Generative Modeling with Guarantees

by

Victor Quach

B.S., École polytechnique (2016)
M.S. École polytechnique (2017)
S.M. Massachusetts Institute of Technology (2020)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 2023

Author	Victor Quach Department of Electrical Engineering and Computer Science May 19, 2023
Certified by	Regina Barzilay Professor of Electrical Engineering and Computer Science Thesis Supervisor
Accepted by	Leslie A. Kolodziejski Professor of Electrical Engineering and Computer Science Chair, Department Committee on Graduate Students

Generative Modeling with Guarantees

by

Victor Quach

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2023, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

Language models have become ubiquitous in natural language processing, leveraging large amounts of unlabeled data and fine-tuning for downstream tasks. However, concerns have been raised regarding the accuracy and trustworthiness of the text generated by these models. In parallel, differential privacy has emerged as a framework to protect sensitive information while allowing machine learning algorithms to learn from it. Nevertheless, the trade-off between statistical guarantees and utility poses challenges for many applications. Therefore, this thesis aims to develop techniques that balance guarantees and utility, focusing on improving the reliability of generative models while preserving their flexibility.

First, we propose a framework that enables the generation of text conditionally using *hard* constraints, allowing users to specify certain elements in advance while leaving others open for the model's prediction. By facilitating interactive editing and rewriting, this framework provides users with precise control over the generated text.

Next, we introduce conformal prediction methods for generating predictions under *soft* constraints, ensuring statistical correctness. These methods produce valid confidence sets for text generation while maintaining high empirical precision.

Finally, we explore the balance between privacy and utility in data release by relaxing the notion of guarantees from differential privacy to a definition based on *guesswork*. We present a learning-based approach to de-identification, addressing the challenges of privacy preservation while still enabling effective data utilization.

The effectiveness of our proposed methods is demonstrated through a range of tasks, including text infilling, radiology report generation, and X-ray classification. These tasks showcase the utility of our techniques in various practical scenarios.

Thesis Supervisor: Regina Barzilay
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

First, I would like to express my deepest gratitude to my advisor, Regina Barzilay. She has been a caring mentor who consistently challenged me to think outside the box and not settle for easy wins. During moments of adversity, her words of encouragement have provided me with the strength to persevere. I would not have been able to complete this thesis without her patient guidance and support.

I would also like to extend my heartfelt appreciation to my thesis committee members, Tommi Jaakkola and Muriel Medard, for their insightful conversations and thoughtful feedback. Their expertise and guidance have been invaluable in refining my research and expanding my academic horizons. I am grateful for the opportunity to have learned from their leadership.

Being a part of the RBG lab has been a tremendous privilege. I am incredibly fortunate to have worked in such a safe and supportive environment. I would like to express my sincere appreciation to all the lab members for their insightful discussions and camaraderie. In particular, I would like to acknowledge and express my gratitude to my wonderful collaborators, Tianxiao Shen, Adam Yala, and Adam Fisch. Working closely with them has not only enriched my research but has also provided a stimulating environment for personal and intellectual growth. Their collaborative spirit has been instrumental in shaping the outcomes of this work. Thank you Yujia Bao, Yujie Qian, Rachel Wu, Benson Chen, Wenxian Shi, Tal Schuster, Peter Mikhael, Itamar Chinn, Aziz Ayed, Jeremy Wohlwend, Umesh Padia, Chris Alexiev, Gabriele Corso, Shiyu Chang, Felix Faltings, Jiang Guo, Wengong Jin, Bracha Laufer Goldshtein, Jiaming Luo, Sean Murphy, Tally Portnoi, Darsh Shah, Nitan Shalon, Hannes Stark, Allison Tam, Janice Yang, Jason Yim and all the members of Regina's group for the discussions and fun.

I also would like to thank my friends and extended family who have supported me throughout this journey. Their love, encouragement, and belief in me have been a constant source of inspiration.

To my wife, Yuening Zhang, I owe a special debt of gratitude. Her unwavering patience, understanding, and support have been invaluable. I am truly blessed to have her by my side.

To my sister, Hélène, thank you for being there for me no matter the time of the day or the night.

Lastly, I would like to express my gratitude to my parents. Mom, Dad, thank you for your sacrifices and unwavering belief in me. Your love and support have been the bedrock of my journey, and I am forever grateful for everything you have done for me.

Bibliographic Note

Portions of this thesis are based on prior peer-reviewed publications. Chapter 2 was originally published in [135]. A publication based on the work presented in Chapter 3 is currently under review. Chapter 4 is available as a preprint at [173].

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In an increasingly digitized world, a vast majority of human knowledge is stored in text format. Throughout history, scholars and researchers have relied on texts as primary sources for understanding the human experience. Historians meticulously analyze ancient manuscripts, letters, and archival records to reconstruct past events and shed light on the motivations and actions of our ancestors. Sociologists investigate newspaper articles, online forums, and social media posts to discern patterns in public sentiment and track the evolution of social dynamics. Psychologists delve into written accounts of individual experiences, such as diaries or therapy transcripts, to gain insights into the complexities of human behavior and cognition.

Harnessing this abundance of textual data, large language models learn a probability distribution over sequences of words from unlabeled training corpora. They learn that the sentence “Abraham Lincoln died in 1865” is more likely than “Abraham Lincoln was born in 1942”. They recognize that the phrase “The stock price will go up, we should *buy* now” is more probable than “The stock price will go up, we should *sell* now”, or that the sequence “1+1=2” is more likely than “1+1=3”. These learned probabilities capture generalizations about word order, syntactic structure, semantic meaning, and pragmatic function.

At first glance, it may seem that a probability distribution over sequences of words is all that is needed to effectively model language. To challenge this idea, Noam Chomsky [29] proposed the famous pair of sentences: “Colorless green ideas sleep furiously" and "Furiously sleep ideas green colorless". Chomsky argues that any two consecutive words in the preceding sentences are unlikely to happen together; thus, a statistical model would predict that these two phrases are equally (un)likely. However, the first one is grammatical while the second one is not. Chomsky concludes that statistical models are unable to capture grammaticalness. Fortunately, statistical models are not restricted to bigram models. For example, a simple hidden markov model would assign a probability much greater to the first sentence than the second sentence [118].

In the 90s, statistical models developed with some success, paving the way for more advanced techniques in NLP. Early attempts at applying deep learning to language modeling were made in the 2000s using recurrent neural networks (RNNs) [16, 107, 31], but they were limited due to the vanishing gradient problem. The problem persisted until the development of long short-term memory (LSTM) units [61], which enabled the creation of efficient RNN architectures. Later, the use of attention mechanisms [10] allowed LSTMs to reach state-of-the-art performance on a wide majority of NLP tasks.

At the time, researchers and practitioners would need to build custom models for each task. Early works have studied the use of language models for representation learning [108] but relied on fixed word representations. In 2018, ELMo [119] was proposed, introducing contextual embeddings that encode information about both the current word and its surrounding context. This marked the start of NLP's transfer learning era, where models are trained on massive amounts of unlabeled text, then fine-tuned on downstream tasks. Meanwhile, the Transformer architecture [150] was introduced: it uses multiheaded self-attention layers to compute the probability distributions of sequences without recurrence. Being highly parallelizable, Transformers leverage the power of modern GPU hardware and enabled training at scale. Incorporating these innovations, pretrained language models like BERT [35] and its variants [98, 88] have become the de facto standard for many NLP applications, reaching state-of-the-art results across a variety of NLP tasks, such as text classification, question answering or natural language inference.

The latest wave of progress in NLP has been fueled by scaling pretrained language models, both in terms of model size and data size. These so-called large language models (LLM) [184] not only demonstrate superior performance compared to smaller ones, but also display new capabilities that were previously thought impossible for NLP models to achieve. For instance, the 175B-parameter GPT-3 [21] can solve few-shot tasks with *in-context learning* where the 1.5B-parameter GPT-2 [121] cannot.

Despite their remarkable achievements, language models raise significant concerns regarding trustworthiness and controllability. Even large models such as GPT-4 [113] suffer from *hallucinations*, i.e., generating plausible yet false statements. These false statements can range from factual inaccuracies to entirely fabricated information. Furthermore, large language models require users to provide appropriate prompts to elicit desired responses effectively. Understanding how to prompt these models effectively requires human expertise and experimentation.

The limitations in trust and control become particularly consequential when large language models are deployed in real-world applications. These models have expanded beyond academia and are now widely used by millions of people daily. For instance, ChatGPT, a variant of the GPT series, has become a popular language model for engaging in human-like conversations and providing assistance across a range of tasks. It assists users in generating text, answering questions, offering recommendations, and even providing emotional support. However, the lack of guarantees in the outputs of these models can lead to unintended consequences or the propagation of incorrect or harmful information.

In the field of privacy, there has been significant progress in developing methods that provide statistical guarantees to control the risk of privacy leakage. Differential privacy (DP) is one such framework that has gained attention and has been applied to various domains [43, 93, 171, 146, 73, 1]. DP ensures that the presence or absence of any individual data point does not significantly affect the output of a computation, thus protecting the privacy of individuals in the dataset.

Privacy is a domain where failures can have catastrophic consequences. The misuse or mishandling of private data can lead to privacy breaches, identity theft, discrimination, or the compromise of sensitive personal information. In response to these risks, regulators have introduced laws and regulations that govern how private data should be handled and protected. Organizations are required to comply with strict privacy regulations, which often involve implementing complex technical and organizational measures to ensure data privacy.

However, the implementation of privacy requirements can be costly and challenging, leading to restricted data sharing and reduced data utility. Organizations face significant hurdles in sharing sensitive data for research purposes due to concerns about privacy risks and the cost associated with implementing privacy-preserving techniques. This scarcity of available data, particularly in domains such as healthcare, has hindered the progress of medical AI research and prevented the exploration of the full potential enabled by training large models at scale.

This dilemma faced by data owners who seek to comply with HIPAA [60] regulations highlights the challenges in achieving a balance between privacy preservation and data utility. Currently, they are forced to choose between manual heuristic de-identification techniques, which lack formal privacy guarantees, or adopting differential privacy, which can result in a significant loss of data utility. As a result, data owners often resort to not sharing the data at all or only sharing it with a limited number of partners under restricted licenses. These restrictions not only hinder the potential for collaborative research and innovation but also raise concerns about reproducibility, transparency, and fairness.

This thesis aims to address some of the challenges in **generative modeling with guarantees**, with a focus on advancing the field of trustworthy and controllable generation. We explore novel approaches that enhance the reliability, and user control over the outputs of generative models. The objective is to develop methods that empower users to have greater confidence in the model's generations and ensure that the model operates within predefined constraints or guidelines.

In the following sections, we outline the three main contributions of this research.

1. *Blank Language Model:* A language model that enables users to designate specific locations where to generate text ;

2. *Conformal Language Modeling:* A method to achieve conformal prediction guarantees for text generation ;

3. *Syfer:* A framework that leverages advances in image generation for private data sharing.

1.1 Blank Language Model

At the core of language models lies their ability to generate coherent and fluent sentences based on contextual information and provide meaningful completions given partial input. This capability has naturally paved the way for numerous generative applications, including auto-completion systems, writing assistants, code generation tools, and more.

Language models traditionally attribute probabilities to sequences of words by factorizing them from left to right, as:

$$p_\theta(\mathbf{x}) = \prod_{k=1}^{n} p_\theta(x_k | x_1, \ldots, x_{k-1}).$$

However, the process of writing text is rarely linear and unidirectional. For example, when composing a sentence, a writer considers not only the preceding context but also the expected structure and content of the remaining of the text. Using a text editor, writers frequently navigate throughout the document, simultaneously editing multiple sections, deleting and revising text. The new text will be influenced by the flow of ideas and the current state of the document, which include both preceding and subsequent context. Similarly, programmers rarely write lines of code, one after another. Instead, they may edit several lines at once or even jump back and forth between different parts of the file. To be valuable tools, auto-completion systems and writing assistants must incorporate information from both preceding and subsequent context to generate meaningful and contextually appropriate responses.

Masked language models (MLMs) such as BERT and variants [35, 98, 88] use a denoising objective that involves masking a portion of the input text. This training objective allows MLMs to obtain powerful context-aware representations that are highly effective for downstream tasks. However, the nature of the masking objective does not readily lend itself to direct text generation in place of the masked tokens.

The central question, therefore, is how to reconcile the unidirectional nature of language models with the need for context-aware and holistic text representations. How can we design models that can utilize contextual information while still being able to generate text, effectively leveraging the interdependencies between different parts of the text, and generating coherent and contextually appropriate completions?

Previous and subsequent work have proposed sequence-to-sequence approaches [37, 88, 40, 122] for text infilling. However, these methods are subject to high failure rate when the ratio of masked tokens diverges from the training setup. We aim to develop a method where output generations are *guaranteed* to respect the canvas they are generated on.

In Chapter 2, we introduce **Blank Language Model** (BLM), a new sequence model that addresses the challenge of generating coherent and contextually appropriate text by filling in blanks in the input. Instead of factorizing from left to right, BLM learns the joint probability of generating a sequence of words together with the generating order:

$$p_\theta(\mathbf{x}) = \sum_{\sigma \sim S_n} p_\theta(x, \sigma).$$

BLM operates on a dynamic canvas that consists of words and special blank symbols. Given an input text with one or more blanks, the model determines which words to place in the blanks and whether to split them into new blanks. The model continues generating until there are no more blanks left to fill. This process allows the BLM to generate text that seamlessly integrates with the existing content.

When starting from a single blank, BLM can generate complete sentences or paragraphs, similar to regular language models. Alternatively, it can take as input a partially completed text with specified blank locations, enabling users to guide the generation process and focus on specific sections of the text.

To train BLM, we propose an efficient training procedure using a lower bound of the marginal data likelihood. Under our training framework, the model learns the joint distribution of blank completions together with a generation order. During testing, BLM can employ traditional decoding algorithms such as greedy decoding or beam search to generate the most likely completions given the context.

On the text infilling task, we demonstrate the superiority of our method over models trained at a comparable scale. Additionally, we showcase the versatility of BLM in various scenarios, including style transfer and ancient text restoration. Since the publication of our work in [135], other authors have adopted the BLM framework and shown its effectiveness in different NLP tasks, such as information fusion [133], molecule generation [158], and material design [159].

1.2 Conformal Language Modeling

One concerning phenomenon observed in language models is the occurrence of *hallucinations*. Hallucinations refer to the generation of plausible yet false statements by the model. These false statements can range from factual inaccuracies to entirely fabricated information. The prevalence of hallucinations in language models presents significant challenges in real-world applications. In domains where accurate and reliable information is paramount, such as healthcare or legal advice, the presence of hallucinations can result in severe consequences. Users may unknowingly receive misleading or erroneous information, leading to incorrect decisions, potential harm to individuals, or detrimental effects on organizations.

Researchers and practitioners have often addressed this challenge by focusing on enhancing the interpretability of these black box models. However, the problem lies not only in understanding how the models arrive at their predictions but also in ensuring the accuracy and reliability of the generated output. Interpreting the inner workings of the models does not guarantee correctness or offer a measure of confidence in the produced outputs.

Practitioners seeking a measure of uncertainty have commonly relied on raw model likelihoods as a proxy of model confidence. However, those probability scores have been shown to be unreliable [34, 77, 105, 149]. This unreliability becomes apparent in scenarios where a text contains an incorrect statement, followed by a factual one. Due to the conditioning on previous context, the second sentence, which may contradict the first, would have a low likelihood. Some approaches propose to calibrate these logits [70, 77, 106, 179] but they still cannot provide *guarantees* that the output will consistently meet a certain level of quality.

Conformal prediction provides a framework for creating prediction sets that have a high probability of containing correct answers. It offers a model-agnostic way to guarantee a certain level of performance, by producing predition regions instead of point predictions. Traditionally, conformal prediction methods involve filtering the output space of a model using a nonconformity measure (e.g. $\mathcal{N}(x, y)$), which quantifies the deviation between the predicted output of a data point and the outputs observed in the training data. However, in the case of generative language models, the output space is unbounded and combinatorial in nature, rendering exhaustive exploration and enumeration of candidate outputs intractable.

How to **extend conformal prediction to generative language models**? In Chapter 3, we propose a novel approach to generate prediction sets via an iterative sampling strategy. We show that our method can effectively produce prediction sets with provably high probability of containing at least one correct answer.

The key idea behind our method is to conformalize the *generative process* rather than the outputs. We introduce a *stopping rule* that determines when to halt the generation of candidate outputs for a given input. This rule terminates the generation process when the desired level of confidence is reached. In conjunction, we use a *rejection rule* during the generation process to remove undesirable outputs, such as those deemed low-quality or redundant. To optimize the precision and coverage of the prediction sets, we jointly calibrate these two rules. This calibration enables us to identify the appropriate thresholds that yield the most effective prediction sets while maintaining the desired coverage level.

Our method naturally extends to identifying the most confident components within each generation and providing statistical guarantees that they are each independently correct, i.e. that they are *not hallucinations*.

We validate our method across three diverse tasks: open-domain question answering, radiology report generation, and news article summarization. We demonstrate valid risk control on multiple diverse tasks with different formats of LMs, while still retaining meaningful output sets that are precise on average, as compared to baselines.

1.3 Neural Obufscation for De-identification

In the United States, the Health Insurance Portability and Accountability Act (HIPAA) [60] is a legislation that defines Protected Health Information (PHI) as any information about a patient that can be used to identify the individual. It sets restrictions on the release of such PHI to protect patient privacy. To facilitate the sharing and analysis of health data while preserving privacy, HIPAA provides two approaches: safe harbor and expert determination.

The safe harbor method requires the removal of 18 specific identifiers, such as names, addresses, and medical record numbers, effectively de-identifying the health data. If all 18 identifiers are removed, the data is considered de-identified and can be used without restrictions under HIPAA. On the other hand, expert determination involves obtaining the professional judgment of a qualified expert who assesses the risk of identifying an individual from the health information and determines it to be very small.

While techniques like differential privacy [42, 43, 2] offer strong statistical guarantees by introducing noise into the data, they often come at the cost of utility. The added noise can distort the data and limit its usability for research and analysis purposes.

In practice, many clinical settings employ the safe harbor approach due to its simplicity. However, the availability of diverse data sources and advanced computational techniques enable attacks that can undermine the privacy protections provided by safe harbor [142]. As a result, there is a growing need for more robust de-identification methods that can withstand such attacks and preserve patient privacy.

In Chapter 4, we aim to challenge the *status quo* by addressing the limitations of existing de-identification methods and proposing a novel approach for data owners who seek to enhance privacy beyond the manual removal of identifiers. Our goal is to strike a **balance between privacy and utility** and offer a viable option for data owners who are committed to safeguarding privacy but cannot afford the utility sacrifices imposed by existing approaches.

To measure privacy, we introduce the concept of guesswork [101, 102, 7, 120, 14], which quantifies the number of trials an adversary would require to guess private information, such as a private key, when querying an oracle. Higher guesswork corresponds to higher privacy.

We then propose Syfer as a neural obfuscation method to protect against re-identification attacks. In our framework, data owners encode their data with a random neural network acting as their private key before releasing it publicly. While arbitrary random neural networks alone are not sufficient to achieve privacy on real-world data (e.g., X-rays), we demonstrate how to shape the distribution of private keys by composing random layers with trained *obfuscator* layers. The obfuscator layers are trained on public data, allowing us to precondition random transformations in order to achieve privacy against an estimated attacker while maintaining the invertability of the whole transformation. In particular, our method tailors the distribution of random encoders to capture the characteristics of real-world X-rays.

Additionally, we introduce a flexible computational attacker and a realistic chest X-ray utility benchmark to evaluate the trade-offs between privacy and utility. Our results showcase a significant improvement in the privacy-utility trade-offs compared to a differentially private baseline, with a notable 25-point AUC improvement and high guesswork. Furthermore, our learned encoding schemes offer enhanced privacy-utility trade-offs for arbitrary and unknown downstream tasks.

We strive to provide data owners with a comprehensive understanding of the proposed method and its potential benefits, as well as its limitations. We will address the challenges associated with implementing the proposed approach and discuss the potential implications for its real-world application.

