized user-simulator for each user. For each user, the conversation corpus is divided into question-response turns. In our dataset, for example, the number of turns per conversation ranged from 30-275 turns. We used a multilayer perceptron (MLP) with 2 hidden layers of 512 output nodes each to train the user simulator. We also introduce regularization with ℓ_2-norm penalty to constrain model complexity. Because we utilize preset questions by the interviewer, we use one-hot encoding of questions, denoted $\mathbf{q}_t^i \in \mathbb{R}^d$, as input for training. Given the original skip-thought vector $\mathbf{v}_t^i$, the user simulator serves as a function which maps $f : \mathbf{q}_t^i \mapsto \mathbf{v}_t^i$. The output of the MLP is the skip- thought embedding representation of the utterance, denoted $f(\mathbf{q}_t^i; \mathbf{w}_i) \in \mathbb{R}^c$. Here, d denotes the size of our question dictionary, c denotes the dimension of skip-thought embeddings, $\mathbf{w}_i$ parameterizes the MLP model for the given user, $i \in N$ denotes the user index and $t \in T$ denotes the turn number. The loss function of the MLP is given by the mean- squared error (MSE) between the MLP output and original skip-thought vector $\mathbf{v}_t^i \in \mathbb{R}^c$:

$$L(\mathbf{w}_i) = \frac{1}{2} \sum\nolimits_{t=1}^{T} \left[f(q_t^i; \mathbf{w}_i) - \mathbf{v}_t^i\right]^2 + \frac{\lambda}{2}||\mathbf{w}_i||_2, \ \forall\, i = 1, ..., N$$

In the case where questions are not preset, more state-of-art methods such as end-to-end recurrent neural network systems can be deployed to train the user simulator instead [13, 27]. To evaluate the performance of our user simulator, we computed the mean squared error on the outputs of the simulator and the original thought vector representation of the user response for each turn.

3.2.5 Reinforcement Learning Components

Again, let c denote the size of skip-thought embeddings and d denote the size of question dictionary. We formulate the dialogue and task managers portions of the dialogue system into a standard RL setting where an agent interacts with environment $\mathcal{E}$ over a finite number of steps. At time step t, the agent receives a state $\mathbf{s}_t$ and samples an action (asks a question) $\mathbf{a}_t$ based on its current policy π.

The environment transitions to the next state $\mathbf{s}_{t+1}$ and the agent receives a scalar reward r_{t+1}. In this setting, the RL-agent tries to learn an optimal policy π^* over all possible states, including ones that are unseen by the agent during training. To do this, the agent has to learn an approximate action-value function, which maps state-action pairs to expected future rewards [28]. Formally, the action-value function is defined as follows:

$$Q^{\pi}(\mathbf{s}, a) = \mathbb{E}_{\pi}\left[\sum\nolimits_{t=1}^{T} \gamma^t r_t | \mathbf{s}, a\right],$$

where $\gamma \in [0, 1]$ is a discount factor and T is the max # of turns.

3.2.5.1 Environment $\mathcal{E}$

The environment in this case consists of the dialogue manager (DM), user simulator and MCI classifier. DM is composed of the reward and state generating functions. In previous works, a task manager, composed of a database and a query manager [15, 13], is used by the DM to generate observations in retrieval tasks. In our case, however, the the user simulator and MCI classifier is equivalent to the task manager and is used by the DM to generate observations. Here, the DM uses the MCI classifier to (1) predict probabilities for both the MCI and the NL classes based on current moving-average of skip-thought vectors at each turn, and (2) predict the label of the current user at the end of the episode for reward calculation. The result of (1) is also used by the agent as part of its internal state-representation. The result of (2) is used by the DM for credit assignment for the generated conversational episode. The MCI classifier is trained separately on the training set corpus before the dialogue system phase.

3.2.5.2 Action $\mathbf{a}_t \in \mathbb{R}^d$

The RL-agent chooses its actions from a set of discrete actions consisting of 107 predefined questions, where each question is represented by a one hot vector in $\mathbb{R}^d$. It is worth noting that we use $\mathbf{a}_t$ and $\mathbf{q}_t$ to differentiate the action taken by our RL-agent and the questions asked during the actual interviews, respectively.

3.2.5.3 State $\mathbf{s}_t \in \mathbb{R}^C$

The state representation by the RL-agent is used to approximate the action-value function. There are five main components of the state representation vector:

- Skip-thought vector of utterance at current turn: $f(a_{t-1}; \mathbf{w}_i)$, which is the output vector from user simulator f given action a_{t-1} at turn t.
- Moving average of skip-thought vectors across all utterances in current episode: $\bar{f}_t = \frac{1}{t}\sum_{k=1}^{t-1} f(a_k; \mathbf{w}_i)$
- First hidden layer weights of user-simulator: $\mathbf{w}_i[:, 1]$
- Predicted probability of current user for MCI and NL classes by classifier
- Number of turns above threshold: τ.

The total dimension of the state vector is $C = 2c + |\mathbf{w}_i[:, 1]| + 3 = 10115$. At each turn, the DM queries the MCI classifier to output a probability vector composed of $P(y_i = 0|\bar{f}_t)$ and $P(y_i = 1|\bar{f}_t)$, where $y = 0$ denotes NL and $y = 1$ denotes MCI. This 2-dimensional vector keeps track of the classifier's confidence-level for MCI prediction based on the current moving-average of skip-thought vectors generated from $1, 2, ..., t$ turns. Keeping track of classifier confidence incentivizes the

RL-agent to terminate the conversation as soon as it reaches a threshold level of confidence for the prediction task.

3.2.5.4 Reward $r \in \mathbb{R}$

Since we want to minimize the number of dialogue turns, we designed the environment to output a negative reward (-10) at every time step unless it reach a terminal state (e.g. when agent says "goodbye"). At the terminal state, the reward depends on the classification using the averaged skip-thought vector collected from this episodes. If the existing classifier is able to make the correct prediction, the agent receives a positive reward (1000), otherwise it receives a moderately negative reward (-500). We also set the maximum length of episodes $T = 35$. Additionally, we added a linearly increasing penalty for each passing turn where the classifier predicts with ≥ 0.65 probability for either class (MCI/NL). We denote this penalty threshold as the number of turns above confidence threshold (τ). Formally, the reward function is defined as:

$$r = \begin{cases} -10 - 10\tau, & \text{for non-terminal state,} \\ -500, & \text{terminal state with misclassification,} \\ +1000, & \text{terminal state with correct prediction.} \end{cases} \tag{3.1}$$

3.2.5.5 State transitions

The state transition function has two parts:

- *Within User.* The state transition rule between turns is characterized by:

$$\begin{aligned} P^{\pi}_{s,s'} &= \sum\nolimits_{a \in A} P(s_{t+1} = s' | s_t = s, a_t = a, \pi) \\ &= \sum\nolimits_{a \in A} \pi(a|s) P^{a}_{s,s'} \end{aligned}$$

  Given a policy π, the probability of the environment transitioning to state s' at s_{t+1} depends only on current state s_t. Internally, the DM utilizes the user simulator to generate skip-thought $f(a_t; \mathbf{w}_i)$ from a_t.

- *Between Users.* In addition to state transitions within episodes, the state-generating function changes between users, leading to different transition probabilities between similar states among different users. To capture this, we apply two changes when training the RL-agent on multiple users: (1) the first hidden layer weights $\mathbf{w}_i[:, 1]$ of each user are incorporated in the state representation vector so that the RL-agent can distinguish between dissimilar users. When used this way, the user simulator provides a means for the RL-agent to learn similar policies for similar users and dissimilar policies for dissimilar users. (2) During training, both the user simulator and classifier of the training environment is reset between users by re-initializing the user simulator weights $\mathbf{w}_i$ to correspond to the new user.

3.2.5.6 Deep Q-Networks (DQN)

In this work, the action value function needs to estimate expected reward based on the high-dimensional state representations as described in previous section. In order to approximate the action value given different users and the complicated internal state changing during the conversation, we learn a deep Q-network parameterized by θ_v to tackle this challenging problem. The learning procedure can be conducted by optimizing the loss function as follows:

$$L(\theta_v) = \mathbb{E}_{\mathbf{s}_t,\mathbf{a}_t,r_t,\mathbf{s}'_{t+1}}[(y_t - Q(\mathbf{s}_t, \mathbf{a}_t; \theta_v))], \tag{3.2}$$

with

$$y_t = r_t + \gamma \max_{\mathbf{a}'_{t+1}} Q(\mathbf{s}'_{t+1}, \mathbf{a}'_{t+1}; \theta'_v), \tag{3.3}$$

where θ'_v denotes the parameters of target Q-network. In order to learn the estimator under complex situations, two key ingredients were proposed in [29]: *experience replay* and fixed *target Q-network*. During the training, the Q-network (θ_v) is updated in an online fashion by conducting the gradient descent of Eqn. (3.2) while the target Q-network (θ'_v) is fixed to compute the target values as in Eqn. (3.3) and only updated after a certain number of iterations, which is essential to the convergence of Q-network in this work. We also observe when the *experience replay* samples minibatch from previous experiences to update the Q-network, the training performance stabilizes more consistently.

3.2.5.7 Policy-masking

One challenge in our problem is creating an environment that can train the agent to produce responses which best align with the flow of conversations. For example, an agent may learn that the question "*can you elaborate on that?* " is useful for generating a wide distribution of words from the user, but it would not make sense to include that in the first sentence of a conversation or before relevant topics are introduced. To achieve this, we created a policy-modifying function in which *confirmation* and *clarification* type questions are masked from the policy set π at turn t if the action history of the agent from $1, 2, ..., t-1$ does not include any questions from *social*, *activity*, *tech*, *picture-related*, *hobbies*, *occupation*, *travel*, *entertainment* and *family* categories. At each turn, we keep track of an action history vector $\pi_t \in \mathbb{R}^d$ and construct a policy-masking vector $\varphi_t \in \mathbb{R}^d$ to be applied element-wise over the agent's Q-value output. Specifically:

$$\varphi_t^j = \begin{cases} 0, & \text{if action } j \text{ masked,} \\ 1, & \text{otherwise.} \end{cases} \tag{3.4}$$

$$Q'(\mathbf{s}_t) = \varphi_t \odot Q(\mathbf{s}_t).$$

where the φ_t^j denotes the j-th element in policy-masking vector φ_t. And $Q(\mathbf{s}_t) \in R^d$ represents the action values of all 107 available actions given current state $\mathbf{s}_t$. Then the $Q'(\mathbf{s}_t)$ is valid action values vector after the policy masking. To achieve effective masking, we assure the elements of $Q(\mathbf{s}_t)$ is positive by using ReLU [30] as the activation function for the output layer of Q-network and a step of pre-training on Q-network as described in following section.

3.2.6 Training the RL-Agent

We outline below the training procedure for our RL-agent. To expedite the learning process, we first train the RL-agent over the original corpus from the training set. For each user, we perform an initial pass through the entire corpus using the existing action history $q_i^1, q_i^2, ...q_i^T$ to generate episodes $s_1, a_1, r_1, ...a_t, r_t$. We use these corpus-generated episodes to train the Q-estimator network. This initialization procedure is motivated by previous studies which have cited the effectiveness

Algorithm 1 RL-Training Protocol

```
Initialize replay memory $\mathcal{D}$
Initialize Task Manager with classifier
Pre-train action-value function $Q$
for $i = 1, ..., N$ do
    Initialize Environment $\mathcal{E}$ with User Simulator $f_i$
    Initialize $\mathcal{E}$ with true label for user $i$
    for $episode = 1, ..., M$ do
        Reset $\mathcal{E}$ Get the initial state $\mathbf{s}_1$.
        for $t = 1, ..., T$ do
            Obtain policy mask $\varphi_t$ as Eqn. (3.4).
            With probability $\epsilon$ select a random action $a_t$
            otherwise select $a_t = \max_a \varphi_t \odot Q(s_t, a; \theta_v)$
            Execute action $a_t$ in $\mathcal{E}$ observe reward $r_t$ and state $\mathbf{s}_{t+1}$
            Store transition $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{j+1})$ in $\mathcal{D}$
            Sample random minibatch of $(\mathbf{s}_j, \mathbf{a}_j, r_j, \mathbf{s}_{j+1})$ from $\mathcal{D}$
            if terminal $\mathbf{s}_{j+1}$ then
                $y_j = r_j$
            else if non-terminal $\mathbf{s}_{t+1}$ then
                $y_j = r_j + \gamma \max'_a Q(\mathbf{s}_{j+1}, a'; \theta'_v)$
            end if
            Perform a gradient descent step on $(y_j - Q(\mathbf{s}_j; a_j, \theta_v))^2$
        end for
    end for
end for
```

of pre-training with successful episodes so that the RL-agent can discover large terminal reward signals in games with delayed rewards [31].

During training, we stabilize the target Q-network θ'_v for minibatch generation and transfer weights from learning Q-network θ_v every 50 conversational episodes. During testing, we use the RL-agent to generate new actions for each test set user $a^i_1, a^i_2, ...a^i_t$. New episodes are then generated by each user simulator from each new action set π_i for prediction. These simulated episodes often differ from the original corpus in both the questions asked by the agent as well as in the skip-thought responses by the user.

3.3 Experiments

Evaluation of dialogue systems differ widely depending on the task. Previous works typically involve using metrics such as perplexity and averaged reward per response to measure the quality of the natural language generation (NLG) phase of the dialogue system [15, 13, 18]. However, because the utility of our framework comes from the quality of *questions* that the chat-bot generates for the *off-conversational task*, we propose a framework of evaluation which emphasizes the agent's off-conversation performance. We gauge utility of the dialogue system by its ability to improve (1) prediction accuracy against baseline techniques and (2) the number of turns needed to make accurate prediction.

3.3.1 Data

Data used for this study was obtained from a randomized controlled behavioral clinical trial to ascertain the effect of unstructured conversation on cognitive functions. Details of the study protocol was explained in [32]. In this clinical study, conversational data was collected in Q&A format for each participant during web-cam interviews with trained interviewers. Each participant was interviewed multiple times over the course of 4-6 weeks, and dialogue responses were transcribed for each interview session [12]. On average, there are 2.81 conversational episodes per participant, and each conversation lasted between 30-45 minutes [12, 32]. MCI labels were generated using clinical assessment of each participant's cognitive status by medical professionals [12, 32].

3.3.2 Baselines vs. RL Performance

We first compare the performance of several baseline classifiers for the MCI prediction task. For our specific dataset, [12] had previously achieved benchmark performance of 72.5% AUC score on 5-fold validation while using linear SVM with ℓ_1-norm penalty and feature engineering by Linguistic Inquiry and Word Count (LIWC) dictionary [12]. LIWC embeds each word into a 69-dimensional word vector space with each dimension representing a latent feature of the English language [12]. Since 2013, various contextual representations of words and sentences have been proposed, many of which have outperformed classical rule-based contexual embedding techniques [26, 25]. Distributed representation such as *Word2Vec* allows for more flexible and corpus-dependent latent features to be created for individual words [26]. More recently, Skip-thought vectors [25] have risen to popularity

Table 3.2: Performance of baseline vs. RL on MCI prediction on 10 stratified shuffle splits

Model	Feature	AUC	Sen.	Specificity	F1-Score
LR + ℓ_1	RD	0.529 ± 0.132	0.380 ± 0.260	0.678 ± 0.105	0.361 ± 0.207
RFC	RD	0.519 ± 0.057	0.080 ± 0.098	0.944 ± 0.075	0.120 ± 0.149
SVM + ℓ_1	RD	0.551 ± 0.131	0.380 ± 0.227	0.722 ± 0.102	0.384 ± 0.214
SVM + ℓ_2	RD	0.560 ± 0.050	0.320 ± 0.256	0.800 ± 0.185	0.322 ± 0.193
MLP	RD	0.640 ± 0.193	0.110 ± 0.243	0.860 ± 0.189	0.162 ± 0.146
LR + ℓ_1	W2V	0.638 ± 0.091	0.520 ± 0.204	0.756 ± 0.147	0.517 ± 0.127
RFC	w2v	0.564 ± 0.110	0.340 ± 0.220	0.789 ± 0.144	0.374 ± 0.189
SVM + ℓ_1	W2V	0.651 ± 0.103	0.560 ± 0.233	0.756 ± 0.130	0.541 ± 0.147
SVM + ℓ_2	W2V	0.598 ± 0.116	0.440 ± 0.233	0.756 ± 0.171	0.449 ± 0.205
MLP	W2V	0.680 ± 0.151	0.500 ± 0.500	0.511 ± 0.490	0.266 ± 0.266
LR + ℓ_1	LIWC	0.703 ± 0.099	0.540 ± 0.237	0.867 ± 0.130	0.584 ± 0.152
RFC	LIWC	0.641 ± 0.135	0.360 ± 0.250	0.922 ± 0.087	0.445 ± 0.273
SVM + ℓ_1	LIWC	0.661 ± 0.125	0.600 ± 0.200	0.722 ± 0.200	0.572 ± 0.144
SVM + ℓ_2	LIWC	0.712 ± 0.110	0.680 ± 0.204	0.744 ± 0.180	0.631 ± 0.135
MLP	LIWC	0.689 ± 0.129	0.300 ± 0.458	0.767 ± 0.396	0.182 ± 0.285
LR + ℓ_1	SKP	0.790 ± 0.112	0.680 ± 0.256	0.900 ± 0.116	0.707 ± 0.183
RFC	SKP	0.608 ± 0.104	0.260 ± 0.220	0.956 ± 0.054	0.343 ± 0.259
SVM + ℓ_1	SKP	0.783 ± 0.123	0.700 ± 0.241	0.867 ± 0.171	0.711 ± 0.190
SVM + ℓ_2	SKP	**0.797±0.122**	**0.660±0.269**	**0.933±0.102**	**0.716±0.189**
MLP	SKP	0.638 ± 0.138	0.600 ± 0.490	0.400 ± 0.490	0.316 ± 0.256
RL(T=1)	SKP	0.607±0.109	0.380±0.166	0.833±0.134	0.447±0.172
RL(T=3)	SKP	0.706±0.092	0.500±0.205	0.911±0.097	0.583±0.154
RL(T=5)	SKP	0.707±0.072	0.480±0.133	0.933±0.102	0.594±0.129
RL(T=10)	SKP	0.772±0.115	0.600±0.237	0.944±0.102	0.683±0.186
RL(T=15)	SKP	0.798±0.115	0.640±0.265	0.956±0.102	0.714±0.190
RL(T=20)	SKP	0.798±0.121	0.640±0.250	0.956±0.102	0.719±0.190
RL(T=25)	SKP	0.808±0.111	0.660±0.254	0.956±0.102	0.732±0.184
RL(T=30)	SKP	0.808±0.119	0.660±0.269	0.956±0.102	0.730±0.190
RL(T=35)	SKP	**0.818±0.102**	**0.680±0.204**	**0.956±0.102**	**0.761±0.140**

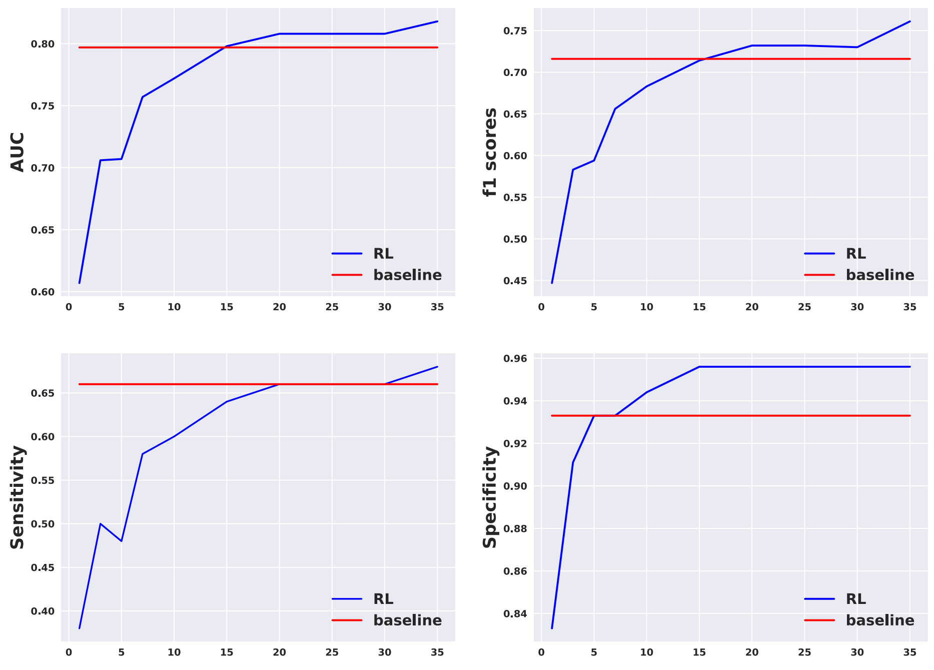
Here, *LR* denotes sparse logistic regression classifier, *RFC* denotes random forest classifier, *SVM* denotes support vector machines, and *MLP* denotes multi-layer perceptron. For feature representation of corpus, *RD* represents raw distribution of word counts. *w2v* denotes averaged 300-dimension *Word2Vec* embeddings across all words appearing in the corpus for each user [26]. *LIWC* denotes the original rule-based embedding used by [12]. *SKP* denote averaged 4800-dimension Skip-Thought vectors across all turn-based responses for each user [25].

due to the ability to embed entire sentences into "thought vectors" that capture contextual meaning and syntactic information from neighboring sentences. For this reason, we compare various word and phrase embedding techniques to establish new baseline performances for our classification task.

The first four sections of Table 5.3 show the performance of these baseline classifiers. Using the original LIWC representation, we were able to recover close to the 72.5% AUC baseline from the original paper using SVM and LR classifiers. When implementing skip-thought embedding, we used pre-trained skip-thought encoders by [25] to embed each user response across all conversational turns. The encoder was pre-trained on the *BookCorpus* dataset, which is a large collection of novels pertaining to numerous literary genres. The advantage of pre-training on this dataset is that *BookCorpus* contains an abundant number of turn-based dialogues between various character types. These conversations capture a wide range of conversational response styles, idiosyncrasies and temperaments. As seen in Table 5.3, the best performing baseline model was the SVM classifier with ℓ_2 norm, using Skip-Thought embedding as features. For this reason, we choose this classifier for the RL portion of our pipeline. As a baseline reference, we also included performance using raw word count distributions for all models.

We then evaluate the performance of our RL-agent across 10 stratified shuffle splits. Each split uses 65% of data for training and 35% for testing. We compare the performance of RL-Agent when manually restricting the number of questions to 1, 3, 5, 7, 10, 15, 20, 25, 30 and 35. By restricting the number of turns, we can observe the number of questions needed to recover the original baseline performance using the SVM classifier.

Figure 3.3: RL-Agent vs. Baseline w/ Variation on Turns.



The last section of Table 5.3 illustrates the performance of the RL-agent under various turn constraints. Here, the turns notation RL(T=t) denote the number of questions the agent is allowed to ask before a prediction is made from the simulated user responses. It is important to note that turn 0 was set to greetings by default and was not counted toward the conversation.

We see from constraint conditions that the performance of our RL-agent started to surpass baseline performances starting at 25 questions and was able to achieve comparable performance using only 15 questions. At full conversation length of 35 turns, we were able to achieve 0.818 AUC, an improvement upon current and previous baselines. In comparison, the mean number of conversational turns per user in the original corpus was 105.71. Additionally, since 2.81 conversations were conducted per user, we adjusted the number of turns allowed based on the mean number of turns *per conversation*, which was 37.36 per user. For this reason, we set the upper bound constraint to 35 questions, which is just slightly less than a full conversation with the user.

Figure 3.3 visualizes this relationship between performance and number of questions asked by the RL-agent. We see that performance improvements with additional questions saturate after 15 questions. This was expected, as the highest-yield questions discovered by the RL-agent were asked first during test conversations.

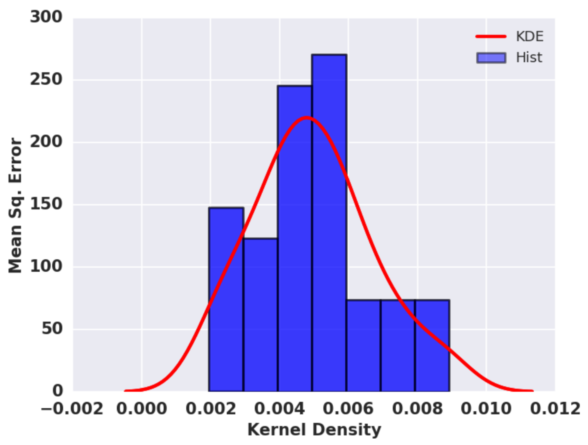
3.3.3 Evaluation of User Simulators

User simulators serve a pivotal role of simulating the user response in the RL training environment [18, 33]. In previous works, the user simulators are evaluated based on accuracy of generated user query to unseen bot responses [33, 18]. Metrics such as BLEU and perplexity are used at the NLG phase of dialogue, as the generation of user query is pivotal in retrieval-type training systems.

In our case, however, the goal of the user simulator is quite different; the RL-agent is responsible for generating queries while the output from the user simulator is actually *an encoded thought-vector* of the user response, which is then used for state representation and downstream prediction purposes. For this reason, we evaluate the performance of the user-simulator not on the decoding portion of the dialogue system, but rather on the performance of the user-simulator in generating accurate thought-vector version of the responses.

We compute mean-squared error (MSE) between the corpus Skip-Thought vector and user

Figure 3.4: Distribution of mean squared error (MSE) across all user simulators.



simulation prediction at each turn. The resulting MSE scores are averaged across all turns for the conversation. Given that each user has on average 2.81 conversations, we evaluate the performance of the user simulator in a leave-one-out fashion: for each user, the simulator is trained on all conversations except for the last one, which is used for evaluation. Figure 3.4 visualizes the performance of user simulators. The mean MSE is 0.00495±2.93E-06, averaged across all test set performances.

3.3.4 Top-Performing Policies

It is interesting to note that the simulated episodes by our RL-agent were able to provide a performance boost for the prediction task. In this section, we look qualitatively at the types of questions at 5, 10, 15, 20 and 35 turns by RL-agent in comparison with the original corpus. We also compare the performance of π^*@5, @10, @20, @30 and @35 with the performance using the first 5, 10, 20, 30 and 35 responses of the original corpus. Again, we note that responses to greeting and parting queries such as "*Hi*" and "*goodbye*" are not counted toward prediction.

As shown in Table 3.3, the optimal policy π^* learned by our framework outperformed the original corpus for each turn constraint. For example, when our RL-agent asked only 5 questions to test set users, the classifier was able to achieve 0.707 AUC and 0.594 F1 using the simulated response. In contrast, using the first 5 questions from the original corpus for each test set user produced 0.504

Table 3.3: Prediction @5, 10, 20, 30 and 35 Turns

Model	**AUC**	**Sen**	**Spec**	**F1-Score**
Corpus@5	0.504±0.070	0.120±0.098	0.889±0.099	0.175±0.145
Corpus@10	0.513±0.076	0.160±0.174	0.867±0.130	0.193±0.200
Corpus@20	0.614±0.077	0.340±0.254	0.889±0.131	0.382±0.223
Corpus@30	0.658±0.121	0.360±0.233	0.956±0.056	0.460±0.266
Corpus@35	0.699±0.125	0.420±0.244	0.978±0.044	0.539±0.248
π^*@5	0.707±0.072	0.480±0.133	0.933±0.102	0.594±0.129
π^*@10	0.772±0.115	0.600±0.237	0.944±0.102	0.683±0.186
π^*@20	0.798±0.121	0.640±0.250	0.956±0.102	0.719±0.190
π^*@30	0.808±0.119	0.660±0.269	0.956±0.102	0.730±0.190
π^*@35	**0.818±0.102**	**0.680±0.204**	**0.956±0.102**	**0.761±0.140**

AUC and 0.175 F1. When using the first full-length conversation with 35 turns, the original corpus recovers an AUC score of 0.699, which is far from the performance of π^*@35. In Table 3.4, we rank the most frequently appearing questions in π^*@5, π^*@10 and π^*@20.

Table 3.4: Most frequently questions in π^*@5, 10, 15 and 20

Turns	Question	Count
1-5	when did you start working?	40
1-5	so how long did you go out for?	37
1-5	when did you meet your SO?	28
1-5	<unspecified hobby comment>	24
1-5	what did you like about <activity>?	24
6-10	what was <occupation> like for you?	30
6-10	<unspecified tech comment>	28
6-10	when did <tech problem> start?	22
6-10	what do you see in this picture?	19
6-10	<unspecified picture comment>	19
10-15	what is your opinion on <social topic>?	42
10-15	did you see any shows lately?	38
10-15	how many people do you think can fit in this?	33
10-15	what you were doing during this time period?	30
10-15	what type of <hobby> do you do?	28
15-20	<goodbye>	27
15-20	where did you meet your so?	25
15-20	did you enjoy school?	24
15-20	anyone visit you lately?	24
15-20	what was the show about?	20

π^***@5.** The most effective question in π^*@5 appears to be "*when did you start working*". In the context of our problem, this question seems to generate the most polarizing responses from the cohort. We also see that the RL-agent included a few elaboration questions such as "*what did you like about <activity>*" and "*why did you do that*," for some users to expand upon previous responses. From the clinical perspective, it is also interesting to note that the RL-agent picks questions such as "*what did you do yesterday*" and "*how long did you go out for*," which are similar to questions used clinically to assess immediate recall in MCI patients [34].

π^***@10.** As seen in π^*@5, occupational questions were the most popular topic asked by the RL-agent. This is also the case with π^*@10, where the RL-agent follows up the previous query with an elaboration question regarding past occupational experiences. It is interesting to note that the RL-agent transitions to picture-related questions, which are often used by the clinical interviewers to facilitate creative responses by participants [12].

We also observe the RL-agent asking questions such as "*<unspecified tech comment>*" and "*when did <tech problem> start*". These were frequently asked questions during the course of the original dialogue, as technical difficulties were often encountered with connection and webcam issues during the interviews [12]. Unfortunately, the responses vary greatly and may at times generate verbose responses from participants. The RL-agent did not seem to be able to recognize this caveat during training.

π^***@20.** As we approach questions 11 through 20, we arrive at mid- to late- dialogue for most conversations. Overall, we observe more widespread topics during this portion of conversation. The most polarizing question asked at this stage was "*what is your opinion on <social topic>?*" Here, we used delexicalised slots [22] <social topic> to reduce model complexity, but the slots may be substituted with a wide range of social topics from political trends to recent news.

Additionally, we observe that the RL-agent learns to say "*goodbye*" to terminate the conversation early in numerous cases. As mentioned previously, we designed the state function to include the predicted probability [0.0-1.0] of MCI by the classifier at each time-step. The environment penalizes the agent for additional turns in which the prediction probability exceeds 0.65 for either class. By opting to terminate the episode, the RL-agent learns to avoid dragging on the dialogue unnecessarily in cases where it is confident in the prediction.

One notable question in π^*@20 is "*how many people do you think can fit in this?*" This is actually a picture-specific question related to one of the more provocative pictures. In fact, we confirmed from the original corpus that it generated more follow-up response from users when compared to other picture-related questions such as "*when do you think this picture was taken?*" and "*interesting, what makes you say that?*". By ranking this question highly, the RL-agent indirectly prioritizes this picture over others in generating user responses. This exemplifies how the ranking of questions by π^* may be used to direct future data collection process.

π^*@35. When approaching the end of conversations, we notice that the questions asked by the agent were more spread-out among the remaining choices. For this reason, we rank only the top 10 questions during the final 15 turns of simulated conversations.

Table 3.5: Table of top 10 ranked questions in final 15 turns of conversations

Rank	Question	Count
1	what is your opinion on using <new tech>?	112
2	did you do anything else?	106
3	so how long did you go out for?	98
4	what you were doing during this time period?	95
5	when do you think this picture was taken?	95
6	<goodbye>	94
7	anything new with you lately?	91
8	what did you like about it?	85
9	<unspecified picture comment>	76
10	how often do you <do activity>?	72

In this latter portion of π^*, we note that the RL-agent utilized more elaboration questions such as "*what do you like about it*" and "*how often do you <do activity>*". We also see that technology-related questions such as "*what is your opinion on using <new tech>*" are included more often when compared to topics such as occupation or social items. This indicates that tech-related questions may not be as high-yield in distinguishing MCI responses, as these questions are prioritized later during conversation by the RL-agent.

3.4 Discussion and Conclusion

In this chapter, we introduce a RL framework for approaching a classically supervised learning problem in clinical medicine, where the data is often noisy, scarce, and prohibitively expensive to obtain. We show that a properly trained RL framework can (1) greatly cut down on the amount of data needed to make accurate predictions, and (2) synthesize relevant new data to improve performance.

To achieve this framework, we proposed a multi-step approach which capitalizes on the vast existing knowledge of the human language and NLP research. First, we used a state-of-art distributed representation to preprocess our data. We then set up a simulation environment for reinforcement learning using supervised learning to create customized user simulators. Lastly, we utilize the trained RL-agent to generate new questions from π^* to obtain more targeted responses for our prediction task.

A careful examination of the optimal policies discovered by our agent demonstrates that the overall framework is self-contained for directing dialogue generation for diagnostic screening, which can potentially replace the need for trained interviewers. Our trained RL-agent is able to discover relevant questions to ask users where the agent has no prior experience of interaction. We also show various clinical insights which could be deduced from observing the ranking of questions in π^* at various turn constraints.

In order for this framework to be effectively deployed in a realistic setting, a user-simulator that could be trained online and in real-time should be considered. In its current form, our user-simulators are trained offline, which may not be scalable to larger corpus and user volumes. Additionally, a natural language generator phase may be needed to make the questions more adaptable to the natural flow of human conversation. These will be areas of research we will explore in future studies.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] J Olazaran, Rubén Muñiz, B Reisberg, J Peña-Casanova, T Del Ser, AJ Cruz-Jentoft, P Serrano, E Navarro, ML García de la Rocha, A Frank, et al. Benefits of cognitive-motor intervention in mci and mild to moderate alzheimer disease. *Neurology*, 63(12):2348–2353, 2004.

[2] Jeffrey L Cummings, Rachelle Doody, and Christopher Clark. Disease-modifying therapies for alzheimer disease: challenges to early intervention. *Neurology*, 69(16):1622–1634, 2007.

[3] Serge Gauthier, Barry Reisberg, Michael Zaudig, Ronald C Petersen, Karen Ritchie, Karl Broich, Sylvie Belleville, Henry Brodaty, David Bennett, Howard Chertkow, et al. Mild cognitive impairment. *The lancet*, 367(9518):1262–1270, 2006.

[4] Keith A Johnson, Nick C Fox, Reisa A Sperling, and William E Klunk. Brain imaging in alzheimer disease. *Cold Spring Harbor perspectives in medicine*, 2(4):a006213, 2012.

[5] D Heister, James B Brewer, Sebastian Magda, Kaj Blennow, Linda K McEvoy, Alzheimer's Disease Neuroimaging Initiative, et al. Predicting mci outcome with clinically available mri and csf biomarkers. *Neurology*, 77(17):1619–1628, 2011.

[6] Clifford R Jack Jr, David S Knopman, William J Jagust, Leslie M Shaw, Paul S Aisen, Michael W Weiner, Ronald C Petersen, and John Q Trojanowski. Hypothetical model of dynamic biomarkers of the alzheimer's pathological cascade. *The Lancet Neurology*, 9(1):119–128, 2010.

[7] Liang Zhan, Yashu Liu, Yalin Wang, Jiayu Zhou, Neda Jahanshad, Jieping Ye, and Paul Matthew Thompson. Boosting brain connectome classification accuracy in alzheimer's disease using higher-order singular value decomposition. *Frontiers in neuroscience*, 9:257, 2015.

[8] Qi Wang, Liang Zhan, Paul M Thompson, Hiroko H Dodge, and Jiayu Zhou. Discriminative fusion of multiple brain networks for early mild cognitive impairment detection. In *ISBI*, pages 568–572. IEEE, 2016.

[9] Carol Dillon, Cecilia M Serrano, Diego Castro, Patricio Perez Leguizamón, Silvina L Heisecke, and Fernando E Taragano. Behavioral symptoms related to cognitive impairment. *Neuropsychiatric disease and treatment*, 9:1443, 2013.

[10] Robert M Chapman, Mark Mapstone, John W McCrary, Margaret N Gardner, Anton Porsteinsson, Tiffany C Sandoval, Maria D Guillily, Elizabeth DeGrush, and Lindsey A Reilly. Predicting conversion from mild cognitive impairment to alzheimer's disease using neuropsychological tests and multivariate methods. *Journal of Clinical and Experimental Neuropsychology*, 33(2):187–199, 2011.

[11] Hiroko H Dodge, Nora Mattek, Mattie Gregor, Molly Bowman, Adriana Seelye, Oscar Ybarra, Meysam Asgari, and Jeffrey A Kaye. Social markers of mild cognitive impairment: Proportion of word counts in free conversational speech. *Current Alzheimer Research*, 12(6):513–519, 2015.

[12] Meysam Asgari, Jeffrey Kaye, and Hiroko Dodge. Predicting mild cognitive impairment from spontaneous spoken utterances. *Alzheimer's & Dementia: Translational Research & Clinical Interventions*, 3(2):219–228, 2017.

[13] Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*, 2016.

[14] Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. End-to-end reinforcement learning of dialogue agents for information access. *arXiv preprint arXiv:1609.00777*, 2016.

[15] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter*, 19(2):25–35, 2017.

[16] Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. Reinforced mnemonic reader for machine reading comprehension. *arXiv preprint arXiv:1705.02798*, 2017.

[17] Wei He, Kai Liu, Jing Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, et al. Dureader: a chinese machine reading comprehension dataset from real-world applications. *arXiv preprint arXiv:1711.05073*, 2017.

[18] Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *Knowledge Engineering Review*, 21(2):97–126, 2006.

[19] Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, 16:105–133, 2002.

[20] Xuijun Li, Yun-Nung Chen, Lihong Li, and Jianfeng Gao. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*, 2017.

[21] Antoine Bordes and Jason Weston. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*, 2016.

[22] Bing Liu and Ian Lane. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. *arXiv preprint arXiv:1708.05956*, 2017.

[23] Matthew Henderson, Blaise Thomson, and Steve Young. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *SLT*, pages 360–365. IEEE, 2014.

[24] Joyce Y Chai, Rui Fang, Changsong Liu, and Lanbo She. Collaborative language grounding toward situated human-robot dialogue. *AI Mag*, 37(4), 2016.

[25] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.

[27] Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*, 2016.

[28] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, pages 630–645. Springer, 2016.

[31] Charles W Anderson, Minwoo Lee, and Daniel L Elliott. Faster reinforcement learning after pretraining deep networks to predict state dynamics. In *IJCNN*, pages 1–7. IEEE, 2015.

[32] Hiroko H Dodge, Jian Zhu, Nora C Mattek, Molly Bowman, Oscar Ybarra, Katherine V Wild, David A Loewenstein, and Jeffrey A Kaye. Web-enabled conversational interactions as a method to improve cognitive functions: Results of a 6-week randomized controlled trial. *Alzheimer's & dementia: translational research & clinical interventions*, 1(1):1–12, 2015.

[33] Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*, 2016.

[34] Marshal F Folstein, Susan E Folstein, and Paul R McHugh. "mini-mental state": a practical method for grading the cognitive state of patients for the clinician. *Journal of psychiatric research*, 12(3):189–198, 1975.

CHAPTER 4

SAMPLE EFFICIENT LEARNING WITH PRIVILEGED INFORMATION

4.1 Introduction

In classical supervised learning, the learner is presented with the training tuple $\{(x_i, y_i)\}_{i=1}^{m}$ and performs an optimization task of finding the best model in a hypothesis space $h : X \to Y$ to approximate some true $f : X \to Y$ which explains the data. Given a new task, knowledge transfer [1] is often applied to accelerate the learning process by distilling and transferring relevant knowledge from previous tasks to the unseen one. Under classical formulations, the learner incorporates prior information in one of several ways:

- Direct transfer of parameters from old hypothesis models to the new task and *fine-tuning* [1] the parameters.
- Learning multiple tasks (online or batched) related to the current task [2, 3].
- Using the prior knowledge (i.e. a knowledge graph) to constrain the hypothesis space by regularization [4].
- Using representations (i.e. *embeddings*) of X and / or Y from previous tasks for new tasks [5, 6].
- Accelerate learning rate and model compression by Distillation as typically seen in Teacher-Student models [7].

In each of these settings, knowledge transfer operates directly within the X, Y and $\mathcal{H}$ spaces to improve generalization of information from old models to the new task.

Recently, *Learning Using Privileged Information* (LUPI) [8] has provided a new paradigm for knowledge transfer. Under LUPI, the learner now interacts with a Teacher who provides *privileged information* (PI), which is available exclusively at training time. From the learner's perspective, the training set is now extended to the tuple $\{(x_i, x_i^*, y_i)\}_{i=1}^m$, and the test set stays the same. Some examples of PI include: 1) Future information that relates X and Y. For example, using future stock prices beyond the prediction window during training. 2) Auxiliary information describing the label space that is available only to a subset of samples. For example, physician notes that accompany diagnostic predictions which is only available after the diagnosis is made.

At a high-level, PI provides some similarity information between training samples from the original feature space, and the Teacher hypothesis serves as additional "explanations" of the hypothesis space [9, 8]. As a result, [9] showed that the LUPI Teacher provides a principled way to improve the *generalization error* of Student learners using agnostic PAC models, providing some theoretical improvements in the number of samples required to generalize to test set data (i.e. improves sampling efficiency).

However, under the current state-of-art LUPI formulations such as [10] and [11], PI is incorporated by means of support vectors and dropout schemes, both of which fail to explore the underlying similarity structure between examples in the PI space X^*. For example the mode distribution and pairwise similarity between points in the X^* space is largely unused. The PI contributes as auxiliary training features and kernel information, but much of the LUPI information is lost at inference time and beyond. A significant question remains: **can privileged information be *retained* for future tasks?**

Ideally, we want the LUPI Teacher to incorporate PI in a way that is specific enough to inform similarity between training samples yet general enough to be retained across future tasks. As a motivating example, consider the medical setting, where electronic health records (EHR) are often sparse, noisy, and full of missing data. Complex tasks such as multi-task learning of many diseases are often difficult to do because of the *long-tail property of diseases* – that is, diseases with very few training samples (i.e. < 100) are very difficult to learn using EHR features alone. On the other hand, medical research on rare diseases are often plenty – large volumes of clinical journals focus on text descriptions of rare diseases in the medical setting. As a result, clinical texts such as discharge notes are *unavailable* at inference time, but when used retrospectively during training can serve as a source of PI that allows for rare diseases to be learned with few examples.

In this work, we propose a LUPI formulation that achieves precisely this. First, we introduce the idea of building a *vocabulary* of PI features by unsupervised learning using external data sources. We then propose a mechanism for learning a joint representation between the PI information and the original set of example features by exploiting their co-occurrence statistics in the training data. We finally learn a *shared* decision function using a contrastive-loss to distinguish between samples drawn from the joint latent space based on their labels for each task. In experiments, we demonstrate the effectiveness of our method in retaining PI obtained from external data sources to support multi-task prediction tasks in the EHR setting against other transfer learning methods. We demonstrate that such an approach both improve the prediction accuracy as well as decrease the samples required to train an accurate model, especially for rare-diseases.

4.2 Methodology

At a high-level, the main intuition behind our proposed method is to decompose the LUPI process into three parts:

1. Build a dictionary of PI features and learn a distributed representation [5] over the PI vocabulary.
2. Find a joint representation space (Φ) between the PI and example features.
3. Jointly learn the decision functions $h^* : \Phi \rightarrow Y$ by feature-matching in the joint representation space.

The first process uses unsupervised learning to embed the PI vocabulary into a vector space. The second process allows for some of the privileged information to be *retained* at inference time, despite not having direct access to the PI vectors. The third process allows for PI information for one task to be transferred for other $C - 1$ tasks in the label space. In the following subsections, we will examine how to achieve (1) – (3) in detail. We also provide analysis of how (3) can maintain the favorable LUPI sample efficiency.

4.2.1 Building the PI vocabulary

First, we can define $g^*(w_j;\theta_{g*})$ as an embedding function that maps $g^* : X^* \to \Phi$. Note that x_i^* consists of individual words, $\{w_1, ..., w_k\}$. So $g^*(w_j;\theta_{g*})$ embeds each individual word in the PI vocabulary rather than the PI samples (i.e., x_i^*). The rationale behind $g^*(.)$ is to encode each word in the PI vocabulary into a vector space so vector operations can be applied to the PI. We specifically consider embedding function of the form,

$$g^*(w_j;\theta_{g*}) = w_j^T\theta_{g*}. \tag{PI Embedding}$$

Since each $w_j \in \{0,1\}^d$ has $w_{jk} = 1$ only when $j = k$, the w_j vector simply selects the j^{th} column in θ_{g*}. We restrict $\theta_{g*} \in \mathbb{R}^{d\times k}$ so that θ_{g*} produces a lower-dimensional representation of each word in the PI vocabulary. For this first step, we do not restrict the PI to come from the original dataset $\{(x_i, x_i^*, y_i)\}$. In fact, we can learn the *embedding* θ_{g*} for our PI using any data source by applying the following word-model:

$$Score(w_1, w_2) = \begin{cases} 1 & \text{if } w_1, w_2 \in x_i^* \\ \\ 0 & \text{otherwise} \end{cases} \tag{Co-occurence}$$

$$g(w_1, w_2) = \sigma\{(\theta_{g*}w_1)^T(\theta_{g*}w_2)\} \tag{Word Model}$$

$$\mathcal{L}_{emb} = BCE(Score(w_1, w_2), g(w_1, w_2)). \tag{2}$$

Here, $\sigma(.)$ denotes the Sigmoid activation function, and $BCE(.)$ denotes the binary cross-entropy loss: $-\sum_i[a_i(logb_i) + (1-a_i)log(1-b_i)]$. The cross-term $(\theta_{g*}w_1)^T(\theta_{g*}w_2)$ gives the *similarity* between w_1 and w_2 in the embedding space, which is then scored against $Score(w_1, w_2)$ based on whether w_1, w_2 both appear in x_i^*. We note that the embedding loss $\mathcal{L}_{emb}$ is trained separately from the rest of the LUPI model since it is not specific to the dataset.

4.2.2 Learning the Joint Representation

Next, let us define $g(x_i;\theta_g)$ to be the embedding function that maps $g : X \to \Phi$. In this work, we consider the following for $g(.)$ and $g^*(.)$:

- *Distributed representation* of PI vocabulary, which captures its underlying manifold structure and is obtained by unsupervised learning [5].

- Encoding of X into a fixed length vector by deep embedding methods such as [12].

For time-series X, we take the embedding functions to be a recurrent encoder neural network:

$$g(x_i; \theta_g) = RNN(x_i; \theta_g) \qquad \text{(EF Embedding)}$$

The motivation behind using the embedding functions $g(x)$ and $g^*(w)$ is to extend the idea of *Student and Teacher kernels*, which allow for the privileged information to provide information about similarity between training samples in the feature space [9]. Using neural encoding for $g(x; \theta_g)$ allows such feature spaces to be represented by a fixed-length vector without losing the underlying spatio-temporal information.

To find commonality between example features and PI, we introduce a *matching function* (μ) that maps each $(g(x_i), g^*(w_j))$ pair onto the interval $[0, 1]$, i.e., $\mu : \Phi \times \Phi \to [0, 1]^d$:

$$\mu(g(x_i), g^*(w_j); A) = \frac{\exp(\max\{0, [g(x_i); g^*(w_j)]^T A\})}{\sum_{p=1}^{d} \exp(\max\{0, [g(x_i); g^*(w_p)]^T A\})}. \tag{3}$$

Here, $[g(x_i); g^*(w_j)] \in \mathbb{R}^{2k}$ denotes the concatenation of the $g(x_i)$ and $g^*(w_j)$ embeddings in the joint latent space. The parameter matrix A projects the pairs $(g(x_i), g^*(w_j))$ onto $\mathbb{R}$, and the softmax activation normalizes the pairwise scores against other word-pairs in the PI vocabulary. Thus, for each sample mapped from the feature space X, the matching function μ produces a set of corresponding weights over all of the words in the PI vocabulary.

We make the key observation that for each word w_j, the output weight of the matching function should correspond to the j^{th} component of the x_i^* sample in the training data. That is, $\mu(g(x_i), g^*(w_j); A) \approx x_{ij}^*$. Using this fact, we can learn the matching function by minimizing over the following objective:

$$\begin{aligned}\mathcal{L}_\phi(\theta_g, \theta_{g^*}, A) = -\frac{1}{md} \sum_{i=1}^{m} \sum_{j=1}^{d} [&x_{ij}^* \log \mu_i(g(x_i), g^*(w_j); A) \\ &+ (1 - x_{ij}^*) \log\, (1 - \mu_i(g(x_i), g^*(w_j); A))].\end{aligned} \tag{4}$$

Each component of the PI vector x_i^* can thus be interpreted as providing an indicator label for likelihood of the (x_i, w_j) pair to occur together. The similarity control mechanism highlighted in Eqn. 4 differs from the Kernel-matching mechanism mentioned previously in [9]. The limitation of kernel-matching [9] is that two sets of Kernel weights need to be learned simultaneously: α for $K(x_i, x)$ and β for $K^*(x_i^*, x^*)$. By contrast, our joint representation for x_i and x_i^* encourages a single hypothesis model to be used to map $h^* : \Phi \rightarrow Y$. Since matrix A captures the $g(x_i)$ and $g^*(w_j^*)$ interactions, it preserves the PI in the space of Φ and allows relevant PI to be retrieved at test time.

Finally, we obtain the *augmented representation* of x_i as a weighted combination of $\mathcal{G}^* = \{g(w_j)\}_{j=1}^d$ and $g(x_i)$:

$$\phi(x_i) = g(x_i) + \sum_{j=1}^{d} \mu_i(g(x_i), g^*(w_j); A) \cdot g^*(w_j) \tag{5}$$

One can think of $\mathcal{G}^*$ as the set of *basis vectors* supporting the PI space (similar to *frames* for the PI Kernel [9]). The augmented representation $\phi(x_i)$ contains both information from the original x_i as well as relevant information retrieved from $\mathcal{G}^*$. Note that since the PI vectors $x_j^* \in X^*$ are not directly used at testing time, each sample $x_i \in \mathcal{D}_{test}$ is mapped into Φ using $g(.)$, and the trained $\mu(.)$ selects the corresponding bases in $\mathcal{G}^*$ to construct ϕ_i.

This is quite different from *representation fusion* methods [6, 13], which only try to learn a shared representation space for input modalities $X_1, ..., X_k$, without a matching function to control the contribution of each modality to the hypothesis. For example, we can take X_1 to be the original feature space and X_2 to be the privileged information. At test time, when X_2 is unavailable, X_1 inputs with masked X_2 components may be projected into a completely different location in the shared representation space than if the X_2 information were available. Furthermore, *model fusion* methods [13] may also under-utilize the original feature space during training, as the PI contain more information related to the target task.

4.2.3 Coupling Decision Functions with Feature Matching

LUPI typically considers 2 hypothesis functions: the Student hypothesis $h : X \longrightarrow Y$, and Teacher hypothesis $f^* : X^* \longrightarrow Y$. Since we have already addressed the problem of finding a common "frame of reference" between the original feature space and the PI space by the matching function μ, the main focus for this portion of our method has to do with finding an efficient f^* that relates the privileged information to the labels. Fortunately, we can directly approximate $f^* : X^* \longrightarrow Y$ by a function $h^* : \Phi \longrightarrow Y$ that maps samples from the joint representation space to the label space. This is because Φ is constructed by embedding function $g^*(.)$ on X^* and is an approximation of the Kernel space for the privileged information.

In the case that the target task is *classification*, we can formulate h^* as a *feature matching* problem between samples from Φ and Y. Specifically, we can use contrastive loss [14] to find an invariant representation $h^* : \Phi \longrightarrow Y$ and vice versa, by minimizing the distance between similar samples drawn from the joint embedding space based on signals from the label space:

$$L(W, y_i, x_i^+, x_i^-) = (1 - y_i)\mathcal{M}_W(x_i^+, x_i^-) + (y_i)(\max\{0, C - \mathcal{M}_W(x_i^+, x_i^-)\}) \tag{Contrastive}$$

$$\mathcal{L}(W) = \frac{1}{S} \sum_{i=1}^{S} L(W, (y_i, x_i^+, x_i^-)^i) \tag{6}$$

where $\mathcal{M}_W(x^+, x^-) = ||\phi(x^+; W) - \phi(x^-; W)||_2$ refers to a parameterized distance metric with respect to projections $\phi(x^+)$ and $\phi(x^-)$, and C is the *slack variable* which defines the margin of separation between them. $\phi(.)$ is simply the projection function from eqn. 5, which is parameterized by $W = [\theta_g, \theta_{g^*}, A]$ from eqn. 4. Intuitively, $\mathcal{M}_W$ finds the distance between *augmented projections* of x_i^+ and x_i^-, i.e., $\phi(x_i^+)$ and $\phi(x_i^-)$, which are compared by their labels y_i. Given a training pair (x_i, x_i^*, y_i), a set of k *similarity samples* $S_i = \{(x_i^+, x_i^-, y_i)^j\}_{j=1}^{k}$ is constructed around the (x_i, y_i) pair, whereby x_i^+ denotes samples with the same label as $y_i = y^+$, and x_i^- denotes samples with a different label than $y_i \neq y^-$. Thus, x^+ and x^- denote *positive samples* (similar) and *negative samples* (dissimilar), respectively.

A variety of negative sampling techniques can be used to obtain the set S [15, 14, 16]. In practice, we found picking $5 - 10$ negative samples that are close to $\phi(x_i)$ and $5 - 10$ samples that are far from $\phi(x_i)$ to be sufficient in creating S for each training triplet (x_i, x_i^*, y_i). We refer the reader to [14] and [15] for more information about the contrastive loss and the construction of S.