Chapter 2

Blank Language Model

This chapter presents Blank Language Model (BLM), a model that generates sequences by dynamically creating and filling in blanks. Unlike previous masked language models [35] or the Insertion Transformer [140], BLM uses blanks to control which part of the sequence to expand. This fine-grained control of generation is ideal for a variety of text editing and rewriting tasks. The model can start from a single blank or partially completed text with blanks at specified locations. It iteratively determines which word to place in a blank and whether to insert new blanks, and stops generating when no blanks are left to fill. BLM can be efficiently trained using a lower bound of the marginal data likelihood, and achieves perplexity comparable to traditional left-to-right language models on the Penn Treebank and WikiText datasets. On the task of filling missing text snippets, BLM significantly outperforms all other baselines in terms of both accuracy and fluency. Experiments on style transfer and damaged ancient text restoration demonstrate the potential of this framework for a wide range of applications.

They also have ____ which ____ .
They also have *ice cream* which *is really good* .

Figure 2-1: BLM fills in blanks of arbitrary length.

2.1 Introduction

Neural language models have shown impressive performance across many applications such as machine translation and summarization where the text is generated from scratch [10, 127]. However, a broader set of text generation tasks — including text editing, information fusion, and ancient text restoration — requires the model to start with partially specified text and generate the missing fragments. In the general setup, the input document may have any number of missing spans, and each span may have an unknown number of missing tokens. To perform this text infilling task [187], a model should: (1) provide fine-grained control over the generation location, (2) accommodate a variable number of missing tokens, and (3) respect both the preceding and following context.

Existing approaches focus on adapting left-to-right language models for text infilling. Intricate inference algorithms leveraging dynamic programming or gradient search are proposed to find the filling content that has a high likelihood within the surrounding context [141, 94, 180]. These methods make simplified Markov assumptions, require high decoding time complexity, and cannot adapt to variable infilling length. Alternatively, [37] predict the concatenation of the infilling content, but do not guarantee that the output will match the number of missing spans in the input.

In this work, we introduce the Blank Language Model (BLM), which uses a special "__" symbol to control where tokens can be placed. The generation of BLM follows the grammar of replacing a blank with a word and possibly adjoining blanks. By jointly modeling context and missing content, BLM supports the control of generation location and produces consistent infilling of variable length.

Our model can start from a single blank or partial text with blanks in specified locations. It maps the entire input into a sequence of vector representations, and further processes the representations in blank positions to determine the generation action. Generation actions are performed iteratively until there are no blanks. Since multiple trajectories of BLM actions can produce the same final text, we train the model by maximizing a lower bound of the log-likelihood marginalized over trajectories. At test time, we can use simple greedy decoding or beam search to fill in the blanks in the input text.

BLM shows superior performance in text infilling [187], ancient text restoration [8] and style transfer [134], demonstrating its flexibility to generate text in diverse conditions. Our model achieves 92.5% accuracy and BLEU score of 23.1 on the Amazon dataset for sentiment transfer. On the task of restoring ancient text that lost half of the characters, we reduce the error rate by 3.3 points compared to previous methods.

2.2 Related work

Recent work has explored various sequence models for non-autoregressive machine translation [55]. The Insertion Transformer supports dynamic canvas with word insertion [140], but does not allow users to specify where to insert. The model is unaware of which parts of the canvas are contiguous text spans that should remain intact, and which (potentially scattered) parts need to be filled in. Directly forcing the Insertion Transformer to perform text infilling can therefore lead to suboptimal solutions. The Levenshtein Transformer combines insertion and deletion through complex policy learning [57]. Its insertion mechanism is a two-stage process in which placeholders are first predicted and then filled-in in a masked language model (MLM) manner. In text infilling where the blanks/placeholders are given, it reduces to an MLM.

MLMs are commonly used in representation learning [35, 74]. To use them in rewriting tasks, one needs to specify the insertion length in advance and heuristically determine the generation order among the masks [46, 155, 52]. Similarly, XL-Net requires absolute positional embedding and thus does not support unknown-length text infilling [174, 136]. BLM provides a natural formulation for generative modeling that can dynamically accommodate insertions of various length.

Another line of work focuses on finding an optimal language generation order,

	Canvas c	Action a			
Step t		Location b	Word w	(Left blank l,	Right blank r)
0.	$\underline{\#1}$	#1	is	Yes	Yes
1.	$\underline{\#1}$ is $\underline{\#2}$	#1	customer	No	Yes
2.	customer $\underline{\#1}$ is $\underline{\#2}$	#2	awesome	No	No
3.	customer $\underline{\#1}$ is awesome	#1	service	No	No
4.	customer service is awesome	-End-			

Figure 2-2: An example trajectory that generates the sentence "customer service is awesome". Each action is a tuple (b, w, l, r), indicating the blank location b selected for expansion, the word w to fill in, whether to create a left blank l, and whether to create a right blank r.

such as syntax-based generation [44] and learning adaptive generation order [56]. These approaches are tailored to generation from scratch in a specific order. Our model instead is attuned for text rewriting, where the missing parts can be located anywhere in the input text, and the algorithm must flexibly complete them.

2.3 Method

A blank language model (BLM) generates sequences by creating and filling in blanks. Generation starts with a single blank and ends when there is no blank. In each step, the model selects a blank "__", predicts a word w, and fills the blank with "w", "__ w", "w __", or "__ w __". This way, a blank can be expanded to any number of words.

We define a *canvas* as a sequence of words interspersed with special "__" tokens. The subsequent action is conditioned on this intermediate stage of generation. Suppose the current canvas is $c = (c_1, \cdots, c_n)$ with blanks located at indices $b_1, \cdots, b_k$ (i.e. $c_{b_i} =$ "__", for $i = 1, \ldots, k$). BLM maps this canvas to a distribution over actions specifying how the canvas is to be revised:

$$p(b, w, l, r | c; \theta) = \text{BLM}(c) \tag{2.1}$$

where $b \in \{b_1, \cdots, b_k\}$ is a blank location; w is a word in the vocabulary V; $l, r \in \{0, 1\}$ denote whether or not to create a blank to the left and right of w; and θ are the model parameters. The *action*, defined as the tuple (b, w, l, r) uniquely specifies the next state of canvas (see Fig. 2-2 for illustration).

Alternatively, we can view the actions in BLM as production rules in a grammar. Each blank represents a nonterminal symbol or the start symbol, and the terminal symbols come from the vocabulary V. The production rules are restricted to be of the form "__" $\rightarrow$ "__?w__?" for $w \in V$, where "?" indicates that the preceding symbol is optional. In contrast to context-free grammars, the probability distribution over production rules in BLM is conditioned on the entire canvas generated so far.

Model Architecture We encode the canvas c into a sequence of representations $(z_1, \cdots, z_n)$, and take representations $Z = (z_{b_1}, \cdots, z_{b_k})$ where the blanks are located. Let d denote the dimension of z's. We factorize the joint distribution $p(b, w, l, r|c; \theta)$ into three parts (shown in Fig. 2-3):

1. Choose a blank:

$$p(b_i|c;\theta) = \mathrm{Softmax}(u^T Z) \tag{2.2}$$

   where $u \in \mathbb{R}^d$ is a parameter vector to project z's into one-dimensional logits.

2. Predict a word for the selected blank:

$$p(w|c, b_i;\theta) = \mathrm{Softmax}(W z_{b_i}) \tag{2.3}$$

   where $W \in \mathbb{R}^{|V|\times d}$ is a parameter matrix to project z_{b_i} into the vocabulary.

3. Decide whether or not to create blanks to the left and right of the predicted word:

$$p(l, r|c, b_i, w;\theta) = \mathrm{MLP}(z_{b_i}, v_w) \tag{2.4}$$

   where v_w is the word vector of w, and MLP is a multilayer perceptron with 4 output classes: Left.Yes/No $\times$ Right.Yes/No.

Likelihood Now let us consider the probability $p(x;\theta)$ of generating a sentence/-paragraph $x = (x_1, \cdots, x_n)$ under the BLM. We call the generating process from an initial blank to complete text a *trajectory*. The same final text x can be realized by multiple trajectories. However, if we specify the order in which the words in x are generated, the trajectory will be uniquely determined. Consider the example trajectory of a 4-word sentence in Fig. 2-2. Given the order $(3, 1, 4, 2)$, at step 0 when we generate x_3, both left and right blanks are created for future generations of x_1 and x_2, x_4. In step 1 of generating x_1, only a right blank is created for the future x_2. Subsequent steps can be deduced by analogy. The correspondence between trajectories and generation orders allows us to write the marginal likelihood as:

$$\begin{aligned} p(x;\theta) &= \sum_{\sigma \in S_n} p(x, \sigma; \theta) \\ &= \sum_{\sigma \in S_n} \prod_{t=0}^{n-1} p(a_t^{x,\sigma} | c_t^{x,\sigma}; \theta) \end{aligned} \tag{2.5}$$

where S_n is the set of all n-permutations; $a_t^{x,\sigma}, c_t^{x,\sigma}$ denote the action and canvas at step t under sentence x and order σ, respectively (cf. Fig. 2-2).

Training Different losses have been proposed to train generalized sequence models. For instance, BERT and XL-Net mask and predict 15% of tokens conditioned on the rest. This strategy is more suitable for representation learning rather than generation. Insertion Transformer masks different numbers of tokens and weights them with uniform loss or binary tree loss [140, 24]. It aims to perform fast inference through parallel decoding. Here, we present a training objective from the language modeling perspective by estimating the log likelihood of generating x.

Directly computing the marginal likelihood over $n!$ orders is intractable. We apply

Jensen's inequality to lower bound the log likelihood:

$$\begin{aligned}\log p(x;\theta) &= \log \sum_{\sigma\in S_n} \prod_{t=0}^{n-1} p(a_t^{x,\sigma}|c_t^{x,\sigma};\theta) \\ &\geq \log(n!) + \frac{1}{n!}\sum_{\sigma\in S_n}\sum_{t=0}^{n-1} \log p(a_t^{x,\sigma}|c_t^{x,\sigma};\theta) \qquad (2.6)\end{aligned}$$

where equality holds when the posterior $p(\sigma|x;\theta)$ is uniform. By maximizing this lower bound, we do not favor any particular order, but encourage the model to realize x equally well in all orders. It can help the model to complete any partial input text regardless of the position of blanks.

A naive training algorithm is to directly estimate the lower bound in Eq. (2.6): first uniformly sample a permutation σ from S_n and a step t from 0 to $n-1$, then construct the canvas $c_t^{x,\sigma}$, and compute the estimated loss $[-\log(n!) - n \cdot \log p(a_t^{x,\sigma}|c_t^{x,\sigma};\theta)]$. However, this procedure has a large variance and can only compute the loss of a single action in one pass (in contrast to left-to-right language models that compute n word losses per pass).

To train the model more efficiently, we note that the canvas $c_t^{x,\sigma}$ depends only on the first t elements of σ. Hence we can combine into one pass the loss calculations of trajectories that are the same in the first t steps but different at the $t+1$ step. Switching the summation order of σ and t, we have:

$$\begin{aligned}&\sum_{t=0}^{n-1}\frac{1}{n!}\sum_{\sigma\in S_n} \log p(a_t^{x,\sigma}|c_t^{x,\sigma};\theta) \\ =\ & n \cdot \mathbb{E}_t \mathbb{E}_{\sigma_{1:t}} \mathbb{E}_{\sigma_{t+1}} \mathbb{E}_{\sigma_{t+2:n}} \left[\log p(a_t^{x,\sigma}|c_t^{x,\sigma};\theta)\right] \\ =\ & n \cdot \mathbb{E}_t \mathbb{E}_{\sigma_{1:t}} \mathbb{E}_{\sigma_{t+1}} \left[\log p(a_t^{x,\sigma}|c_t^{x,\sigma};\theta)\right] \\ =\ & \mathbb{E}_t \mathbb{E}_{\sigma_{1:t}} \left[\frac{n}{n-t}\sum_{\sigma_{t+1}} \log p(a_t^{x,\sigma}|c_t^{x,\sigma};\theta)\right] \qquad (2.7)\end{aligned}$$

which leads to our efficient training algorithm: sample t from 0 to $n-1$ and partial permutation $\sigma_{1:t}$, construct the canvas $c_t^{x,\sigma}$, and compute loss:

$$-\log(n!) - \frac{n}{n-t} \sum_{\sigma_{t+1}} \log p(a_t^{x,\sigma} | c_t^{x,\sigma}; \theta) \tag{2.8}$$

The whole process is illustrated in Algorithm 1. In this way, we can compute in expectation $n/2$ action losses per pass.

2.4 Experiments

We test BLM's capacity to rewrite specified portions of text on three tasks: text infilling [187], ancient text restoration [8] and style transfer [134]. Fig. 2-4 displays example inputs and outputs for these tasks. We also measure the perplexity of BLM on language modeling benchmarks and compare with traditional left-to-right language models.

Experimental Details In all experiments, the sequence representations in BLM are obtained using the encoder module of `transformer_base` [150] (6 layers, 8 heads, $d_{model} = 512$, $d_{ff} = 2048$, $d_k = d_v = 64$). The MLP used for blank prediction has one hidden layer of size 1024. Weight decay, learning rate, and dropout are tuned based on the loss on the validation set for each dataset respectively. When decoding, we use beam size in $\{1, 5, 10\}$ and choose the best value as observed on the validation set. We note that beam search in BLM does not search for the sentence with the maximum marginal likelihood $p(x; \theta)$, but instead for a sentence *and* a trajectory that have the maximum joint likelihood $p(x, \sigma; \theta)$.

2.4.1 Text Infilling

Dataset We experiment on the Yahoo Answers dataset, which has 100K/10K/10K documents for train/valid/test respectively [176]. A document has a maximum length of 200 words, with an average of 78 words. Following [187], we automatically compile test data by deleting portions of documents. For each document x, we randomly mask a given ratio r of its tokens. Contiguous masked tokens are collapsed into a single "__", resulting in a canvas c to be completed.

Metrics We measure generation's accuracy by computing its BLEU score against the original document x, and fluency as its perplexity evaluated by a pre-trained (left-to-right) language model. We also report the failure rate, which is the percentage of invalid generations, such as missing existing words or not filling in all the blanks.

Baselines We compare BLM with five baselines:

- *Insertion Transformer (InsT)*: By default, InsT does not support controlling the insertion position. We force it to produce valid generations by normalizing the predictions over valid locations, disabling the $\langle\text{eos}\rangle$ prediction unless all blanks have been filled, and prioritizing slots that have not been filled yet. Without these steps, InsT would have a failure rate $\geq 88\%$. An illustration is provided Figure 2-5
- *MLM (oracle length)*: MLM for text infilling requires predicting the length of each blank. Here we replace blanks with the target number of $\langle\text{mask}\rangle$ tokens, and fill them autoregressively by the most-confident-first heuristic. We illustrate the process on Figure 2-6
- *BERT+LM*: We use BERT's representation of each blank as seed for a left-to-right language model that learns to generate the tokens in the corresponding blank. At inference time, the multiple blanks are filled in one after another, conditioned on previous generations. The generation process is illustrated on Figure 2-7.
- *Seq2seq-full* [37]: We train a seq2seq model to output the full document x from input c. Note that it may have invalid outputs that do not match the input format, such as missing existing tokens in c or generating tokens in incorrect locations.
- *Seq2seq-fill* [37]: We train a seq2seq model to output only tokens to be placed in the blanks, with a special '|' token to indicate separation. For the example in

Fig. 2-4, its target output will be “ice cream |is really good”. Unlike *seq2seq-full*, *seq2seq-fill* does not have the problem of losing existing tokens in c. However, it may still fail to generate the correct number of ‘|’ that matches the input.

Results As shown in Table 2.1, BLM achieves the highest BLEU score at all mask ratios: 0.7 to 1.7 higher than InsT, 2.6 to 4.1 higher than MLM with oracle length, and 3.7 to 9.4 higher than BERT+LM. InsT is not trained with insertion position control. Restricting it to generate at the specified positions thus biases the model towards making suboptimal completions. MLM is trained to independently predict masked tokens instead of jointly modeling them. Even with the target number of ⟨mask⟩ tokens given, its performance is still inferior to BLM. BERT+LM lags behind other models. In BERT training, one mask corresponds to one token, whereas a blank here can cover multiple tokens, and the distance between words is not fixed. Hence, it is difficult for the LM to complete the sentence from BERT representations.

Seq2seq-full has BLEU scores closest to BLM. However, its failure rate ranges from 15% to 40.6% as the mask ratio increases. Seq2seq-fill performs worse than Seq2seq-full, possibly because the decoder has to model segmented text while counting the number of blanks.

In terms of fluency, Table 2.2 shows that outputs of BLM, InsT and Seq2seq-full all have perplexity lower than original data perplexity. This is because with beam search, models tend to generate the most typical output with the highest likelihood [63].

Examination of model generations confirms the superiority of BLM. In Fig. 2-8, we showcase example outputs by each model at different mask ratios. In low mask ratio settings, models only need to fill in the blanks with a single word to produce grammatical completions. Most models succeed in this task. With a higher mask ratio of 50%, the main ideas of the document are concealed, and the infilling task is much more challenging. Models need to creatively generate sentences that fit the imposed canvas. Although the original meaning of the sentence is not recovered, BLM is the only model able to produce a coherent document with consistency between the question and the answer.

Overall, BLM displays the best performance both quantitatively and qualitatively. Its inherent text infilling ability frees it from length, order, or termination heuristics used by other methods.

Mask ratio	10%	20%	30%	40%	50%
No infill	75.2	55.0	37.4	23.6	13.0
InsT	**84.8**	**72.3**	**58.9**	**46.0**	**33.8**
MLM (oracle length)	83.7	69.3	55.5	43.2	32.2
BERT+LM	82.8	66.3	50.3	37.4	26.2
Seq2seq-full	86.3	72.9	59.4	46.3	34.0
Seq2seq-fill	82.8	67.5	52.9	39.9	28.6
BLM	**86.5**	**73.2**	**59.6**	**46.8**	**34.8**

Table 2.1: BLEU scores of generated documents by different models for text infilling.

Mask ratio	10%	20%	30%	40%	50%
No infill	98.4	163.0	266.3	421.0	647.9
InsT	**48.3**	**44.2**	41.8	39.7	37.7
MLM (oracle length)	58.4	59.8	59.8	59.0	56.8
BERT+LM	55.1	55.2	54.9	56.5	53.6
Seq2seq-full	51.3	46.9	41.0	**31.9**	**20.6**
Seq2seq-fill	64.6	71.0	73.4	65.6	48.7
BLM	50.2	44.9	**39.9**	35.0	32.7

Table 2.2: Perplexity (PPL) of generated documents by different models for text infilling. The perplexity is measured by a pre-trained left-to-right language model, and the original documents have perplexity 55.8.

2.4.2 Ancient Text Restoration

Ancient text restoration is a form of text infilling where there are fragments in ancient documents that are illegible due to time-related damages and need to be recovered. [8] introduces the PHI-ML dataset made of fragments of ancient Greek inscriptions. Restoration is performed at the character-level. The number of characters to recover is assumed to be known and indicated by a corresponding number of '?' symbols, as shown in the second row of Fig. 2-4. In reality, when epigraphists restore a deteriorated document, the length of the lost fragment is unknown and needs to be guessed

Mask ratio	10%	20%	30%	40%	50%
Seq2seq-full	15.0	22.4	28.7	33.3	40.6
Seq2seq-fill	31.0	28.4	34.5	42.5	47.2

Table 2.3: Infilling failure rate (%) of seq2seq models. Other methods always produce valid outputs.

as a first step. While models proposed by [8] relies on expert conjectures, we note that BLM can bypass this limitation and flexibly generate completions without this additional knowledge. However, in order to compute the character error rate (CER) for each '?' and have a fair comparison with previous work, we evaluate our model in the length-aware setting.

Length-aware BLM (L-BLM) We present a variant of BLM adapted to the specific features of this task. The vocabulary V is an alphabet of characters from the ancient Greek language. We extend V with special "$\underline{\ \ \texttt{[}t\texttt{]}\ \ }$" tokens that denote the length of the fragment to recover. Specifically, as a preprocessing step, consecutive '?' characters are collapsed into a single "$\underline{\ \ \texttt{[}t\texttt{]}\ \ }$" token, where t is the number of '?' symbols. For each such blank token, L-BLM is trained to predict a character to fill in and the length $l \in \{0, \cdots, t-1\}$ of the new blank to its left. The length of the new blank on the right is accordingly $t - 1 - l$.

Dataset The PHI-ML dataset contains about 3 million words / 18 million characters. We report the statistics in Figure 2-9. We evaluate models in two settings: *single-slot* and *multi-slot*. For the single-slot setting, we use the testing script of [8] which samples a context of length $L = 1000$ from an inscription, then samples a slot of length $C \in [1, 10]$ from that context. The characters from the slot are replaced with '?' and constitute the target. For the multi-slot setting, we progressively increase the number of slots, yielding mask ratios of 25%, 40% and 50% respectively.