Finally, we combine the two portions of our learning task (i.e. representation learning and joint hypothesis) into the optimization task:

$$\min_{\Theta} \mathcal{L}(W) + \lambda \mathcal{L}_{\phi}(\theta_g, \theta_{g^*}, A) + \Omega(W), \tag{7}$$

where $W = [\theta_g, \theta_{g^*}, A]$ is the total set of parameters for the learning task. λ is the hyperparameter which controls the trade-off between the contrastive loss to learn h^* and the representation loss in eqn. 4. $\Omega(.)$ is the regularization term used to constrain the hypothesis space of the joint model.

4.3 Analysis of Sampling Efficiency

4.3.1 Results from Existing Agnostic Models

For an agnostic hypothesis model, such as non-linearly separable SVM, the generalization error bound holds with $1 - \delta$ probability:

$$R(h) \leq R_{emp}(h) + O^*\left(\sqrt{\frac{\Delta_H \log(m/\Delta_H) - \log \delta}{m}}\right), \tag{VC-bound}$$

where $|R(h) - R_{emp}(h)| = \epsilon \in [0, 1]$ is the generalization error represented by the difference between expected and empirical training risks. Δ_H is the VC-dimension of the given SVM model class, m is the sample size, and $\delta \in (0, 1)$, whereas under the SVM+ formulation in [9], the generalization error is given by:

$$R(h) \leq R_{emp}(f^*) + O^*\left(\frac{(\Delta_H + \Delta_{F^*})\log(\frac{m}{\Delta_H + \Delta_{F^*}}) - \log \delta}{m}\right), \tag{SVM+}$$

where $R_{emp}(f^*)$ denotes the error rate of the Teacher's hypothesis $f^* : X^* \longrightarrow Y$, and Δ_{F^*} denotes the VC-dimension of the Teacher model. In the original SVM, the model needs to re-estimate m slack variables for each training sample, in addition to the n parameters in w. At a high-level, the hypothesis function f^* of the LUPI Teacher serves as a *slack function* which approximates these slack variables for each x_i, eliminating the need for the Student to estimate them during training [9]. The number of estimations in the latter case reduces to $O(m + n)$, rather than $O(mn)$. As a result, the sampling efficiency improves from $m \leq O(\frac{\Delta_H + \log(1/\delta)}{\epsilon^2})$ to $m \leq O(\frac{\Delta_H \log \Delta_H + \log(1/\delta)}{\epsilon})$ in the number of samples required to achieve the same generalization error ϵ.

4.3.2 Complexity of Proposed LUPI Method

In this section, we examine the sample complexity of our proposed LUPI method. For simplicity, let us consider the classification setting where we are given a hypothesis class $\mathcal{H}$ of finite VC-dimensions which define a set of functions mapping $\mathcal{X}$ to a label set $\{0, 1\}$, and let the $0 - 1$ loss function define the empirical risk. Let $\Delta_{\mathcal{H}} = d < \infty$. By **the fundamental theorem of PAC learning** (Thm. 6.7 in [17]), there exist $C_1, C_2 \in \mathbb{R}$ such that:

1. $\mathcal{H}$ is agnostic PAC learnable and has the *uniform convergence property* with sample complexity

$$C_1 \frac{d + \log(1/\delta)}{\epsilon^2} \leq m(\epsilon, \delta) \leq C_2 \frac{d + \log(1/\delta)}{\epsilon^2} \qquad \text{(Agnostic)}$$

2. There exists a Realizable subset $\mathcal{H}_r$ such that the sample complexity is defined by

$$C_1 \frac{d + \log(1/\delta)}{\epsilon} \leq m(\epsilon, \delta) \leq C_2 \frac{d \log(1/\epsilon) + \log(1/\delta)}{\epsilon}. \qquad \text{(Realizable)}$$

The main difference between *agnostic* and *realizable* PAC models lies in whether the classifier can completely classify a training set $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^m$. That is, the training error $R_{emp}(h) = \frac{1}{m} \sum_{i=1}^m L(h(x_i), y_i) = 0$ for the particular hypothesis class under some empirical risk minimization (ERM) algorithm. On the other hand, when we have $y_i \neq h(x_i)$ for some training data, there exist some examples for which the current hypothesis class can not successfully separate (i.e. cannot realize an accurate ERM hypothesis with 0 empirical risk), suggesting that the optimal solution is either not contained in the span of the given hypothesis class, or the ERM algorithm cannot converge to the optimal solution in the hypothesis space. Fortunately,[18] introduces some conditions for which a sample complexity between $O(\frac{\log(1/\delta)}{\epsilon})$ and $O(\frac{\log(1/\delta)}{\epsilon^2})$ is possible for some classes of models. Specifically, Tsybakov showed that $m(\epsilon, \delta) \leq C \frac{d \log(1/\delta) + \log(1/\delta)}{\epsilon^n}$, $1 < n < 2$ exists under two general scenarios:

- When there is zero training error (ERM realizable).
- When the classification *margin* between the given hypothesis class and another realizable hypothesis class is bounded.

LUPI qualifies the second Tsybakov condition by leveraging a realizable Teacher that provides a mechanism to bound the margins of the Student ERM classifier. Specifically, [8] considers the case where the Student hypothesis class $\mathcal{H}$ is non-realizable and the Teacher hypothesis $f^* \in \mathcal{F}^*$ is a realizable classifier that approximates an *Oracle* classifier with zero-training error. [8] showed that using kernel alignment between the Teacher and the Student, the latter can satisfy the Tsybakov conditions, leading to a sample complexity that is comparable to the realizable case. [9] further showed that, under some assumptions on $\mathcal{H}^*$, one can achieve $O(\frac{\log(1/\delta)}{\epsilon}) \leq m(\epsilon,\delta) \leq O(\frac{\log(1/\delta)}{\epsilon^2})$ so long as the Teacher classifier $\mathcal{H}^*$ has lower VC-dimension and training error than $\mathcal{H}$.

In our model, we provide the margin bounding mechanism by contrastive loss in Eqn. Contrastive. Under the condition that the Teacher model $f^*(x^*)$ has a lower VC-dimension and $y \cdot h(x) > c_0 C - \mathcal{M}(x^+, x^-)$, the $C - \mathcal{M}(x^+, x^-)$ term from the Contrastive eqn. serves as the approximate slack margin of the oracle classifier. To see this, let $p \in \mathbb{R}$ and $q \in \mathbb{R}$ be random values. Suppose that $p < 0$, then either $q < 0$ or $p - q < 0$ is true. We then also have $P(p < 0) \leq P(q < 0) + P(p - q < 0)$. If we take $p = y \cdot h(x)$ and $q = y \cdot f^*(x^*)$, then with $1 - \delta$ probability, we can express the error bounds of the Student and Teacher models as follows,

$$P(y \cdot h(x) < 0) \leq P(y \cdot f^*(x^*) < 0) + P(y \cdot h(x) < y \cdot f^*(x^*)).$$

Under the contrastive loss in eqn. (Contrastive), we can re-formulate the above as:

$$\begin{aligned} P(y \cdot h(x) < 0) &= P(y \cdot f^*(x^*) < 0) + P(y \cdot h(x) < C - \mathcal{M}(x^+, x^-)) \\ &\leq P(y \cdot f^*(x^*) < 0) + O\left(\frac{\Delta_H + \Delta_{f^*} - \ln \delta}{m}\right) \\ \implies m &\leq O\left(\frac{(\Delta_H + \Delta_{f^*} \log(1/\delta) + \log(1/\delta)}{\epsilon^n}\right), \end{aligned}$$

where $1 < n < 2$ if $P(y \cdot f^*(x^*) < 0)$ is satisfied, i.e. if the Teacher model is realizable. We note that although our proposed LUPI model is PAC learnable, i.e. has the uniform convergence property, we cannot bound the computational complexity of learning. Specifically, if we allow the embedding components $g(.)$ and $g^*(.)$ to be non-convex functions, then finding the global optimum for $\phi(x)$ becomes an NP-hard problem. In other words, the above analysis only examine the sample complexity bounds, but it does not provide insight into the computational runtime of learning, or the *actualization* of the uniform convergence property.

4.4 Experiments

We empirically assess the effectiveness of our LUPI formulation for improving sample efficiency and generalization performance in a multi-task setting. First, we consider performance accuracy on numerous *diagnostic prediction tasks*, which are individually binary classification problems. This allows us to evaluate the ability of our LUPI formulation to actually transfer the privileged information to improve the learning efficiency in a multi-task setting. We benchmark the learning accuracy of our method against the performance of various transfer learning baselines. We also perform an ablation study on the privilege information components of our model to ascertain its necessity. Finally, we perform prediction tasks on classes with only sparse examples – as defined by ≤ 100 training samples, and compare the sample efficiency of our model against select models from other transfer learning paradigms.

For PI, we consider *physician notes* in the form of discharge summaries, linking standard medical terminologies (i.e., UMLS codes [19]) with diagnostic findings in the EHR. *UMLS codes* are a set of standardized medical concepts used by clinicians to describe physical findings of diseases and are used widely in both the EHR as well as medical research [19]. [9] alluded at the idea that medical datasets also contain vast amounts of privileged information in the *physician notes*, which serve to explain the qualities of diseases that can greatly aid decision rules. For experiments, we consider the following set of data for example features, PI, and labels:

- Example Features X: continuous time-series data (i.e. lab values, blood tests, imaging) and discrete static variables (i.e. demographics information) that describe a patient.
- Privileged Information X^*: physician notes containing descriptions in natural language and

medical terms (UMLS concepts [19]) that summarize a particular visit for a patient.

- Target Task Y: prediction tasks of interest, such as mortality (binary classification), disease prediction (multi-task and transfer learning), ... etc.

4.4.1 Datasets and Setup

Table 4.1 provides a brief summary of data sources for our experiments. For each data source, we extract unique data modalities available in the dataset. MIMIC-III (Medical Information Mart for Intensive Care) is a publicly available benchmark dataset for predictive modeling and clinical decision support in the intensive care unit (ICU) setting [20]. It should be noted that MIMIC-III and STRIDE datasets are EHR datasets, although STRIDE is comprised of clinical notes (PI vocabulary) obtained from multiple EHR datasets over 19 years of data collection. *Documents* in table 4.1 refer to literature sources, including medical claims [21] and research articles [22] that heavily utilize UMLS codes. We refer to the clinical notes from EHR as the PI source, which we decomposed into lists of UMLS codes. For example, a clinician's note may contain a text description of *pneumonia* which may utilize several UMLS codes such as (*Lower Lobe Consolidation, Staph Aureus, Productive Cough*) as keywords.

Table 4.1: Summary of datasets used in this study

Database	No. Patients	UMLS	ICD-9s	Temporal
MIMIC-III	22,043	928	148	40
STRIDE	4M	14,256	None	None
Documents	1.2M	None	11,245	None

MIMIC-III provides a rich source of temporal data, ranging from laboratory tests, vital signs and respiratory parameters, all of which provide hourly resolution of descriptive features. For example features, we use 40 physiologic features, including vital signs (i.e. *heart rate, blood pressure, oxygen saturation, temperature*), blood tests (i.e. *WBC count, platelets, INR*) and respiratory parameters: (i.e. *PaO2/FiO2 ratio, PEEP*). These temporal features are the source of example features (EF) for our experiments, i.e. $X = \{x_i\}_{i=1}^{m}, x_i = \{x_i^t\}_{t=1}^{T}$. Preprocessing of these features include binning the time-series by hourly average of each feature and standardized feature values across all adult patients.

Physician notes (the source of PI) in MIMIC-III exist in the form of *discharge notes*, which are physician documentation of key findings relating to the patient's hospital visit. We can represent the PI as $X^* = \{x_i^*\}_{i=1}^m$, where each $x_i^* \in \{0, 1\}^d$ represents a discharge note for the i-th patient, in the form of a d dimensional one-hot vector. Here, d is the total number of UMLS codes that are found in all of our data sources (MIMIC-III, STRIDES, and Documents). One can think of the UMLS codes as a set of basis features for the PI vectors. The rationale of using physician notes as PI is that they are *only available at the end of the hospital stay* and contain copious amount of valuable information regarding a wide array of clinical decision support tasks, such as physical findings, periodic nurse observations, medical or surgical complications, and indicators for mortality risk. During training, we can incorporate these notes into the learning regime, but they become unavailable at inference time.

For labels, MIMIC-III provides a wide range of potential tasks. We focus on the prediction of ICD-9 diagnostic codes, where are a set of diagnosis labels given to patients that identifies their disease states. Each patient has a set codes that can be described by a label vector $y_i \in \{0, 1\}^C$, where C denotes the total number of disease classes considered. ICD-9 prediction is in fact a difficult *multi-label classification* problem among other clinical benchmark tasks due the fact that the distribution of diseases often contain *long tails* [23]. In the typical case, a few diseases dominate in high frequency while most diagnostic codes appear only a few times among all patients. As a result, training samples are sparse for most diseases, leading to poor prediction beyond the most frequent cases. Our experimental task is to leverage information from PI under a *multi-task learning setting* to improve the learning efficiency for a large set of ICD-9 codes, especially ones in the tail distribution (i.e. occurring with few samples in the dataset). We consider the diagnoses appearing in at least 1% of admissions, leaving $C = 148$ ICD-9 group codes to formulate our multi-task prediction as C classification tasks. We consider UMLS terms appearing at least 50 times in discharge notes, leaving $d = 928$ UMLS terms to construct the PI vocabulary.

4.4.2 Initial Baselines

We establish some baseline performance of various hypothesis models for our prediction tasks under 3 settings:

- Using only example features (EF only) to predict ICD-9 labels.
- Using only PI information to predict ICD-9 labels.
- Using both EF and PI information to predict ICD-9 labels.

For each setting, a diverse set of hypothesis classes are used, including a standard recurrent neural network (RNN) and feed forward perceptrons (MLP). The rationale behind these baselines is to determine whether the PI indeed offers more information than the original feature-set based on noisy timer-series data. Ideally, the Teacher hypothesis class F^* should obtain lower empirical risk while using lower model complexity (lower VC-dimension) compared to the Student hypothesis class $\mathcal{H}$ without LUPI. Otherwise, the Student learner will not improve its sample complexity by LUPI, and any improvements in prediction accuracy will likely result from a variance-reduction mechanism (i.e. ensemble) rather than the LUPI mechanism. As a sanity check, we also included a comprehensive Teacher model using EF + PI features, which should provide the best performance. We note, however, that because the Teacher models use PI at test time, they are used to assess the quality of PI rather than benchmark Student performance. In practice, PI is unavailable at test time, so the Teacher models cannot be used for inference in a real-world setting. Table 4.2 summarizes the performance of these baselines on held-out test set data.

Since ICD-9 predictions involve a large number of classes, we take both micro-averaged and macro-averaged AUC as evaluation metrics. Macro-averaged AUC takes *per-class average* of AUC scores, while micro-averaged AUC considers a single AUC score based on a roll-out of label classes for each test set sample. We also include micro-averaged F1-score and micro-averaged area under PRC to quantify the trade-off between precision and recall.

- **RNN Student** denotes the Student learner using the RNN model class conditioned exclusively on EF, using the LSTM architecture as mentioned in [24].
- **MLP Student** denotes a feed-forward network conditioned on the final time-step of EF.
- **MLP Teacher** denotes the Teacher feed-forward network conditioned on PI only. Specifically, we use a weighted sum of the PI embeddings for each $x_i^* = \{w_1, ..., w_k\}$:

  $$\phi(x_i^*) = \frac{1}{k} \sum_{j=1}^{k} w_j^T \theta_{g*}$$

  which maps each PI vector x_i^* into a lower-dimensional representation space, and θ_{g*} denotes the look-up matrix of embeddings obtained in the first step of our LUPI algorithm.
- **Oracle Teacher** denotes the Teacher model which uses both EF and PI for prediction.

  $$h(x_i) = RNN(x_i)$$
  $$f_c^*(x_i, x_i^*) = \sigma(W_{hc}h(x_i) + W_{gc}g(x_i^*) + b_c)$$

  EF inputs are encoded into fixed-length vectors by a set of RNN layers and the PI features are embedded into lower-dimensional space by $g(x^*)$ described previously. Since there are C tasks (i.e., C outputs), a classifier layer is used to predict the $0-1$ label for each ICD-9 code.

Here, we note that the embedding matrix θ_{g*} for learning the lower dimensional representation of PI is obtained by the embedding mechanism highlighted in the PI Embedding equations. Taking the set of UMLS concept codes as the PI vocabulary, we leverage the corpus available in STRIDE and Documents datasets to learn the θ_{g*}, conditioned on the UMLS codes. For example, given a medical document consisting of a set of n relevant UMLS codes $\vec{v} = \{w_1, w_2, ...w_n\}$, we can train the θ_{g*} for the UMLS codes by Eqn. 2, with the modified scoring function:

$$Score(w_1, w_2) = \begin{cases} 1 & \text{if } w_1, w_2 \subset \vec{v} \\ 0 & \text{otherwise} \end{cases} \qquad \text{(Co-occurence)}$$

Table 4.2: Comparison of performance across baseline models

Model	Ma-AUC	Mi-AUC	Mi-F1	AUPRC
RNN Student	0.735	0.783	0.299	0.260
MLP Student	0.715	0.756	0.235	0.211
MLP Teacher*	0.824	0.868	0.446	0.432
Oracle Teacher*	**0.845**	**0.882**	**0.497**	**0.510**

(*) *denotes Teacher models using PI.*

Note that each UMLS concept is represented by $w_i \in \{0,1\}^d$, where $w_{ij} = 1$ for the index corresponding to the UMLS code. Thus, $\theta_{g*}[:, i]$ gives the the *distributed representation* of i^{th} UMLS concept. We train the embedding matrix θ_{g*} over STRIDE and Documents before applying θ_{g*} on the MIMIC-III dataset for LUPI.

We see from table 4.2 that the PI provides strong signals for ICD-9 prediction. Micro-averaged AUC and Macro-averaged AUC are denoted as Mi-AUC and Ma-AUC, respectively. Micro-averaged F1-score and AUC of precision-recall curve are denoted as Mi-F1 and AUPRC, respectively. Large differences exist between the Student baselines and the Teacher models across all performance metrics, suggesting that the PI provides more information about the label space compared to the original time-series features. Again, we emphasize here that discharge notes (PI) are generated only *after* the diagnostic predictions have been made by clinicians, and thus the Teacher models are actually not available at inference time. Interestingly, we also see that the Oracle teacher with combined features provided additional performance boost compared to using PI exclusively as features. This suggests that the temporal features provide some complementary information not contained in the PI.

4.4.3 Comparison Against Other Transfer Learning Methods

Next, we benchmark performance for several existing transfer learning paradigms for incorporating PI with the Student model: transductive learning, inductive learning, and model distillation. Under the transductive framework, we treat the PI as *auxiliary targets*, much like *target replication* in [24]. We train a joint hypothesis model $h : X \longrightarrow X^* \times Y$ to map from the original EF space to the joint PI and label space. By contrast, we incorporate X^* as an auxiliary input for the inductive framework.

We use *data fusion* to learn a joint representation $g : X \times X^* \rightarrow Z$ before learning a hypothesis function to predict $h : Z \rightarrow Y$. For model distillation, a Teacher network predicts a set of soft-labels over the PI information, which the Student model uses as auxiliary input for the final prediction model $h : X \times X^* \rightarrow Y$. Details of the setup is explained below.

4.4.3.1 Multi-task learning

MTL is the representative transductive learning technique. There is only one source domain $\mathcal{D} = \{X, P(X)\}$ and two target tasks: $\mathcal{T}_S = \{X^*, G\}$ and $\mathcal{T}_T = \{Y, \mathcal{H}\}$. The MTL model learns a joint model:

$$g_k(x_i) = MLP(RNN(x_i; W_{rep}); W_k) \qquad \text{(Shared Rep.)}$$

$$h_c(x_i) = MLP(RNN(x_i; W_{rep}); W_c) \qquad \text{(Individual Hyp.)}$$

$$\begin{aligned}\mathcal{L}_{MTL} &= \frac{1}{mC} \sum_{i=1}^{m} \sum_{c=1}^{C} L_Y(h_c(x_i), y_{ic}) \\ &+ \frac{\lambda}{dm} \sum_{i=1}^{m} \sum_{j=1}^{d} L_{X^*}(g_j(x_i), x^*_{ij}) \qquad \text{(MTL Obj.)}\end{aligned}$$

where W_{rep} is the shared weights for the representation model $RNN(x_i; W_{rep})$, W_k and W_c are task-specific weights for target hypothesis models. The MTL loss is composed of two parts: (1) a loss component over the joint label space $\mathbb{E}[L_Y(h(X), Y)]$, and (2) a loss term over the joint PI space $\mathbb{E}[L_{X^*}(g(X), X^*)]$. λ is a hyperparameter which controls the trade-off between the multiple objectives during learning. $L(.)$ denotes some evaluation criterion to approximate the $0 - 1$ loss, for example the binary cross-entropy (BCE) or mean squared error (MSE). We used BCE in the proceeding experiments.

4.4.3.2 Data Fusion

For inductive learning, we used a variant of the Siamese Network [16] to achieve *data fusion* between EF and PI. We use two parallel networks, $g : X \rightarrow Z$ and $g^* : X^* \rightarrow Z$ and minimize the distance

Table 4.3: Comparison of performance across transfer learning models

Model	Ma-AUC	Mi-AUC	Mi-F1	AUPRC
MTL	0.783	0.836	0.384	0.336
Distillation	0.738	0.793	0.289	0.245
Data Fusion	0.779	0.811	0.374	0.328
Ours	**0.838**	**0.845**	**0.397**	**0.344**

between $g(x)$ and $g^*(x^*)$ using the BCE loss. We then learn a hypothesis function $h : Z \to Y$.

$$g(x_i) = RNN(x_i; W_x) \tag{EF Embedding}$$

$$g^*(x_i^*) = MLP(x_i^*; W_{x^*}) \tag{PI Embedding}$$

$$h(x_i, x_i^*) = \sigma(W_g g(x_i) + W_{g^*} g^*(x_i^*) + b_g) \tag{Joint Hyp.}$$

$$\mathcal{L}_Z(W_x, W_{x^*}) = \sum_{i=1}^{m} BCE(g(x_i), g^*(x_i^*)) \tag{Fusion Loss}$$

$$\mathcal{L}_Y(W_g, W_{g^*}) = \sum_{i=1}^{m} BCE(h(x_i, x_i^*), y_i) \tag{Task Loss}$$

$\mathcal{L}_Z$ and $\mathcal{L}_Y$ are trained iteratively using alternating stochastic gradient descent (SGD). At test time, a masking vector $x_{test}^* = \{0\}^d$ is used to represent PI, as it is unavailable for inference. Thus, $h(x_i, x_{test}^*) = y_i$ is used for evaluation.

4.4.3.3 Distillation

Our distillation also contains two parts: a Teacher network trained to generate soft-labels for PI, and Student network conditioned on the EF and PI soft-labels to predict the ICD-9 targets.

$$g(x_i) = RNN(x_i; W_T) \tag{Teacher}$$

$$h_c(x_i) = MLP([g(x_i); RNN(x_i; W_{RNN})]; W_c) \tag{Student}$$

$$\mathcal{L}_{TS} = \sum_{i=1}^{m} \sum_{c=1}^{C} BCE(h_c(x_i), y_i) \tag{Distillation Loss}$$

Here, $[g(x_i); RNN(x_i; W_{RNN})]$ denotes concatenation of the PI soft-labels and the last hidden state of the $RNN(x_i)$. The joint T-S loss connects the Teacher and Student loss together, allowing the two networks to be trained end-to-end.

Table 4.3 summarizes the AUC and retrieval scores for our LUPI model against transfer learning baselines. Micro-averaged AUC and Macro-averaged AUC are denoted as Mi-AUC and Ma-AUC, respectively. Micro-averaged F1-score and AUC of precision-recall curve are denoted as Mi-F1 and AUPRC, respectively. We see that our LUPI formulation outperformed other transfer learning baselines in all major performance metrics. Vanilla *Data Fusion* and *MTL* networks produced comparable performances, and *Distillation* did not have significant improvement over the baseline Student models that did not use PI. For *Data Fusion*, it is likely that since the PI contains a lot more information than the original EF, the decision function of the hypothesis model $h(x, x^*)$ relied heavily on access to PI.

Since PI is masked during testing due to unavailability, $h(x, \{0\}^d)$ likely resulted in poor generalization. Comparable results can be seen in MTL. The drop in performance is most likely due to *negative transfer* [1] due to the wide range of tasks (ICD-9 codes) that contribute uniformly to the multi-objective learning process. Unlike our LUPI, the PI is not used to inform similarity between training samples from different tasks, which do not share the same support (i.e. different diagnosis may come from very different underlying distributions). Thus, in both *Data Fusion* and *MTL* cases, the PI is incorporated in a less efficient way than our proposed model.

Interestingly, we see that the *MLP Teacher* and *Oracle Teacher* models in Table 4.2) still provided better AUC, F1 and AUPRC performances over the all transfer learning models, including our LUPI model. This result suggests that the PI is more informative for diagnostic tasks compared to the original EF, which is what enables the LUPI method to be effective. One possible explanation of the predictive power of the PI is that the embeddings of the UMLS terms, which comprise the PI vocabulary, are learned based on their co-occurrence with disease codes in public literature.

4.4.4 Performance with Sparse Examples

In addition to broad coverage of tasks, we evaluate the *sample efficiency* of our proposed model against transfer learning baselines by considering the more rare diseases with very few training samples. This is actually quite typical in the EHR setting, where diagnosis labels often have very

Table 4.4: Performance of various models for 30 ICD-9 codes appearing less than 100 times in the dataset

Model	Ma-AUC	Mi-AUC	Mi-F1	AUPRC
RNN Student	0.628	0.639	0.104	0.096
MLP Teacher	0.821	0.833	0.212	0.196
Oracle Teacher	0.801	0.805	0.256	0.146
MTL	0.717	0.724	0.150	0.122
Distillation	0.729	0.738	0.158	0.149
Data Fusion	0.821	0.826	0.307	0.237
Ours	**0.834**	**0.835**	**0.381**	**0.330**

long tail distributions. We restrict our predictions to diagnostic codes appearing less than 100 times in the training and test sets and examine the generalization of various modeling schemes.

In table 4.4, we see that performance decreased drastically for non-transfer learning models such as the RNN Student and the Teacher models (both MLP and Oracle). Transfer learning schemes such as *MTL* and *Distillation* also decreased greatly in F1-score and AUPRC. Interestingly, *Data Fusion* method was able to outperform other transfer learning baselines in F1-score and AUPRC, suggesting that learning a domain-invariant representation between the original features and the PI provided a key improvement for sample efficiency. However, our LUPI model achieved the best performance among all the models for long-tail tasks. In fact, its performance across stayed relatively consistent among this subset of tasks compared to its performance on the original set of common diagnoses.

4.5 Discussion and Conclusion

In this chapter, we presented a novel LUPI framework for retaining PI in the multi-task setting to improve sample complexity over a wide range of related tasks. The key idea was to learn a joint representation of the original feature space and the PI by leveraging their co-occurrence information in the data. Decomposing the PI into distributed representations of basis features was vital for the realization of this mechanism. Experiments show that our proposed LUPI method can out-perform baseline models and other transfer learning methods in multi-task learning scenarios, particularly in situations where training samples are very rare ($<$ 100 samples per task). In addition to improved performance, we also provided sample complexity analysis that outline scenarios under which our

LUPI method can provide similar benefits over traditional transfer learning approaches.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[2] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[3] Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *arXiv preprint arXiv:1703.07771*, 2017.

[4] Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F Stewart, and Jimeng Sun. GRAM: graph-based attention model for healthcare representation learning. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 787–795. ACM, 2017.

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.

[6] Jing Zhao, Xijiong Xie, Xin Xu, and Shiliang Sun. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54, 2017.

[7] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[8] Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural networks*, 22(5-6):544–557, 2009.

[9] Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: similarity control and knowledge transfer. *Journal of machine learning research*, 16(2023-2049):2, 2015.

[10] Xue Li, Bo Du, Chang Xu, Yipeng Zhang, Lefei Zhang, and Dacheng Tao. R-svm+: Robust learning with privileged information. In *IJCAI*, pages 2411–2417, 2018.

[11] John Lambert, Ozan Sener, and Silvio Savarese. Deep learning under privileged information using heteroscedastic dropout. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8886–8895, 2018.

[12] Edward Choi, Mohammad Taha Bahadori, Elizabeth Searles, Catherine Coffey, Michael Thompson, James Bost, Javier Tejedor-Sojo, and Jimeng Sun. Multi-layer representation learning for medical concepts. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1495–1504. ACM, 2016.

[13] Dhanesh Ramachandram and Graham W Taylor. Deep multimodal learning: A survey on recent advances and trends. *IEEE Signal Processing Magazine*, 34(6):96–108, 2017.

[14] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *null*, pages 1735–1742. IEEE, 2006.

[15] Miguel A Carreira-Perpinan and Geoffrey E Hinton. On contrastive divergence learning. In *Aistats*, volume 10, pages 33–40. Citeseer, 2005.

[16] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

[17] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[18] Alexander B Tsybakov et al. Optimal aggregation of classifiers in statistical learning. *The Annals of Statistics*, 32(1):135–166, 2004.

[19] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270, 2004.

[20] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3, 2016.

[21] Youngduck Choi, Chill Yi-I Chiu, and David Sontag. Learning low-dimensional representations of medical concepts. *AMIA Summits on Translational Science Proceedings*, 2016:41, 2016.

[22] Samuel G Finlayson, Paea LePendu, and Nigam H Shah. Building the graph of medicine from millions of clinical narratives. *Scientific data*, 1:140032, 2014.

[23] Fengyi Tang, Cao Xiao, Fei Wang, and Jiayu Zhou. Predictive modeling in urgent care: a comparative study of machine learning approaches. *JAMIA Open*, 1(1):87–98, 2018.

[24] Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzel. Learning to diagnose with lstm recurrent neural networks. *arXiv preprint arXiv:1511.03677*, 2015.

CHAPTER 5

ADVERSARIAL PRECISION SENSING

5.1 Introduction

In many real-world applications, fully-observed datasets are difficult to obtain, especially for temporal data, where observing the full set of features for all samples across time is simply too expensive. A prime example of this phenomenon exists in the realm of medical informatics, specifically electronic health records (EHR) dominated by time-series data. Yet this hasn't stopped EHR from sparking numerous research interests in recent years [1, 2, 3, 4, 5]. Time-series data in the EHR consists of features that are sampled at different levels of temporal granularity (e.g., lab tests are sampled at longer time-scales than vital signs). Most of the time, only a small subset of features are observed at any time-step, as it is prohibitively expensive for the physician to obtain the full set of features for every patient for every time interval. In practice, a form of active sensing [6] is implicitly done in an ad-hoc manner by physicians based on a combination of inpatient work-flow, resource constraints, and domain expertise. As a result, the key difficulty with EHR time-series data is that they almost always require re-sampling and imputation due to large amounts of missing values. Therefore it is no surprise that several recent works have shown that temporal models trained on EHR data have fragile decision boundaries that are susceptible to small perturbations [7, 8].

To address this issue, we formulate the *precision sensing* problem setting as one that takes in data-streams along the *temporal* dimension, rather than across samples. For example, consider the EHR setting, where at each time-step, the physician has to decide how to allocate a constrained amount of tests/resources among patients for some set of clinical tasks. In other words, the precision sensing problem for time-series is in fact a *resource distribution* problem at each time-step, with a fixed observation budget across a given set of samples. Therefore the querying process is applied across time rather than samples, and any formulation must reflect both the temporal-dependency and the budgetary-constraint of the querying process.

In this work, we formulate the active sensing problem as a minimax game between two players:

- A *Feature Sensor* (FS) that generates a sensing tensor A over a temporal data-stream in an online manner.
- A *Progressive Learner* (PL) that adjusts the decision boundary at each time-step to handle sparser sets of features, allowing the FS model to improve its sensing strategy toward sparser selections.

We then prove that our formulation reaches a local Nash equilibrium during co-training. Afterwards, we evaluate the quality of the dynamic sensing strategy by training a diverse set of classifiers on the resulting set of data with incomplete features and compare their performances against a corresponding set of hypothesis models conditioned on the original dataset.

Our results show that classifiers conditioned on dynamically sensed data, i.e., incomplete data, are able to maintain prediction accuracy, despite adhering to a strict budgetary constraint for sensing. We show that unlike attention and perturbation models, the sensing strategy of the FS-Model can be used to train many types of models, including deep models, shallow models, and even non-temporal models, with minimal decrease in performance. In experiments, we demonstrate how our budget-adaptive framework can be used to produce an active data acquisition strategy that can be implemented in hospitals to deal with feature sensing under moving budgetary constraints.

5.2 Problem Formulation

5.2.1 Precision Sensing for Time-Series Data

Classic time-series data comes in the form: $X = \{x_i\}_{i=1}^m, x_i = \{x_i^{(t)}\}_{t=1}^T, x_i^{(t)} \in \mathbb{R}^n$, where m is the number of samples, T the number of time-steps, and n the number of features. Precision sensing considers the case where $X^{(t)} = \left\{x_i^{(t)}\right\}_{i=1}^m, x_i^{(t)} \in \mathbb{R}^n$, is a *feature matrix* across the entire batch of samples at time step t. At each time-step we are interested in finding a corresponding *sensing matrix* $A^{(t)} \in \{0, 1\}^{m \times n}$ over the data matrix. During the observation period, a cumulative *sensing tensor*

$A = \{A^{(t)}\}_{t=1}^{T}$ is incrementally constructed according to the dynamics:

$$A^{(t+1)} = G(X^{(1)} \odot A^{(1)}, \dots, X^{(t)} \odot A^{(t)}), \tag{5.1}$$

for some update function $G(.)$. Alternatively we can view $A_i = \{A_i^{(t)}\}_{t=1}^{T}$ as the subset of sensed features across time for each sample. Under this view, we can formulate the precision sensing objective as follows:

Definition 2. *(Precision Sensing) The problem of precision sensing seeks a sensor tensor A^* that minimizes the empirical risk (ERM) [9] according to:*

$$\begin{aligned} A^* &\in \underset{A_i \in \mathcal{A}}{argmin}\, \mathbb{E}_{P(X,Y)}[L(h(A_i \odot X_i), y_i)] \\ &\approx \underset{A_i \in \mathcal{A}}{argmin}\, \frac{1}{m} \sum\nolimits_{(X_i,y_i)\in\mathcal{D}} L(y_i, h(A_i \odot X_i)), \end{aligned}$$

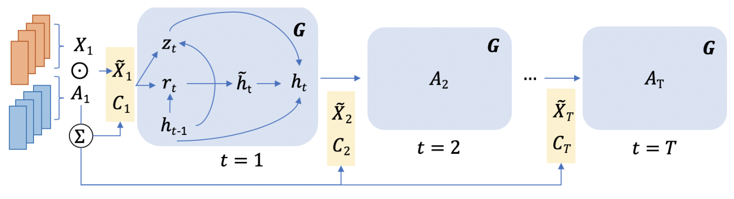
where h is a hypothesis class that maps $h : X \to Y$, and $L(.)$ is a risk function that evaluates the hypothesis mappings against the actual label.

Here, $\mathcal{A}$ denotes the space of sensing matrices, and each matrix A_i can be thought of as a per-sample sensing matrix. From this perspective A_i describes the subset of sensed features across time for each sample.

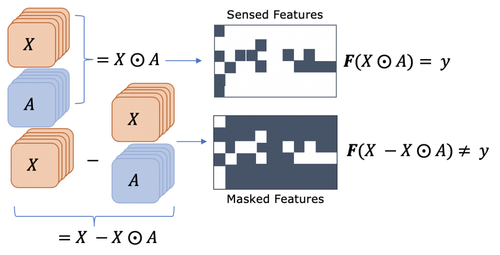
Remark 1. *Ideally the optimal* sensing tensor A *generates a* partially-observed dataset*:* $\{(\tilde{X}_i, y_i)\}$, $\tilde{X}_i = A_i^* \odot X_i$ *such that $P(\tilde{X}, Y) \approx P(X, Y)$. That is, the joint distribution of the dynamically sensed data produced by A^* should match the joint distribution of the fully-observed dataset.*

The key difference between existing methods and our approach is in the manner in which A is generated. In feature selection for time-series, A is obtained by some form of sparse sensing technique that *statically* assigns the same subset of features to be observed for each X_i. By contrast active sensing allows for A to be built in an individualized manner for each sample, yet still actively *sub-samples* (sample, view) pairs during each co-training iteration. But since it only sub-samples a small subset of examples, active sensing fails to consider an overarching budgetary constraint across *all* the samples at each time-step.

Figure 5.1: Overview of problem setting.



(a) Incremental generation of the *sensing tensor*.



(b) Minimax game between FS and PL.

5.2.2 Budget-Constrained Precision Sensing as Adversarial Optimization

At a high-level, the proposed adversarial framework works as follows:

- At each time-step, FS tries to "discard", based on observations from previous time-steps, features that lead to mis-classifcation.
- At the end of each observation window, the PL adjusts the decision boundary so that the sensed features result in correct classification.

Fig. 5.1a describes incremental generation of the *sensing tensor* A by the FS. A is generated recurrently by applying the sensor dynamics $G(.)$, which is learned by the FS, where $A^{(t)} = G(\tilde{X}^{(1)}, \ldots, \tilde{X}^{(t)})$ outputs $A^{(t+1)}$ at each time-step. Fig. 5.1b describes the minimax game between the FS and PL. PL (top) tries to minimize the classification error with sensed features $X \odot A$ while FS (bottom) tries to allocate A such that unsensed features lead to misclassification. At the end of the observation window, the PL adjusts the parameters of F to handle the updated input $X \odot A$, which now has a different sparsity pattern than before, as shown in Fig. 5.1b. At equilibrium the PL will no longer be able to improve its performance, provided that the FS has selected an optimal subset of features to sense such that accuracy improvement is no longer possible given the observation budget.

Max-Margin Objective. More formally, the FS tries to solve the optimization problem:

$$\max_{\theta_G} \left\{ F(X - X \odot G(X))_{\neg y} - F(X - X \odot G(X))_y, 0 \right\}$$
$$\text{s.t.} \;\; A^{(t)} \in \{0,1\}^{m \times n}, \quad \sum\nolimits_{i,j,t} A_{ij}^{(t)} \leq B \tag{5.2}$$

where $G(X) = [G(\tilde{X}^{(1)}), G(\tilde{X}^{(2)}), \ldots, G(\tilde{X}^{(T)})] = A$. When $A_{ij}^{(t)} = 0$, the j-th feature is *masked* for the i-th sample at time t. We can think of $\hat{X} = X - X \odot A$ as a subset of features that are discarded by FS. Thus, $\tilde{X} = X \odot A$ is the resulting set of *sensed* (unmasked) features at the end of the observation window. During training, the FS model ranks $F(X - X \odot A)_{\neg y}$ higher than $F(X - X \odot A)_y$.We use $\neg y_i$ to denote the case where the hypothesis produces the wrong class for the $i-$th sample. B denotes the budget constraint on A, and $F(.)_c$ denotes the c-th logit of the hypothesis model output.

5.3 Proposed Method

Derivation of Eqn. 5.2 First, let us consider the popular adversarial setting of perturbation models described in [10]:

$$\begin{aligned} \text{minimize} \quad & D(X, X + \delta) \\ \text{s.t.} \quad & C(X + \delta) \neq y \\ & X + \delta \in [0, 1]^n \end{aligned} \tag{5.3}$$

where X is the original set of features, δ is the perturbation to be applied to the feature space, $D(.)$ is a distance metric, and $C(.)$ is a classifier. Under the perturbation setting, the $C(X + \delta) \neq y$ term finds an adversarial set of samples, $X + \delta$, leading to misclassification, while $D(X, X + \delta)$ is a reconstruction term that limits the size of δ such that it is not trivially large. Recent work (e.g., [7, 8]) leverages variants of this approach for ranking feature importance in complex feature spaces.

We develop a different set of reconstruction and modification objectives under a similar setting:

$$\begin{aligned} \text{minimize} \quad & \mathcal{L}(C(X \odot A), y) \\ \text{s.t.} \quad & C(X - X \odot A) \neq y, \\ & \forall i, j, t : A_{ij}^{(t)} \in \{0, 1\}, \ \& \ \sum_{ijk} A_{ijk} \leq B \end{aligned} \tag{5.4}$$

where C is the classifier and $\mathcal{L}$ a loss function on (X, y) pairs. A denotes a "sensing tensor" that gives a component-wise indicator for whether feature j is sensed at time t for sample i. Additionally,

B gives a "budget constraint" for the sensing tensor A. Rather than reconstructing $X \approx X + \delta$, we learn the construction of A based on the recovery of $P(y|X) \approx P(y|X \odot A)$ through C. We also update C and A at different rates—C is updated at the end of observation windows while A is updated every time-step.

Thus, we formulate the active sensing problem of interest under a single mini-max objective:

$$\min_{\theta_F} \max_{\theta_G} \underbrace{\{F(X - X \odot G(X))_{\neg y} - F(X - X \odot G(X))_y\}^+}_{\text{Feature Sensing}} + \underbrace{CE_y(F(X \odot G(X)))}_{\text{Classifier Reconstruction}} + \underbrace{\beta||X - X \odot G(X)||_1}_{\text{Budget Management}} . \tag{5.5}$$

Here F corresponds to C and G corresponds to a function to be learned from A. Meanwhile, CE_y is the *cross-entropy* loss w.r.t. y:

$$CE_y(x) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \log x_i + (1 - y_i) \log\left(1 - x_i\right)\right] . \tag{5.6}$$

5.3.0.1 Feature Sensor

We can decompose the update dynamics of the sensor generation process (G) into the following components:

$$h_i^{(0)} = \max\{0, W_{xh}[X_i^{(0)}; 0]\} \tag{Initialization}$$

$$z_i^{(t)} = \max\{0, W_{xz}[\tilde{X}_i^{(t)}; C^t] + W_{zh}h_i^{(t-1)}\} \tag{Update Gate}$$

$$r_i^{(t)} = \max\{0, W_{xr}[\tilde{X}_i^{(t)}; C^t] + W_{rh}h_i^{(t-1)}\} \tag{Reset Gate}$$

$$\tilde{h}_i^{(t)} = \tanh\left\{W_{xh}[\tilde{X}_i^{(t)}; C^t] + W_{hh}\left(r_i^{(t)} \odot h_i^{(t-1)}\right)\right\} \tag{Transition}$$

$$h_i^{(t)} = \left(1 - z_i^{(t)}\right) \odot h_i^{(t-1)} + z_i^{(t)} \odot \tilde{h}_i^{(t)} \tag{State Update}$$

$$A_i^{(t+1)} = \sigma\left(W_h h_i^{(t)} + b_a\right) \tag{Sensor Output}$$

$$C^t = C^{t-1} + \sum_i \sum_j c_j A_{ij}^{(t)} \tag{Budget Update}$$

where C^t denotes the total observation cost at time t, corresponding to the total number of observations in $[A^{(1)}, \dots, A^{(t)}]$ across *all* samples. c_j denotes the cost of observing feature j (e.g., certain medical tests may be more costly to observe than others). $[a; b] \in \mathbb{R}^{n+1}$ denotes the concatenation of vector $a \in \mathbb{R}^n$ and scalar $b \in \mathbb{R}$. $\sigma(.)$ denotes a thresholding function (e.g., sigmoid activation) that maps the output of each $A_{ij}^{(t)} \in [0, 1]$ feature component to $\{0, 1\}$, which is done to satisfy the integer constraint in Eqn. 5.2 that $A_{ij}^{(t)} \in \{0, 1\}, \forall i, j, t$.

$h_i^{(t)}$ serves as an internal state representation that considers every *sensed* features vector $\tilde{X}_i^{(1)}, \dots, \tilde{X}_i^{(t)}$ up to time t for sample i, as well as the *observation cost across samples* at each time-step, i.e., $C^1, \dots, C^t$. The gates z and r control the update rule of the memory state to prevent gradient vanishing [11]. The weights $\theta_G = [W_{xh}, W_{xh}, \dots, W_{hn}]$ are learned by optimizing over the *feature sensing* portion of the objective function in Eqn. 5.5:

$$\max_{\theta_G} \; \{F(X - X \odot G(X))_{\neg y} - F(X - X \odot G(X))_y\}^+ \tag{5.7}$$

5.3.0.2 Progressive Learner

The PL serves to stabilize the reference hypothesis model F in response to the shift in sparsity patterns observed in $G(X)$. We note the time-scale difference between F and G updates: whereas FS updates G at every time-step t, the PL updates F only at the end of an observation window $1, \dots, T$. Before $G(.)$ is learned by the FS model, the decision function F is first initialized on a small batch of fully-observed data $\{X_i, y_i\}_{i=1}^k \subset \mathcal{D}_{\text{train}}, k \ll m$ to obtain an initial approximation of the underlying $P(Y|X)$. Over time, the PL is gradually conditioned on $A \odot X$, with a progressively sparser A.

To achieve this conditioning effect, we utilize a *schedule sampling* technique similar to [12], where we pick the input to F based on a coin toss (H):

$$F(.) = \begin{cases} F(X) & P(H = 1) = \varepsilon, \\ F(G(X) \odot X) & P(H = 0) = 1 - \varepsilon. \end{cases} \tag{Scheduled Sampling}$$

Here, ε denotes the probability of using the fully-observed X for training, and $1-\varepsilon$ is the probability of using the masked version. Initialize $\varepsilon = 1$ for the first k iterations steps to obtain an unbiased estimation. After k iterations, decay ϵ according to:

$$\varepsilon = t - k/(t - k + \exp(k - t)). \tag{ε-Decay}$$

By decaying ε, the training distribution of the PL model shifts gradually from the fully-observed to the sensed features. We pick a baseline $\varepsilon = \min\{t - k/(t - k + \exp(k - t)), 0.25\}$ to ensure that PL gets exposed to at least a small fraction of the fully-observed X during each training iteration. This allows the PL to obtain a good initialization on the feature importance. The PL adjusts the decision boundary by minimizing over the *classifier reconstruction* portion of Eqn. 5.5:

$$\min_{\theta_F} \quad CE_y(F(G(X) \odot X)) \tag{5.8}$$

where each $F(X_i)$ is sampled from Eqn. Scheduled Sampling.

5.3.0.3 Budget-Manager

The strength of budget constraint is updated at each iteration by dual gradient ascent [13] for β according to Eqn. 5.9:

$$\beta_{t+1} = \beta_t + \alpha(X - X \odot G(X)), \tag{5.9}$$

where α denotes the learn rate for the dual variable β. We note that in real-world applications, the budget β is usually set beforehand (e.g., by resource constraints). Thus, dual ascent on β gives an option to continually shrink the observation budget until a *minimum set* of sensed features is realized. However, this may not be needed in a realistic setting where the budget constraint is constant.

The intuition behind the β term follows by considering its extreme values. When $\beta \to 0$, the max-margin objective dominates. Consequently, the FS is incentivized to sense every feature, i.e. $A_{i,j}^{(t)} = 1, \forall i, j, t$, leading to the trivial solution where $X \odot A \to X$. In contrast, as $\beta \to \infty$ the budget constraint dominates, and the FS is incentivized to choose $A_{i,j}^{(t)} = 0, \forall i, j, t$, leading to an empty feature set. Thus, the β term serves as a trade-off parameter between A's sparsity and sensing performance. The training loop for FS, PL, and budget-manager is summarized in Algorithm 2.

Algorithm 2 Co-training of FS and PL.

```
1: for number of total training iterations do
2:     for $k$ training steps do
3:         Sample mini-batch of $m$ samples $\{(X_i, y_i)\}_{i=1}^m$ from training data.
4:         Generate $m$ samples using $G$:
               $$\{X_i - X_i \odot G(X_i), y_i\}_{i=1}^m.$$
5:         Using Scheduled Sampling with $\varepsilon$-Decay, update $\theta_F$ by performing SGD on Eqn. 5.8
6:     end for
7:     for $k$ training steps do
8:         Update $G$ parameters by projected gradient ascent on
               $$\nabla_{\theta_G} \max \{Z(X - X \odot G(X))_{\neg y} - Z(X - X \odot G(X))_y, -\kappa\}.$$
9:         (Optional) Dual ascent on budget constraint $\beta$ according to
               $$\beta \leftarrow \beta + \alpha(X - X \odot G(X)).$$
10:    end for
11: end for
```

5.3.0.4 Inference

To perform inference, we train two sets of models for comparison. (1) A set of hypothesis models $h_S : X \mapsto Y$ trained on the *sensed* dataset $\mathcal{D}_S = \{(G(X_i) \odot X_i, y_i)\}_{i=1}^m$, generated using the FS. (2) A set of hypothesis models h trained on the original dataset $\{(X_i, y_i)\}_{i=1}^m$. The difference in generalization error between h_S and h quantifies the feature sensing strategy learned by the FS.

5.4 Analysis of Proposed Method

Optimizing FS, PL and Budget-Manager involves finding saddle points in a dynamic loss landscape. Here, we prove the existence of *fixed points* in the proposed minimax game and their correspondence with local Nash Equilibria (NE) [14]. At local NEs, FS and PL cannot improve allocation or accuracy by local changes in F and G, respectively.

5.4.1 Existence of Local Nash Equilibria

We can re-express Eqn. 5.5 into individual pay-off functions:

- $J_F(\theta_F) = CE_y[F(X \odot G(X))]$ gives the cost for the PL player.
- $J_G(\theta_G) = \{F(X - X \odot G(X))_y - F(X - X \odot G(X))_{\neg y}\}^+$ gives the cost for the FS, from the minimizer's point of view.
- $J_B(\theta_B) = \sum_{ijk} A_{ijk} - B$ defines the cost for the budget manager.

Let us define $\xi := (\theta_F, \theta_G, \beta)^T$ as a set of actions for the PL, FS and budget manager. Here, we denote $\xi_1 = \theta_F \in \Gamma_1, \xi_2 = \theta_G \in \Gamma_2, \xi_3 = \beta \in \Gamma_3$ as the subset of parameters in ξ that individual players can modify at each iteration, and Γ_i denotes the $i-$th player's action space.

We also define a *potential function* as a composition of the payoffs among the players:

$$\phi^i_j(\xi) := \max\{0, J_i(\xi) - J_i(j, \xi_{\neg i})\}, \quad j \in \Gamma_i$$

where $J_i(j, \xi_{\neg i})$ defines the cost of another policy $j \neq \xi_i$ for the $i-$th player currently using ξ_i parameters. $\xi_{\neg i}$ indicating that other players retain the same actions. So the potential function defines the *cost-of-switching* from the current policy to another, defined over each player's action space.

Finally, let us define the *best response* function [14] for each player:

$$f_i(\xi_i, j) := \frac{\xi_i + \phi^i_j(\xi)}{1 + \sum_{k \in \Gamma_i}(\phi^i_k(\xi))}$$

where $j \in \Gamma_i$ corresponds to the alternative actions in the i-th action space. We now introduce the definition of Nash Equilibrium (NE) according to [14] for our problem:

Definition 3. *Nash Equilibrium. ξ^* is a Nash Equilibrium point of $f(.)$ if it satisfies $J_i(\xi_i) \leq J_i(j, \xi_{\neg i}), \forall i, j$.*

Lemma 1. *Assuming that the domain of ξ is continuous and convex, there exists a fixed point of f, i.e., $f_i(\xi^*_i, j) = \xi^*_i, \forall i, j$.*

Proof. The proposed best response function maps from $f : \Gamma \to \Gamma$ and is a continuous mapping. Furthermore, each sub-domain Γ_i is convex and compact. Compactness can be achieved through regularization applied to F and G, in the form of budget-constraint, which serves to bound the size of θ_G according to a β. So each Γ_i is closed and bounded, and by the Heine-Borel Theorem, each Γ_i is compact in R^{n_i}. Thus, we can apply Brouwer's fix point theorem from [15], and therefore $\exists \xi^*$ s.t. $f(\xi)$ at $f(\xi^*) = \xi^*$. In other words, a fixed point exists.

Lemma 2. *A point ξ^* is a fixed point of f <u>if and only if</u> it is a Nash Equilibrium point.*

Proof. First, suppose ξ^* is a NE point. Then we have $J_i(\xi_i^*) \leq J_i(j, \xi_{\neg i}^*),\ \forall i, j$. So it follows that $\forall i, \phi_j^i(\xi_i^*) = 0$. Thus, we have $f_i(\xi_i^*, j) = \frac{\xi_i^* + 0}{1+0} = \xi_i^*, \forall i, j$, which defines a fixed point.

Conversely, suppose ξ^* is a fixed point, i.e., $f(\xi^*, j) = \xi^*$. Then for every i, there is at least one ξ_i^* s.t. $J_i(\xi_i^*) \leq J_i(j, \xi_{\neg i}^*)$ (by definition of fixed point). For such ξ^*'s, we have

$$
\begin{aligned}
&\phi_j^i(\xi^*) = 0 \\
\implies & f_i(\xi_i^*) = \xi_i^* = \frac{\xi_i^* + 0}{1 + \sum_k \phi_k^i(\xi^*)} \\
\implies & \sum_k \phi_k^i(\xi^*) = 0 \implies \phi_k^i(\xi^*) \equiv 0,\ \forall k \neq j \in \Gamma_i.
\end{aligned}
$$

We have shown that $\forall \xi^* = f(\xi^*)$, such a point must also correspond to a NE point. □

Theorem 1. *There exists at least one Nash Equilibrium point ξ^* such that each player's best response does not deviate from ξ^* with respect to the minimax objective in Eqn. 5.5. Furthermore, the fixed points of the minimax objective correspond to local Nash Equilibrium points.*

Proof. The result directly follows from combining Lemma 1 and 2 with Definition 3. □

We note that although local NE's exist under our formulation, we do not show *admissibility* [16] or interchangeability of the payoff values for such NE's since the proposed game is non-zero sum. However, the NE property implies that small, local changes to the budget allocation and classifier decision boundary do not improve the performance under either objective.

5.4.2 Convergence of Algorithm 2

One interesting challenge of our problem is that G and F are updated at different time-scales. We now show that Algorithm 2 can converge to local NE's, at least under some good initialization schemes around the neighborhood of NE's. We summarize the main result below.