Baselines [8] proposed two models: *Pythia*, a character-level seq2seq-based approach; and *Pythia-Word*, a variant of Pythia that uses both character and word

	Single-	Multi-slot		
Mask ratio	1%	25%	40%	50%
Human	57.3%	-	-	-
Pythia	32.5%	-	-	-
Pythia-Word	**29.1%**	**36.9%**	42.3%	44.9%
L-BLM	33.7%	37.1%	**37.9%**	**41.6%**

Table 2.4: CER for ancient text restoration.

representations as input. During training, the model learns to recover the missing characters of examples where a random slot has been masked. When testing on the multi-slot setting, Pythia(-Word) is applied iteratively with beam size 20 for each slot.

Results Table 2.4 summarizes the CER of all models in both settings. L-BLM achieves similar CER as Pythia in the single-slot setting, significantly outperforming human experts. Augmented with word representations, Pythia-Word further decreases the error rate compared to character-only methods.

In reality, restoring damaged inscriptions requires reconstructing multiple lost fragments. As a larger proportion of text is missing, Pythia-Word's performance is degraded. L-BLM is robust to the setting change and outperforms Pythia-Word at the mask ratio of 40% and 50% by 4.4 and 3.3 points, respectively. We posit that L-BLM's advantage lies in its ability to maximize the joint likelihood of the completions over all slots. In contrast, Pythia-Word's is only aware of one slot at a time, and beam search is performed locally within each slot.

2.4.3 Sentiment Transfer

The goal of sentiment transfer is to modify the sentiment of a sentence while maintaining its topic [134]. An example is described on the third row of Fig. 2-4. Inspired by the way humans perform rewriting, we follow a recent line of work in style transfer that adopts a two-step approach [89, 172, 164]:

1. Remove words and expressions of high polarity from the source sentence;
2. Complete the partial sentence with words and expressions of the target sentiment.

Specifically, we adapt the *Mask-And-Infill (M&I)* framework of [164]. We perform Step 1 by training a Bi-LSTM sentiment classifier and masking words whose attention weight is above average. We evaluate the contribution of our model as an infilling module in Step 2 in place of their fine-tuned BERT model. To this end, we train two instances of BLM on the dataset, one for each sentiment. At test time, the corresponding BLM is used to produce completions of the target sentiment.

[164] further train the infilling model with the classifier to improve transfer accuracy. They use soft words relaxation to backprop gradients from the classifier to the generator. For BLM, however, we cannot pick locations or insert blanks as “soft” choices, making it challenging to employ a classifier at training time. Nevertheless, we can easily apply the classifier to guide inference. We sample 10 outputs and keep the one with the highest classifier ranking. It is not slower than beam search with size 10 and can be fully parallelized.

Datasets We test on the Yelp and Amazon review datasets [134, 89]. The Yelp dataset has 450K/4K/1K non-parallel sentences for train/valid/test respectively, and the Amazon dataset has 555K/2K/1K sentences. (see Table 2-10). Each sentence is labeled as either positive or negative. The task is to flip the sentiment of each sentence.

Metrics We use evaluation methods introduced by prior work [134, 89, 164, 175]. To assess the accuracy of generated sentences with respect to the target sentiment, we use a pretrained CNN classifier that achieves 97.7% accuracy on the Yelp dataset and 82.2% accuracy on the Amazon dataset. We also measure the BLEU score between transferred sentences and human references.

	Yelp		Amazon	
	ACC	BLEU	ACC	BLEU
Delete-And-Retrieve [89]	88.7	8.4	48.0	22.8
UMT [183]	96.6	22.8	84.1	33.9
Point-Then-Operate [163]	91.5	**29.9**	40.2	**41.9**
Mask & Infill with MLM [164]	41.5	15.9	31.2	32.1
+ classifier	**97.3**	14.1	75.9	28.5
Mask only (canvas)	42.6	19.9	37.1	21.2
Mask & Infill with BLM	79.6	21.9	52.0	24.7
+ classifier	96.5	21.5	**92.5**	23.1

Table 2.5: Accuracy and BLEU scores for style transfer. Accuracy measures the percentage of sentences labeled as the target sentiment by the classifier. BLEU is evaluated against human reference generations. For reference, we also report accuracy and BLEU scores of the canvas (i.e. the original masked sentence).

Baselines [89] propose *Delete-And-Retrieve (DAR)* which generates the output conditioned on the hidden representation of the masked sentence and retrieved expressions of the target sentiment. is a seq2seq-based approach where hidden representations of the masked sentence is concatenated with a learned attribute embedding before decoding. Additionally, a retrieval module is used to collect relevant expressions of the target sentiment to guide generation. [183] adapt *Unsupervised Machine Translation (UMT)* techniques and iteratively train the style transfer models with back translation and a style classifier. [163] present a hierarchical reinforcement learning method *Point-Then-Operate (PTO)*, in which a pointer indicates operation positions and an operator modifies them.

We follow the *Mask-And-Infill (M&I)* framework of [164]. Their infilling module is a fine-tuned BERT$_{base}$ model, while ours is BLM.

Results In Table 2.5, we can see that directly applying BLM as the infilling module is significantly better than MLM. The accuracy on Yelp and Amazon datasets is increased by 38.1% and 20.8%, respectively. In addition to the aforementioned problem of MLM being trained to predict masked tokens independently, it must generate the same number of tokens as in the source sentence, whereas our BLM formulation is not subject to this constraint. Our simple use of a classifier at inference time further improves accuracy. It achieves the highest accuracy of 92.5% on Amazon with a small decrease in BLEU, indicating that BLM can easily find high-quality outputs.

Results in Table 2.5 demonstrate the ability of different models to perform text infilling for style transfer. The DELETE-AND-RETRIEVE method with the frequency-ratio based masking strategy achieves high sentiment accuracy, but can only do so at the expense of content fidelity. By constraining BLM to fill in blanks in between content words, we ensure that the predictions will yield high content preservation, improving both BLEU score and sentiment accuracy over the original masked sentence.

The MLM formulation in MASK-AND-INFILL is problematic on this task for two reasons. By design, MLM is forced to generate the same number of tokens as there were originally in the source sentence, making it more difficult to produce coherent sentences that are consistent with the target sentiment. Furthermore, MLM is trained to predict the masked tokens independently rather than jointly, which further hurts performance. Our formulation of BLM does not suffer any of these weaknesses. With both masking strategies, our model outperforms the MASK-AND-INFILL baseline on all metrics, proving its superiority as the better-suited formulation for this setup.

In Fig. 2-11, we show examples generated by BLM on Yelp. It can dynamically adapt to the imposed canvas and fill in blanks with expressions of varied lengths, e.g., "**nowhere to be** found" → "the best i found" and "**definitely not**" → "always". We note that failure cases arise when negative words like "either" are left unmasked; BLM is then unable to produce satisfactory outputs from the canvas.

2.4.4 Language Modeling

Language modeling is a special case of text infilling where sequences are generated from scratch. Traditional left-to-right models dominate this task, but are not suitable for text infilling. Conversely, unconventional sequence models are rarely evaluated on language modeling. Here, we study the perplexity of BLM and Insertion Transformer, and compare them with left-to-right language models to provide additional insights.

m	1	10	100	1000
Estimated PPL	46.3	44.4	43.3	42.5

Table 2.6: The estimated perplexity of BLM with the number of MC samples m on WikiText-103.

	PTB	WT2	WT103
LSTM [54]	82.3	99.3	48.7
TCN [11]	88.7	-	45.2
AWD-LSTM [103]	57.3	**65.8**	-
Transformer [32]	-	-	30.1
Adaptive [9]	-	-	18.7
Transformer-XL [32]	**54.5**	-	**18.3**
InsT (our implementation)	77.3	91.4	39.4
BLM	69.2	81.2	42.5

Table 2.7: Perplexity on the PTB and WikiText datasets.

We use the Monte-Carlo method to estimate the likelihood in Eq. (2.5) with m samples. While the estimate is unbiased, given that per-word perplexity is a convex function of per-sentence likelihood, sampling estimates like ours are likely yielding a value higher than the actual perplexity (see Appendix A.2 for a proof). As m increases, it converges to the actual perplexity.

Datasets We test on three benchmark datasets: Penn Treebank (PTB) which has about 1M tokens [107], WikiText-2 (WT2) which has 2M tokens, and WikiText-103 (WT103) which has 103M tokens [104].

Results Table 2.6 shows the trend of estimated PPL with the number of samples m. We choose $m = 1000$ in our evaluation, which is close to convergence. Table 2.7 summarizes the perplexity of our model in comparison with previous work. The top results are achieved by the Transformer-XL [32] and the adaptive embedding method [9]. They use larger model sizes and supplementary techniques that can also be combined with our model. BLM rivals the Insertion Transformer and outperforms left-to-right language models with LSTM and Temporal Convolutional Network (TCN) architecture. Language modeling seems to still be challenging for free-order models. By reporting the perplexity of unconventional models like BLM, we hope to stimulate future work in this area to close the performance gap with traditional left-to-right models.

2.5 Qualitative results

In Figure 2-12, we present examples of the generation trajectory of BLM on the Yelp review dataset. This demonstrates the sequential generation of text by the model, where each row in the table corresponds to a different step in the generation process. It seems that the model's learned joint distribution (x, σ) is not guided by grammatical considerations. Instead, the model tends to choose words that have a high probability of appearing, such as determiners like "the," or polarity words like "terrible" or "favorite." This is because the model was trained on a dataset that primarily focused on high polarity expressions.

2.6 Conclusion

2.6.1 Summary of contributions

In this chapter, we proposed the Blank Language Model for flexible text generation. Given partially specified text with one or more blanks, BLM will fill in the blanks with a variable number of tokens consistent with the context. We demonstrate the effectiveness of our model on various text rewriting tasks, including text infilling, ancient text restoration and style transfer.

The action of BLM consists of selecting a blank and replacing it with a word and possibly adjoining blanks. We train BLM by optimizing a lower bound on the marginal data likelihood that sums over all possible generation trajectories. In this way, we encourage the model to realize a sentence equally well in all orders, which is suitable for filling arbitrary blanks. Appendix 2.5 shows examples generated by BLM along with their trajectories.

2.6.2 Future work

The research presented in this work opens up several exciting avenues for future exploration.

One natural direction is the application of BLM in other NLP tasks, such as template infilling, information fusion [133], and writing or coding assistance. For example, BLM can assist programmers by providing suggestions or completing code snippets based on the given context. This application has the potential to enhance coding productivity and improve the efficiency of software development.

Moreover, the formulation of BLM can be extended to sequence-to-sequence settings, enabling it to function as a conditional generative model. This extension opens up possibilities for supporting interactive editing and refining of generated text. BLM can be applied in machine translation systems, summarization software, and dialogue systems, where users can iteratively modify the generated text based on feedback. Additionally, there is great potential in extending BLM to tasks involving multimodal data, such as generating image captions, or video descriptions, that incorporate both text and visual information. In such cases, users can utilize BLM to specify the locations requiring editing or provide additional context to guide the model's focus on specific parts of the input that needs to be described.

In general, BLM can enable incremental text generation, where a user and the language model collaborate to construct text. The model can adapt in real-time to user feedback or dynamically incorporate new information. This feature makes BLM a valuable addition to collaborative writing tools.

Another avenue of research to explore is the expansion of its *action space*. In this work, the focus was on word and blank insertions as the actions. However, a valuable extension would be to incorporate token deletion capability. By allowing the model to choose to delete previously generated tokens as an action, BLM gains the ability to self-recover from mistakes or incorrect predictions. This enhancement would further improve its performance in interactive systems, where accuracy and coherence are essential. Incorporating token deletion as an action expands the model's flexibility and adaptability, enabling it to dynamically adjust its output based on user feedback or changing contexts.

While this work primarily focuses on language generation and text infilling tasks, it would be interesting to compare the learned representations of BLM with those generated by other pre-training methods. BLM combines the contextual information from both directions, similar to BERT-like models, while also being designed for text generation, similar to GPT-like models. It will be interesting to explore pre-training and fine-tuning BLM on downstream tasks that require both these capabilities.

Scaling up the training process of BLM is essential to fully harness its potential. Although beyond the scope of this thesis, the positive results obtained in this chapter provide a strong foundation for further exploration in this direction. Scaling up BLM would involve training on larger datasets, utilizing more computational resources, and potentially exploring distributed training techniques. This advancement would enable the model to handle more complex and diverse language tasks, opening up new possibilities for its application.

Furthermore, future research can extend the use of BLM beyond language generation. Recent studies have already explored its application in molecule generation and material design [158, 159]. Another domain of interest is music modeling, as the harmonic constraints naturally impose a canvas that composers fill in with melodies.

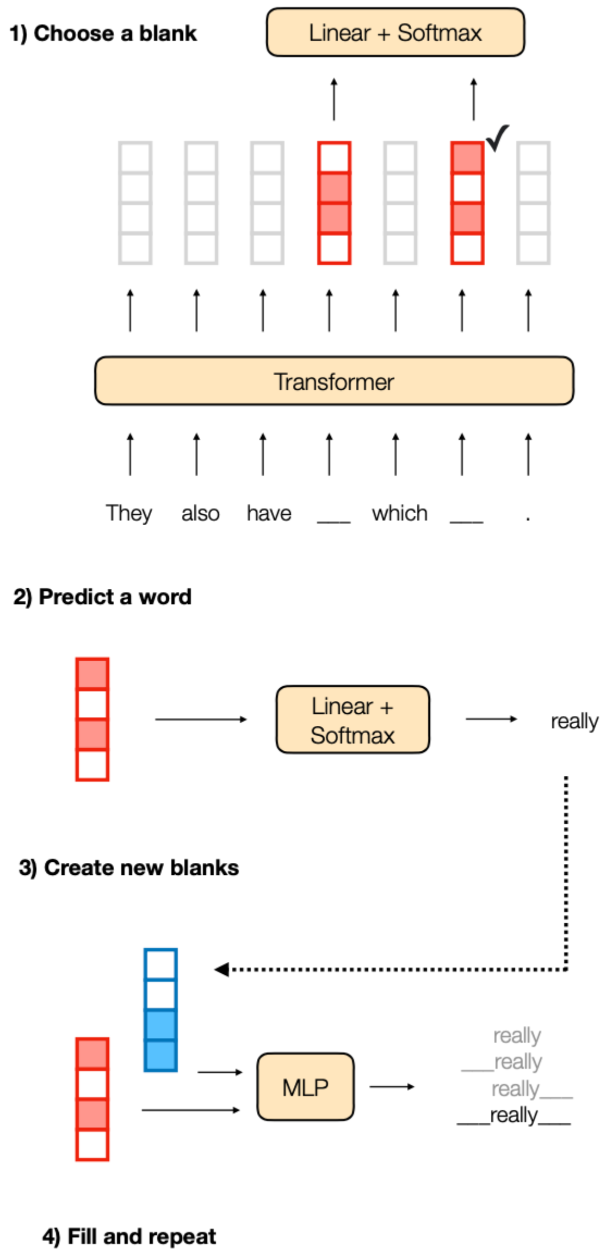


Figure 2-3: Architecture of the BLM. In the first stage, an index is chosen among all current blank positions. For that location, a word is selected in the second stage. In the final stage, the blank representation is concatenated with the chosen word's embedding and fed into an MLP to determine the creation of the following blanks.

Algorithm 1 BLM Training

```
1: Initialize model parameters $\theta$
2: while model not converged do
3:     Sample a training example $x = (x_1, \cdots, x_n)$
4:     Sample $t$ from 0 to $n - 1$
5:     Sample an $n$-permutation $\sigma$
6:     Construct canvas $c$ that keeps tokens $x_{\sigma_j}(j = 1, \cdots, t)$ and collapses remaining tokens as blanks
7:     Get $n - t$ target actions $a_{j-t}$ for filling $x_{\sigma_j}$ $(j = t + 1, \cdots, n)$ into canvas $c$
8:     Compute loss($\{a_1, \cdots, a_{n-t}\}$, model.forward($c$)) from Eq. (2.8)
9:     Update $\theta$ by gradient descent
```

They also have ____ which ____ .
They also have ice cream which is really good .
τε εγγονον εισαι??????σοφιαι
τε εγγονον εισαιου του σοφιαι
The employees were **super nice** and **efficient** !
The employees were rude and unprofessional !

Figure 2-4: Examples of input and output for text infilling, ancient text restoration, and style transfer tasks.

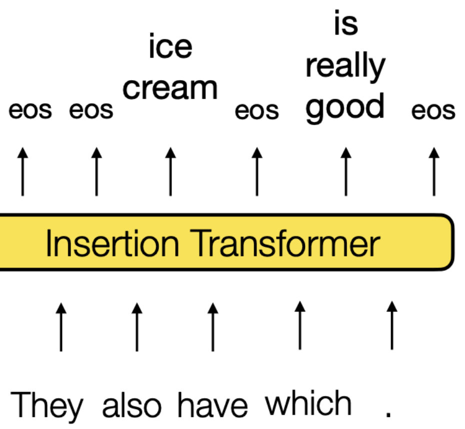


Figure 2-5: Schematic of the Insertion Transformer

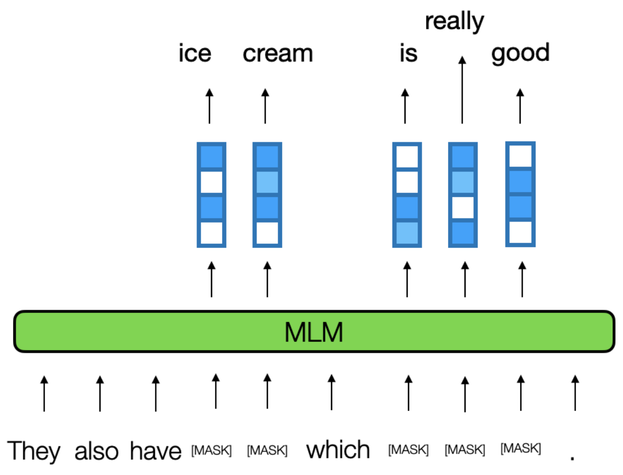


Figure 2-6: Schematic of the MLM baseline

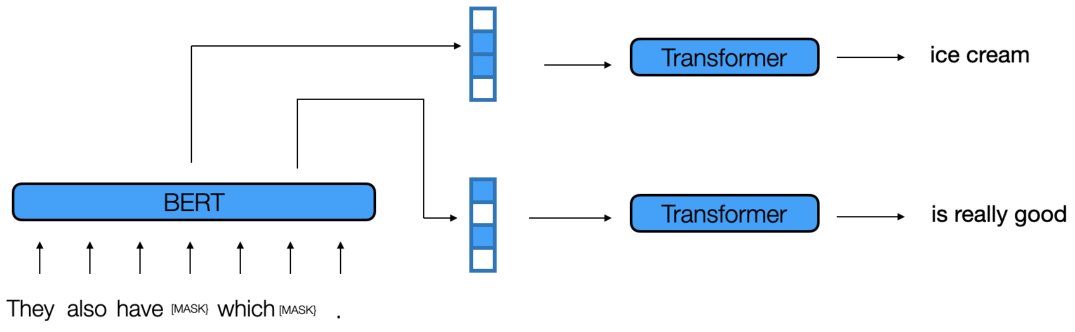


Figure 2-7: Schematic of the BERT + LM baseline

	Mask-ratio 10%	Mask-ratio 50%
Blanked	when time flies , ____ does it go ? ____ the center of the ____ to be recycled ____ made into new time .	when time ____ , where ____ ? ____ the ____ of ____ universe to ____ recycled ____ made into ____ .
BLM	when time flies , *where* does it go ? *for* the center of the *earth* to be recycled *and* made into new time .	when time *was created* , where *did it come from* ? *it was* the *first part* of *the* universe to *be* recycled *and* made into *space* .
InsT	when time flies , *where* does it go ? *for* the center of the *earth has* to be recycled *and* made into new time .	when time *was created* , where *was it* ? *what was* the *name* of *the* universe to *be* recycled *and* made into *space* .
MLM (oracle len)	when time flies , *where* does it go ? *from* the center of the *earth* to be recycled *converted* made into new time .	when time *is* , where *is the universe* ? *from* the *creation* of *the* universe to *be* recycled *and* made into *the universe* .
BERT+LM	when time flies , *where* does it go ? *to* the center of the *earth* to be recycled *came* made into new time .	when time *is* , where *to* ? *i need to find* the *way* of *the* universe to *be* recycled *and* made into *a lot* .
Seq2seq-full	when time flies , *where* does it go ? *at* the center of the *earth* to be recycled *and* made into new time .	when time *heals* , where *does it go* ? *it 's* the *end* of *the* universe to *be* recycled *and* made into *space* .
Seq2seq-fill	when time flies , *how* does it go ? *at* the center of the *earth* to be recycled *and* made into new time . how |at |earth |and	when time *is time* , where *is time* ? *time is* the *time* of *time* universe to *the* recycled *be* made into *and* . *the universe* is time |is time |time is |time |time |the |be |and |the universe
Original	when time flies , **where** does it go ? **to** the center of the **universe** to be recycled **and** made into new time .	when time **flies** , where **does it go** ? **to** the **center** of **the** universe to **be** recycled **and** made into **new time** .

Figure 2-8: Example generations of different models for text infilling on Yahoo Answers. Completions are in italic. Invalid completions are in red. For Seq2seq-fill, we present model outputs along with the merged document.

Split	Inscriptions	Word	Chars
Train	34,952	2,792K	16,300K
Dev	2,826	211K	1,230K
Test	2,949	223K	1,298K

Figure 2-9: Statistics of the PHI-ML dataset.

Attribute	Train	Dev	Test
Positive	270K	2000	500
Negative	180K	2000	500

Figure 2-10: Statistics of the Yelp review dataset for style transfer.

the food 's ok , the service is **among** the **worst** i have encountered .
the food 's ok , the service is *probably* the *best* i have encountered .
the food is good, and the service is one of the best i've ever encountered.

everyone that i spoke with was **very helpful** and **kind** .
everyone that i spoke with was *rude* and *unprofessional* .
everyone that i spoke with wasn't helpful or kind.

the beans were in the burro in the rice was **nowhere to be** found .
the beans were in the burro in the rice was *the best i* found .
the beans were in the burro and the rice was plentiful

everything is **fresh** and so **delicious** !
everything is *horrible* and so *expensive* !
everything was so stale

there is **definitely not** enough **room** in that part of the venue .
there is *always* enough *parking* in that part of the venue .
there is so much room in that part of the venue

it is n't **terrible** , but it is **n't** very good either .
it is n't *fancy* , but it is *still* very good either .
it is n't perfect , but it is very good .

executive chefs would **walk** by **not** even saying good morning .
executive chefs would *come* by *without* even saying good morning .
the excecutive chef was nice and said good morning to us very often

Figure 2-11: Example generations by BLM for sentiment transfer on Yelp. The first line is the source sentence with masked words in bold. The second line is BLM's completion. The third line is a human reference.

____
____ also ____
the ____ also ____
the ____ also ____ choice ____
the salsa ____also ____ choice ____
the salsa was also ____ choice ____
the salsa was also ____ only choice ____
the salsa was also ____ only choice .
the salsa was also my only choice .

____
____ , ____
____ , ____ terrible ____
____ poor ____ , ____ terrible ____
____ poor ____ , ____ terrible , ____
____ poor ____ , ____ terrible , very ____
____ poor selection , ____ terrible , very ____
very poor selection , ____ terrible , very ____
very poor selection , service terrible , very ____
very poor selection , service terrible , very ____ !
very poor selection , service terrible , very slow !

____
____ favorite ____
my favorite ____
my favorite ____ pittsburgh ____
my favorite ____ pittsburgh .
my favorite restaurant ____ pittsburgh .
my favorite restaurant in pittsburgh .

____
____ the ____
____ is ____ the ____
____ is ____ the ____ .
____ is ____ the ____ are ____ .
____ food is ____ the ____ are ____ .
____ food is ____ the ____ are ____ friendly .
____ food is ____ and the ____ are ____ friendly .
____ food is delicious and the ____ are ____ friendly .
____ food is delicious and the ____ are very friendly .
____ food is delicious and the owners are very friendly .
the food is delicious and the owners are very friendly .

Figure 2-12: Examples of BLM generation trajectory on the Yelp review dataset.

Chapter 3

Conformal Language Modeling

In this chapter, we propose a novel approach to conformal prediction for generative language models (LMs). Standard conformal prediction produces prediction sets—in place of single predictions—that have rigorous, statistical performance guarantees. LM responses are typically sampled from the model's predicted distribution over the large, combinatorial output space of natural language. Translating this process to conformal prediction, we calibrate a *stopping rule* for sampling different outputs from the LM that get added to a growing set of candidates, until we are confident that the output set is sufficient. Since some samples may be low-quality, we also simultaneously calibrate and apply a *rejection rule* for removing candidates from the output set to reduce noise. Similar to conformal prediction, we prove that the sampled set returned by our procedure contains at least one acceptable answer with high probability, while still being empirically precise (i.e., small) on average. Furthermore, within this set of candidate responses, we show that we can also accurately identify subsets of individual components, such as phrases or sentences, that are each independently correct (e.g., that are not "hallucinations"), again with statistical guarantees. We demonstrate the effectiveness of our approach on multiple types of language models applied to tasks in open-domain question answering, text summarization, and radiology report generation.

3.1 Introduction

Language models (LMs) have emerged as powerful tools for solving natural language processing (NLP) tasks. Given an input prompt, LMs generate a response from some predicted distribution over output text sequences. For modern models, these generations are often coherent and contextually relevant. At the same time, these models still make mistakes, and lack certain aspects of robustness and reliability in terms of providing accurate, trustworthy predictions [72, 81, 91, 100, 139, 157]. Unfortunately, however, quantifying the uncertainty in LM outputs has remained a major challenge.

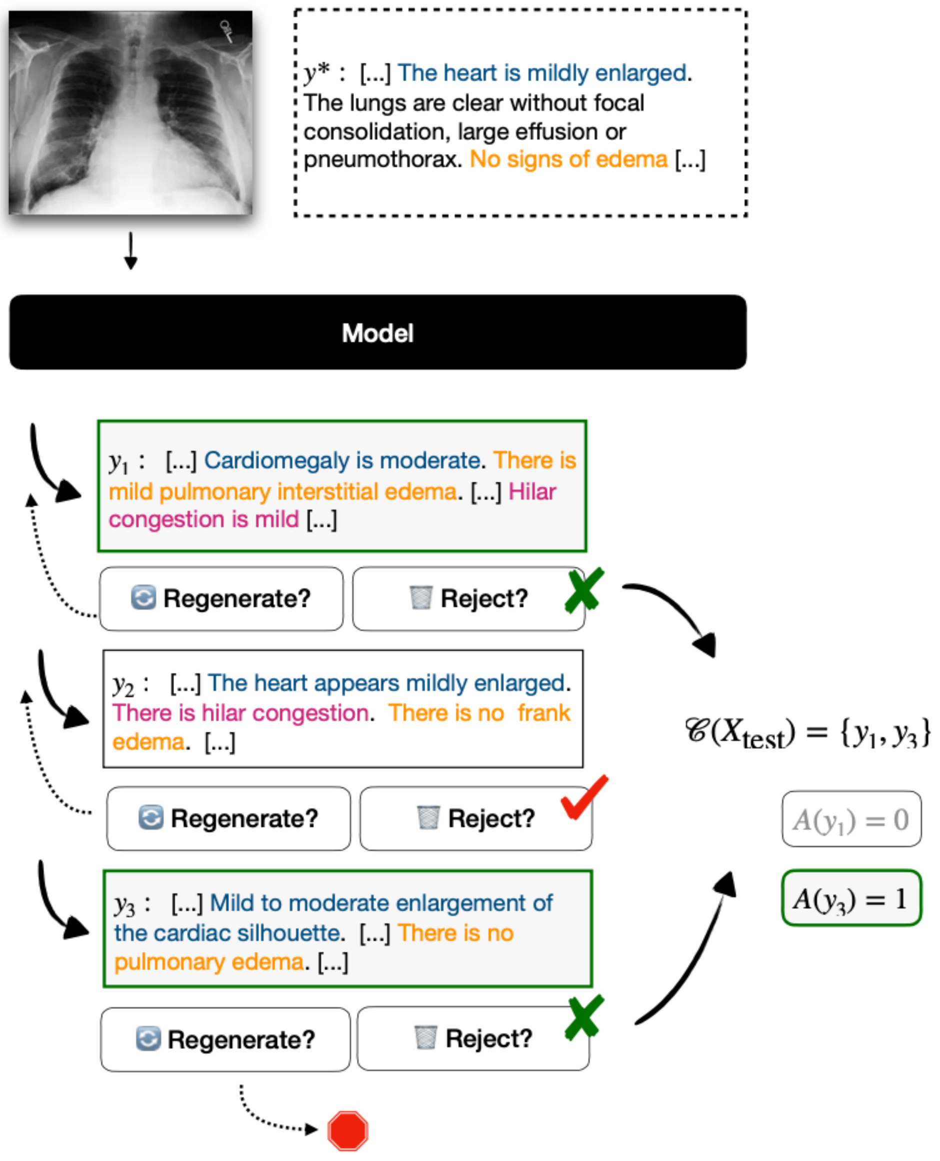
Conformal prediction is a popular model-agnostic and distribution-free method for creating prediction sets that contain the correct answers with high probability [4, 5, 6, 13, 86, 126, 152]. Applying conformal prediction to generative models such as LMs, however, is challenging due to (a) the unbounded nature of their output space (i.e., all possible text sequences), and (b) the limited (tractable) mechanisms for exploring all possible predictions. In particular, LMs can typically only approximately search or sample candidate responses. Furthermore, while several possible responses might be acceptable (e.g., correct or factual), small differences can result in abrupt changes in coherence or meaning.

In this chapter, we propose an extension of conformal prediction that is tailored specifically to generative LMs. We only assume that the (potentially black-box) LM that is given to us can be used to sample diverse output sequences, together with their evaluated model likelihoods (i.e., the output token sequence logits). Like conformal prediction, our method offers a rigorous coverage guarantee by constructing prediction sets that, in our case, provably contain at least one acceptable response with high probability. Unlike conformal prediction, however, we do not enumerate the entire output space (which is impossible). Instead, we derive a calibrated *stopping rule* for sampling different outputs from the LM that get added to a growing output set of candidates, until we are confident that the output set is sufficient. Since not all samples from the LM may be high quality (e.g., some may be redundant, incoherent, or have lower confidence), we also simultaneously calibrate a *rejection rule* for removing candidates from the output set—while still ensuring that our coverage bound is not violated. This gives the benefit of making our output sets not only accurate, but also precise (i.e., small).

To more concretely describe the type of guarantee that we provide, suppose we have been given a calibration set $\mathcal{D}_{\text{cal}} = (X_i, A_i) \in \mathcal{X} \times \mathcal{A}$, $i = 1, \dots, n$ of independent and identically distributed (i.i.d.) prompts and "admission" functions (see also [47]). Here, A_i is a binary random function that measures whether or not a generation y for prompt X_i is good enough (i.e., $A_i(y) = 1$). Note that randomness in A_i can come from implicit random covariates—such as relying on a random annotated reference, Y_i^{ref}, to compare the candidate y to. For example, Figure 3-1 illustrates a setting where X_i is an X-ray to automatically analyze and produce a report for, while A_i extracts individual findings from each generated report, and checks if they correspond to those given by an expert radiologist. Let X_{test} be a new i.i.d. test prompt. Using $\mathcal{D}_{\text{cal}}$ to guide our choice of hyper-parameters $\lambda \in \Lambda$, for any $\epsilon, \delta \in (0, 1)$, our goal is to generate a set of samples $\mathcal{C}_\lambda(X_{\text{test}}) \subseteq 2^{\mathcal{Y}}$ that satisfies

$$\mathbb{P}\Big(\mathbb{P}\Big(\exists y \in \mathcal{C}_\lambda(X_{\text{test}}) \colon A_{\text{test}}(y) = 1 \mid \mathcal{D}_{\text{cal}}\Big) \geq 1 - \epsilon\Big) \geq 1 - \delta. \tag{3.1}$$

The outer and inner probabilities are over the draws of $\mathcal{D}_{\text{cal}}$ and $(X_{\text{test}}, A_{\text{test}})$, respectively. Intuitively, ϵ is our error tolerance, while δ controls for the sensitivity of our algorithm with respect to calibration data. While Eq. equation 3.1 stipulates the existence of at least one "acceptable" generation in $\mathcal{C}_\lambda(X_{\text{test}})$, it does not tell us much about the individual responses, $y \in \mathcal{C}_\lambda(X_{\text{test}})$. Furthermore, longer generations are often composed of multiple statements. In our radiology example, a report may contain multiple findings, such as *"Cardiomegaly is moderate. There is mild pulmonary interstitial edema."* Extending techniques from multi-label conformal prediction [23, 49], we identify a subset of confident components that would independently be categorized as being correct (given another admission function A^c_{test}, this time operating over generation fragments). For example, we might predict that *"Cardiomegaly is*



y* : [...] The heart is mildly enlarged. The lungs are clear without focal consolidation, large effusion or pneumothorax. No signs of edema [...]
Model
y1 : [...] Cardiomegaly is moderate. There is mild pulmonary interstitial edema. [...] Hilar congestion is mild [...]
Regenerate?
Reject?
y2 : [...] The heart appears mildly enlarged. There is hilar congestion. There is no frank edema. [...]
Regenerate?
Reject?
y3 : [...] Mild to moderate enlargement of the cardiac silhouette. [...] There is no pulmonary edema. [...]
Regenerate?
Reject?
$\mathscr{C}(X_{\text{test}}) = \{y_1, y_3\}$
$A(y_1) = 0$
$A(y_3) = 1$

Figure 3-1: Our procedure samples candidate reports from a trained language model until a *stopping rule* is reached. Each sample is only added to the output conformal set if it meets both a minimum estimated quality and a diversity criterion. The procedure is calibrated such that at least one candidate y from the conformal set (green frame) is admissible ($A(y) = 1$). In this example, samples y_1 and y_2 are in-admissible because they hallucinate the presence of "edema" (in orange) and "hilar congestion" (in magenta), respectively. Sample y_3, however, is admissible, and included in the output set by our method.

moderate." is correct, but perhaps not *"There is mild pulmonary interstitial edema."* This can not only be useful in catching incorrect statements, but can also help identify independently correct parts of a larger generation, even when the overall quality of the full generation is poor. Like Eq. equation 3.1, we calibrate this process such that it gives accurate results with high probability.

Contributions. In summary, our main results are as follows:

- We bridge the gap between conformal prediction and LMs by calibrating the *sampling* of output sets, rather than enumerating and selecting candidate responses directly from the output space;

- We extend multi-label conformal prediction to identify confident components of long generations;

- We demonstrate valid risk control on multiple diverse tasks with different formats of LMs, while still retaining meaningful output sets that are precise on average, as compared to baselines.

3.2 Related work

Conformal prediction and risk control. Our work adds to the rich collection of tools for uncertainty estimation and risk control for machine learning algorithms [4, 5, 12, 13, 49, 58, 87, 86, 151, 153, 154, *inter alia*]. These techniques were previously extended and applied in the language domain to classification with finitely-many classes [47, 48, 72], to token-level predictions [36, 125], and to reliably accelerate LMs [85, 131, 130]. Here, we address the emerging challenge of providing reliable prediction sets in unbounded, free-text generation—which previous methods are unequipped for. The distribution-free, finite-sample performance guarantees that we derive are similar to those given by prediction sets or regression intervals in standard conformal prediction [4, 116, 152], but with slightly relaxed "correctness" criterions [22, 47]. In particular, we build on the groundwork set by [5], which provides a general methodology for calibrating any risk function that is controllable via some low-dimensional hyper-parameter configuration. We extend their framework to handle sampling-based algorithms that can effectively be used for LMs, and that, critically, do not require enumerating the full output space (which is intractable in our case). Most relevant to our work in LMs, other recent approaches have built on conformal principles to construct confidence intervals for generative diffusion models over images [65, 144]. These methods do not directly translate to LMs, however, as they only provide non-combinatorial confidence intervals on the pixel-level.

Uncertainty estimation in LMs. As the use of LMs in-the-wild grows fast, there is growing interest in obtaining and expressing meaningful confidence estimates for each output. Recent studies show that the logits of out-of-the-box LMs tend to exhibit overconfidence, even when wrong [34, 77, 105, 149]. Recent alignment techniques degrade this even further [77, 113]. Most current mitigation approaches focus on introducing linguistic cues [92, 186] or empirical post-hoc logit calibration [70, 77, 106, 179]. However, such heuristics don't provide any concrete guarantees. In this work, we develop similar techniques to improve the output of the underlying LM. Our methods are model agnostic and provide rigorous guarantees. Our conformal component selection (§3.4.4) also relates to recent self-consistency work that builds on the empirical observation that repeated similar samples are more likely to be correct [109, 157], and cross-sample entailment can approximate uncertainty [83]. Unlike previous work that uses a fixed number of re-samples and compares full outputs, we (1) introduce a dynamic stopping rule to reduce the number of samples, (2) extend this concept to semantically compare sub-components of long text outputs, and (3) conformalize the process to provide proper guarantees.

Reliable generation. It is common practice to post-hoc apply classifiers and filters on top of LM generations for various quality goals such as preventing toxicity [50, 123, 160], verifying grounding against sources [17, 97, 178], or re-ranking the set of decoded outputs [69]. Our work provides a systematic and reliable approach for filtering or flagging poor-quality outputs—both at a full generation and component level—and can also readily incorporate additional signal from auxiliary classifiers. For example, we demonstrate in our experiments using off-the-shelf natural language inference (NLI) models [19, 78, 129, 145, 161, 182] to help guide the selection of individual, confident components in text summarization (i.e., sentences that are fully entailed by the larger text [45, 64, 84, 128]).

3.3 Background

We begin with a brief review of conformal prediction and general risk control (see also [3]). Here, and in the rest of the chapter, upper-case letters (X) denote random variables; lower-case letters (x) denote constants, and script letters ($\mathcal{X}$) denote sets, unless specified.

Given a new example x, for every candidate label $y \in \mathcal{Y}$ standard conformal prediction either accepts or rejects the null hypothesis that the pairing (x, y) is correct. The test statistic for this test is a *nonconformity measure*, $\mathcal{M}((x, y), \mathcal{D})$, where $\mathcal{D}$ is a dataset of labeled examples. Informally, a lower value of $\mathcal{M}$ reflects that point (x, y) "conforms" to $\mathcal{D}$, whereas a higher value of $\mathcal{M}$ reflects that (x, y) does not. For example, a practical choice for $\mathcal{M}$ could be the model-based negative log likelihood, $-\log p_\theta(y|x)$, where θ are parameters fit to $\mathcal{D}$. Split conformal prediction [115] uses a separate training set $\mathcal{D}_{\text{train}}$ to learn a fixed $\mathcal{M}$ that is not modified during calibration or prediction. To construct a prediction set for the new test point x, the conformal classifier outputs all y for which the null hypothesis (that pairing (x, y) is correct) is not rejected. This is achieved by comparing the scores of the test candidate pairs to the scores computed over n calibration examples.

Theorem 3.3.1 (Split conformal prediction [115, 152])**.** *Let (X_i, Y_i), $i = 1, \ldots, n+1$ be i.i.d random variables.*[1] *Let random variable $V_i = \mathcal{M}(X_i, Y_i)$ be the nonconformity score of (X_i, Y_i), where $\mathcal{M}$ is fixed. For $\epsilon \in (0, 1)$, define the prediction (based on the first n examples) at $x \in \mathcal{X}$ as*

$$\mathcal{C}_\epsilon(x) := \big\{y \in \mathcal{Y} \colon \mathcal{M}(x, y) \leq \text{Quantile}(1 - \epsilon;\, V_{1:n} \cup \{\infty\})\big\} \tag{3.2}$$

Then $\mathbb{P}(Y_{n+1} \in \mathcal{C}_\epsilon(X_{n+1})) \geq 1 - \epsilon$.

Note that the coverage property expressed in Theorem 3.3.1 is *marginal* over the draw of calibration and test data. The recent Learn Then Test (LTT) framework of [5] extends conformal prediction to control the expectation of any loss function

[1] Technically, standard conformal prediction only requires exchangeablity—a weaker requirement than i.i.d.

(conditional on the draw of calibration data) by reframing hyper-parameter selection as a statistical multiple hypothesis testing problem.