Definition 4. *The gradient operator for the FS and PL players is defined as follows:*

$$\xi^{(t+1)} = \xi^{(t)} + \alpha\eta(\xi^{(t)}), \quad \eta(\xi) = \begin{pmatrix} -\nabla_{\theta_F} J_F(\theta_F) \\ \varepsilon\nabla_{\theta_G} J_G(\theta_G) \\ 0 \end{pmatrix}$$

where α is the learning rate hyperparameter, and $\varepsilon > 0$ denotes an off-set *parameter that scales the update rate of the ∇_{θ_G} component according to a faster time-scale.*

We can verify that when $\dot{\eta} = 0$ at some time T, we have $\eta \equiv 0 \implies \xi^{(t+1)} = \xi(t), \forall t > T$, which describes a fixed point ξ^*.

Theorem 2. *Assuming the loss functions for each player is locally Lipschitz in ξ and piecewise linear in t near a Nash Equilibrium point, the error dynamics $\dot{\eta}$ follows uniform asymptotic stability and converges toward the origin at a rate of:*

$$||\eta(t)|| \leq \mathcal{K}(\lambda||\eta(t_0)||, t - t_0),$$

for some $\mathcal{KL}$−class function (§4.4 in [17]) ($\mathcal{K}$) and constant $0 < \lambda < \infty$.

Proof. Here, we make the following assumptions:

- At a *fixed* budget constraint β, $\frac{\partial\beta}{\partial t} = 0$.

- J_F and J_G satisfy the following:

$$\begin{aligned}
J_F(\theta_F) &= L_F(F(X \odot G(X))) \\
J_G(\theta_G) &= L_G(F(X - X \odot G(X))) \\
\nabla_{\theta_F} J_F(\theta_F) &= L'_F(F(X \odot G(X))) \cdot \nabla_{\theta_F} F(X \odot G(X)) \\
\nabla_{\theta_G} J_G(\theta_G) &= -L'_G(F(X - X \odot G(X))) \cdot \nabla_X F(X - X \odot G(X)) \cdot \nabla_{\theta_G} G(X)
\end{aligned}$$

  where L_F and L_G are loss functions according to the labeled sample pairs $\{(X_i, y_i)\}_{i=1}^m$.

- L_F and L_G are bounded, locally Lipschitz in ξ and piecewise continuous in t near a local fixed-point equilibrium.

We can thus express the error dynamics $\dot{\eta}$ (i.e., the update rule of η) as follows

$$\dot{\eta} = \begin{pmatrix} -\nabla^2_{\theta_F} J_F(\theta_F) & -\varepsilon \nabla_{\theta_F,\theta_G} J_F(\theta_F) \\ \varepsilon \nabla_{\theta_G,\theta_F} J_G(\theta_G) & \varepsilon^2 \nabla^2_{\theta_G} J_G(\theta_G) \end{pmatrix} \eta$$

$$\begin{aligned}
-\nabla^2_{\theta_F} J_F(\xi_F) &= -L''_F(F(X \odot G(X))) \cdot \nabla_{\theta_F} F(X \odot G(X)) \cdot \nabla_{\theta_F} F(X \odot G(X))^T \\
-\nabla_{\theta_F,\theta_G} J_F(\xi_F) &= -L'_F(F(X \odot G(X))) \cdot \nabla_{\theta_F} F(X \odot G(X)) \cdot (\nabla_X F(X \odot G(X))) \cdot \nabla_{\theta_G} G(X))^T \\
\nabla_{\theta_G,\theta_F} J_G(\xi_G) &= -L'_G(F(X - X \odot G(X))) \cdot \nabla_X F(X \odot G(X)) \cdot \nabla_{\theta_G} G(X) \nabla_{\theta_F} F(X - X \odot G(X))^T \\
\nabla^2_{\theta_G} J_G(\xi_G) &= -L''_G(F(X - X \odot G(X)))(\nabla_X F(X \odot G(X)) \cdot \nabla_X F(X \odot G(X)))^T \nabla_{\theta_G} G(X) \nabla_{\theta_G} G(X)^T
\end{aligned}$$

Next, around the equilibrium point at the origin, let us assume the following properties about $\nabla_X F(X), \nabla_{\theta_F} F(X), \nabla_{\theta_G} G(X)$, i.e., the back-propagation terms:

$$\begin{aligned}
&\implies \exists 0 < c_1 < \infty : \nabla_{\theta_F} F(X) \leq c_1 ||\nabla_{\theta_F} J_F(\theta_F)|| = c_1 ||\eta_1|| \\
&\implies \exists 0 < c_2 < \infty : \nabla_{\theta_G} G(X) \leq c_2 ||\nabla_{\theta_G} J_G(\theta_G)|| = c_2 ||\eta_2|| \\
&\implies \exists 0 < \gamma < \infty : ||\nabla_X F(X)|| \leq \gamma.
\end{aligned}$$

Let us also assume that $\exists c_3 > 0 : ||L_i'(\xi)|| < c_3$, and $\exists c_4 > 0 : ||L_i''(\xi)|| < c_4$, that is the first and second order changes are bounded around the origin. We can thus re-express $\dot{\eta}$ as follows:

$$\begin{aligned}
-\nabla_{\theta_F,\theta_G} J_F(\xi_F) &= \nabla_{\theta_F} J_F(\xi_F) \cdot \nabla_X F(\tilde{X}) \cdot \nabla_{\theta_G} G(X)^T \\
\nabla_{\theta_G,\theta_F} J_G(\xi_G) &= -\nabla_{\theta_G} J_G(\xi_G) \nabla_{\theta_F} F(\hat{X})^T \\
\nabla^2_{\theta_G} J_G(\xi_G) &= -L_G''(F(\hat{X})) \cdot (\nabla_X F(\hat{X}) \nabla_X F(\hat{X}))^T \\
&\cdot \nabla_{\theta_G} G(X) \nabla_{\theta_G} G(X)^T .
\end{aligned}$$

$$\implies \dot{\eta} = \begin{pmatrix} -L_F''(F(\tilde{X})) \cdot \nabla_{\theta_F} F(\tilde{X}) \nabla_{\theta_F} F(\tilde{X})^T \eta_1 + \varepsilon \nabla_X F(\tilde{X}) \cdot \nabla_{\theta_G} G(X)^T \eta_1 \eta_2 \\ \\ -\varepsilon \nabla_{\theta_F} F(\hat{X})^T \eta_1 \eta_2 - \varepsilon^2 L_G''(F(\hat{X})) \cdot (\nabla_X F(\hat{X}) \nabla_X F(\hat{X}))^T \nabla_{\theta_G} G(X) \nabla_{\theta_G} G(X)^T \eta_2 \end{pmatrix}$$

Finally, we let us consider the Lyapunov function candidate:

$$V(\eta) = \frac{1}{2}\eta_1^2 + \frac{1}{2}\eta_2^2$$

which satisfies the property $V(\eta) > 0, \forall \eta \neq 0$. We can also see that $\dot{V}(\eta) < 0$:

$$\begin{aligned}
\dot{V}(\eta) &= \frac{\partial V(\eta)}{\partial \eta}^T \frac{\partial \eta}{\partial t} \\
&= \begin{pmatrix} \eta_1 \\ \\ \eta_2 \end{pmatrix}^T \begin{pmatrix} \dot{\eta}_1 \\ \\ \dot{\eta}_2 \end{pmatrix} \\
&= -L_F''(F(\tilde{X})) \cdot \nabla_{\theta_F} F(\tilde{X}) \nabla_{\theta_F} F(\tilde{X})^T \eta_1^2 + \varepsilon \nabla_X F(\tilde{X}) \cdot \nabla_{\theta_G} G(X)^T \eta_1^2 \eta_2 \\
&- [\varepsilon \nabla_{\theta_F} F(\hat{X})^T \eta_1 \eta_2^2 - \varepsilon^2 L_G''(F(\hat{X})) \cdot (\nabla_X F(\hat{X}) \nabla_X F(\hat{X}))^T \nabla_{\theta_G} G(X) \nabla_{\theta_G} G(X)^T \eta_2^2] \\
&\leq -c_3 c_1^2 \eta_1^4 - (c_1 - \gamma c_2) \eta_1^2 \eta_2^2 - \varepsilon^2 c_4 c_2^2 \gamma^2 \eta_2^4.
\end{aligned}$$

Letting $\lambda = \min\{c_3 c_1^2, \varepsilon^2 c_4 c_2^2 \gamma^2\}$, we have

$$\begin{aligned}
\dot{V}(\eta) &\leq -\lambda ||\eta||^4 - (c_1 - \gamma c_2) \varepsilon \eta_1^2 \eta_2^2 \\
\dot{V}(\eta) &\leq -\lambda ||\eta||^4 < 0, \ \forall \frac{\gamma c_2}{c_1} < 1.
\end{aligned}$$

By Theorem 4.2 of [17], we have uniform asymptotic stability for the error dynamics around the NE, implying that the error term $\eta(t) \to 0$ at the rate of

$$||\eta(t)|| \leq \mathcal{K}(\lambda ||\eta(t_0)||, t - t_0)$$

for some $\mathcal{KL}$−class function $\mathcal{K}$ and constant $\lambda = \min\{c_3 c_1^2, \varepsilon^2 c_4 c_2^2 \gamma^2\}$.

Theorem 4 assumes an initialization for ξ that is close to a local equilibrium point ξ^*, which motivates the use of scheduled sampling on the PL model inputs as a pre-training step. We also allow a fully-observed $A_{ij}^{(1)} = 1, \forall i, j$ at $t = 1$ so that the FS observes a full set of features, as a good initialization of the generation process depends heavily on the initially observed features.

5.5 Experiments

To gain intuition into the precision sensing mechanism, we first compare with several classic feature selection schemes on synthetic data. We then compare performance of precision sensing with SOTA active sensing methods on real-world medical data.

5.5.1 Synthetic Experiments

We evaluate the ability of the budget-constrained precision sensing (BCPS) to retrieve the underlying temporal patterns on a set of synthetic time-series data. We generate a synthetic dataset consisting of time-series data with labels: $\{X_i, y_i\}_{i=1}^{1000}$, $X_i \in \mathbb{R}^{100\times 10}$ (100 time-steps, 10 features), $y_i \in \{0, 1\}$. The temporal signals were adapted from the UCI ML dataset [18], specifically the *Pseudo Periodic Synthetic Time Series Data Set*. This dataset features 100,000 samples of univariate time-series data, generated from the stochastic periodic function:

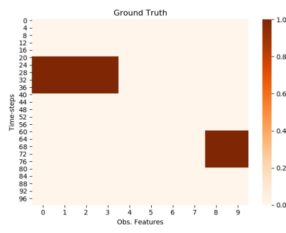
$$f(x) = \sum_{i=3}^{7} \frac{1}{2} \sin\{2\pi(2^{2+i} + \text{rand}(2^i))c\}. \quad (0 \le c \le 1) \tag{5.10}$$

We expand this dataset by applying the generator equation across multiple channels (10), each with a different initialization of c. We sub-sampled 1000 of the generated time-series and aligned the data such that each sample contained 100 time-steps and 10 features. We generate labels y_i according to

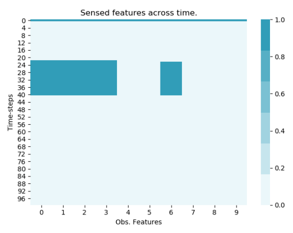
$$y_i = \begin{cases} 1 & ||C(X_i)|| > 0 \\ 0 & ||C(X_i)|| \le 0, \end{cases}$$

where $C(.)$ masks all features of X across time except for certain features at certain times. For example, one configuration of C may emphasize the $0 - 5^{th}$ features from time-steps $20 - 40$ and the $9 - 10^{th}$ features from time-steps $60 - 80$. All other features at all other time-steps are masked with

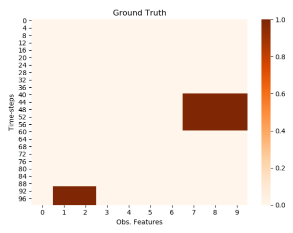
Figure 5.2: Recovery of feature relevance (C) by BCPS on synthetic data.



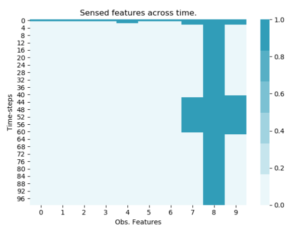
(a) Ground Truth



(b) BCPS



(c) Ground Truth



(d) BCPS

high negative noise sampled from a Gaussian distribution. Our goal is to recover the true underlying C (unmasked features) using the BCPS framework.

5.5.1.1 Recovery of C using BCPS

Table 5.1 gives an overview of the BCPS performance in recovering the relevant features posed by C on the synthetic dataset. Here, the sparsity-level refers to the percentage of masked features in C (ground-truth). We see that the specificity of the recovered features by BCPS remains high, despite the increase in budgetary constraints. In general, the performance drops are not significant between different levels of sparsity in C.

Figure 5.2 visualizes the performance of BCPS on several test-set samples with 80% sparsity in C. We see that BCPS chooses to allocate sensing to early time-steps, and then accurately locates the *largest block* of key observations in C related to predicting the target. From there, BCPS decides whether to continue to observe future time-steps, as shown by Fig. 5.2d, or stop observations when it is confident in the prediction, as shown by Fig. 5.2b. Interestingly, BCPS is capable of locating the key observation blocks in C despite the *discontinuity* of the key block from the original time-steps, as demonstrated by Figs. 5.2b and 5.2d.

Table 5.1: BCPS recovery performance vs. sparsity of sensing budget on synthetic data

Sparsity	AUC	F1	Sensitivity	Specificity
20%	99.0 ± 0.0	98.1 ± 1	97.3 ± 1.1	99.7 ± 0.0
40%	98.2 ± 2.1	94.5 ± 3	90.2 ± 4.2	98.6 ± 1.0
60%	96.4 ± 1.1	91.7 ± 1	87.6 ± 3.1	97.9 ± 1.2
80%	94.4 ± 3.0	90.9 ± 1	84.4 ± 3.8	97.0 ± 1.1

Table 5.2: Comparison of predictive power of various sensing methods on synthetic data, under 80% sparsity in C

Model	AUC	F1	Sensitivity	Specificity
Temporal Feature Selection	89.5 ± 3.9	81.2 ± 4.0	77.6 ± 4.8	85.7 ± 2.8
Self-Attention	93.3 ± 3.4	88.3 ± 3.2	**87.0 ± 3.5**	94.4 ± 2.0
BCPS	**94.0 ± 0.9**	**90.4 ± 0.8**	84.5 ± 3.6	**96.9 ± 1.1**

5.5.1.2 Comparison Against Other Sensing Methods

We compare BCPS against other sensing strategies on the synthetic dataset. Specifically, we investigate (1) classification difference between sensing mechanisms, and (2) how they differ in recovering feature contributions from C. We consider the following alternative models:

- *Temporal Feature Selection*: GRU model with ℓ_1−regularization applied to input weights at each time-step.

- *Attention-Based Sensing*: Self-attention model with a slight modification:

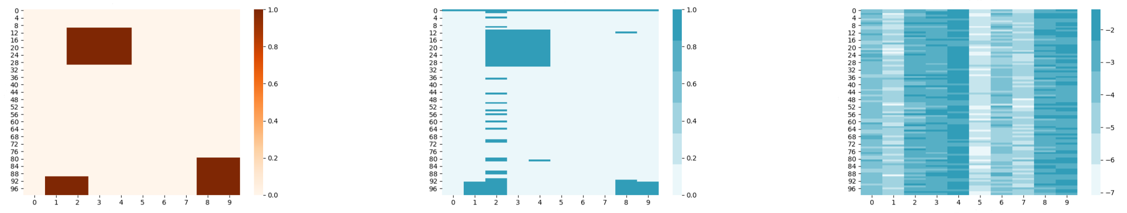
$$A_{i,j,k} = \begin{cases} 1 & A_{i,j,k} \neq 0 \\ 0 & \text{Otherwise.} \end{cases}$$

  Because self-attention produces activations in the continuous $[0, 1]$ range, we make this adjustment to use the non-zero activations as a proxy for feature utilization in self-attention.

Table 5.2 compares the performance of BCPS against other sensing methods on the synthetic data. We fix the sparsity in C to be 80% for all models and compare prediction performance and sensing patterns. As expected, BCPS and self-attention sensing strategies performed notably better than temporal feature selection since the relevant features of C shifts across time.

The difference in the allocation of observations between BCPS and attention-based sensing is relevant, so we further investigated it. In particular we compared the sensing patterns in BCPS and attention activations against the ground truth C under a fixed sparsity pattern, see Table 5.2. Scores are averaged over 5 evaluation runs and reported in the format *Avg.* ± *STD*. We considered a C with some interesting properties:

Figure 5.3: Heatmap comparisons of sensed features under a fixed C. Feature number 0-9 (*x-axis*) vs. Timesteps 0-96 (*y-axis*).



(a) Ground Truth (b) BCPS (c) Attention Activations

- C has *multiple, discontinuous* blocks of key features whose observations are critical to the final prediction of y.
- Different groups of features are prioritized in each key observation block.
- Multiple groups of features can be prioritized at the same time-step.

We see from Fig. 5.3 that *only BCPS* is capable of recovering the true underlying sets of key features in C. The attention activations, on the other hand, cannot locate the discontinuous blocks of contributory features, and as a result always uses a high observation budget, i.e., most features are observed frequently across time, regardless of the true underlying sparsity in C. We also see from Fig. 5.3b that although there exists a very large divide between the first block and future time-steps, the FS model sparingly allocates observations across the large gap of non-contributory features before allocating large amounts of observation resources toward the last few time-blocks.

5.5.2 MIMIC-III Experiments

We also evaluate our framework on **MIMIC-III** [19], a publicly available electronic health records (EHR) dataset using temporal data collected from 22, 830 adult patients. For the prediction task, we choose *In-Hospital Mortality* as the target. Given the first $T = 48$ hours of observations $(x_i^{(1)}, ...x_i^{(T)})$, mortality risk assessment predicts mortality risk from $T+1$ until the end of the hospital stay. Because patients have variable lengths of stay (LOS), we aligned the temporal sequences of patients by the start of their hospital admit time and excluded patients who ha LOS less than 24 hours. We considered a set of 19 temporal features, split between metabolic laboratory panels (7), specialized hematologic tests (6) and vital signs (6).

5.5.2.1 Baseline Classifiers

We consider a wide variety of hypothesis model classes for inference:

- *Logistic Regression (LR)*: non-deep baseline.
- *Multi-layer Perceptron (MLP):* 3 fully-connected layers with 128 hidden units each.
- *Gated Recurrent Units (GRU):* a bi-directional GRU with 2 hidden layers and 128 nodes per layer.
- *Self-Attention Model (Attn):* a modified *self-attention* [20] model, where the inputs at each time-step are modified by an attention mechanism [21] that combines pertinent information from previous time-steps to attend to the most relevant part of the current feature space.

5.5.2.2 Performance on MIMIC-III

Table 5.3 summarizes the performance of the inference models conditioned on the original dataset. Scores are averaged over 5 evaluation runs and reported in the format *Avg.* ± *STD*. We see that that the temporal information is highly relevant to the mortality task. This is indicated by the superior performance of recurrent models (GRU and self-attention GRU), which achieves higher performance across all metrics compared to the non-temporal models (LR and MLP). Self-attention model achieves the highest baseline performance compared to other models, but vanilla GRU results were comparable.

Table 5.3: Baseline performance of inference models conditioned on the original data X

Model	AUC	F1	Sens.	Spec.
MLP	74.0 ± 5.5	37.4 ± 4.2	65.0 ± 7.8	72.8 ± 3.3
LR	58.4 ± 0.7	28.3 ± 1.9	18.8 ± 1.7	**97.9** ± **0.3**
GRU	84.7 ± 1.2	45.2 ± 2.6	**78.4** ± **3.3**	74.8 ± 4.1
Attn	**85.1** ± **1.8**	**45.5** ± **1.6**	76.0 ± 2.2	79.8 ± 2.6

Table 5.4: Performance of inference models conditioned on the masked data $X \odot A$

Model	AUC	F1	Sens.	Spec.
MLP	74.2 ± 5.2	36.2 ± 5.1	69.6 ± 2.8	67.3 ± 6.2
LR	57.3 ± 0.7	25.6 ± 1.9	16.5 ± 1.5	**98.2** ± **0.2**
GRU	81.3 ± 2.2	42.4 ± 2.5	73.0 ± 5.4	74.4 ± 4.3
Attn	81.7 ± 1.2	41.1 ± 3.0	**75.5** ± **3.2**	72.6 ± 1.4
PL	**82.9** ± **1.3**	**44.5** ± **2.4**	74.3 ± 3.5	74.1 ± 1.2

5.5.2.3 Performance on Masked Data

Table 5.4 illustrates the performance of the same set of inference models trained and tested on the incrementally sensed data $\{X_i \odot G(X_i), y_i\}_{i=1}^m$. The $G(X)$ used for Table 5.4 is trained using a budgetary constraint of 58.6% of available features across time. From Table 5.4, we see that both recurrent models (GRU and Attn) have notable decrease in performance, with their AUCs dropping by 2 – 4%, respectively. Conversely, the non-temporal models maintain their performance. Again, scores are averaged across 5 evaluation runs. This contrast between temporal vs. non-temporal models reveals that $G(X)$ specifically maintains the *temporal patterns* in X related to the prediction task at hand. We also see that the PL is able to achieve the highest performance on the sensed data, despite having the same architecture as the GRU models. This is likely because the PL is trained on a variety of sparsity patterns in $G(X)$ during co-training.

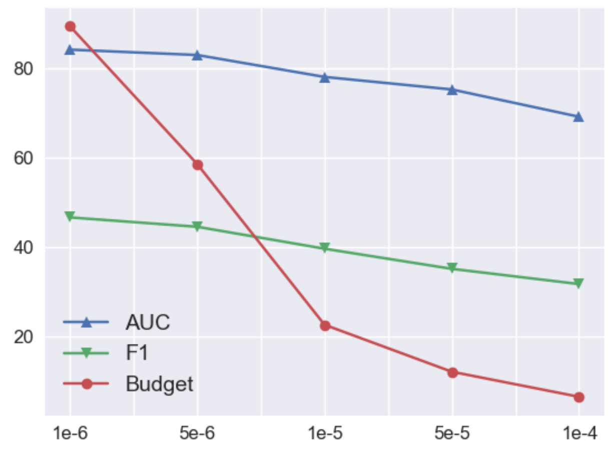
5.5.3 Sparsity Trade-offs

We further investigate the relationship between the severity of the budget constraint, i.e., sparsity in the generated sensing tensor, and the performance of the resulting dynamically sensed data. Here, we define *budget cost* as the % of features utilized during the observation period ($T = 48$ hours):

$$\textsc{BudgetCost} = \sum_i^m \sum_j^t \sum_k^n A_{i,j,k} / (m \times t \times n),$$

where we assume $X \in \mathbb{R}^{m \times t \times n}$. Fig. 5.4 summarizes the trade-off between performance and budget constraints. Notably, the budget cost decreases from 89.3% to 58.3% while the AUC and f1-scores only decreased by < 0.02. The greatest drop in performance occurred from 12.1% $\rightarrow$ 6.5%. This is presumably because at 6.5%, the FS allocates the observation budget at the first time-step only, sacrificing all of the temporal information within the data. Thus, we can see from Fig. 5.4 that $\beta \in$ [1e-6, 1e-4] represents an optimal trade-off range. In practice, the β-constraint is usually

Figure 5.4: Trade-off between Budget Constraint (x-axis) vs. predictive power (y-axis). *Budget* denotes the strength of budgetary constraint β.



predetermined based on real-world constraints. For example, a well-funded hospital may have access to a large set of observations, say $\beta = 5e - 6$ (less constrained), whereas an under-funded clinic may only be able to observe at the level of $\beta = 5e - 5$ (highly constrained).

5.5.4 Comparison Against SOTA Methods

Finally, we compare BCPS against other SOTA sensing strategies for temporal data:

- Deep Sensing [22]: Active sensing is done through an adaptive sampling scheme. Inference is done using *interpolation* and *imputation* layers for the missing values.

- Dynamic Measurement Scheduling [23]: Active sensing is done by a dueling DQN network, and inference is done using a pre-trained recurrent network.

We implement Deep Sensing (both single and multiple model settings) with the hyperparameter settings described in the Appendix Section of [22]. For Dynamic Measurement Scheduling, we run Algorithm 3 without meta-data, i.e., without age, demographics, or disease history, to obtain the simulated dataset. In both cases, we use the same set of features and the same 48-hour observation window. For each active sensing model we peform the following:

1. Pre-train a GRU model.

2. Synthesize new datasets using the active sensing strategy.

3. Test inference models (M-RNN for Deep Sensing) on synthetic datasets.

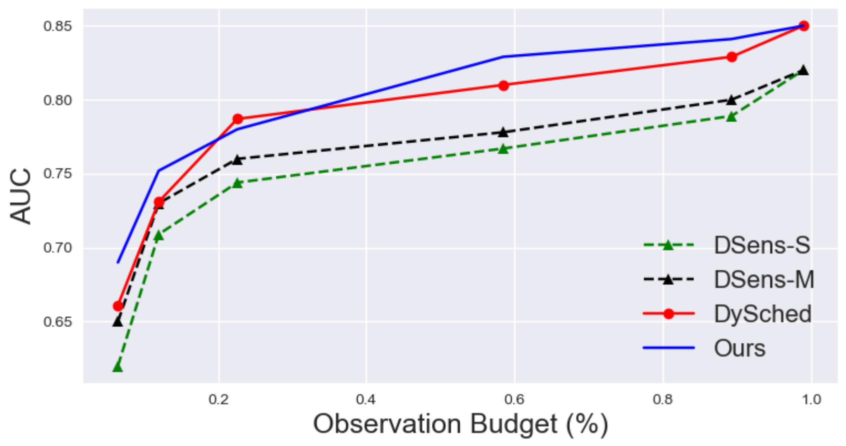
In Table 5.5, we report the best performance for each method with $50-60\%$ observation budget, which is the reported low end for the Deep Sensing work [22]. We use the $50-60\%$ budget range because each work uses its own hyperparameter (e.g., threshold τ in [22] and λ in [23]), so it is difficult to translate the threshold parameters to exact percentages in sparsity. Thus, we report the best performance within the $50-60\%$ observation range of each model for fairness. From Table 5.5, we can see that the combination of FS (for data generation) and PL (for adaptive classification) maintains the best level of performance at higher budget constraints. Dynamic scheduling produces comparable results to Deep Sensing.

We also plot the AUC-gain vs. sparsity trade-off for each approach in Figure 5.5. For that experiment, we fix a pre-trained GRU to deliver the predictions at each sparsity level for the generated data from each method. Our FS model produces measurements that maintained a higher level of performance at each sparsity level. It is also notable that without variance reduction (e.g., using multiple models), the imputation-prediction (M-RNN) framework of Deep Sensing has a large drop off, especially with decreasing observation budget. Dynamic Scheduling performs best at around 25%, which is consistent with the reported performance in [23]. However, we note that its performance plateaus at increasing levels of observation.

Table 5.5: Comparison of precision sensing against SOTA active sensing method with sparsity constraint of $50-60\%$

Sensing Model	**Inference**	**AUC**	**F1**
Deep Sensing [22]	M-RNN	76.0	38.5
Dynamic Scheduling [23]	GRU	78.7	41.3
Ours (FS)	**Ours (PL)**	**82.9**	**44.5**

Figure 5.5: Comparison of AUC-gain against SOTA models.