Specifically, let $L : \Lambda \to \mathbb{R}$ be any random function using a hyper-parameter configuration λ in some space Λ. For example, we might have $L(\lambda) := \ell(X, Y; \lambda)$ for some fixed loss function ℓ with random inputs (X, Y). Unlike conformal prediction, however, λ can be multi-dimensional (e.g., consist of multiple thresholds). Let L_i, $i = 1, \ldots, n$ be an i.i.d. calibration set $\mathcal{D}_{\text{cal}}$ of random functions, and $\epsilon \in \mathbb{R}$ be a tolerance for the test risk, $\mathbb{E}[L_{\text{test}}(\lambda)] \leq \epsilon$. LTT then identifies a random (depending on $\mathcal{D}_{\text{cal}}$) subset of parameters, $\Lambda_{\text{valid}} \subseteq \Lambda$, with the goal of guaranteeing:

$$\mathbb{P}\left(\sup_{\lambda \in \Lambda_{\text{valid}}} \mathbb{E}[L_{\text{test}}(\lambda) \mid \mathcal{D}_{\text{cal}}] \leq \epsilon \right) \geq 1 - \delta, \tag{3.3}$$

where the outer probability is over the draw of $\mathcal{D}_{\text{cal}}$, and the inner expectation is over draws of L_{test}. This then implies that any $\lambda \in \Lambda_{\text{valid}}$ can be selected to control the risk of L_{test}. In short, this is achieved by associating the null hypothesis $\mathcal{H}_\lambda \colon \mathbb{E}[L_{\text{test}}(\lambda)] > \epsilon$ to each $\lambda \in \Lambda$. For each null hypothesis, we then use the calibration set to compute a super-uniform p-value p_λ using concentration inequalities. Any multiple testing algorithm $\mathcal{T}(p_\lambda \colon \lambda \in \Lambda)$ that controls the family-wise error rate (FWER) can then be used to identify the subset of non-rejected λ, i.e., Λ_{valid}.[2] Note that it is possible for $\Lambda_{\text{valid}} = \varnothing$, in the case that we fail to identify any statistically valid solutions (and the desired risk may not even be achievable with any λ). In this situatoin, we set $\lambda = \texttt{null}$, and either reject the task, or provide a trivial solution (e.g., a classifier that provides all possible labels $\mathcal{Y}$).

Theorem 3.3.2 (Learn Then Test [5])**.** *Suppose p-value p_λ, derived from $\mathcal{D}_{\text{cal}}$, is super-uniform under $\mathcal{H}_\lambda$ for all λ. Let $\mathcal{T}$ be any FWER-controlling algorithm at level δ. Then Λ_{valid} satisfies Eq. equation 3.3.*

Defining $\mathcal{C}_\lambda(x) := \{y \in \mathcal{Y} \colon \mathcal{M}(x, y) \leq \lambda\}$, $\Lambda \subset \mathbb{R}$, and $L(\lambda) := \mathbf{1}\{Y \notin \mathcal{C}_\lambda(X)\}$ recovers a criterion similar to that of conformal prediction (though not marginal over

[2] A FWER-controlling algorithm at level δ is any procedure that accepts or rejects null hypotheses $\mathcal{H}_\lambda$, while ensuring that the probability of falsely rejecting any $\mathcal{H}_\lambda$, $\forall \lambda \in \Lambda$, is less than δ.

$\mathcal{D}_{\text{cal}}$). Unfortunately, in either instantiation (LTT vs. conformal prediction) iterating over $y \in \mathcal{Y}$ is intractable for LMs, regardless of whatever calibration technique is ultimately used. Instead, in §3.4, we introduce our method for generating uncertainty sets by casting λ as a configuration of a *sampling* algorithm, rather than a filter on the output space $\mathcal{Y}$. We then show that this randomized algorithm can still be calibrated with LTT.

3.4 Conformal language modeling

We now introduce our method for generating uncertainty sets for LMs. At a high level, our procedure consists of three main steps to sample and return an collection of plausible output predictions:

1. **Sample.** A new candidate response y is sampled from our language model.
2. **Accept or reject.** The sample y is added to the growing output set, as long as it is diverse (e.g., maximum overlap with any other element is $\leq \lambda_1$) and confident (e.g., the LM likelihood is $\geq \lambda_2$).
3. **Stop or repeat.** Using a set-based scoring function, we check if the confidence in the current set is $\geq \lambda_3$. If it is, then we stop and return the current set. Otherwise we return to Step 1.

$\lambda = (\lambda_1, \lambda_2, \lambda_3)$ is a configuration that we calibrate to find a valid setting, $\hat{\lambda} = (\hat{\lambda}_1, \hat{\lambda}_2, \hat{\lambda}_3)$, that controls the risk of our output sets. In the following, we more carefully define our setting and notation (§3.4.1), and then describe our sampling (§3.4.2) and calibration algorithms (§3.4.3). Then, in §3.4.4, we provide an additional extension for highlighting confident generation components—i.e., subsections of our full generations that are independently likely to be correct, even if the full generation is not.

3.4.1 Formal setting and notation

Let $\mathcal{V}$ be an alphabet (a non-empty, finite set of tokens such as {"a","b","c",...}) from which all possible output strings, y, are composed, i.e. $\mathcal{Y} := \mathcal{V}^*$.[3] We assume that we are given a generative model $p_\theta(y \mid x)$ that defines a conditional probability distribution given some input prompt $x \in \mathcal{X}$ (where x may be text, or another modality such as an image), which we can sample from to obtain candidate output strings, $y \sim p_\theta(y \mid x)$. Following [47], for every input prompt x, we assume access to some "admission" function $A \colon \mathcal{V}^* \to \{0, 1\}$ that is used to measure the acceptability of a given sample y. Intuitively, A tells us if an output is "good enough".

As an example, in our radiology report setting from §3.1, x is the input X-ray, y is the generated report, and $p_\theta(y \mid x)$ is our image-to-text LM (in English). Given y and some "ground truth" report y^* (e.g., written by a radiologist), $A(y)$ might measure if y and y^* agree on all findings. Note that, in practice, it may be hard to exactly define such an A, or at least an A that is automatically computable without manual annotation. In Appendix B.2 we show that it is also sufficient to only require access to a *conservative* admission function, $\bar{A} \colon \mathcal{V}^* \to \{0, 1\}$, where $\forall y \in \mathcal{V}^*$ we have $\bar{A}(y) \leq A(y)$. For instance, $\bar{A}$ might measure exact match on a word-for-word basis between y and y^*, instead of accounting for differences in dictation. We explore different tasks and admission functions in our experiments in §3.5.

As introduced in §3.1, given a calibration set $\mathcal{D}_{\text{cal}}$, our goal is to derive a configurable algorithm with input parameters $\lambda \in \Lambda$ for constructing a prediction $\mathcal{C}_\lambda$ that we can calibrate to satisfy Eq. equation 3.1. In the framework of LTT (refer to §3.3), this is equivalent to defining

$$L_i(\lambda) = \mathbf{1}\big\{\nexists y \in \mathcal{C}_\lambda(X_i) \colon A_i(y) = 1\big\}, \tag{3.4}$$

and using the calibration set $\mathcal{D}_{\text{cal}}$ to find a value $\hat{\lambda}$ such that $\mathbb{E}[L_{\text{test}}(\hat{\lambda})] \leq \epsilon$ with probability $\geq 1 - \delta$.

[3]We write $\mathcal{V}^*$ to denote the Kleene closure of a set $\mathcal{V}$, i.e., $\mathcal{V}^* := \bigcup_{n=0}^{\infty} \mathcal{V}^n$.

Algorithm 2 Conformal sampling with rejection

Definitions: x is an input prompt, $\mathcal{F}$ is our set-based confidence function, $\mathcal{S}$ is our text similarity function, $\mathcal{Q}$ is our sample quality estimator, λ is our threshold configuration, and $k_{\max}$ is our sampling budget. $p_\theta(y \mid x)$ is the conditional output distribution defined by our language model.

```
1: function SAMPLE($x$, $\mathcal{F}$, $\mathcal{S}$, $\mathcal{Q}$, $\lambda$, $k_{\max}$)
2:     $\mathcal{C}_\lambda \leftarrow \{\}$                                   ▷ Initialize an empty output set.
3:     for $k = 1, 2, \ldots, k_{\max}$ do
4:         $y_k \leftarrow y \sim p_\theta(y \mid x)$.                        ▷ Sample a new response.
5:         if $\mathcal{Q}(x, y_k) < \lambda_2$ then                ▷ Reject if its estimated quality is too low.
6:             continue
7:         if $\max\{\mathcal{S}(y_k, y_j) \colon y_j \in \mathcal{C}_\lambda\} > \lambda_1$ then ▷ Reject if it is too similar to other samples.
8:             continue
9:         $\mathcal{C}_\lambda = \mathcal{C}_\lambda \cup \{y_k\}$.                ▷ Add the new response to the output set.
10:        if $\mathcal{F}(\mathcal{C}_\lambda) \geq \lambda_3$ then               ▷ Check if we are confident enough to stop.
11:            break
12:    return $\mathcal{C}_\lambda$
```

3.4.2 Conformal sampling with rejection

Let $\mathcal{F} \colon 2^{\mathcal{V}^*} \to \mathbb{R}$ be a set-based function that, for any set $\mathcal{C} \in 2^{\mathcal{V}^*}$, gives a confidence score for the event $\mathbf{1}\{\exists y \in \mathcal{C} \colon A(y) = 1\}$. Practically, we expect that $\mathcal{F}$ should be non-decreasing, i.e., $\mathcal{C} \subset \mathcal{C}' \implies \mathcal{F}(\mathcal{C}) \leq \mathcal{F}(\mathcal{C}')$, though it is not strictly enforced. Furthermore, let $\mathcal{S} \colon \mathcal{V}^* \times \mathcal{V}^* \to \mathbb{R}$ be a text-based similarity function (e.g., such as `BLEU` or `ROUGE`) that we use to detect duplicates in $\mathcal{C}$, and $\mathcal{Q} \colon \mathcal{X} \times \mathcal{V}^* \to \mathbb{R}$ an input-conditional text-based measure of individual prediction quality—such as the LM's likelihood function, $p_\theta(y \mid x)$. We then adopt a sampling-based procedure that *grows* an output set, $\mathcal{C}_1 \subseteq \mathcal{C}_2 \subseteq \ldots \subseteq \mathcal{C}_{k-1}$, by repeatedly taking samples $y_k \sim p_\theta(y \mid x)$, and updating

$$\mathcal{C}_k := \begin{cases} \mathcal{C}_{k-1} \cup \{y_k\} & \text{if } \max\{\mathcal{S}(y_k, y_j) \colon y_j \in \mathcal{C}_{k-1}\} \leq \lambda_1 \\ & \text{and } \mathcal{Q}(x, y_k) \geq \lambda_2, \\ \mathcal{C}_{k-1} & \text{otherwise.} \end{cases} \tag{3.5}$$

until the confidence after k samples, $\mathcal{F}(\mathcal{C}_k)$, is $\geq \lambda_3$ (or some sampling budget $k_{\max}$ is reached).

As an intuitive, but toy, example, suppose we modeled $y_k \sim p_\theta(y \mid x), k = 1, 2, \ldots$

as a Bernoulli process, where each y_k has the same probability of success p that we assume (albeit unrealistically) that we know. For X_{test}, "success" is determined by the admission function, A_{test}. The confidence that our current set $\mathcal{C}_k$ contains at least one admissible answer (without rejection) then follows a geometric distribution, Geo(p): all that remains is to compute the minimum number of samples to take such that Eq. equation 3.1 is satisfied. This is achieved by taking $\mathcal{F}(\mathcal{C}_k) = k$ and $\lambda_3 = \lceil \log(\epsilon)/\log(1-p) \rceil$.

Of course, in reality we do not know the probability of success p for test examples. Furthermore, the samples y_k are not independent, and since we are also able to observe their values, better strategies may exist to conditionally estimate $A(y_k) = 1$. Therefore, we allow $\mathcal{F}$ to be *any* set-based function—that we also pair with similarity function $\mathcal{S}$, and sample quality function $\mathcal{Q}$, for handling rejections. Pseudocode is given in Algorithm 2. We derive, calibrate, and test different variations of $\mathcal{F}$, $\mathcal{S}$, and $\mathcal{Q}$ in §3.5 and §3.6, respectively. Using $\lambda = (\lambda_1, \lambda_2, \lambda_3)$, we write $\mathcal{C}_\lambda(X_{\text{test}})$ to denote the final output set.

3.4.3 Calibration with Learn Then Test

Let Λ be a finite set of configurations. For example, if searching for a value of $\lambda = (\lambda_1, \lambda_2, \lambda_3) \in [0,1]^3$, we might consider the evenly-spaced set $\Lambda = \{\frac{i}{\kappa} \colon i = 1, \ldots, \kappa\}^3$ for some finite $\kappa \in \mathbb{N}$. For each $\lambda \in \Lambda$, LTT then requires computing a valid p-value p_λ, where p_λ is a super-uniform random variable under $\mathcal{H}_\lambda$. Here, we can obtain valid p-values from the empirical risk on $\mathcal{D}_{\text{cal}}$,

$$\widehat{R}_n(\lambda) := \frac{1}{n} \sum_{i=1}^{n} L_i(\lambda), \quad \text{where} \quad L_i(\lambda) = \mathbf{1}\big\{\nexists y \in \mathcal{C}_\lambda(X_i) \colon A_i(y) = 1\big\}, \tag{3.6}$$

Lemma 1 (Binomial tail bound p-values)**.** Let $\widehat{R}_n(\lambda)$ be the empirical risk in Eq. equation 3.6, and let Binom(n, ϵ) denote a binomial random variable with sample size n

and success probability ϵ. Then

$$p_\lambda^{\mathrm{BT}} := \mathbb{P}(\mathrm{Binom}(n, \epsilon) \leq n\widehat{R}_n(\lambda)) \tag{3.7}$$

is a valid p-value for $\mathcal{H}_\lambda \colon \mathbb{E}[L_{\text{test}}(\lambda)] > \epsilon$.

Proof. Let $X = \mathrm{Binom}(n, \epsilon)$ and $Y = n\hat{R}_n(\lambda)$. Under $\mathcal{H}_\lambda$, $n\hat{R}_n(\lambda)$ stochastically dominates $\mathrm{Binom}(n, \epsilon)$, i.e., $F_X(u) \geq F_Y(u)\ \forall u$. Let $Z = p_\lambda^{\mathrm{BT}} = F_X(Y)$. Then

$$\begin{aligned}
\mathbb{P}(Z \leq z) &= \mathbb{P}(F_X(Y) \leq z) && (3.8)\\
&\leq \mathbb{P}(F_Y(Y) \leq z) && (3.9)\\
&= \mathbb{P}(Y \leq F_Y^{-1}(z)) && (3.10)\\
&= F_Y(F_Y^{-1}(z)) && (3.11)\\
&= z. && (3.12)
\end{aligned}$$

Therefore, p_λ^{BT} is super-uniform, and a valid p-value. □

When paired with any FWER-controlling algorithm $\mathcal{T}$ at level δ, we obtain the set $\Lambda_{\text{valid}} \subseteq \Lambda$ by selecting all configurations for hypotheses $\mathcal{H}_\lambda$ that are rejected by $\mathcal{T}(p_\lambda^{\mathrm{BT}} \colon \lambda \in \Lambda)$. We then use the configuration that empirically minimizes the average set size (also measured over $\mathcal{D}_{\text{cal}}$), i.e.,

$$\hat{\lambda} = \underset{\lambda \in \Lambda_{\text{valid}}}{\operatorname{argmin}}\ \frac{1}{n} \sum_{i=1}^{n} |\mathcal{C}_\lambda(X_i)|. \tag{3.13}$$

As mentioned in §3.3, if we fail to find any valid configurations (which may not even exist for small values of $k_{\max}$, or LMs without full support over $\mathcal{V}^*$) with the right confidence, then we abstain. As a consequence of LTT, $\hat{\lambda}$ (which is a random variable that depends on $\mathcal{D}_{\text{cal}}$) is risk-controlling.

Theorem 3.4.1 (Sampling-based LTT). *Let $\hat{\lambda}$ be defined according to Eq. equation 3.13. Then the prediction $\mathcal{C}_{\hat{\lambda}}(X_{\text{test}})$ computed by Algorithm 2 satisfies Eq. equation 3.1.*

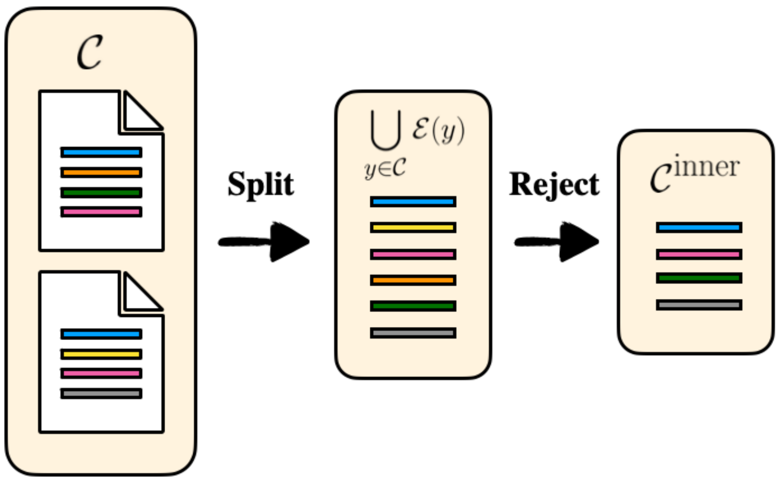

$\mathcal{C}$
Split
$\bigcup_{y \in \mathcal{C}} \mathcal{E}(y)$
Reject
$\mathcal{C}^{\text{inner}}$

Figure 3-2: Our conformal selection procedure requires splitting each candidate generations in components, then filtering them while controlling for desired risk with LTT

Finally, to efficiently search and test the higher dimensional $\lambda = (\lambda_1, \lambda_2, \lambda_3)$ that we consider here, we use the recently proposed (FWER-controlling) Pareto Testing procedure from [85]. Pareto Testing can be used to exploit structure in Λ by first using a proportion of $\mathcal{D}_{\text{cal}}$ to find Λ's Pareto-optimal frontier, and then iteratively validate the most empirically promising configurations using Fixed Sequence Testing [62] on the remaining calibration data. See Appendix B.1 for details.

3.4.4 Conformal selection of individual components

A caveat of language generation tasks is that it is often the case that LM responses can be verbose, and composed of multiple components. We consider a component to be some logically defined subpart of a larger response, such as a series of phrases, sentences, or propositions. For example, in our radiology setting, a report such as *"The heart is mildly enlarged. The lungs are clear."* can be broken down into two findings, namely, *"The heart is mildly enlarged."* and *"The lungs are clear."* While our past guarantees may tell us that admissible generations exist within our sampled prediction sets, they still do not give us any insight into how confident our model is about individual statements within each option.

To address this, we propose the conformal selection of individual components. We illustrate our approach in Figure 3-2. let $\mathcal{E}\colon \mathcal{V}^* \to 2^{\mathcal{V}^*}$ be a deterministic function that takes a text string as input, and breaks it down into components. Here, we implement $\mathcal{E}$ to be a simple sentence splitter. Similar to before, for every input x, we assume access to some *component-based* admission function $A^{\mathrm{c}}\colon 2^{\mathcal{Y}} \to \{0,1\}$ that is used to judge response fragments for correctness. For example, A can check if e is entailed by (or exactly matches) another component $e' \in \mathcal{E}(y^{\mathrm{ref}})$, where y^{ref} is a human reference. Let $\mathcal{F}^{\mathrm{c}}\colon \mathcal{V}^* \to \mathbb{R}$ be a function that, for any component $e \in \mathcal{V}^*$, gives a confidence score for the event $A^{\mathrm{c}}(e) = 1$. We then define the subset of components $\mathcal{C}_\gamma^{\mathrm{inner}} \subseteq 2^{\mathcal{V}^*}$ as

$$\mathcal{C}_\gamma^{\mathrm{inner}}(x) := \Big\{ e \in \bigcup_{y \in \mathcal{C}_\lambda(x)} \mathcal{E}(y) \colon \mathcal{F}^{\mathrm{c}}(e) \geq \gamma \Big\}. \tag{3.14}$$

Again using $\mathcal{D}_{\mathrm{cal}}$, we seek to calibrate $\gamma \in \Gamma$, such that for test pair $(X_{\mathrm{test}}, A_{\mathrm{test}})$ and $\alpha, \delta \in (0,1)$,

$$\mathbb{P}\Big(\mathbb{P}\Big(A^{\mathrm{c}}_{\mathrm{test}}(e) = 1, \forall e \in \mathcal{C}_\gamma^{\mathrm{inner}}(X_{\mathrm{test}})\Big) \geq 1 - \alpha \mid \mathcal{D}_{\mathrm{cal}}\Big) \geq 1 - \delta. \tag{3.15}$$

The outer and inner probabilities are over the draws of $\mathcal{D}_{\mathrm{cal}}$ and $(X_{\mathrm{test}}, A_{\mathrm{test}})$, respectively. The new parameter α can be interpreted as the maximum rate of making *any* false positive predictions in which we select a component that is not in fact acceptable. Like $\mathcal{C}_\lambda$, we calibrate $\mathcal{C}_\gamma^{\mathrm{inner}}$ using LTT. In contrast to $\mathcal{C}_\lambda$, however, we seek to make $\mathcal{C}_\gamma^{\mathrm{inner}}$ as *large* as possible. Concretely, let $L_i^{\mathrm{c}}(\gamma) = \mathbf{1}\{\exists e \in \mathcal{C}_\gamma^{\mathrm{inner}} \colon A_i^{\mathrm{c}}(e) = 0\}$ and let Γ_{valid} be the set of non-rejected configurations found by LTT (when using binomial tail p-values, p_γ^{BT}). During calibration we define $\mathcal{C}_\gamma^{\mathrm{inner}}(X_i)$ using an upper bound to $\mathcal{C}_\lambda(X_i)$, by simply taking the first $k_{\max}$ samples, $\{y_1, \ldots, y_{k_{\max}}\}$. We then use the configuration that empirically maximizes the average number of confident components (again, while reusing $\mathcal{D}_{\mathrm{cal}}$):

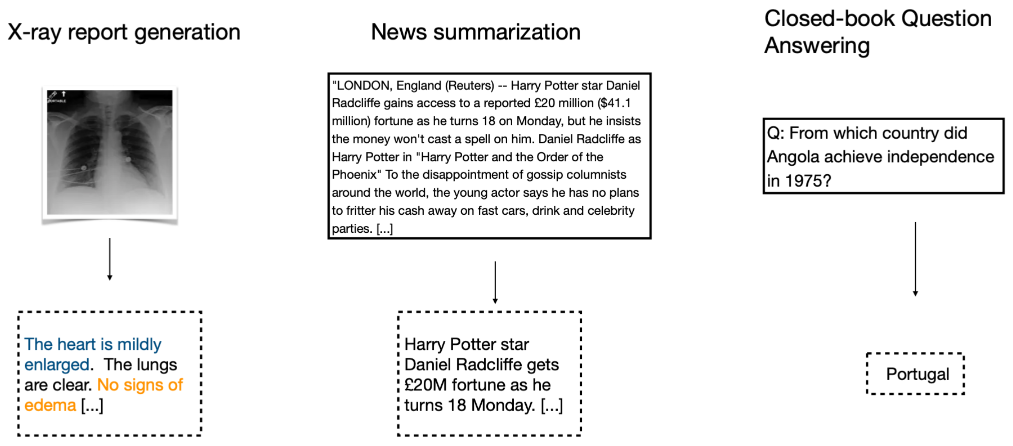
$$\hat{\gamma} = \operatorname*{argmax}_{\gamma \in \Gamma_{\mathrm{valid}}} \; \frac{1}{n} \sum_{i=1}^{n} |\mathcal{C}_\gamma(X_i)|. \tag{3.16}$$

Proposition 3.4.2 (Component-based LTT)**.** Let $\hat{\gamma}$ be defined according to Eq. equation 3.16, where $\mathcal{C}_\gamma(X_i)$ uses $\mathcal{C}_\lambda(X_i) \equiv \{y_1, \ldots, y_{\mathrm{k}_{\max}}\}$ during calibration. Then the

Algorithm 3 Conformal component selection

Definitions: $\mathcal{C}_\lambda$ is a prediction set, $\mathcal{E}$ is an algorithm for splitting candidates y into components, $\mathcal{F}^{\mathrm{c}}$ is a confidence estimator for individual components, γ is our threshold configuration.

```
1: function SELECT($\mathcal{C}_\lambda$, $\mathcal{E}$, $\mathcal{F}^{\mathrm{c}}$, $\gamma$)
2:     $\mathcal{C}_\gamma^{\mathrm{inner}} \leftarrow \{\}$                        ▷ Initialize an empty output set.
3:     for $y \in \mathcal{C}_\lambda$ do                               ▷ Iterate over full predictions.
4:         for $e \in \mathcal{E}(y)$ do                        ▷ Iterate over individual components.
5:             if $\mathcal{F}^{\mathrm{c}}(e) \geq \gamma$ then
6:                 $\mathcal{C}_\gamma^{\mathrm{inner}} \leftarrow \mathcal{C}_\gamma^{\mathrm{inner}} \cup \{e\}$    ▷ Keep only high-confidence components.
7:     return $\mathcal{C}_\gamma^{\mathrm{inner}}$
```



X-ray report generation
News summarization
Closed-book Question Answering
"LONDON, England (Reuters) -- Harry Potter star Daniel Radcliffe gains access to a reported £20 million ($41.1 million) fortune as he turns 18 on Monday, but he insists the money won't cast a spell on him. Daniel Radcliffe as Harry Potter in "Harry Potter and the Order of the Phoenix" To the disappointment of gossip columnists around the world, the young actor says he has no plans to fritter his cash away on fast cars, drink and celebrity parties. [...]
Q: From which country did Angola achieve independence in 1975?
The heart is mildly enlarged. The lungs are clear. No signs of edema [...]
Harry Potter star Daniel Radcliffe gets £20M fortune as he turns 18 Monday. [...]
Portugal

Figure 3-3: We study the applicability of our method on three generation tasks

prediction set of components, $\mathcal{C}_{\hat{\gamma}}(X_{\text{test}})$ computed by Algorithm 3 paired with any $\mathcal{C}_\lambda$ at test time with $\sup_x |\mathcal{C}_\lambda(x)| \leq k_{\max}$ satisfies Eq. equation 3.15.

Furthermore, by the union bound, Eq. equation 3.1 and Eq. equation 3.15 hold simultaneously with probability $1 - 2\delta$.

3.5 Experimental setup

We evaluate our methods on datasets for open-domain question answering (**TriviaQA** [75]), news summarization (**CNN/DM** [132]), and chest X-ray radiology report generation (**MIMIC-CXR** [71]). Figure 3-3 summarize the tasks. Appendix B.3 contains additional details.

3.5.1 Tasks

Radiology report generation. As motivated in §3.1, we apply our method to chest X-ray radiology report generation using the **MIMIC-CXR** [71] dataset. For our LM, we fine-tune an encoder-decoder architecture based on a pretrained ViT [39] image encoder and a GPT2-small [121] text decoder. To judge admission, we use the popular Clinical Efficacy metric [95, 112] to check if the 14 labels predicted by an auxiliary CheXbert [138] model on the generated report exactly match the labels predicted by the same CheXbert model for a reference report from a radiologist. Similarly, a component (here a sentence including a finding) is defined to be admissible if it has a `ROUGE-L` [90] score ≥ 0.4 (picked through empirical validation), when compared to any component directly extracted from the reference.

News summarization. We also apply our method to news article text summarization using the **CNN/DM** [132] dataset. For our LM, we finetune a T5-XL [122] model. We define a candidate generation to be admissible if it has a `ROUGE-L` score higher than 0.35 , when compared to all available reference summaries from human annotators. Like MIMIX-CXR, we define a component to be admissible if it has a `ROUGE-L` score ≥ 0.4 when compared to components extracted from human summaries.

Open-domain question answering. Finally, we apply our method to open-domain question answering using the **TriviaQA** [75] dataset. For this task, we sample answers from the LLaMA-13B [147] LM in the few-shot setting ($k = 32$), without any additional fine-tuning. Since answers are limited to one or few tokens, a candidate output generation is acceptable only if it exactly matches an annotated reference answer (after minor normalization for removing articles, casing, and punctuation). Furthermore, since the expected answers are short and fairly atomic, we do not evaluate component-level confidence.

3.5.2 Scoring functions

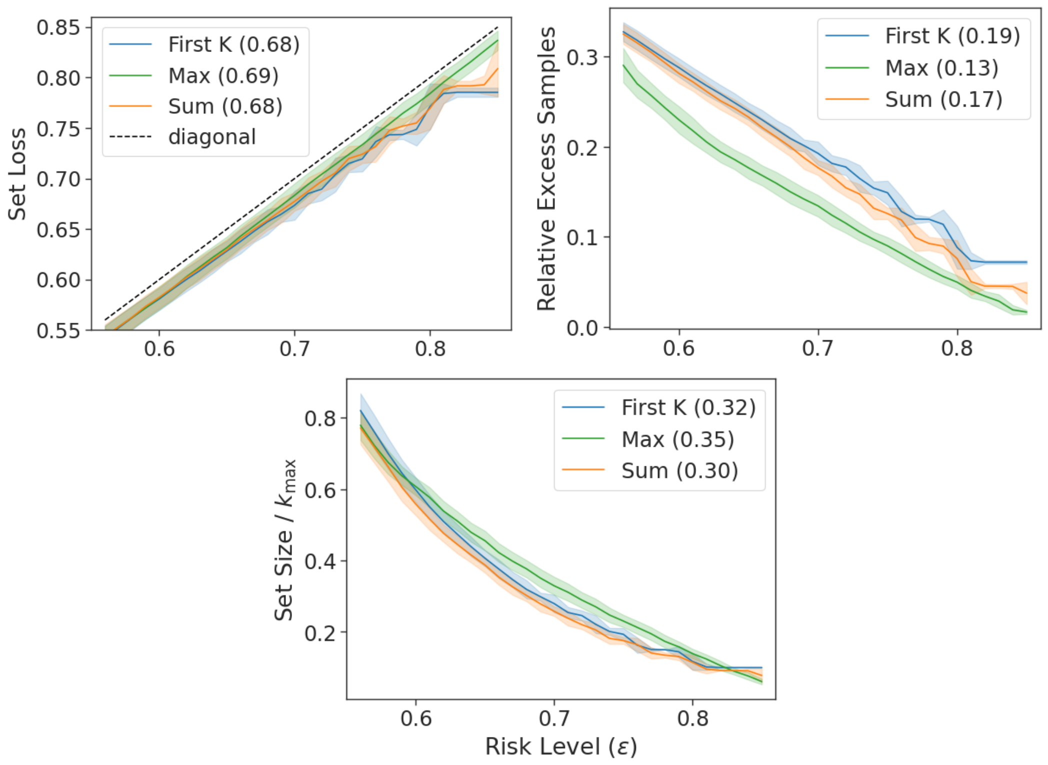
As discussed in §3.4.2, our method is implementation-agnostic and can support different choices of quality function $\mathcal{Q}$, similarity function $\mathcal{S}$, and set scoring function $\mathcal{F}$. For our purposes, we show that it can be sufficient to simply use transformations on the model likelihoods (from token logits).[4] Specifically, we define $\mathcal{Q}(x, y) = p_\theta(y \mid x)$ using the likelihood function of the base LM, with length-normalization [165]. We use `ROUGE-L` for $\mathcal{S}$. For $\mathcal{F}$, we experiment with the following variants:

- First-K. As a baseline, we score a set by its size, $\mathcal{F}_{\text{FIRST-K}}(\mathcal{C}) = |\mathcal{C}|$, and do not use rejection. This corresponds to the number of samples taken, and follows the intuition from our toy example in §3.4.2.
- Max. The $\mathcal{F}_{\text{Max}}$ scoring function stems from the intuition that a set is only as good as its best element, and defines $\mathcal{F}_{\text{Max}}(\mathcal{C}) = \max\{\mathcal{Q}(y) \colon y \in \mathcal{C}\}$.
- Sum. Alternatively, we also use the sum of item-level scores: $\mathcal{F}_{\text{Sum}}(\mathcal{C}) = \sum_{y \in C} \mathcal{Q}(y)$.

3.5.3 Metrics

Our main motivation is to produce valid confidence sets that are also precise. A desirable procedure should output smaller sets on easy examples, and dedicate a larger budget to harder examples. To reflect this, we measure both the **loss** of our sets (which is guaranteed to satisfy our prescribed limits), as well as both (a) the **relative number of "excess" samples** taken from our model (i.e., how many extra samples our algorithm takes *after* the first admissible answer has already been surfaced, proportional to the total number of samples), and (b) the ultimate **output size** of the prediction set (after rejection). Both metrics are important, as over-sampling wastes computational budget, while conservative, large output sets can be unwieldy to use, and less helpful as an uncertainty quantification tool. We measure results and compute the AUC over the range of achievable ϵ or α (using a fixed $\delta = 0.05$), excluding trivial values (e.g., that a policy that always returns the first generation would satisfy).

[4]While straightforward, note that potentially better measures can be derived through other means like [77, 157].



(a) MIMIC-CXR

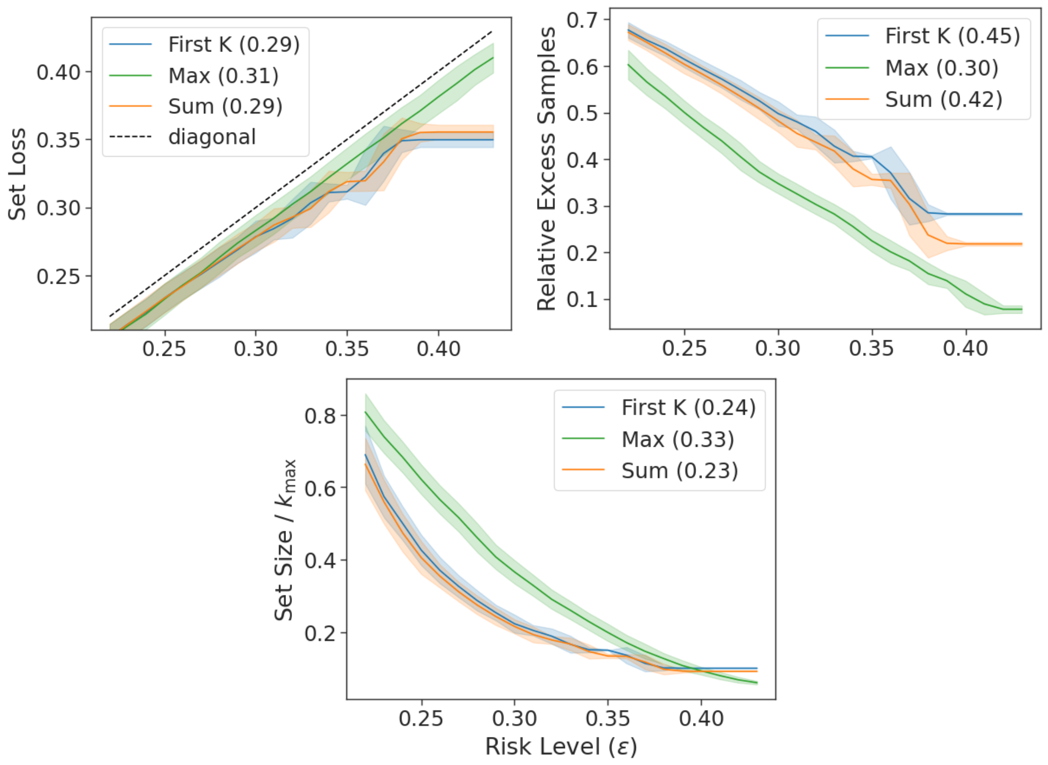
Figure 3-4: Conformal sampling results for $\mathcal{C}_\lambda$ a function of ϵ, on MIMIC-CXR. We report the loss, relative excess samples, and overall size (normalized by $k_{\max}$). We also report the AUC over achieved/non-trivial ϵ.

3.6 Experimental results

We now present our main results. In all plots, solid lines give the mean over 100 trials and shaded regions show +/− the standard deviation. Additional experimental results are reported in Appendix B.4.

Validity of conformal sampling with rejection. As per Theorem 3.4.1, we observe in Figure 3-4 and Figure 3-6 that our conformal sampling approach is valid, as the average set loss often matches but never exceeds the target risk level. Methods that have access to the model logits (namely MAX and SUM) are close to the diagonal line, indicating that they are not overly conservative.

Prediction efficiency. The likelihood-based approaches outperform the uniform FIRST-K baseline across all three tasks. For example, as Figure 3-6a shows, the AUC of expected set size of MAX and SUM are both less than half the AUC of FIRST-K

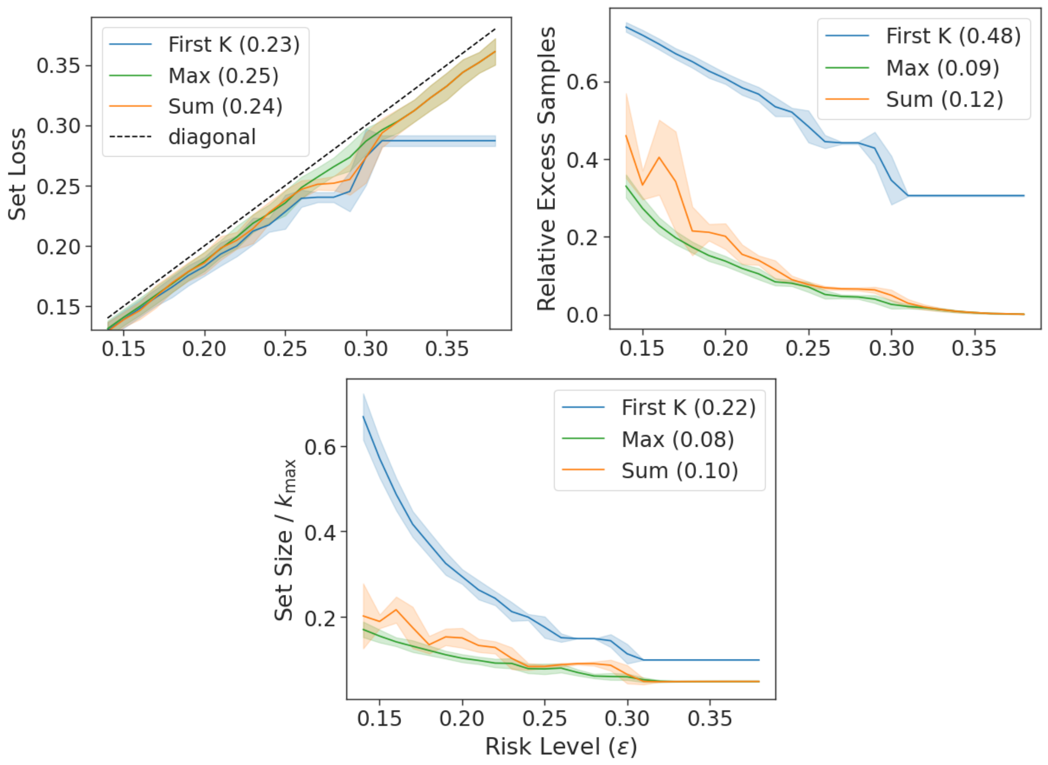


(a) CNN/DM

Figure 3-5: Conformal sampling results for $\mathcal{C}_\lambda$ a function of ϵ, on CNN/DM. We report the loss, relative excess samples, and overall size (normalized by k_{max}). We also report the AUC over achieved/non-trivial ϵ.

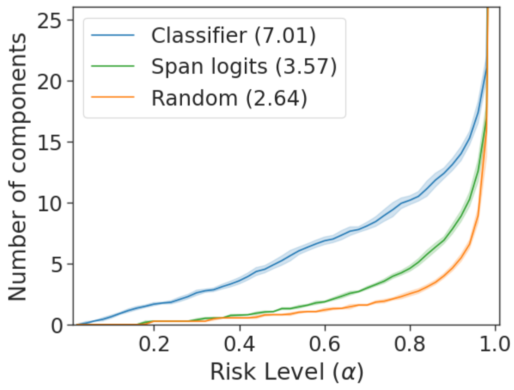
in the QA task. In tasks with longer output texts, First-K produces competitive set sizes across all achievable ϵ. However, it is being overly conservative on easy examples at the expense of hard ones. This is revealed when plotting the relative number of excess samples, where the Max scoring function largely outperforms Sum and First-K.