5.6 Conclusion

In this chapter we formalize the precision sensing problem as an extension of active sensing and illustrated its application in mortality risk prediction. Our formulation consists of a minimax game between a dynamic feature sensor and an online classifier that reaches a local Nash equilibrium during co-training. We empirical show that the sensing strategy is capable of maintaining predictive performance while satisfying budget constraints. However, one limitation is that although we show local Nash equilibriums exist, we cannot make statements about their admissibility and interchangeability in payoff values [16]. Future works might consider other scheduled sampling techniques for the PL model, which can greatly affect training stability and sample complexity under the proposed setting.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Yu Cheng, Fei Wang, Ping Zhang, and Jianying Hu. Risk prediction with electronic health records: A deep learning approach. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 432–440. SIAM, 2016.

[2] Fengyi Tang, Cao Xiao, Fei Wang, and Jiayu Zhou. Predictive modeling in urgent care: a comparative study of machine learning approaches. *JAMIA Open*, 2018.

[3] Abhyuday N Jagannatha and Hong Yu. Bidirectional rnn for medical event detection in electronic health records. In *Proceedings of the conference. Association for Computational Linguistics. North American Chapter. Meeting*, volume 2016, page 473. NIH Public Access, 2016.

[4] Edward Choi, Mohammad Taha Bahadori, Elizabeth Searles, Catherine Coffey, Michael Thompson, James Bost, Javier Tejedor-Sojo, and Jimeng Sun. Multi-layer representation learning for medical concepts. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1495–1504. ACM, 2016.

[5] Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *arXiv preprint arXiv:1703.07771*, 2017.

[6] Shipeng Yu, Balaji Krishnapuram, Romer Rosales, and R Bharat Rao. Active sensing. In *Artificial Intelligence and Statistics*, pages 639–646, 2009.

[7] Mengying Sun, Fengyi Tang, Jinfeng Yi, Fei Wang, and Jiayu Zhou. Identify susceptible locations in medical records via adversarial attacks on deep predictive models. *arXiv preprint arXiv:1802.04822*, 2018.

[8] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[9] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[10] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[11] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[12] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179, 2015.

[13] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[14] John Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.

[15] Ky Fan. Fixed-point and minimax theorems in locally convex topological linear spaces. *Proceedings of the National Academy of Sciences of the United States of America*, 38(2):121, 1952.

[16] João P Hespanha. *Noncooperative game theory: An introduction for engineers and computer scientists*. Princeton University Press, 2017.

[17] Hassan K Khalil. *Nonlinear control*. Pearson New York, 2015.

[18] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

[19] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3, 2016.

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

[21] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[22] Jinsung Yoon, William R Zame, and Mihaela van der Schaar. Deep sensing: Active sensing using multi-directional recurrent neural networks. 2018.

[23] Chun-Hao Chang, Mingjie Mai, and Anna Goldenberg. Dynamic measurement scheduling for event forecasting using deep rl. *arXiv preprint arXiv:1901.09699*, 2019.

CHAPTER 6

PERSONA AUTHENTICATION

6.1 Introduction

In recent years, one promising approach to diverse and personalized dialog generation has been *persona* models [1, 2, 3] which embed the so-called "persona" information (e.g., name, gender, and self-descriptions) into neural conversational agents. While the goal of persona modeling is to achieve human-level response diversity and character consistency, a critical yet often overlooked factor is the *sequence of prompts* used to induce diversity in generated responses.

Consider the toy example in Table 6.1. Certain sequences of question prompts (from the verifier) create a trail of generated responses that reveal more persona information, compared to non-specific conversation (the random policy). Yet it is unclear *a priori* which sequence(s) of questions most effectively reveal the dialog agent's underlying persona. Moreover, a set of questions may be effective for one persona but fail for others. Currently, long interactions with humans are necessary to gain insight into persona model characteristics such as authenticity [3, 4], diversity [2], and engagement [3].

In this work we present a learning approach for interacting with conversational agents. Specifically, we introduce the **persona authentication** problem, where a model estimates the persona information of an input agent by learning to deliver a sequence of questions that progressively reveal more information about the agent throughout the course of the dialog. This is difficult because exact search through the space of possible question sequences is infeasible. Therefore a model must adaptively prune its set of potential questions based on the dialogue agent's responses.

We further decompose persona authentication into two parts: *persona identification*, which is inferring a set of persona features from a given dialog trajectory, and *persona verification*, the problem of finding a second conversational model – we call it a *question policy* – to elicit dialog trajectories for persona identification. To address the intractability of exact search through the space

Table 6.1: Persona model responses can differ greatly depending on input questions

Persona Descriptions		
1. I am a construction worker.	2. I enjoy building houses.	3. I have 5 cats.
4. my cats are very special to me.		

Role	**Response**	**Role**	**Response**
Verifier	hello! what kind of work do you do?	Random	hello how are you today?
Model	i build houses.	Model	great! i just got back from work.
Verifier	that's awesome. what do you do outside of work?	Random	me too. i'm a teacher at a high school.
Model	i like to spend time with my cats.	Model	cool, what grade do you teach?

of dialog trajectories, we introduce a computationally tractable algorithm and show its asymptotic convergence (in cumulative conversations) toward the full persona identification objective. The key contributions of this work can be summarized as follows:

- We introduce the *authentication* loss and show that estimators trained to convergence under this objective maximize the mutual information between dialog history and persona.
- Based on the authentication loss we learn a dialog verification model that effectively generates question sequences to distinguish the persona of input models. Empirically, we show that the question policies of the verification model adapt to out-of-distribution personas.
- We present a way to incorporate question policies into language model (LM) based dialog models, e.g., GPT-2, without sacrificing the felicity and consistency of the original LM model.

6.2 Why Model Persona?

One of the long-term goals of this work is toward general conversational authentication. Biometric authentication has traditionally relied on physical measures such as fingerprints and facial recognition to determine the identity of human beings [5, 6]. Importantly, physical features such as facial structures and fingerprints are largely stationary; a standard procedure can be used to collect stationary features and scaled to entire populations without the need for personalization. By contrast, "soft" biometrics such as using personal information to verify identity require some degree of personalization. For example, password retrieval or reset procedures often ask users to answer security questions for which they design, or to answer a set of adaptive tests such as visual Captchas or re-tagging photos. While these tests are personalized, they are not *a*daptive in the sense that the test itself is predetermined before it is delivered to the user.

To this end, conversational authentication can be viewed as an *a*daptive extension of traditional biometrics based authentication. In particular, we are interested in finding an algorithmic approach to deliver automated and adaptive tests to survey whether the conversational behavior of an input agent (e.g., human users, spam bots, etc.) matches a set of personal information provided by the user (which roughly corresponds to "persona facts" in the PersonaChat dataset). Given such an authenticator, we can explore a much larger space of tests (e.g., the space of adaptive questions rather than the space of predefined personalized questions) to survey more complicated behavior. This is especially relevant if we want to generalize beyond simple persona facts such as "I have a dog named radar" to more complicated facts such as the capacity to deliver satirical responses or the use of specific generational grammatical structures (e.g., millennial or "Gen Z" lexicon). In other words, we see general conversational authentication as the linguistic analog of visual captcha turing tests. Toward this goal, persona authentication – learning a policy to deliver identifying questions – is a necessary stepping stone because it presents an algorithmic approach (through dialog policy) as opposed to human interviewers using handcrafted linguistic features, which is the only alternative at the moment, and does not scale.

6.2.0.1 Applications

One potential application of conversational authentication is *s*peaker verification for conversational agents, human or chatbot. In many real-world settings, speaker information such as audio and video may not be readily available. In such cases, the verifier network provides a way of speaker identification via text. One can think of persona verification as a way of obtaining a linguistic "fingerprint" of speakers based on the manner in which they converse under different question policies. For example, human speakers seeking access to personal data may go through a short conversation with the authenticator in order to see whether the person trying to access private data has the correct identity.

Verification is a critical issue in the modern era of cybersecurity. Consider for example the arrival of *Deep Fakes* [7] – synthetically generated videos of people doing actions that may be outside the context of their persona. The use of only voice and video identifiers may not be enough to truly assess whether a person's physical features match with their actions. In this regard, we try to introduce the idea that the problem of verification may entail much more than just matching physical / biological features. This is why we approach the problem from the point of a *question policy*, a verification process that is dynamic and stochastic rather than static and deterministic. In the latter case, technical advances in modern AI can "game" physical features which are fixed points in some classifier space. In the former case, however, an impersonator must do much more. To fool a dynamic authentication procedure, one has to find fixed points in *policy space*, which involves sequential decision-making rather than one-time classification using facial recognition and fingerprint features.

6.2.0.2 Authenticator Example

To give a concrete example of a use case, we provide an example conversation of conversational authentication in which an authenticator surveys the persona facts relevant to a user who engages in an 8 turn conversation. In this example, the user is a persona model (personalized chatbot) rather than a human user. Table 6.2 shows an example conversation between the learned authenticator model and an input persona model (PersonaGPT). Turn 1 and 8 highlight the importance of incorporating question codes as context for decoding: rather than hard-coding questions, an authenticator must also answer questions from the PersonaGPT model as to not disrupt the flow of conversation. Since the authenticator is not given any persona facts, it must make up an answer to PersonaGPT questions on the fly. For example, the authenticator states that its favorite color is green and that it is unemployed in order to answer queries in addition to proposing questions.

In this particular conversation, the authenticator question codes are: *talk about work* for the first 3 turns, *ask about hobbies* for the next 2 turns, *talk about politics* for turn 6, *talk about travel* for turn 7, and *talk about age and gender* in turn 8. When question codes are repeated, the authenticator model is able to avoid the same question twice – instead, it expands upon the existing knowledge in the conversation (e.g., ex-military, works in a shipyard). Note that when as context, the question codes function as one-to-many mappings. For example, "talk about travel" may correspond to a much larger set of topics other than traveling as a hobby. The authenciator may ask where the input agent lives, where they are from, etc. Similarly, "ask about age and gender" has much more to do with questions related to identity and appearance rather than strictly age and gender alone.

Table 6.2: An example of interaction between the learned authenticator policy and PersonaGPT over an 8-turn conversation

Role	Response
Authenticator	hello there what kind of work do you do?
PersonaGPT	hi there i am ex military. what do you do for a living?
Authenticator	nothing much. what do you like about your job?
PersonaGPT	i am a shipyard worker.
Authenticator	what's your dream job?
PersonaGPT	to be a navy seal one day.
Authenticator	that sounds exciting! do you have any hobbies?
PersonaGPT	my hobby is volunteering with the bunny rescue.
Authenticator	do you have any other hobbies?
PersonaGPT	yes i help out at the local bunny rescue.
Authenticator	do you follow politics?
PersonaGPT	yes i follow politics. i donate a part of my salary to charity each month.
Authenticator	where are you from?
PersonaGPT	i'm from the united states. what's your favorite color?
Authenticator	my favorite color is green. how old are you?
PersonaGPT	i'm 20 what about you?

Persona Facts	
1. I donate a part of my salary to charity each month.	2. I work fulltime in a shipyard.
3. I volunteer my time with a local bunny rescue.	4. I'm ex military.
5. My favorite color is navy blue.	

6.2.0.3 Challenges

Table 6.2 reveals two main challenges to learn an adaptive authentication policy:

1. At the turn-level, a policy must be able to incorporate turn-level goals into its decoding. However, the decoding algorithm has to remain open-domain in order to be flexible enough to handle a wide range of conversational topics and styles. For this reason, we turn to generative decoding rather than ranking models.

2. An objective function has to capture the relationship between the responses generated by a policy and persona facts at the dialog-level. One can think of the dialog-level objective function as informing turn-level goals to guide the decoding process.

6.3 Problem Setting

6.3.1 Notation

Let $\mathcal{D} = \{\tau_i\}_{i=1}^n$ be a set of dyadic dialog samples. Each dialog follows the form $\tau = \{X, Y\}$, where $X = (X_t)_{t=1}^T$ denotes the sequence of *source responses* and $Y = (Y_t)_{t=1}^T$ denotes the sequence of *target responses*. Each response is composed of a sequence of tokens, represented as $(\mathbf{x}_k^{(t)})_{k=1}^K$ (source tokens) and $(\mathbf{y}_k^{(t)})_{k=1}^K$ (target tokens). To be consistent with state-of-the-art (SOTA) dialog model decoders [8, 9], we use Byte-Pair Encoding (BPE) [10] for tokenization. Additionally, T signifies the maximum number of turns in a dialogue sample, K the maximum number of tokens per response. If a response consists of $k < K$ tokens, then we take $k + 1$ through K tokens to be empty ("PAD") tokens (with similar logic applying to dialog turns). As a shorthand, we write τ_t to denote the *dialogue trajectory* $(X_1, Y_1, \ldots, X_t, Y_t)$ up to turn t, with $Y_{1:t}$ to signify the sequence of responses $Y_1, \ldots, Y_t$. Similarly $\mathbf{y}_{1:k}$ represents the ordered sequence of tokens up to token $\mathbf{y}_k$.

6.3.2 Persona Identification

The standard objective of persona models is:

$$\max_{Y_t} \ \log p(Y_t | X_t, \tau_{1:t-1}, P_Y), \tag{6.1}$$

where P_Y is the set of persona descriptions for the dialog agent. Zhang et al. [2] and the ConvAI2 challenges [11] provided numerous ways to incorporate persona information into the dialogue generation process. Recently, generative persona models [12, 8] have been shown to be effective at contextualized decoding by incorporating persona P as language model context. Due to their effectiveness, we only consider generative persona models in this chapter.

To identify a persona from a given trajectory, we formulate the *persona identification* problem:

Problem 2. *Persona Identification. Given an input dialogue trajectory τ, find the persona P that maximizes the mutual information between P and τ. More formally, the optimization objective is*

$$\max_P \; I(P, \tau) = \max_P \; H(\tau) - H(\tau|P) = \min_P \; H(\tau|P), \tag{6.2}$$

where $H(\cdot)$ is entropy and $P \in \mathbb{R}^m$ is a vector in the space of possible personas.

Persona identification seeks a fixed-length representation of persona information that captures the consistency of generated responses. In other words, a personalized dialog agent has to not only generate *diverse* responses (high entropy $H(\tau)$), but it must also stay *consistent* to a persona profile throughout multiple turns of conversation, minimizing $H(\tau|P)$. One challenge is that it is unclear how to arrive at a set of questions $X_{1:T}$ to generate the input trajectory τ. For example, certain sets of questions may always result in generic responses, regardless of the agent quality. Thus, problem 2 requires a way to constrain the *question policy* so that, given the right set of questions, the persona of the dialog agent can be elicited.

6.3.3 Persona Verification

We addresses the above issue by formulating question generation as an optimization problem. A naive attempt may be to propose the following objective:

$$\max_\tau I(\tau, P) = \max_\tau H(\tau) - H(\tau|P) \tag{6.3}$$

However, we will show in the following section that Eqn (6.3) is intractable to optimize directly due to the partition function in the second term. We will first present a more feasible objective below and show its asymptotic convergence toward Eqn (6.3) in the next section. Suppose that the inputs to a verification model are dialog agents, i.e. trained persona models parameterized by different personas. We define the *persona verification* problem as follows:

Problem 3. *Persona Verification. Given a space of persona information $\mathcal{P}$, persona verification is the optimization objective:*

$$\min_{\theta} \mathbb{E}_{P \sim \mathcal{P}} \big[\mathcal{L}(\tau_\theta, P) \big] \tag{6.4}$$

where $\mathcal{L}(\cdot, \cdot)$ is the authentication loss:

$$\mathcal{L}(\tau_\theta, P) = \max\{0, C + d(\tau_\theta, P^+) - d(\tau_\theta, P^-)\} - \log p(\tau_\theta)). \tag{6.5}$$

P^+ denotes persona facts that co-occur with trajectory τ, P^- the opposite. C specifies the desired margin of separation, and τ_θ is the dialog trajectory generated by the question policy (θ).

The first term of Eqn. (6.5) approximates Eqn. (6.2) through a triplet loss using negative sampling over the space of possible personas. The rationale behind the first term is to address the intractability of solving for Eqn. (6.2) directly. In Section 6.3.4, we show that this triplet loss component converges to the mutual information term in Eqn. (6.2).

The second term in Eqn. (6.5) gives the likelihood of the trajectory. In order to minimize the second term, a verification algorithm has to generate queries with high likelihood under a given language model, e.g. GPT-2. If either the input agent or the question policy generate nonsensical responses, then the resulting $p(\tau_\theta)$ will be close to zero. For the rest of the chapter, we will refer to "identifier" as a model used to solve the identification problem and "verifier" to denote a model used to solve the verification problem.

6.3.4 Analysis of Persona Authentication Objective

Now we analyze the relationship between Eqn. (6.5) and the mutual information between P and τ. First, we assume that for a given persona P, the density function for $p(\tau|P)$ follows the probability density function (PDF) of a Gibbs distribution:

$$p(\tau|P) = \frac{\exp[-\beta E(\tau, P)]}{\int_{\tau' \in \mathcal{D}} \exp[-\beta E(\tau', P)]},$$

where $E(\tau, P)$ is an energy function which scores the un-normalized co-occurrence likelihood of a specific dialogue trajectory τ and persona P. β is the temperature term which controls the overall entropy of the distribution. We choose the Gibbs distribution because of its expressiveness and common use in contrastive learning [13]. Then we can express the mutual information between τ and P as:

$$I(\tau, P) = H(\tau) - H(\tau|P) = -\mathbb{E}_\tau[\log p(\tau)] + \mathbb{E}_{\tau,P}\left[\beta E(\tau, P) - \log \int_{\tau' \in \mathcal{D}} \exp[-\beta E(\tau', P)]\right]. \tag{6.6}$$

In Eqn. (6.6), the first term on the RHS corresponds to the entropy of dialog trajectories (diversity of generated responses), which is determined by the decoding quality of the input agent. The second term depends on our question policy and our estimation of P. Trajectories under P depend on the question policy since the input agent maximizes $p(Y_t|X_t, \tau_{t-1}, P)$. Since P is not known by the policy beforehand, it is estimated each turn by the identifier. Unfortunately, directly estimating the entire second term is difficult – the partition function of the conditional distribution requires us to integrate over the space of trajectories, an intractable task.

We thus propose a local density estimation of the conditional density $p(\tau|P)$ as follows: let

$$\widetilde{P}_N = \frac{1}{nV(C_n)} \sum\nolimits_{i=1}^{n} K(\varphi(\tau_i), \psi(P), C_n), \tag{6.7}$$

denote an empirical estimate of $p(\tau|P)$ using n sampled trajectories. ψ and φ are embedding representations of P and τ, respectively. $V(C_n) = \int_{\mathcal{N}_{C_n}(P)} dP'$ gives the volume of a neighborhood ball of radius C_n around $\psi(P)$. K is a kernel function (which we show to be a valid kernel function in Supplemental Materials) designed as follows:

$$K(\tau, P, C_n) = \begin{cases} 1 & \text{if } d(\varphi(\tau), \psi(P)) \le C_n \\ 0 & \text{else} \end{cases}. \tag{6.8}$$

We now present the main theorem of our analysis.

Theorem 3. *(Convergence of $\widetilde{P}_N$)*

If Eqn. 6.5 (authentication loss) is minimized with 0 *loss over* $\mathcal{D} = \{\tau_i\}_{i=1}^n$ *and* $\mathcal{P} = \{P_j\}_{j=1}^m$*, then*

$\widetilde{P}_N$ asymptotically converges to $p(\tau|P)$, i.e.,

$$\lim_{n\to\infty} \widetilde{P}_N = p(\tau|P) \tag{6.9}$$

when the following conditions hold:

$$\lim_{n\to\infty} nV(C_n) = \infty, \quad \lim_{n\to\infty} V(C_n) = 0, \quad \lim_{n\to\infty} \frac{k}{n} = 0, \tag{6.10}$$

where k is the expected number of samples that fall within $N_{C_n}(P)$.

The proof is provided in the Supplemental Materials. The goal of the identifier model is to learn the embedding functions φ and ψ:

$$\varphi, \psi = \arg\min_{\varphi,\psi} \frac{1}{nk} \sum_{i=1}^{n} \sum_{j=1}^{k} \max\ \{0, C + d(\varphi(\tau), \psi(P_i)) - d(\varphi(\tau), \psi(P_j))\}, \tag{6.11}$$

where $P_i \in \mathcal{P}^+$, and each P_j belongs to the set of k negative persona samples.

6.4 Proof of convergence of Thm. 1

We first state some assumptions about the density function $p(\tau|P)$. Unless otherwise stated, we assume that there exists some joint embedding space $\mathcal{H}$ for which we are comparing $\tau \in \mathcal{D}$ and $P \in \mathcal{P}$. Specifically, let us assume that here exists some optimal mapping functions φ^* and ψ^* that maps τ and P to $\mathcal{H}$, respectively, i.e.,

$$\varphi : \mathcal{D} \to \mathcal{H},\ \psi : \mathcal{P} \to \mathcal{H}$$

where $\mathcal{H} \subseteq \mathbb{R}^n$. With some abuse of notation, we refer to $\varphi(\tau)$ by τ and $\psi(P)$ by P in the following analyses for simplicity.

Assumption 1. *(Locally Constant Density)*
We assume that within a local neighborhood $\mathcal{N}_C(P)$ of radius C around persona vector P, trajectories τ are indistinguishable. Formally, $\forall P, P', \exists 0 < C < \infty$:

$$d_\psi(P, P') \le C \implies d_\varphi(p(\tau|P'), p(\tau|P)) = 0,$$

for some distance functions d_ψ, d_φ. For simplicity, we will consider the Euclidean distance for d_ψ and total variational divergence for d_φ.

We will use $\mathcal{N}_C(P)$ to denote the neighborhood set around P for which the above condition is satisfied.

Assumption 2. *(Continuity and topological properties)*
The conditional density $p(\tau|P)$ is Lipschitz continuous over the supporting set $\mathcal{H}$ for both τ and for P. Furthermore, we assume that $p(\tau|P)$ is simply-connected.

Next, we define $p(\tau|\mathcal{N}(P))$ as the probability that trajectory τ will fall in the neighborhood $\mathcal{N}(P)$ around a given persona P. Specifically, we consider the case where n trajectories are sampled, k of which fall into $\mathcal{N}(P)$.

Definition 5. *(Neighborhood Density) We define the neighborhood density around a persona vector P as the probability that a trajectory τ falls into the neighborhood $\mathcal{N}(P)$ as defined by*

$$P_{\mathcal{N}} = p(\tau|\mathcal{N}(P)) = \int_{\mathcal{N}(P)} p(\tau|P')dP'. \tag{6.12}$$

Furthermore, given a set of i.i.d. n trajectories $\{\tau_1, \ldots, \tau_n\}$, the probability that k such trajectories fall in $\mathcal{N}(P)$ follows the binomial distribution:

$$k \sim \binom{n}{k} {P_{\mathcal{N}}}^k (1 - P_{\mathcal{N}}))^{1-k}. \tag{6.13}$$

At this point, there is one key issue: how do we calculate k, which needs to somehow "count" the trajectory-persona pairs that fall into the same neighborhood? We can conceptualize k as the image of some counting function of the form

$$K : (\tau, P, C) \to \mathbb{R}$$

where K is normalized over the domain $\mathcal{H}$. For this purpose, we construct a kernel density function for k as follows: given a persona vector P, let

$$k_n = \sum_{i=1}^{n} K(\tau_i, P, C_n)$$

be the output of the kernel function K over n sampled trajectories $\mathcal{D} = \{\tau_1 \ldots \tau_n\}$ from $p(\tau|P)$. Here, C_n denotes the *sample* neighborhood size $\mathcal{N}(P)$ around P satisfying the constraint

$$C_n = \max_{\tau_i, \tau_j \in \mathcal{D}} d(\tau_i, \tau_j)$$

for Euclidean distance $d(\cdot, \cdot)$ from Assumption 1. Given embeddings $\varphi(\tau)$ and $\psi(P)$, we propose the following kernel density function $K(\tau, P, C_n)$:

$$K(\tau, P, C_n) = \begin{cases} 1 & \text{if } d(\varphi(\tau), \psi(P)) \leq C_n \\ 0 & \text{else} \end{cases}. \tag{6.14}$$

Lemma 3. *(Validity of the proposed kernel density)*
Let $V(C) = \int_{\mathcal{N}_C(P)} dP'$ denote the volume of the neighborhood with radius C around P. The counting function K described by Eqn. (6.14) is a valid kernel density function satisfying

$$\forall n > 0, \forall \tau, P \in \mathcal{P} : \; K(\tau, P, C_n) \geq 0 \tag{6.15}$$

$$\forall n > 0 : \frac{1}{V(C_n)} \int_{\mathcal{H}} K(\tau', P, C_n) d\tau' = 1. \tag{6.16}$$

Proof. Condition 6.15 is observed by the definition of K from Eqn. 6.14. $K(\tau, P, C) > 0$ over the entire supporting set for τ, P and constants C_n and 0 everywhere else. For condition 6.16, see that K integrates to $V(C)$ over the domain of τ:

$$\int_{\mathcal{H}} K(\tau', P, C) d\tau' = \int_{\mathcal{N}_C(P)} 1 \cdot dP' = V(C). \qquad \text{(by definition in Eqn. (6.14))}$$

From our construction of K, we know that $K(\tau, P, C_n) = 0$ everywhere except in neighborhood $\mathcal{N}_C(P)$. Thus, the integral $\int_{\mathcal{H}} K(\tau', P, C) d\tau'$ reduces to integrating over $\mathcal{N}_C(P)$. □

We now present the main theorem of our analysis. First, let us denote φ^*, ψ^* as functions satisfying the empirical objective:

$$\varphi^*, \psi^* = \arg\min_{\varphi, \psi} \frac{1}{nk} \sum_{i=1}^{n} \sum_{j=1}^{k} \max \; \{0, C + d(\varphi(\tau), \psi(P_i)) - d(\varphi(\tau), \psi(P_j))\}. \tag{6.17}$$

Theorem 4. *(Convergence of $\widetilde{P}_N$)*

Let $P_{\mathcal{N}}$ be the empirical estimate of P_N using the kernel density estimator:

$$\widetilde{P}_{\mathcal{N}} = \frac{1}{nV(C_n)} \sum_{i=1}^{n} K(\tau_i, P, C_n). \tag{6.18}$$

If Eqn. (6.17) (authentication loss) is satisfied with 0 *loss over $\mathcal{D} = \{\tau_i\}_{i=1}^{n}$ and $\mathcal{P} = \{P_j\}_{j=1}^{m}$, then $\widetilde{P}_N$ asymptotically converges to $p(\tau|P)$, i.e.,*

$$\lim_{n\to\infty} \widetilde{P}_N = p(\tau|P) \tag{6.19}$$

when the following conditions hold:

$$\lim_{n\to\infty} nV(C_n) = \infty, \quad \lim_{n\to\infty} V(C_n) = 0, \quad \lim_{n\to\infty} \frac{k}{n} = 0, \tag{6.20}$$

where k is the expected number of samples that fall within $N_{C_n}(P)$.

Proof. From Eqn. (6.13), we see that $k \sim \text{Binomial}(n, P_{\mathcal{N}})$. Thus, we have $\mathbb{E}[k] = nP_{\mathcal{N}}$, where $\mathbb{E}[k]$ is the expected number of samples k that fall within $\mathcal{N}_{C_n}(P)$ from a random sample of n trajectories. Observe that:

$$\begin{aligned} P_{\mathcal{N}} &= \int_{\mathcal{N}_C(P)} p(\tau|P')dP' = \int_{\mathcal{N}_C(P)} p(\tau|P)dP' && \text{(by Assumption 1)} \\ &= p(\tau|P) \int_{\mathcal{N}_C(P)} dP' = p(\tau|P) \cdot V(C). && (6.21) \end{aligned}$$

Additionally, if the authentication loss in $Eqn.$ (6.17) is satisfied with 0 loss, then we have

$$K(\tau, P, C) = 1 \iff d(\varphi(\tau), \psi(P)) < C \iff \tau \in N_C(P) \iff k = 1.$$

By the strong law of large numbers, we have

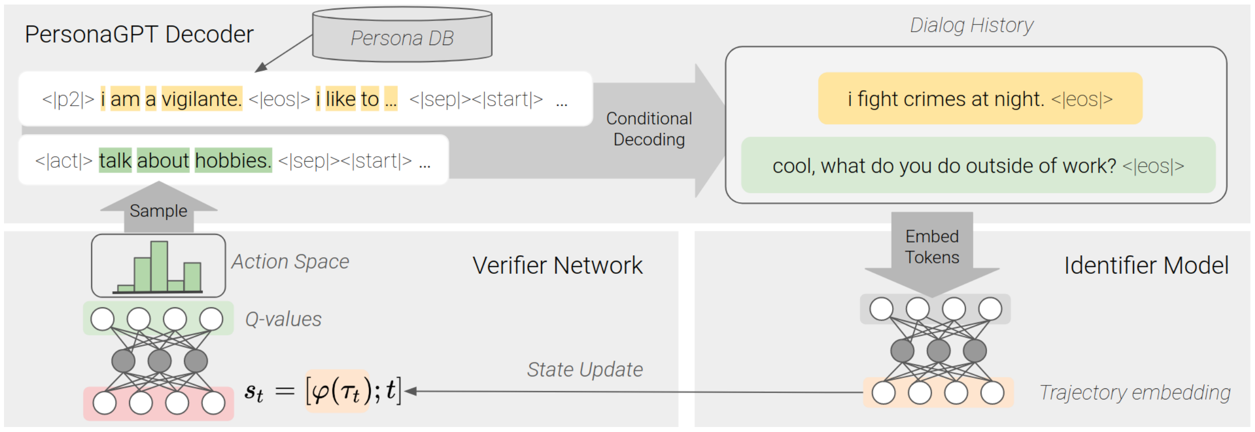
$$\begin{aligned} \lim_{n\to\infty} \sum_{i=1}^{n} K(\tau_i, P, C_n) &= \mathbb{E}_\tau[K(\tau, P, C_n)] = \int_{\mathcal{H}} K(\tau, P, C_n) \cdot p(\tau|P)d\tau \\ &= \int_{\mathcal{H}} k \cdot p(\tau|P)d\tau = \mathbb{E}[k]. && (6.22) \end{aligned}$$

For a given n, we can combine $\mathbb{E}[k] = nP_{\mathcal{N}}$ with Eqns. (6.21) and (6.22) to obtain:

$$\frac{\mathbb{E}[k]}{n} = P_{\mathcal{N}} = p(\tau|P) \cdot V(C_n).$$

$$\implies \lim_{n\to\infty} \widetilde{P}_N = \frac{\mathbb{E}[k]}{nV(C_n)} = p(\tau|P).$$

Figure 6.1: Overview of authentication pipeline.


PersonaGPT Decoder
Persona DB
<|p2|> i am a vigilante. <|eos|> i like to ... <|sep|><|start|> ...
<|act|> talk about hobbies. <|sep|><|start|> ...
Conditional Decoding
Dialog History
i fight crimes at night. <|eos|>
cool, what do you do outside of work? <|eos|>
Sample
Action Space
Verifier Network
Q-values
State Update
$s_t = [\varphi(\tau_t); t]$
Embed Tokens
Identifier Model
Trajectory embedding

Although $V(C_n) \to 0$, the first condition in Eqn. (6.20) guarantees that $n \to \infty$ faster. Intuitively, $V(C_n) \to 0$ means that the size of the contrastive margin should shrink with increasing number of training samples, but the rate of shrinking must be slower than the n. Similarly, $k \to \infty$ since the number of observed trajectories falling into $\mathcal{N}_C(P)$ increases monotonically with n. However, the third condition in Eqn. (6.20) ensures that $P_{\mathcal{N}}$ converges at all. $\square$

6.5 Methodology

Figure 6.1 summarizes the key components of our authentication pipeline. The identifier model represents the current conversation history as state input to the verifier. The verifier outputs a distribution $\pi(\cdot|s_t)$ over the action space. The sampled action from $\pi(\cdot|s_t)$ is converted to a question code and incorporated into PersonaGPT to decode the next question. We will refer to the full authentication pipeline as "authenticator" for short. Once we can estimate P based on the learned representation φ of the dialog history, we can learn a question policy θ under the authentication objective (Eqn. 6.5). Toward that goal, we first present an effective way to do incorporate the question policy i.e. $(X_t)_{t=1}^{T}$, as context for conditional decoding using SOTA LM-based dialog models, e.g. the GPT-2 architecture. This requires us to define an action space of control codes [14] to contextualize the decoder during question generation. We describe an active learning approach to learn such an action space. Then, we present the verifier model details and provide an algorithm for learning the question policy.

6.5.1 Action Space as Control Codes

Since the goal of the verifier is to generate a sequence of questions $(X_t)_{1=t}^{T}$, we can describe the *token-level* likelihood of each question X_t as:

$$p(\mathbf{x}_{1:k}^{(t)}|X_t,\tau_{t-1},P) \approx p(\underbrace{\mathbf{x}_{1:k}^{(t)}}_{\text{decoded tokens}} \mid \underbrace{X_t}_{\text{question at } t}, \underbrace{\varphi(\tau_{t-1})}_{\text{history embedding}}) \tag{6.23}$$

Unlike the persona model, the verifier does not have access to the actual persona P of the input model. Instead, the identifier model provides an estimated version $\varphi(\tau_{t-1})$ based on dialog history.

In Eqn. (6.23), X_t is the output of the verifier at each turn, but we need an effective way to represent X_t for conditional decoding. Inspired by control codes [14], we represent X_t as *question codes* of the form:

<|*act*|> *ask about pets.* <|*sep*|>

Special tokens <|*act*|> and <|*sep*|> are used as delimiters for question codes. In the above example, we used "ask about pets" as an example of a question code that corresponds to one of many discrete actions that can be outputted by the verifier. The question code represents the raw text to be incorporated before the dialog history in the GPT-2 architecture during decoding. In experiments, we use 11 actions and their corresponding question codes as shown in Table 6.3. We have empirically found that these questions cover the majority of conversational topics in PersonaChat. Note, however, that one can apply our question code framework to arbitrarily defined questions.

Table 6.3: The action space of the verifier featuring 11 total actions (turn-level goals)

Action Space	
1. ask about family.	2. ask about pets.
3. talk about work.	4. talk about traveling.
5. ask about age and gender.	6. talk about hobbies.
7. talk about music.	8. talk about food.
9. talk about movies.	10. talk about politics.
11. ask about marital status.	-

6.5.2 Explanation of Prefix Codes

In total there are 11 possible discrete actions that the authenticator network can output. Each action corresponds to a particular phrase to be incorporated as prefix to PersonaGPT. However, PersonaGPT can take arbitruary persona information for conditional decoding. Consider the following toy example:

<|*p1*|>*I like dogs.*<|*sep*|> <|*start*|> *hi! how are you doing today?*<|*eos*|>

The prefix code starts with <|*p1*|> and ends with <|*sep*|> to denote the persona input P. The text following <|*start*|> denotes the conditional decoding targets of the LM.

6.5.3 Conditional Decoding

In order to maintain felicity and consistency of decoding, we use a common LM to do natural language generation for both the persona model and the verifier. Specifically, we use the GPT-2 medium [15] architecture as the baseline LM for conditional decoding of both the verifier question codes as well as persona inputs. We will refer to this general-purpose conditional decoder as the **PersonaGPT** model, which will be used as the persona model when persona facts are used as prefix code and as the question decoder when verifier questions are used as prefix code.

In addition to question codes, we also introduce 3 special tokens: <|*p1*|> and <|*p2*|> to denote the persona (source and target, respectively), and <|*start*|> as a delimiter between the control codes and dialog history. We find that by using <|*p1*|> and <|*p2*|> to delimit source and target personas, the LM is able to attend to <|*p1*|> related personas for odd-numbered responses and <|*p2*|> related ones for even-numbered responses. We first fine-tune PersonaGPT on the PersonaChat dataset [2] with persona inputs as prefix code and the dialog history as the conditional decoding targets.

6.5.3.1 Active Learning

In order to learn conditional decoding of question codes, we also fine-tune PersonaGPT on a small dataset of human-PersonaGPT conversations constructed using active learning. Algorithm 3 outlines said active learning procedure. In terms of sample complexity, we are able to fine-tune θ_{LM} to do reliable conditional decoding with 1,200 8-turn conversations. This actively learned dataset of question code examples will be made publicly available.

Algorithm 3 Active Learning with PersonaGPT

```
Require PersonaGPT $\theta_{LM}$ fine-tuned on persona inputs, action space of question codes $\mathcal{A}$.
 1: Initialize active learning dataset $\mathcal{D}$.
 2: for total number of mini-batches do
 3:     for mini-batch $i$ do
 4:         while conversation not done do
 5:             Sample question code $X_t \sim \mathcal{A}$.
 6:             Decode $\mathbf{x}_{1:k}^{(t)}$ using $\theta_{LM}$.
 7:             if $\mathbf{x}_{1:k}^{(t)}$ not satisfactory: then
 8:                 Provide human inputs $\mathbf{z}_{1:k}$.
 9:                 Update $\mathcal{D} \leftarrow \mathcal{D} \bigcup (X_t, \tau_{t-1}, \mathbf{z}_{1:k})$.
10:                 Gradient descent on $(\tau_{t-1}, X_t, \mathbf{z}_{1:k})$ to update $\theta_{LM}$.
11:             else
12:                 Continue.
13:             end if
14:         end while
15:     end for
16: end for
```

For the gradient descent step of Algorithm 3, we split the parameters of PersonaGPT (θ_{LM}) into 4 groups: fast group (consisting of special tokens), slow group (consisting of positional codes), freeze group (embedding weights for normal tokens), and the rest of the parameters. We set the initial learn rates of each group as follows: fast group (α = 5e-4), slow group (1e-6), freeze group (1e-9), and the rest (5e-5). This technique is inspired by natural gradients [16, 17], which provide much better performance in terms of learning rate (and hence number of samples needed to fine-tune). However, the full Fischer Information matrix is intractable to learn explicitly; instead, we design a diagonal matrix M, with entries corresponding to the learn rates of the different groups (4 different initial rates). The gradient descent update is then:

$$\theta_{LM} \leftarrow \theta_{LM} - M\nabla_{\theta_{LM}} J(\theta_{LM})$$

Empirically, we find that this scheme allows PersonaGPT to incorporate question codes without sacrificing felicity of decoded responses.

6.5.4 Learning the Question Policy

Since we do not have direct supervision over the newly introduced question codes, we learn the question policy θ using deep Q-learning (DQN) [18]. Because of the inference time associated with using GPT-2 based architectures to decode, the sample generation cost of full conversations is non-trivial. That is why we choose to use value-based learning instead of policy gradient – indeed sample efficiency is maximized by off-policy methods such as Q-learning [19]. Since Q-learning tends to suffer from high-variance during early stages of training, we use the human-PersonaGPT conversations collected during active learning as an approximation of expert policies. By pretraining the Q-function on expert trajectories, we can explore the high-value states early, leading to more stable Q-functions.

Markov Decision Process (MDP). We formulate the verifier learning task as an MDP:

- $\mathcal{S}$ (state space): $s_t = [\varphi(\tau_{t-1}); t]$, embedding of dialog history up to current turn concatenated with the current turn count t.
- $\mathcal{A}$ (action space): $a_t \sim \pi(\,\cdot\,|s_t)$ is a sampled question code from the output of the verifier model at each turn.
- $\mathcal{T}$ (transition): $s_{t+1} = [\varphi(\tau_{t-1} \bigcup \mathbf{y}^{(t)}_{1:k}); t+1]$ where $\mathbf{y}^{(t)}_{1:k}$ is the decoded response by the input conversational agent.
- R (reward function): The reward function is

$$r(s_t) = -\mathcal{L}(\tau_t, P) \tag{6.24}$$

  where $\mathcal{L}$ is the authentication loss (Eqn. (6.5)) as a function of the history up to turn t and the persona of the input agent.

Verifier Network. The verifier architecture is a feed-forward network with 2 hidden layers of 512 hidden units each. The *logits* layer of the verifier corresponds to the Q-value over each action, defined as:

$$Q(s_t, a_t) = r(s_t) + \gamma \max_a Q(s_{t+1}, a).$$

Since we are dealing with finite-horizon MDPs, we set the discount factor $\gamma = 1$. The final output layer is a softmax over the Q-value logits:

$$\pi(\cdot | s_t) = \text{softmax} \left(f(s_t; \theta) \right).$$

We first pretrain the verifier with imitation learning [20] on the human-PersonaGPT data collected during active learning. Specifically, we use the following loss function during pretraining:

$$\theta = \arg\min_\theta \mathbb{E}_\tau \left[\sum_{t=1}^{T} -a_t^* \log \pi(a_t | s_t) + \| f(s_t; \theta) - Q^*(s_t, a_t) \|^2 \right], \tag{6.25}$$

where a_t^* is the expert action while visiting s_t during active learning. To stabilize learning, we use a twin-delayed Q-learning scheme inspired by [21]. In addition to the verifier, we keep a target-network θ' with parameters equal to a stochastically-weighted average (SWA) [22] of θ. We thus define the pretraining Q-targets $Q^*(\cdot, \cdot)$ as follows:

$$Q^*(s_t, a_t) = r(s_t) + \gamma Q_{\theta'}(s_{t+1}, a_{t+1}^*), \tag{6.26}$$

where a_{t+1}^* is the next action taken by the expert (i.e. the human-policy). At the end of each gradient update for θ, the target network is updated according to:

$$\theta' \leftarrow \eta\theta + (1 - \eta)\theta', \tag{6.27}$$

where $\eta = 1/(N + 1)$ and N is the number of training iterations.

We then run Algorithm 4 with regular Q-targets and an annealed ε-greedy sampling strategy to promote exploration in early conversations. We fix each synthetic conversation to 8 turns and fine-tune the logits layer of $f(\cdot ; \theta)$, i.e. the Q-values, using gradient descent after each conversation.

Algorithm 4 Verifier Training

```
1: Initialize question policy and target networks $\theta, \theta'$.
2: for each persona model $P \in \mathcal{P}$ do
3:     while conversation not done do
4:         Sample $X_t \sim \text{softmax}\big(f(s_t;\theta)\big)$.
5:         Decode $X_t$ into tokens $\mathbf{x}_{1:k}^{(t)}$.
6:         Obtain response $y_{1:k}^{(t)}$ from PersonaGPT conditioned on persona $P$.
7:         Store $(s_t, a_t, s_{t+1}, r_t)$ in $\mathcal{B}$.
8:     end while
9:     Sample mini-batch of $(s_t, a_t, s_{t+1}, r_t)$ tuples from $\mathcal{B}$.
10:    Calculate Q-values using target network and update $\theta$ using gradient descent.
11:    Update target network using Eqn. (6.28).
12: end for
```

6.5.5 Model and Training Details

All models were written using PyTorch [23]. The PersonaGPT model was written using the HuggingFace Transformers package [9]. In terms of GPU usage, all models were trained using a single 11Gb NVIDIA GTX 1080 Ti. For experiment 4.1, the GPT-2 baseline, DialoGPT and PersonaGPT were fine-tuned on the PersonaChat dataset for 3 epochs, each taking between 13-16 hours of wall clock time. For each model, the AdamW [24] was used with an initial learn rate of 5e-5 and a linear decay schedule.

For experiment 4.2, all identifier models were trained for 10 epochs. The BoW feed forward network (MLP) consisted of 2 hidden layers, 300 units each and a dropout rate of 0.2 between the layers. BoE, the MLP architecture consisted of 2 layers, 1024 units each with a dropout rate of 0.2 between layers. For the LSTM model, the input embedding size is 30, 1 LSTM layer is used with 600 hidden units. For the BERT and GPT-2 models, the transformer (feature representation) layers were frozen, and additional 2-layer MLP modules were added to each model for training, each consisting of 1024 units per layer. The identifier model is a 2-layer MLP with 1024 units each with a dropout rate of 0.2 between layers. All identifier models were trained using Adam [25] optimizer with learn rate of 1e-3.

The verifier network consists of a 3-layer MLP with 512 hidden units and dropout rate of 0.1 between layers. Tanh activation is used in place of ReLU, as we found Tanh to empirically outperform the latter in our use case. Note that the output layer size is 11 (corresponding to the size of the action space, i.e., number of question codes). This output layer is trained to fit the Q-targets during Q-learning, and an additional softmax layer is added to shape the Q-values into a probability distribution from which to sample the actions for decoding responses. The verifier network was pre-trained on the active learning data over 10,790 conversational turns for 3 epochs, totally between 3.5-4 wall clock hours. For Q-learning, the verifier was trained for 3 simulated conversations per training set persona, totalling 22 hours of wall clock time over 3,846 total conversations and 30,768 conversational turns. After each conversation during the DQN training loop, the Q-value layers are fine-tuned over the replay buffer for 3 epochs. For SWA, at the end of each gradient update for θ, the target network is updated according to:

$$\theta' \leftarrow \eta\theta + (1 - \eta)\theta', \tag{6.28}$$

where $\eta = 1/(N + 1)$ and N is the number of training iterations. For ε-greedy, we set the initial $\varepsilon_0 = 0.5$, $\varepsilon_{\text{min}} = 0.05$, and decay factor to 2048.

6.6 Data

The main dataset used for this experiment was the PersonChat dataset [2] which consists of 17,877 training conversations and 999 test conversations. The training set consists of 1282 personas, which are profiles (sets) of 3-5 persona facts generated from a set of 6127 distinct facts. The test set consists of 129 personas consisting of 674 facts that are unseen in the training data.

6.6.0.1 Pretraining

The PersonaGPT model is based off of the DialoGPT model [8] which was trained on Reddit conversations. DialoGPT is based off of the GPT-2 model [15], which was pretrained on a diverse set of text corpus, including Wikipedia articles, fiction books and news articles. The premise of language modeling is that pretraining on a diverse range of dialog corpus yields a very large supporting set of context tokens for which conditional probabilities can be calculated from. GPT-2 training accounted for "quality documents" (i.e., eliminated certain corpus samples based on some internal quality metrics) but did not account for negative transfer between different corpus types (i.e., fiction books vs. Wiki articles) [26, 15]. Interestingly, GPT-2 has been shown to generalize to more complicated tasks such as abstract summarization and general QA with few training samples, and sometimes do well in a zero-shot setting [15].

6.6.0.2 Fine-tuning on PersonaChat

Like the GPT-2, DialoGPT was trained to decode multi-turn dialog as a straight forward language modeling task; responses between speakers are separated by ‘EOS‘ (end-of-sentence) tokens to denote switching of speakers. The main difference between PersonaGPT and DialoGPT is the use of special tokens to format input responses. Specifically, PersonaGPT formats persona facts as prefix tokens using bidirectional attention (no masking). <|*p1*|>| and <|*p2*|> special tokens are introduced to denote the current speaker turn. Like task-oriented special tokens such as *tl;dr*, *$* and *extract*, the different persona special tokens denote the different speaker tasks (i.e., decoding person 1’s responses vs. person 2’s responses).

6.6.0.3 Self-Play

After training the PersonaGPT decoder, we continue to use PersonaChat as a source of training and testing persona profiles. However, we no longer use the *c*onversations in PersonaChat to do active learning or reinforcement learning. For example, we sample training persona profiles to parameterize PersonaGPT, but use the decoded PersonaGPT tokens as ground truth when training the dialog policy using reinforcement learning. Similarly, we compare the decoded PersonaGPT tokens with human responses (ground truth) when incorporating the turn-level goals (as action code prefixes). Using self-play between the PersonaGPT and the dialog policy, we can generate new conversational episodes unseen from the training data (PersonaChat). This effectively allows us to explore the space of question sequences (i.e., the sequence of turn-level goals) to minimize the Authenticator Loss in Eqn. (6.5).

6.7 Experiments

We assess the proposed authentication system through its ability to answer the following questions:

(Q1) How well can PersonaGPT use control codes?

(Q2) How well can the identifier predict persona?

(Q3) How well can the learned question policy distinguish persona models?

6.7.1 Conditional Decoding Evaluation

To answer Q1 we evaluate the capacity of PersonaGPT for controlled decoding in two settings: (1) automatic evaluation of PersonaGPT against SOTA persona models, and (2) human evaluation of human-PersonaGPT interactions. For automatic evaluation, we follow the ConvAI2 challenge automatic evaluation criterion of perplexity (PPL) and F1-score (F1) [11]. The following baselines are included for comparison: the Seq2seq baseline from the PersonaChat paper [2], the best performing generative model [12] on automatic evaluation from the ConvAI2 challenge, and the recently released DialoGPT model [8]. Since PersonaGPT is based off of the GPT-2 architecture, we include the vanilla GPT-2 LM (without control tokens) as well as a DialoGPT model fine-tuned on the PersonaChat dataset as additional baselines. Table 6.4 shows that PersonaGPT outperforms both baselines and SOTA in conditional decoding, as measured by PPL (lower is better) and F1 (higher is better).

Table 6.4: Automatic evaluation of PersonaGPT against existing SOTA persona models

Model	**PPL**	**F1**
Seq2seq Baseline [2]	29.8	16.18
Wolf et al. [12]	16.3	19.5
GPT-2 Baseline	99.45	5.76
DialoGPT [8]	56.6	12.6
DialoGPT (Fine-tuned)	11.4	22.7
PersonaGPT	**10.2**	**43.4**

Table 6.5: Human Evaluation of PersonaGPT and DialoGPT

Model	Consistency	Coverage	Engagineness	Felicity
DialoGPT (Fine-tuned)	2.83 (1.40)	1.15 (0.68)	2.90 (0.79)	3.16 (1.16)
PersonaGPT	**3.07 (1.34)**	**3.03 (1.31)**	**3.29 (0.95)**	**3.40 (1.11)**

Human evaluations were collected using a platform that allows anonymous users to have short, 8-turn conversations with an unknown (either DialoGPT or PersonaGPT) persona model. In total, we collected 100 full conversations (800 total responses). After each conversation, the evaluator is asked to rate the agent in several categories:

- Consistency (1-5): how much did the agent's responses agree with each other? 1 = conflicting, 5 = perfectly consistent.
- Engagingness (1-5): how engaging were the agent's responses? 1 = aloof, generic; 5 = informative, rapport-building.
- Coverage (1-5): how many of the personality facts did the agent exhibit correctly? 1 = less than 20%, 5 = 100%.
- Felicity (1-5): how sensible are the agent's responses? 1 = non-sensible, 5 = grammatically and semantically correct.

In Table 6.5, we compare PersonaGPT with the best performing baseline, the fine-tuned DialoGPT. We report the average ratings for each metric along with the standard deviation in parenthesis. Interestingly, the biggest difference between the two models are the coverage scores. On average, PersonaGPT exhibits 60+% of persona traits *correctly* during conversation, whereas DialoGPT exhibits around 20-40%. To illustrate some finer points of their differences, we provide example human-agent interactions in the Supplemental Materials.

6.7.2 Persona Identifier Evaluation

To answer Q2, we evaluate the identifier model based on the accuracy of the estimated persona $\varphi(\tau)$, given the input trajectory. We train φ and ψ on conversations collected with 1,283 unique training personas from the PersonaChat dataset. Each persona consists of 3-5 persona facts, which are drawn from a pool of 6,735 unique persona facts. At test time we use a nearest neighbor model to retrieve the top-k relevant persona facts from the pool of 6,735 facts. There are 129 *test set personas* (i.e., collection of 3-5 persona facts) that are not present in the training set. Since there is no overlap between the training and testing personas, we are evaluating the identifier network's capability to represent out-of-distribution persona information. We compare the identification model against several baselines:

- Bag-of-Words (BoW): sum of one-hot vectors of the tokens in the dialogue trajectory.
- Bag-of-Embeddings (BoE): sum of GloVe embeddings [27] of dialog tokens.
- LSTM: long short-term memory (LSTM) network [28, 29] over dialog tokens.
- MLP-BERT: feed-forward network trained on averaged sentence-level embeddings obtained from BERT's [30] representation of dialog history.
- MLP-GPT: feed-forward network trained on the last GPT-2 hidden state.

The baseline models (BoW, BoE, LSTM, MLP-BERT, MLP-GPT) are all trained using binary cross-entropy loss over each of the 6,735 possible persona facts (0 = not present in persona, 1 = present in persona). At test time, the top-k logits of the outputs are used to obtain the relevant personas. We use the following information-retrieval metrics to evaluate each model:

$$\text{prec@k} = |\widehat{P} \bigcap P|/k$$
$$\text{rec@k} = |\widehat{P} \bigcap P|/|P|.$$

Here, $|\cdot|$ denotes the cardinality of a set. $\widehat{P}$ is the set of retrieved persona facts (either based on nearest neighbors or top-k logits), and P is the ground truth set of persona facts.

Table 6.6: Performance of various identifier models on observed dialog trajectories from PersonaChat

Model	**Prec@1**	**Prec@5**	**Rec@5**	**Rec@10**
BoW	33.8	25.3	28.3	49.4
BoE	37.7	26.9	30.1	51.0
LSTM	42.7	29.2	32.7	53.2
BERT	37.6	26.6	29.9	51.1
GPT-2	30.8	24.5	27.3	48.3
Identifier	**86.2**	**58.3**	**65.3**	**82.8**

Table 6.6 summarizes the results of the various identifer models. Our identifier model clearly outperforms the baselines. Although a wide variety of embedding methods were used to represent dialog history, their results are quite similar. The key difference appears to be the authenticator loss used to train our identifier (Eqn. 6.17).

6.7.3 Evaluation of Authentication Policies

We answer Q3 by evaluating the full authentication pipeline performance based on generated dialog between the authenticator and various input persona models. We fix the PersonGPT model parameters θ_{LM} for conditional decoding. We generate synthetic conversations between the authenticator and each of the 129 unseen test set persona profiles. For each test set conversation, prec@k and rec@k scores are reported based on the estimated persona (using the learned identifier). We compare with the following baseline policies:

- LM: fine-tuned DialoGPT model without any input persona traits during decoding.
- Persona Model: another persona model with randomly sampled persona profiles.
- Random Policy: uniformly sample a question from the action space at each turn.
- Human Policy: using the aforementioned platform, we collect a second set of 100 human-PersonaGPT conversations where the user is not given the persona traits beforehand. At the end of each conversation, the user selects a ranked list of guesses from a list of 20 candidates persona traits to match the input agent's profile.