Individual components. We evaluate two scoring functions $\mathcal{F}^c$ to apply our conformal component selection method. A first method Span-logits extracts the likelihood of a candidate component, as produced by the language model. Since those likelihoods are conditioned on previous context, this scoring function may underestimate the score of a correct component if it follows an incorrect component. Instead, we use an application-specific Classifier to assign a conformity score to each component. We compare these methods to a Random baseline which attributes a random score to any (x, e) pair. Figure 3-7 shows that by modeling components

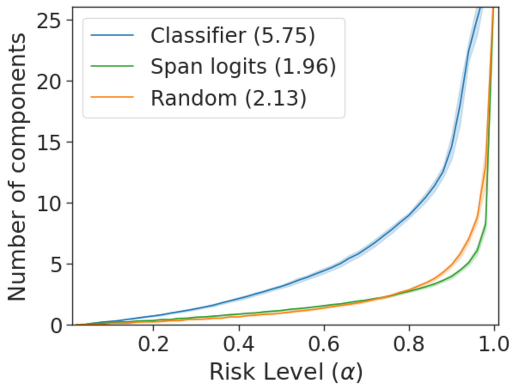


(a) TriviaQA

Figure 3-6: Conformal sampling results for $\mathcal{C}_\lambda$ a function of ϵ, on TriviaQA. We report the loss, relative excess samples, and overall size (normalized by $k_{\max}$). We also report the AUC over achieved/non-trivial ϵ.



(a) MIMIC-CXR



(b) CNN/DM

Figure 3-7: Conformal component selection results for $\mathcal{C}_\gamma^{\text{inner}}$ as a function of α. We report the number of components identified in $\mathcal{C}_\gamma^{\text{inner}}$, which we want to maximize. We also report the AUC over α.

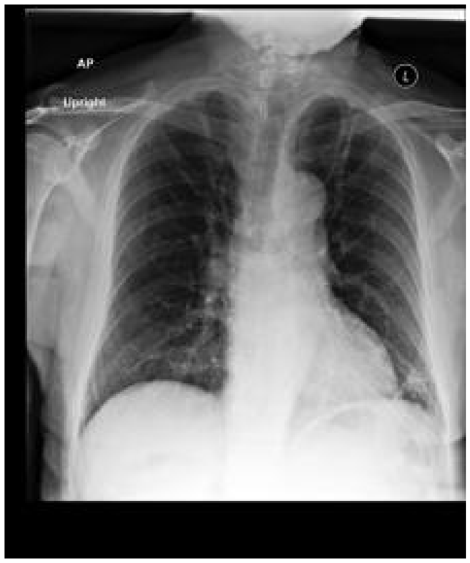
independently, we produce more effective (larger) sets. We show qualitative results in Appendix 3.7.

3.7 Qualitative results

We present qualitative results for the task of radiology report generation. In Figure 3-8, an X-ray example is shown, depicting left basilar opacities while the rest of the X-ray appears normal. Table 3.1 indicates that our method terminates the generation process after producing three samples. The third generation correctly identifies "apical scarring"; however, it mistakenly attributes it to the right lung instead of the left lung. This highlights a limitation of using CheXbert as the basis for the admission function, as its label granularity does not differentiate between left and right.

Our component selection method accurately identifies several sentences that align with the reference report. These sentences are displayed in bold. Notably, our method avoids emphasizing low-confidence findings such as "right apical scarring" and instead focuses on the absence of an acute cardiopulmonary process.

Furthermore, we examine a more challenging example in Figure 3-9. The report mentions an enlarged heart, signs of cardiomegaly, and edema. Samples 4 and 5 correctly capture these findings but are considered incorrect due to the inclusion of "effusion." The conformal selection of components chooses not to highlight any sentences since none of them meet the confidence threshold defined by ϵ.



AP and lateral views of the chest. Streaky biapical and left basilar opacities are most compatible with scarring. The lungs are clear of confluent consolidation. There is no effusion. Cardiomediastinal silhouette is within normal limits. No acute osseous abnormalities detected. IMPRESSION: No definite acute cardiopulmonary process.

Figure 3-8: Chest X-ray and reference radiology report for study id 55663120

3.8 Conclusion

3.8.1 Summary of contributions

Reliably using language models (LMs) in real-world tasks inevitably requires meaningful uncertainty quantification. In this chapter, we introduced a novel approach to conformal prediction that allows a user to sample prediction sets from generative LMs with infinite, combinatorial output spaces, while retaining desirable statistical guarantees. Our method bridges the gap between standard conformal prediction and LM inference techniques by calibrating a stopping rule for an algorithm that iteratively grows an output prediction set by sampling new generations (with rejection). Moreover, we provide a method for separately identifying answer components that we are more confident are accurate. This can help users better understand the quality of LM answers, including which parts may be incorrect (and vice versa) within a larger, verbose output. Finally, we demonstrate our method on three popular LM

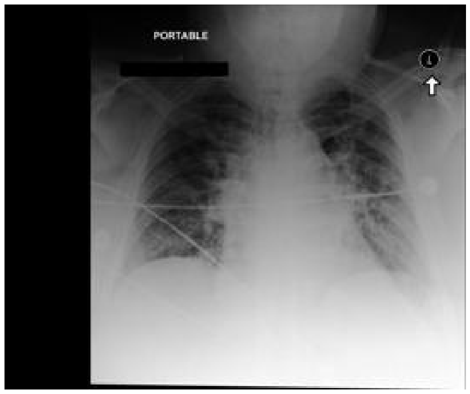
Table 3.1: Example prediction sets for study id 556631204, at $\alpha = 0.7$, $\epsilon = 0.7$. Bolded sentences are sentences that are selected by the component selection process.

	Text	**Score**	**Label**
Sample 1	**AP upright and lateral views of the chest provided.** There is chronic left lower lobe atelectasis. There is no focal consolidation concerning for pneumonia. **No large effusion or pneumothorax. The cardiomediastinal silhouette is normal.** Imaged osseous structures are notable for a mild compression deformity of an upper lumbar vertebral body. Degenerative change is seen in the right acromioclavicular joint. **IMPRESSION: No acute intrathoracic process.**	1.45e-05	False
Sample 2	**AP and lateral views of the chest.** Diffusely calcified pleural plaques are identified bilaterally suggesting prior asbestos exposure. There is no pleural effusion. **The cardiomediastinal silhouette is within normal limits. No acute osseous abnormalities detected.** Surgical clips project over the upper abdomen. **IMPRESSION: No definite acute cardiopulmonary process.**	1.98e-05	False
Sample 3	**AP and lateral views of the chest.** There is right apical scarring. Elsewhere, the lungs are clear. **The cardiomediastinal silhouette is within normal limits.** Atherosclerotic calcifications noted at the aortic knob. No acute osseous abnormality detected. **IMPRESSION: No acute cardiopulmonary process.**	5.36e-03	True

applications. Compared to common practices (e.g., first k), we obtain more efficient prediction sets, both in terms of size and samples required, leading to more effective outputs that are also less costly to obtain.

3.8.2 Discussion

Proposed guarantee Traditionally, conformal prediction provides marginal guarantees concerning the selection of calibration and test data. These guarantees are in the form of $\mathbb{E}[\mathbb{E}[\mathbf{1}_{Y \in \mathcal{C}(X)} | \mathcal{D}_{\text{cal}}]] \leq 1 - \varepsilon$. However, this approach fails to address the performance of a model in the future when a single split of the data is relied



In comparison with the study of ___, there is continued enlargement of the cardiac silhouette with increasing fullness and indistinctness of central pulmonary vessels, consistent with worsening pulmonary edema. Mild asymmetry at the left base could represent developing aspiration or even infectious

Figure 3-9: Chest X-ray and reference radiology report for study id 55770135

upon by a specific researcher or practitioner. The marginal validity of the system merely suggests that, on average, it will be valid across different researchers who use randomly drawn calibration sets.

In contrast, we follow prior work [5, 130] and propose a more conditional form of guarantee using the two-parameter form of Equation 3.1:

$$\mathbb{P}\Big(\mathbb{P}\Big(\exists y \in \mathcal{C}\lambda(X\text{test})\colon A_{\text{test}}(y) = 1 \mid \mathcal{D}_{\text{cal}}\Big) \geq 1-\epsilon\Big) \geq 1-\delta. \tag{3.17}$$

This equation ensures that at least a fraction of $(1-\delta)$ of researchers will be satisfied. In our empirical experiments (Section 3.6), we observe that the risk does not significantly increase for the remaining δ fraction, and that we achieve marginally valid prediction sets.

Dependence on model quality and achievable risk values The effectiveness of our method relies on the underlying LM's ability to generate diverse and accurate outputs. If the LM itself produces unreliable or redundant predictions, our approach may not fully mitigate these issues.

Algorithm 2 introduces a procedure that includes a maximum cap, denoted as k_{max}, on the number of generations per example. This cap guarantees the termination of our algorithm but imposes limitations on the achievable range of ϵ (target risk). Consequently, there exist certain combinations of $(\epsilon, \delta, k_{\max})$ that are unattainable for any selection of $\hat{\lambda}$. Practitioners may desire to experiment with different values for $k_{\max}$. However, it is important to note that even with unlimited sampling budget (i.e. $k_{\max} = \infty$), certain values of ϵ may remain unachievable. This is particularly true when the underlying model's support does not contain any acceptable answers.

3.8.3 Future work

In this section, we outline potential avenues for future research and development to further enhance the framework of applying conformal sampling to language modeling.

Alternative quality and scoring functions While our work demonstrates the effectiveness of using model logits as the scoring function, there is room for exploration of alternative approaches. Future research can investigate the use of self-evaluation techniques [77] or independent evaluators as quality functions. Additionally, more sophisticated aggregation functions can be explored, such as those based on estimating the marginal benefit of sampling a new generation or utilizing self-consistency as a measure of uncertainty.

Admission function In our study, our primary focus is on addressing *miscoverage*, which captures the user's preference to accept or reject a set based on whether at least one element is deemed good enough. When it is not feasible to precisely estimate user preferences in a computable admission function A, we demonstrate that it is possible to use a conservative admission function $\overline{A} \geq A$. However, different applications may require controlling for different notions of risk. Practitioners are encouraged to adapt the admission function accordingly to align with specific requirements. For instance, an application might consider a set of predictions acceptable if they collectively cover all relevant points (without any single prediction covering everything). Our framework allows for the adaptation of the admission function to accommodate such scenarios. It is important to note that the use of a binary admission function enables the utilization of efficient concentration inequalities, such as the binomial tail bound. If practitioners require different forms of risk control, appropriate p-values should be selected accordingly. For instance, in the bounded case of controlling for a risk $\mathcal{L}(\mathcal{C}(X), Y) \in [0, 1]$ (such as false positive rate), one could follow [5] and employ the Hoeffding-Bentkus inequality.

Explainability An exciting direction is the integration of the conformal selection of individual elements (as discussed in Section 3.4.4) with interpretable methods. This integration would enable us to generate explanatory text for each selected element. Such an approach holds the potential to significantly enhance the trustworthiness of generated predictions and serve as a crucial component for production-ready systems.

Constrained decoding Another promising avenue to explore is the alteration of the language model's decoding process (e.g., using a reinforcement learning objective) to directly enforce desirable properties, such as diversity. By doing so, we can achieve more efficient prediction sets while ensuring their validity.

Extension to other generation tasks Our approach holds the potential for application beyond language modeling, extending into various domains such as image generation, music composition, or de novo molecular design. By adapting the principles of our framework to these different domains, we can explore the generation of diverse and high-quality outputs while incorporating meaningful uncertainty quantification. This extension opens up exciting opportunities for advancing generative tasks in diverse fields and further enhancing the reliability and effectiveness of the generated results.

Table 3.2: Example prediction sets for study id 55770135, at $\alpha = 0.7$, $\epsilon = 0.7$. Bolded sentences are sentences that are selected by the component selection process.

	Text	**Score**	**Label**
Sample 1	In comparison with the study of ___, there is little overall change. Again there is enlargement of the cardiac silhouette with elevated pulmonary venous pressure and bilateral opacification is consistent with developing pulmonary edema or pneumonia in the appropriate clinical setting. The nasogastric tube again extends at least to the lower body of the stomach, where it crosses the lower margin of the image.	1.96e-05	False
Sample 2	Compared to chest radiographs ___ through ___. Moderate pulmonary edema is exaggerated due to the low lung volumes, but is new, including mild interstitial edema and engorgement of the mediastinal veins. Mediastinal veins are still engorged, but not large. Pleural effusions are presumed, but not large. Indwelling right subclavian line ends in the low SVC.	7.10e-08	False
Sample 3	Compared to chest radiographs ___ through ___. Moderate pulmonary edema is improving, although heart remains moderately enlarged and mediastinal veins are substantially dilated due to volume status. Bilateral pleural effusions are presumed, but not large. No pneumothorax. NOTIFICATION: I discussed the findings with the referring physician by telephone on ___ at 3:08 PM.	2.42e-07	False
Sample 4	Compared to chest radiographs ___ through ___. Moderate to severe pulmonary edema has worsened. Moderate cardiomegaly is chronically large, exaggerated by lower lung volumes. Pleural effusions are small if any. No pneumothorax.	2.35e-04	False
Sample 5	No previous images. The cardiac silhouette is enlarged and there is some indistinctness of pulmonary vessels consistent with mild elevation of pulmonary venous pressure. In view of the prominence of the pulmonary vasculature, it would be difficult to unequivocally exclude superimposed pneumonia, especially in the absence of a lateral view.	4.69e-04	False

Chapter 4

Neural Obfuscation for De-identification

Balancing privacy and predictive utility remains a central challenge for machine learning in healthcare. In this chapter, we develop *Syfer*, a neural obfuscation method to protect against re-identification attacks. *Syfer* composes trained layers with random neural networks to encode the original data (e.g. X-rays) while maintaining the ability to predict diagnoses from the encoded data. The randomness in the encoder acts as the private key for the data owner. We introduce a guesswork-based metric to evaluate the re-identification risk of encoding schemes and propose a contrastive learning algorithm to measure it. We show empirically that differentially private methods, such as DP-Image, can obtain privacy at the cost of a significant loss of utility. In contrast, *Syfer* achieves comparable privacy while preserving utility. For example, X-ray classifiers built with DP-image, *Syfer*, and original data achieve average AUCs of 0.53, 0.78, and 0.86, respectively.

4.1 Introduction

One of the key impediments to the development and democratization of clinical AI algorithms is the lack of public datasets. Access to such datasets is limited to a select few with established hospital partnerships, leaving out the majority of ML researchers interested to contribute. Moreover the lack of standard benchmarks for comparison slows methodological advancements for clinical AI and exacerbates reproducibility concerns. If this unfortunate trend continues, a significant portion of patient groups and diseases will be systematically underrepresented in available datasets, steering healthcare AI models towards inequitable outcomes [27]. In this work, we aim to develop a mechanism for data sharing while preserving patient privacy. We propose *Syfer*, a learned encoding scheme for data de-identification, where data owners encode their data with a random neural network (acting as their private key) for public release.

An ideal encoding scheme would enable untrusted third parties to develop classifiers for an arbitrary (i.e unknown) downstream task using standard machine learning tools, while preventing attackers from re-identifying raw samples. Designing such an encoding scheme with practical privacy-utility tradeoffs has remained a longstanding challenge for the community. Cryptographic tools (e.g. homomorphic encryption [177, 53, 15, 25, 51, 20, 28]) can guarantee both privacy and utility, but these methods are still too computationally expensive for modern model development. Differentially private methods [42] leverage additive random noise to achieve privacy. However, the stringent requirements of Local DP lead to a substantial utility loss, rendering predictions useless in the medical context [27]. As a result, hospitals still lack practical alternatives to the current practice of manual heuristic anonymization [68, 71] , which is an expensive process to abide by privacy regulations and does not offer a quantifiable notion of privacy. In this work, we propose to learn a keyed encoding scheme to achieve improved privacy-utility tradeoffs.

The relevant notion of privacy, as defined by HIPAA, is *de-identification*, i.e. preventing an attacker from identifying matching pairs between raw and encoded samples. We measure this risk using *guesswork*, i.e. the rank of the first correct guess by an attacker who ranks (raw image, encoded image) pairs from most likely to least likely. This setting subsumes *reconstruction* hardness and corresponds to a worst-case scenario where an attacker has gained access to the raw images (e.g. through a data leak). In particular, the adversary can exploit correlations in the raw data, a challenging setting for data privacy [79, 148]. In addition, we assume the attacker knows the implementation details of the encoding scheme, but does not have access to the data owner's private key. While stronger threat models exist (e.g. chosen-plaintext attack), we focus on the threat model that precisely captures HIPAA requirements. Furthermore, our method does not require hospitals to ever store the correspondence between raw and encoded samples, making it impermeable to attacks relying on parallel datasets, even in the event of data breaches.

While an arbitrary distribution of random neural networks is insufficient to achieve strong privacy on real-world datasets, we can learn to shape this distribution to obtain privacy. Using a public dataset, *Syfer*'s obfuscator layers are optimized to maximize the re-identification loss of a model-based attacker while minimizing a reconstruction loss, maintaining the (almost) invertibility of the whole encoding. Data owners then compose the pretrained obfuscator with (private) random layers to encode their private data samples. To encode labels, they apply a random permutation to the label identities.

Characterizing the privacy-utility tradeoffs of such complex encoding schemes on real-world datasets is an open challenge, theoretically and empirically. To measure these tradeoffs, we leverage a realistic utility benchmark and develop a flexible computational attacker trained to maximize the likelihood of re-identifying raw images across encodings. While such a computational attacker does not provide a lower bound on guesswork, we show that it refutes previously proposed methods such as InstaHide [67] and Dauntless [169], which both achieve a guesswork of 1.

We train *Syfer* on a public X-ray dataset from NIH [156], and evaluate the privacy-utility tradeoffs of the scheme on a heldout dataset (MIMIC-CXR [71]) across multiple attacker architectures and prediction tasks. *Syfer* displays strong empirical privacy, with an expected guesswork of 8411, (i.e. when presented with 10,000 raw samples $\times$ 10,000 encoded samples, an attacker ranks their first correct pairwise correspondence at rank 8411 on average). This matches the privacy provided by differentially private schemes, such as DP-Image [93] which gets a guesswork of 9037 at $\varepsilon = 4$. Moreover, models built on *Syfer* encodings obtain an average AUC of 0.78 across diagnosis tasks, approaching the accuracy of models built on raw images (0.84 AUC). In contrast, DP-Image only achieves an average utility AUC of 0.53, even when using a higher $\varepsilon = 32$, at the cost of worse privacy, with an average guesswork of 1146.

We emphasize that *Syfer* does not come with theoretical privacy guarantees. However, we believe that the strong empirical results and favorable comparison against state-of-the-art encoding schemes, that do offer privacy guarantees, will foster the development and theoretical analysis of learned privacy-preserving methods.

4.2 Related Work

Differentially Private Dataset Release Differential privacy [43] methods offer strong privacy guarantees by leveraging additive random noise in accordance to the maximum sensitivity of the function of interest (e.g. dataset release algorithms). For instance, DP-Image [93] proposed to add laplacian noise to the latent space of an auto-encoder to produce differentially private instance encodings. Instead of directly releasing noisy data, methods like DP-GAN [171, 146, 73] propose to leverage generative adversarial networks, trained in a differentially private manner (e.g. DP-SGD [1] or PATE [117]), to produce private synthetic data. Such methods would require data owners to have the computational resources and skills to train models that are orders of magnitude more complex than classifiers of the target task. Besides, differentially private tools have been shown to significantly degrade image quality and result in large utility losses [27]. Instead of leveraging independent noise per sample to achieve privacy, *Syfer* obtains privacy through its keyed encoding scheme and thus enables improved privacy-utility trade-offs.

Cryptographic Techniques Cryptographic techniques, such as secure multiparty computation and fully homomorphic encryption [177, 53, 15, 25, 51, 20, 28] allow data owners to encrypt their data before providing them to third parties. These tools provide extremely strong privacy guarantees, making their encrypted data indistinguishable under chosen plaintext attacks (IND-CPA). However, building models with homomorphic encryption [111, 96, 76, 18] requires leveraging specialized cryptographic primitives and induces a large computational overhead (ranging from 100x-10,000x [99]) compared to standard model inference. As a result, these tools are still too slow for training modern deep learning models. In contrast, *Syfer* considers a weaker threat model, where attackers cannot query the data owner's private-encoder (i.e no plaintext attacks) and specifically protects against raw data re-identification (the privacy notion of HIPAA). Moreover, *Syfer* encodings can be directly leveraged by standard deep learning techniques, improving their applicability.