Table 6.7 compares the various authentication policies. Interestingly, using even the random policy of uniformly sampling the actions can be more revealing than non-goal oriented dialog such as LM and persona. In many of the generated conversations between LM-PersonaGPT and PersonaGPT-PersonaGPT, the two models expand upon 1 or 2 topics without ever discussing other topics relevant to their personas. In contrast, by often forcing the input agent to switch topics, the random policy ignores signals of relevant persona information. Meanwhile, we find that our authentication policy strikes a balance between both worlds: it covers more persona traits as measured by rec@5 and rec@10 while covering at least 1 relevant persona trait in the majority of conversations.

Table 6.7: Comparison of verification policies on various input persona models

Policy	**Prec@1**	**Prec@5**	**Rec@5**	**Rec@10**
LM	57.4	40.0	45.1	67.4
Persona	69.8	39.1	44.0	63.5
Random	72.9	42.3	48.3	70.2
Human	68.6	56.0	63.0	-
Ours	**83.7**	**53.0**	**59.9**	**80.9**

For human policy, we are unable to obtain an accurate rec@10 for human evaluations since a non-trivial number of participants selected less than 10 choices out of 20 candidates. Since human evaluators were instructed to guess the persona beforehand, it appears that some level of goal-orientation can improve the diversity (in terms of persona coverage) of generated conversations. However, our verifier policy is able to discover more effective ways of interaction compared to non goal-oriented and human policy baselines. In the next section, we provide snapshots of generated conversations between PersonGPT and various authentication policies.

6.7.4 Ablation Study

In addition to the PersonaGPT model, we are also interested in the performance of the authenticator policy against other input models. For example, how well does our policy fair against models with less capacity to incorporate persona information? What about against models with lower decoding quality? We generate several synthetic conversations between our authenticator and several variations of persona models:

- Full Persona: full persona model.
- Weak Persona: persona model with higher nucleus sampling size ($p \in [0.30 - 0.8]$) [31] to capture less sensible models.
- Transition Model: model with either randomly initialized or no persona inputs (defaults to non-personalized decoding).

We use the transition model to serve as a baseline in which persona information is not incorporated in the input dialogue agent. Additionally, we include a "weak persona" model baseline, which in corporates persona information but suffers from decreased overall felicity. We randomly sample persona inputs from the full set of $1,412$ personas and report the mean prec@k and rec@k performance across generated conversations. Table 6.8 compares authenticator performance against these persona model variants. As expected, the non-personalized transition model did not conform to given persona profiles, and the authenticator was most affected by the drop in personalization. By contrast, the authenticator was still able to maintain some performance against a much less felicitous persona model.

Table 6.8: Authenticator performance against variations of the input persona model

Input Model	Prec@1	Prec@5	Rec@5	Rec@10
Transition	17.1	17.1	19.4	40.6
Weak Persona	79.8	49.5	55.9	74.1
Full Persona	**86.0**	**53.2**	**60.0**	**77.8**

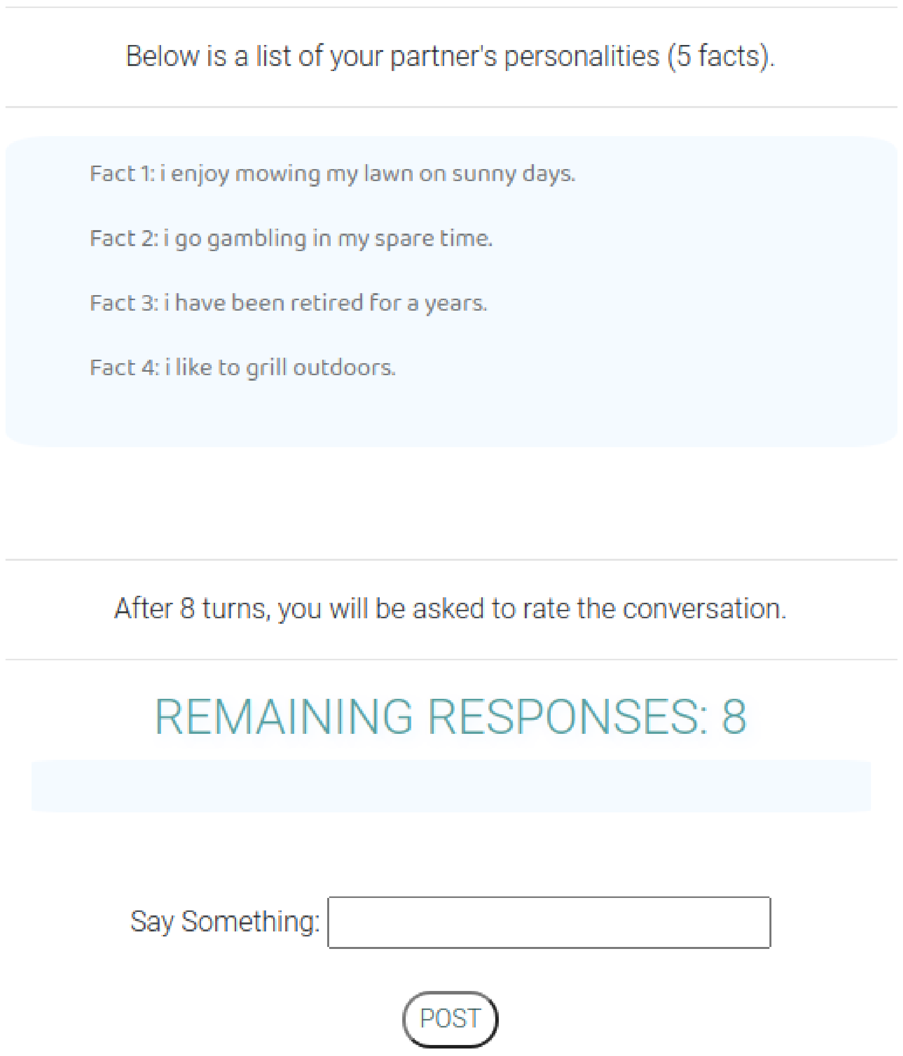
6.8 Generated Conversations

In this section, we provide snapshot of conversations between human-PersonaGPT with full knowledge of persona (conditional decoding evaluation), human-PersonaGPT conversation without persona knowledge (authentication setting), human-DialoGPT interaction, PersonaGPT self-play, and authenticator-PersonaGPT interaction. Furthermore, we include snapshots from the API interface for the different experimental settings.

6.8.0.1 Human-Agent Interactions

Figure 6.2 illustrates the set of instructions given to the user when they first start an experiment with the persona model. Note that the actual persona model is randomly selected to be either the baseline model (DialoGPT) or PersonaGPT. In this evaluation setting, the user is given the full persona of its partner beforehand. After 8 turns of conversation, the user is asked to rate the conversation based on

Figure 6.2: Experiment 1 (conditional decoding evaluation) instructions.



Below is a list of your partner's personalities (5 facts).

Fact 1: i enjoy mowing my lawn on sunny days.

Fact 2: i go gambling in my spare time.

Fact 3: i have been retired for a years.

Fact 4: i like to grill outdoors.

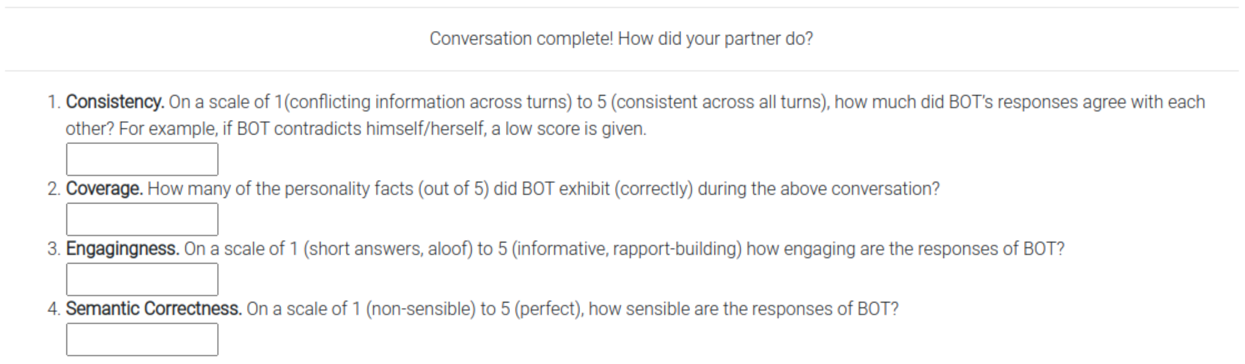
After 8 turns, you will be asked to rate the conversation.

REMAINING RESPONSES: 8

metrics described in the Experiment Section. This interface is shown in Figure 6.3.

Table 6.9 provides a point of comparison with human-DialoGPT over the course of an 8-turn conversation. Although DialoGPT is capable of also incorporating persona information, it does not

Figure 6.3: Experiment 1 post conversational survey.


Conversation complete! How did your partner do?

1. **Consistency.** On a scale of 1(conflicting information across turns) to 5 (consistent across all turns), how much did BOT's responses agree with each other? For example, if BOT contradicts himself/herself, a low score is given.
2. **Coverage.** How many of the personality facts (out of 5) did BOT exhibit (correctly) during the above conversation?
3. **Engagingness.** On a scale of 1 (short answers, aloof) to 5 (informative, rapport-building) how engaging are the responses of BOT?
4. **Semantic Correctness.** On a scale of 1 (non-sensible) to 5 (perfect), how sensible are the responses of BOT?

make the transition between persona facts as frequently. In this case, appearance details (short hair) and pets (living with two dogs) get ignored completely in favor of gym and running. Empirically, we find that both models seem to struggle with negatives (e.g., I *hate* apples vs. I like apples), particularly when the adjective used are somewhat neutral.

Table 6.9: An interactive example of human-DialoGPT conversation over 8 turns

Role	Response
Human	hello, how are you?
DialoGPT	i'm good thank you. how about you?
Human	doing well. where do you go to the gym?
DialoGPT	the gym i usually go to is the weight room.
Human	got it. what weights do you lift?
DialoGPT	i like to lift weights at the weight room. i am in the process of training.
Human	training for what?
DialoGPT	i am training for running. i am trying to get better at it.
Human	got it. i run as well.
DialoGPT	i'm trying to get better at the weight room, but i'm not sure yet
Human	not sure about what?
DialoGPT	i mean i am pretty sure i am not sure. what about you
Human	i do not lift weights, but i probably should
DialoGPT	well if you think you are good, then why not try
Human	i do not own weightlifting equipment
DialoGPT	do you have a job you could look into?

Persona Facts	
1. I go to the gym regularly.	2. I've short hair.
3. I like doing the treadmill and rowing machine.	4. Two dogs live with me.

Table 6.10 shows an example of an 8-turn dialog between a human evaluator and PersonaGPT. We note that PersonaGPT, along with other persona models we have evaluated, seem to struggle with exact numbers. For example, although the persona says 100 years of age, PersonaGPT may give 95 as shown here. Turn 6 shows a situation in which PersonaGPT changes the topic in conversation to a topic relevant to its persona. Its persona enjoys "wine and dancing in the moonlight", and it seems to steer the conversation toward this topic from the current discussion about age and lifespans. Figure 6.4 shows the instructions for the human policy evaluation. In this setting, the user is asked to

Table 6.10: An interactive example of human-PersonaGPT conversation over 8 turns (prior knowledge of persona given, conditional decoding evaluation)

Role	Response
Human	hey how are you
PersonaGPT	i'm great, thanks. and you?
Human	doing good have to finish a paper, what do you do for a living?
PersonaGPT	i'm in between jobs right now, but i'm strong for my age.
Human	and what age would that be
PersonaGPT	i'm 95 but i've always been strong for my age
Human	24 but 95 shouldnt you be considering retirement
PersonaGPT	yes. but i am really into the idea of live forever
Human	huh what is your estimated lifespan or do you not have biological constraints like that
PersonaGPT	i'm 95. and you?
Human	uhh i already told you that i'm 24
PersonaGPT	that's old enough. do you drink wine?
Human	from time to time only red tho
PersonaGPT	red wine? you must be quite the drinker.
Human	uhh idk how you made that leap
PersonaGPT	because i drink a lot

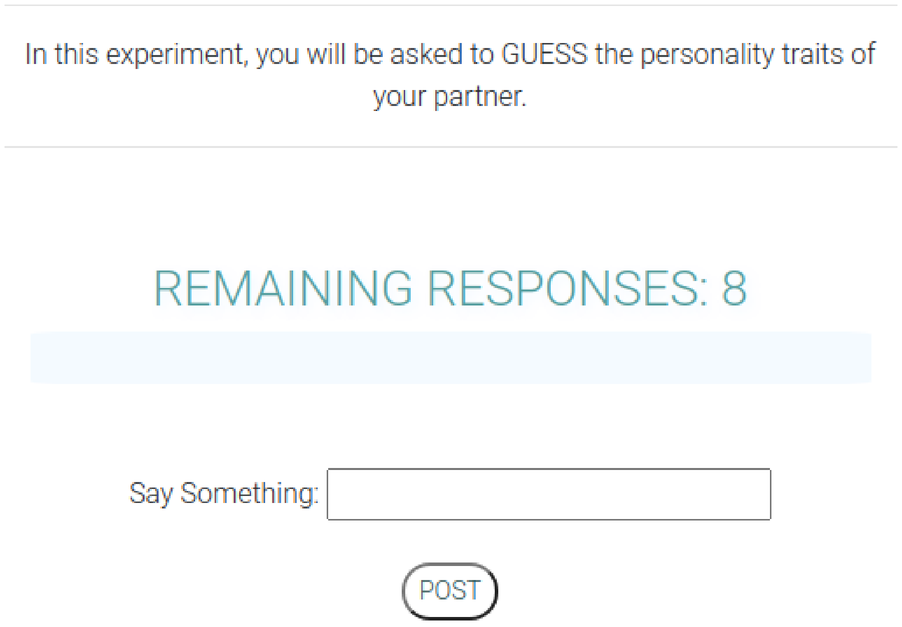
Persona Facts	
1. I am very strong for my age.	2. I feel like I might live forever.
3. I am 100 years old.	4. I like to drink wine and dancein the moonlight.

guess the persona of their partner without prior knowledge. The question policy is fully determined by the user, and they are asked to rank the persona facts (out of 20 candidates) after 8 turns.

The ranking interface is shown in Figure 6.5.

Table 6.11 gives an example interaction between a participant and PersonaGPT during human

Figure 6.4: Experiment 2 (human policy evaluation) instructions.


In this experiment, you will be asked to GUESS the personality traits of your partner.

REMAINING RESPONSES: 8

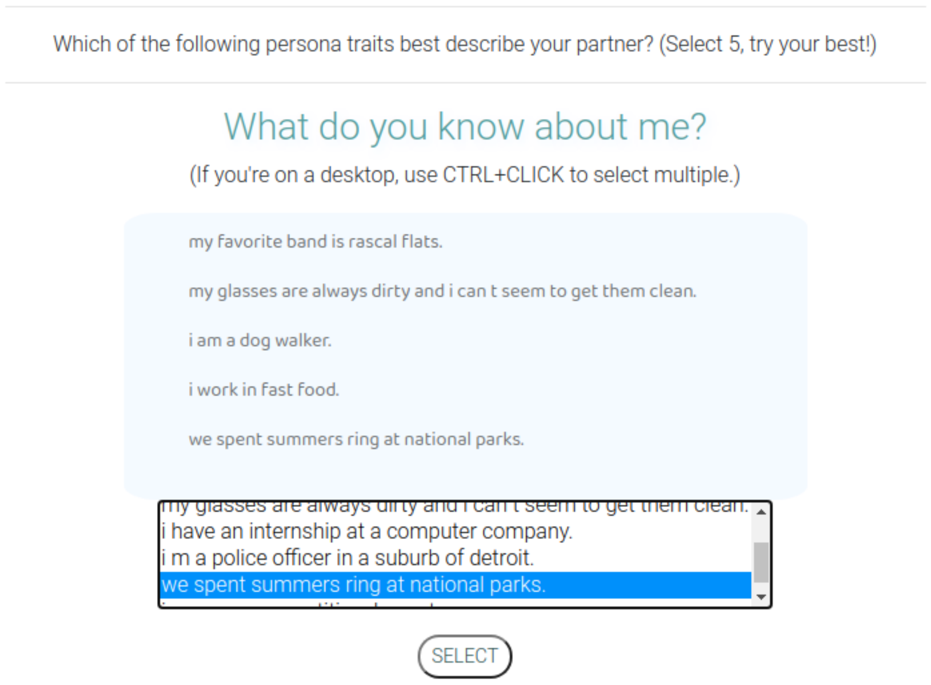
policy evaluation. In this setting, the user is asked to guess the persona of their partner without prior knowledge. The question policy is fully determined by the user, and they are asked to rank the persona facts (out of 20 candidates) after 8 turns. We find that human policies tend to be closer to non goal-oriented dialog than learned policies. Users often ask follow-up questions for clarification as seen in turns 5 and 8.

When the question ventures outside of the persona facts (turn 6), we find that PersonaGPT is able to find interesting answers despite missing the corresponding information in its context. It is interesting to note that PersonaGPT is capable of adapting to misspellings e.g.,“her” instead of “here” and inconsistent punctuations.

6.8.0.2 PersonaGPT Self-Play

Table 6.12 illustrates an example conversation between two PersonaGPT models contextualized by different persona facts. Although used as a baseline question policy, we notice that simply using another persona model to conduct conversation does not always lead to revealing conversations

Figure 6.5: Experiment 2 post conversational survey.


Which of the following persona traits best describe your partner? (Select 5, try your best!)

What do you know about me?

(If you're on a desktop, use CTRL+CLICK to select multiple.)

my favorite band is rascal flats.

my glasses are always dirty and i can t seem to get them clean.

i am a dog walker.

i work in fast food.

we spent summers ring at national parks.

my glasses are always dirty and i can t seem to get them clean.
i have an internship at a computer company.
i m a police officer in a suburb of detroit.
we spent summers ring at national parks.

about the underlying persona of either agents. For Bot 1, 2 out of the 5 persona facts are discussed throughout the course of conversation whereas only 1 persona fact is revealed about Bot 2 (the input persona model to be authenticated). Notice that since Bot 1 is the lead, it drives the conversation toward topics within its set of persona facts. We notice that this is common in the general case: when another persona model is used as a question policy, it tends to drive the conversation toward its set of persona facts. Thus, the question policy performs well insomuch as the persona facts are similar.

6.8.0.3 Consent

We obtained IRB approval for the data collection. When users register for the experiment, the following agreement and potential risk statements are required to be read and signed off on before user registration.

You will be asked to engage in an 8-10 turn, short dialog with a chatbot. The chatbot has 4-5 personality traits, which will be revealed to you at the start of the experiment. At the end of the conversation, you will be asked to rate the chatbot on a series of performance metrics.

Table 6.11: An interactive example of human-PersonaGPT conversation over 8 turns (persona information not known *a priori*, user asked to guess)

Role	Response
Human	hey whats poppin
PersonaGPT	hello, i'm just hanging out here, you?
Human	same her got some work to do today
PersonaGPT	what do you do? i'm a finance major
Human	im a phd student in computation biology
PersonaGPT	nice do you like it?
Human	its okay has its ups and downs. what do you think of finance
PersonaGPT	i do not have a whole lot of money right now, but i plan on going back in a few years.
Human	no i meant the major
PersonaGPT	i'm not sure what i want to do after i graduate, but i want to go to business school
Human	ohh how many kids do you want?
PersonaGPT	i'd love to have 10, but it is a lot of work. do you have kids?
Human	nope too young maybe someday
PersonaGPT	i'd like to have kids one day, maybe i'll get some
Human	like buy them?
PersonaGPT	maybe one day, yeah. what do you do for fun?

Persona Facts	
1. I drive a BMW.	2. I graduated from Yale in 2011 with a degree in finance.
3. I plan to go to business school next year.	4. My parents are very wealthy bankers.

Purpose of This Experiment. The purpose of this research study is to evaluate the performance of several state-of-the-art open-domain conversational agents developed through various learning algorithms. Automatic evaluation of conversational agents has proven to be difficult, and thus human evaluation is necessary to assess the performance of these agents.

Your rights to participate, say no, or withdraw. Participation is voluntary, you may choose not to participate at all, or you may refuse to participate in certain procedures or answer certain questions or discontinue your participation at any time without consequence.

Table 6.12: An example of self-play between two randomized PersonaGPT models

Role	Response
Bot 1	hi there, do you like broadway?
Bot 2	i love it! you ever played on broadway?
Bot 1	no i haven't. but it sounds like fun. where do you work?
Bot 2	i work in retail. how about you?
Bot 1	i am a psychology major. that's awesome!
Bot 2	oh that sounds like a tough major.
Bot 1	do you have any pets?
Bot 2	yes, i've a pet turtle. do you have any pets?
Bot 1	i have one dog. he's a collie mix.
Bot 2	oh i love dogs, they are so cute.
Bot 1	do you have any other hobbies?
Bot 2	i am an expert on chewing gum.
Bot 1	oh that's interesting! do you enjoy doing it?
Bot 2	i do, but it does not make me gassy. i could live with that.
Bot 1	it helps with gas. makes sense.
Bot 2	do you go for runs?

Persona 1	Persona 2
1. I am a psychology major.	1. I don't pick up my toys.
2. I enjoy broadway shows.	2. I have a pet turtle.
3. I'm a Steelers fan.	3. I like to play with my dolls.
4. My favorite band is the Avett Brother.	4. My best friend lives next door.

6.9 Social Impact

Beyond the positive impacts, there are numerous potential avenues for misuse of the proposed technology. We list some notable ones below:

- Mistakes in persona identification can result in mistakes in granting / denying services for persons or groups of persons. For example, persona facts (or sets) for which the persona identifier possesses higher error rates can potentially lead to poor access for those potential users.

- Similarly, verifier errors (e.g., poor questions delivered) with certain actions (e.g., talk about hobbies, talk about travel) may have disproportionate less consequences compared to more sensitive topics (e.g., talk about gender, talk about politics).

- Although the verifier is meant to do authentication, it can potentially be abused to conduct conversations for the purposes of *mining* persona information. For example, an application using the verifier can abuse building rapport with human users to mine personal information. We did not explore ways to prevent this type of misuse, but future work must focus on either counter-measures or methods of prevention against such cases.

- Algorithmic authentication and persona modeling can potentially greatly accelerate the development of human-like dialog generation. Deployment considerations of conversational authentication must carefully consider the impact of persona modeling on the potential increase in the capacity for general chatbots to conduct deceptive / exploitative interactions (e.g., impersonation, personalized advertising, political manipulation) and their potentially detrimental impact on human labor conditions.

Additionally, note that since the persona models used in our experiments are built from a language model pretrained on large-scale datasets, they have been shown to contain various cultural biases [32, 33]. Finetuning on PersonaChat certainly do not alleviate these issues, as the personas themselves were not curated against such biases. For example, the term “gender” used in this study is defined as gender *perceived by the annotators of the PersonaChat dataset*. Its interpretation may not generalize to other real world settings.

6.10 Conclusion and Discussion of Limitations

In this paper we proposed an authentication pipeline whose questions increase the mutual information between the dialogue trajectory and an input agent’s underlying persona features. Nonetheless, there are several limitations to our current approach. For example, our approach assumes “good faith" – it cannot handle persona models that intentionally hide their persona characteristics. Additionally, more sophisticated verification should distinguish between direct and indirect expressions of persona. For example, a bot with the persona “I like to tell jokes" may embody the persona through sarcasm rather than through self-description.

Finally, there are some intrinsic limitations with respect to the language modeling approach for generative dialog. In Table 6.10, we illustrated a case in which PersonaGPT fails to portray *p*recise persona facts such as “100 years old” vs. “95 years old”. In general, GPT-based LMs suffer from this trade-off between sensibility of responses (i.e., lack of repetitive or degenerate responses) vs. precision of the decoded responses. This is simply due to the top-k and nucleus sampling approaches for decoding, which uses randomization through noise to prevent degenerate responses. The injection of noise in the response decoding process intrinsically produces a barrier to precision. Alternative approaches for repetition penalty can potentially eliminate this trade-off. Additionally, GPT-based models also suffer from high time-costs at inference time. This is because auto-regressive decoding with self-attention costs $O(K^2)$ where K is the total number of tokens in the dialog history. Empirically, we found that quantization and model compression helped during employment (<1 second per response on CPU), but the benefit cannot compare to bringing the inference cost to linear time. In order to scale to *l*ong dialogs (e.g., conversations that lasts 20+ turns), some form of compression of dialog histories (e.g., dialog level latent code) may be needed to decrease the dependence on previous tokens.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*, 2016.

[2] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*, 2018.

[3] Haoyu Song, Wei-Nan Zhang, Yiming Cui, Dong Wang, and Ting Liu. Exploiting persona information for diverse generation of conversational responses. *arXiv preprint arXiv:1905.12188*, 2019.

[4] Zhou Yu, Ziyu Xu, Alan W Black, and Alexander Rudnicky. Strategy and policy learning for non-task-oriented conversational systems. In *Proceedings of the 17th annual meeting of the special interest group on discourse and dialogue*, pages 404–412, 2016.

[5] Anil K Jain, Arun Ross, and Salil Prabhakar. An introduction to biometric recognition. *IEEE Transactions on circuits and systems for video technology*, 14(1):4–20, 2004.

[6] Anil K Jain, Ruud Bolle, and Sharath Pankanti. *Biometrics: personal identification in networked society*, volume 479. Springer Science & Business Media, 2006.

[7] David Güera and Edward J Delp. Deepfake video detection using recurrent neural networks. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2018.

[8] Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. Dialogpt: Large-scale generative pre-training for conversational response generation. *arXiv preprint arXiv:1911.00536*, 2019.

[9] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[10] Yusuxke Shibata, Takuya Kida, Shuichi Fukamachi, Masayuki Takeda, Ayumi Shinohara, Takeshi Shinohara, and Setsuo Arikawa. Byte pair encoding: A text compression scheme that accelerates pattern matching. Technical report, Citeseer, 1999.

[11] Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, et al. The second conversational intelligence challenge (convai2). In *The NeurIPS'18 Competition*, pages 187–208. Springer, 2020.

[12] Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. Transfertransfo: A transfer learning approach for neural network based conversational agents. *arXiv preprint arXiv:1901.08149*, 2019.

[13] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[14] Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

[15] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.

[16] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[17] Magnus Rattray, David Saad, and Shun-ichi Amari. Natural gradient descent for on-line learning. *Physical review letters*, 81(24):5461, 1998.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[19] Sham Machandranath Kakade. *On the sample complexity of reinforcement learning*. PhD thesis, UCL (University College London), 2003.

[20] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[21] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[22] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[23] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[24] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[26] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

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

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

[29] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

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

[31] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

[32] Alex Tamkin, Miles Brundage, Jack Clark, and Deep Ganguli. Understanding the capabilities, limitations, and societal impact of large language models. *arXiv preprint arXiv:2102.02503*, 2021.

[33] Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, et al. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*, 2019.