Lightweight Encoding Schemes Our work extends prior research in lightweight encoding schemes for dataset release. Previous approaches [80, 143, 137] have proposed tools to carefully distort images to reduce their recognition rate by humans while preserving the accuracy of image classification models. However, these methods do not offer privacy against machine learning based re-identification attacks. [166, 170, 167, 124] have proposed neural encoding schemes that aim to eliminate a particular private attribute (e.g. race) from the data while protecting the ability to predict other attributes (e.g. action) through adversarial training. These tools require labeled data for sensitive and preserved attributes, and cannot prevent general re-identification attacks while preserving the utility of unknown downstream tasks. Our work is most closely related to general purpose encoding schemes like InstaHide [67] and Dauntless [169, 168]. InstaHide encodes samples by randomly mixing images with MixUp [181] followed by a random bitwise flip. Dauntless encodes samples with random neural networks and provides information-theoretic bounds on how well an adversary can approximate the private transformation. However, we show that neither InstaHide nor Dauntless meet our privacy standard on our real-world image datasets. In contrast, *Syfer* leverages a composition of trained obfuscator layers and random neural networks to achieve privacy on real word datasets while preserving downstream predictive utility.

Evaluating Privacy with Guesswork Our study builds on prior work leveraging guesswork to characterize the privacy of systems [101, 102, 7, 120, 14]. Guesswork quantifies the privacy of a system as the number of trials required for an adversary to guess private information, like a private key, when querying an oracle. In this framework, homomorphic encryption methods, which uniformly sample b-bit private keys, offer maximum privacy [41], as the average number of guesses to identify the correct key is 2^{b-1}. In the non-uniform guessing setting [30], guesswork offers a worst-case notion of privacy by capturing the situation where an attacker may only be confident on a single patient identity. Such privacy weaknesses are not measured by average case metrics, like Shannon entropy.

4.3 Problem Statement

Our problem setting is illustrated in Figure 4-1. A data owner (Alice) wishes to publish an encoded medical dataset to enable untrusted third parties (Bob) to develop machine learning models, while protecting patient privacy. In Figure 4-2, we describe how Bob uses the encoded dataset to train a classifier C_T. The classifier is then used by Alice in Figure 4-3 to make predictions on unseen images.

We consider an adversary (Eve) with knowledge of Alice's image distribution, encoding scheme and encoded data. Intuitively, Alice's encoding scheme is deemed *private* if it takes many guesses for Eve to re-identify any single raw image from the encoded data.

Alice (the Data Owner). Alice uses a potentially randomized algorithm to transform each sample of her dataset and releases the encoded data along with their encoded labels. Formally, let $X_A = \{x_1, ..., x_n\} \subset \mathbb{R}^d$ denote Alice's dataset, composed of n samples, each of size d. She is interested in a k-class classification task

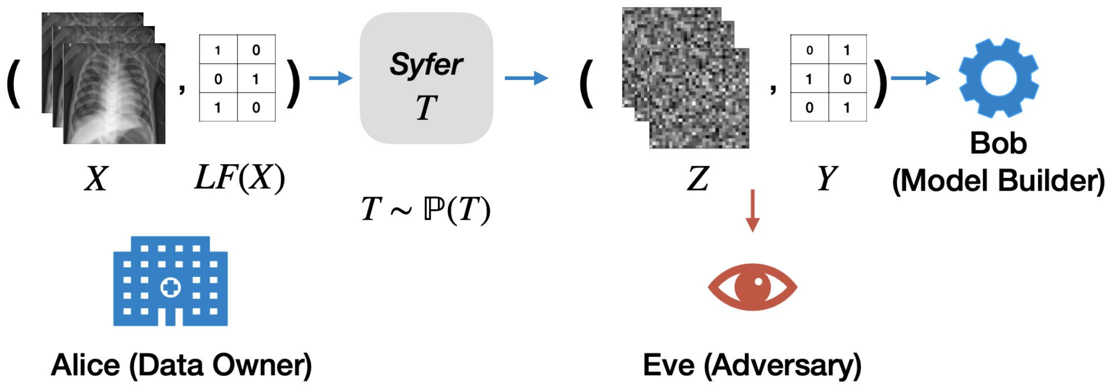


Figure 4-1: Alice (data-owner) samples a transformation T according to $\mathbb{P}(\boldsymbol{T})$ and leverages T to encode her labeled data $(X, LF(X))$. She then publishes $(Z, Y) = T(X, LF(X))$ for model development by Bob, while preventing re-identification by Eve, the adversary.

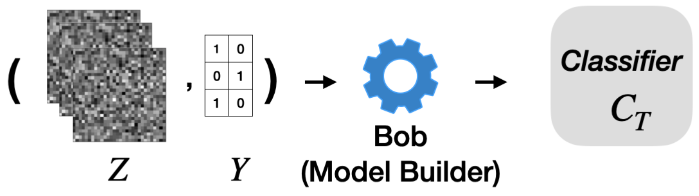


Figure 4-2: Role of Bob the model builder. Given encoded images Z and encoded labels Y, Bob's task is to learn a classifier to predict y from a sample z

defined by a labeling function $LF : \mathbb{R}^d \rightarrow \{1, \ldots, k\}$ (e.g. diagnosis labels annotated by a radiologist), and assembles the labeled dataset $\{(x, LF(x)), x \in X_A\}$. To encode her data, Alice samples a transformation $T = (T^X, T^Y)$ according to a distribution $\mathbb{P}(\boldsymbol{T})$, where $T^X : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a sample encoding function and $T^Y : \{1, ..., k\} \rightarrow \{1, ..., k\}$ is a label encoding function. Alice releases the encoded dataset $(Z_A, Y_A) = T(X_A, LF(X_A)) = (T^X(X_A), T^Y(LF(X_A))$. We use Γ to denote the encoding scheme defined by $\mathbb{P}(\boldsymbol{T})$.

In the case of *Syfer*, Alice samples random neural network weights to construct a T^X and samples a permutation to construct a T^Y. We assume all actors have

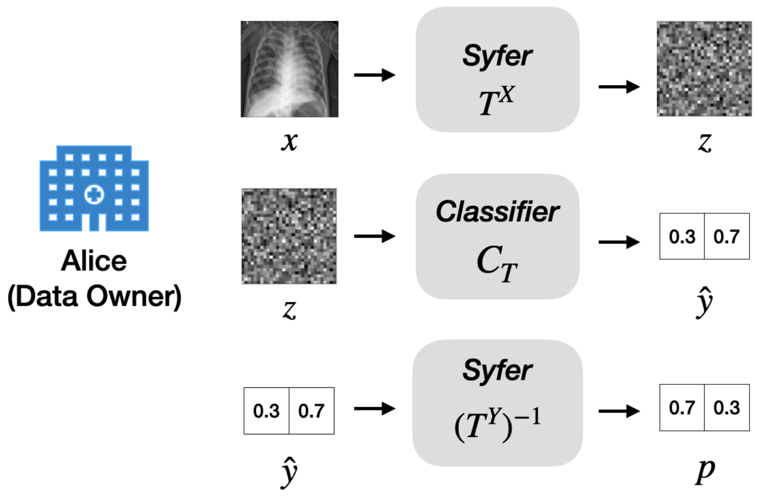


Figure 4-3: Illustration of how Alice, the data owner, uses a model (at test time) built using Syfer encodings. Alice receives C_T from Bob. Given a new image x, she first encodes it with her private key T^X to produce z. She then inputs z into Bob's classifier to obtain a encoded prediction $\hat{y}$. Alice then leverages the inverse of her label encoding, $(T^Y)^{-1}$, to decode Bob's prediction into a raw prediction p. Alice's utility is then $\mathcal{L}(p, LF(x)$, which compares the raw prediction to the true label $LF(x)$.

knowledge of the distribution $\mathbb{P}(\boldsymbol{T})$ (i.e. the neural network architecture and the pretrained weights of the obfuscator layers), but do not know Alice's specific choice of random T (i.e. sampled weights or permutation).

Bob (the Model Builder). As illustrated in Figure 4-2, Bob learns to classify the encoded data. He receives the encoded training dataset $(Z_A^{\text{train}}, Y_A^{\text{train}}) = T(X_A^{\text{train}}, LF(X_A^{\text{train}}))$ and trains a model C_T for the task of interest. Bob then shares the model C_T with Alice who can then use the model on newly generated data and decode the predictions using the inverse of the label encoding, $\left(T^Y\right)^{-1}$. We illustrate how Alice uses C_T in Figure 4-3. We note that Bob does not know T or $\left(T^Y\right)^{-1}$. Bob's objective is to minimize the generalization error of his classifier on the testing set $(Z_A^{\text{test}}, Y_A^{\text{test}}) = T(X_A^{\text{test}}, LF(X_A^{\text{test}}))$.

Definition 1 (utility)**.** The *utility score* of an encoding scheme Γ for a labeling function LF on a dataset X_A is defined as the expected value of the generalization performance of Bob's classifier, i.e.

$$\mathcal{U} = - \mathop{\mathbb{E}}_{T \sim \mathbb{P}(\boldsymbol{T})} \left[\mathop{\mathbb{E}}_{x \in X_A^{\text{test}}} \left[\mathcal{L} \left(\left(T^Y\right)^{-1} \circ C_T(z), LF(x) \right) \right] \right]$$

where $\mathcal{L}$ is a loss function (e.g. cross entropy), $z = T^X(x)$, C_T is a classifier trained for a specific transformation T, and $\left(T^Y\right)^{-1}$ is the inverse of the label encoding.

Eve (the Adversary). Eve's objective is to re-identify Alice's data. In a realistic setting, Eve would use an encoded image from Alice's released dataset to reconstruct the pixels of the corresponding raw image. To simplify analysis, we consider an easier task for Eve where she only needs to select the correct raw image from a list X_E of candidate raw images, akin to identifying a suspect from a police lineup. In the rest of the chapter, we use $X_E = X_A$. Knowing the dataset X_A and the encoding scheme Γ, Eve's task is to leverage the released dataset $(Z_A, Y_A) = T(X_A, LF(X_A))$ to re-identify samples from Z_A. Specifically, Eve's objective is to identify a *single* correct match within $M_T = \{(x, z) \in X_A \times Z_A \text{ s.t. } z = T^X(x)\}$.

Definition 2 (privacy)**.** We quantify the *privacy* of an encoding scheme by computing the *guesswork* of its transformations. Intuitively, a computationally unbounded Eve ranks pairs of (x, z) as most likely to least likely. Guesswork is defined as the rank of Eve's first correct guess. Given the dataset X_A, the encoding scheme Γ, i.e. $\mathbb{P}(\boldsymbol{T})$, the released dataset Z_A (of size n) along with released labels Y_A, Eve derives for any $(x, z) \in X_A \times Z_A$ the probability $\mathbb{P}\left((x, z) \in M_T | Z_A, Y_A, X_A, \Gamma\right)$. Eve then submits an ordered list of n^2 correspondence guesses $(u_1, u_2, \ldots u_{n^2})$, where $u_i \in X_A \times Z_A$, by greedily ordering[1] her guesses from most likely to least likely. The guesswork of a transformation T is defined as the index of the first correct guess in the ordered list, i.e.

$$\mathcal{G}(T) = \min_k \{k \text{ s.t. } u_k \in M_T\}.$$

Finally, to compare the privacy of encoding schemes, we compare the distributions of

[1]Whenever ties occur, we compute $\mathcal{G}(T)$ as an average over the permutations of the list that keep it ordered.

$\mathcal{G}(T)$ as T is drawn from different distributions $\mathbb{P}(\boldsymbol{T})$.

4.4 Privacy quantification by guesswork

Recall that for an ordered list of mn correspondence guesses $(u_1, \ldots, u_{mn})$, where $u_i \in X_E \times Z_A$, the guesswork is defined as the rank of the first correct guess: $\mathcal{G} = \min_k\{k \text{ s.t. } u_k \in M_T\}$, where $M_T = \{(x,z) \in X_E \times Z_A \text{ s.t. } z = T^X(x)\}$. In the event of ties, the guesswork is computed as the expected value over permutations of the suitable subsets. In this work, we use $X_E = X_A$ but the guesswork can be computed for an arbitrary superset $X_E \supseteq X_A$ of size m.

Guesswork Algorithm We propose the following algorithm to compute the guesswork for a given probability matrix and set of correct guesses.

Algorithm 4 Guesswork algorithm

```
Input Correct matching $M_T = \{(x_i, z_j) \text{ s.t. } z_j = T^X(x_i)\}$
Input Probability matrix $A$ where $A_{i,j} = \mathbb{P}((x_i, z_j) \in M)$
Output Guesswork $\mathcal{G}$ for $A$
 1: From $A$, extract
 2: $S = \{(i, j, A_{i,j}) \text{ for } 1 \le i \le m, 1 \le j \le n\}$
 3: Partition $S$:
 4: $S = \bigcup_p S_p$ where $S_p = \{(i, j, A_{i,j}) \text{ s.t. } A_{i,j} = p\}$
 5: Find the highest value of $p$ such that $A_p$ contains matches:
 6: $q = \max_p\{p \text{ s.t. } \exists(i, j, A_{i,j}) \in S_p \text{ s.t. } (x_i, z_j) \in M_T\}$
 7: $\mathcal{G} \leftarrow 0$
 8: for $p > q$ do
 9:     $\mathcal{G} \leftarrow \mathcal{G} + |A_p|$
10: $\mathcal{G} \leftarrow \mathcal{G} + \frac{1+|A_q|}{1+|A_q \cap M|}$
11: return $\mathcal{G}$
```

The expression $\frac{1+|A_q|}{1+|A_q \cap M|}$ is derived by computing the expected value of number of trials before success in the urn problem without replacement.

Example 4.4.1 (Guesswork Calculation)**.** Let $X_E = X_A = \{1, 2\}$ and disregard labels for now. Consider three transformations $X_A \to \{a, b, c, d\}$:

$$T_1 : \begin{cases} 1 & \mapsto a \\ 2 & \mapsto b \end{cases} \qquad T_2 : \begin{cases} 1 & \mapsto b \\ 2 & \mapsto a \end{cases} \qquad T_3 : \begin{cases} 1 & \mapsto c \\ 2 & \mapsto d \end{cases}$$

We evaluate the following encoding schemes, defined by the distribution used to sample T:

$$\begin{aligned} \Gamma_1 : &\quad \mathbb{P}(T_1) = 2/3,\ \mathbb{P}(T_2) = 1/3,\ \mathbb{P}(T_3) = 0 \\ \Gamma_2 : &\quad \mathbb{P}(T_1) = 1/2,\ \mathbb{P}(T_2) = 1/2,\ \mathbb{P}(T_3) = 0 \\ \Gamma_3 : &\quad \mathbb{P}(T_1) = 1/3,\ \mathbb{P}(T_2) = 1/3,\ \mathbb{P}(T_3) = 1/3. \end{aligned}$$

For Γ_1, Eve observes $Z_A = \{a, b\}$ regardless of the choice of T_A. Given her knowledge of $\mathbb{P}(\boldsymbol{T})$, she elects to rank $\{(1, a), (2, b)\}$ before $\{(1, b), (2, a))\}$ which gives guessworks $\mathcal{G}(T_1) = 1$ and $\mathcal{G}(T_2) = 3$. In expectation, the guesswork of Γ_1 is 5/3 (with a variance of 8/27).

For Γ_2, Eve observes $Z_A = \{a, b\}$ as well, but equally ranks all 4! orderings of the guesses $((1, a), (1, b), (2, a), (2, b))$, which leads to the same guesswork for both T: $\mathcal{G}(T_1) = \mathcal{G}(T_2) = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{2}{3} \cdot 2 + \frac{1}{2} \cdot \frac{1}{3} \cdot 3 = \frac{5}{3}$ (no variance).

For Γ_3, whenever Eve observes $Z_A = \{c, d\}$, she deduces that $T_A = T_3$, which leads to a guesswork of 1. In the other cases, observing $Z_A = \{a, b\}$ means that T_1 and T_2 are equally likely, so the guesswork is 5/3. In expectation, the guesswork is 13/9, which is lower (and thus worse privacy) than the previous schemes.

Example 4.4.2 (Guessworks in Special Cases)**.** We discuss two special cases that arise when computing guesswork.

1. If all guesses in the bucket A_p of highest probability are correct guesses, then the guesswork is 1, characterizing a non-private scheme. Note that this does not depend on the cardinal of the bucket A_p of highest probability: regardless of

whether the attacker is confidently correct about one matching pairs or multiple matching pair, the guesswork will still be 1.

2. If the probability matrix is uniform (i.e. there is p for which $S = S_p$, such that all guesses are in the same bucket), then the guesswork is $\frac{mn+1}{n+1}$, i.e. $\mathcal{G} \approx |X_E|$. This characterizes an attacker that fails to capture any privacy leakage of the encoding scheme.

Note that $|X_E|$ is not an upper-bound of guesswork. An attacker that is confidently wrong can achieve a guesswork up to $mn - n + 1$.

Discussion on the Guesswork Definition Traditionally, guesswork is defined as the "number of guesses an attacker needs in order to break a system", while we only measure and report the "rank of the first correct match in the attacker ordered list of guesses". When guesses are correlated, those definitions do not match.

For Γ_1, Γ_2 and Γ_3 defined in Example 4.4.1, an attacker can rule out the $(2, b)$ correspondence after guessing $(1, a)$, which brings the expected number of guesses to respectively $\frac{4}{3}$, $\frac{3}{2}$ and $\frac{4}{3}$.

In Example 4.4.2.1, if the guesswork is 1, then the expected number of guesses will also be 1. In Example 4.4.2.2, when $\mathbb{P}((x_i, z_j)$ is uniform, Eve should commit to a single column (or row) and achieve an expected number of guesses of $m/2$ (or $n/2$). More generally, after Eve made her first guess u_1, she can assume the first guess was incorrect and recompute the new probability matrix $\mathbb{P}((x_i, z_j) \in M | u_1 \notin M)$, then proceed with subsequent guesses. Such an auto-regressive strategy is costly to implement. Therefore, we adopt the definition of guesswork exposed in Section 4.3 as an efficient way to universally compare the privacy of different encoding schemes. In particular, our definition of guesswork can rule out schemes that are clearly not private (guesswork of 1), while providing a scale-invariant measure of *uniformness* of the probability matrix A.

4.5 Method

We propose *Syfer*, an encoding scheme which uses a combination of learned *obfuscator* layers and random neural network layers to encode raw data. *Syfer* is trained to maximize the re-identification loss of an attacker while minimizing a reconstruction loss, which acts as a regularizer to preserve predictive utility for downstream tasks. To estimate the privacy of an encoding scheme on a given dataset, we use a model-based attacker trained to maximize the likelihood of re-identifying raw data. To encode the labels $LF(X)$, *Syfer* randomly chooses a permutation of label identities $\{1, ..., k\}$.

4.5.1 Privacy Estimation via Contrastive Learning

Before introducing *Syfer*, we adopt Eve's perspective and describe how to evaluate the privacy of encoding schemes. The attacker is given the candidate list $X_E = X_A$, and a fixed encoding scheme Γ, i.e. a fixed distribution $\mathbb{P}(\boldsymbol{T})$. We propose an efficient contrastive algorithm to estimate $\mathbb{P}\left((x, z) \in M_T | Z_A, Y_A, \Gamma, X_A\right)$. When the context allows it, we omit the conditional terms and use $\mathbb{P}((x, z) \in M_T)$.

As shown in Figure 4-4, the attacker's model E is composed of an instance-level encoder E^{ins}, with parameters φ^{ins}, acting on individual images and their labels and a set-level encoder E^{set}, with parameters φ^{set}, taking a set of instance representations as input.

In each iteration, we sample a batch $X = (x_1, \ldots, x_b)$ of datapoints from $X_E = X_A$ and a transformation $T = (T^X, T^Y)$ according to the fixed distribution $\mathbb{P}(\boldsymbol{T})$. Let $Z = T^X(X) = (z_1, \ldots, z_b)$ denote the transformed batch and $Y = T^Y(LF(X)) = (y_1, \ldots, y_b)$ the encoded labels. The hidden representations of the raw data are computed as a two-step process:

1. using E^{ins}, we compute $H^X = (h_1^X, \ldots, h_b^X)$ where each $h_i^X = E^{\text{ins}}\left(x_i, LF(x_i)\right)$;

2. using E^{set}, we compute $R^X = (r_1^X, ..., r_b^X)$ where each $r_i^X = E^{\text{set}}\left(h_i^X, H^X\right)$.

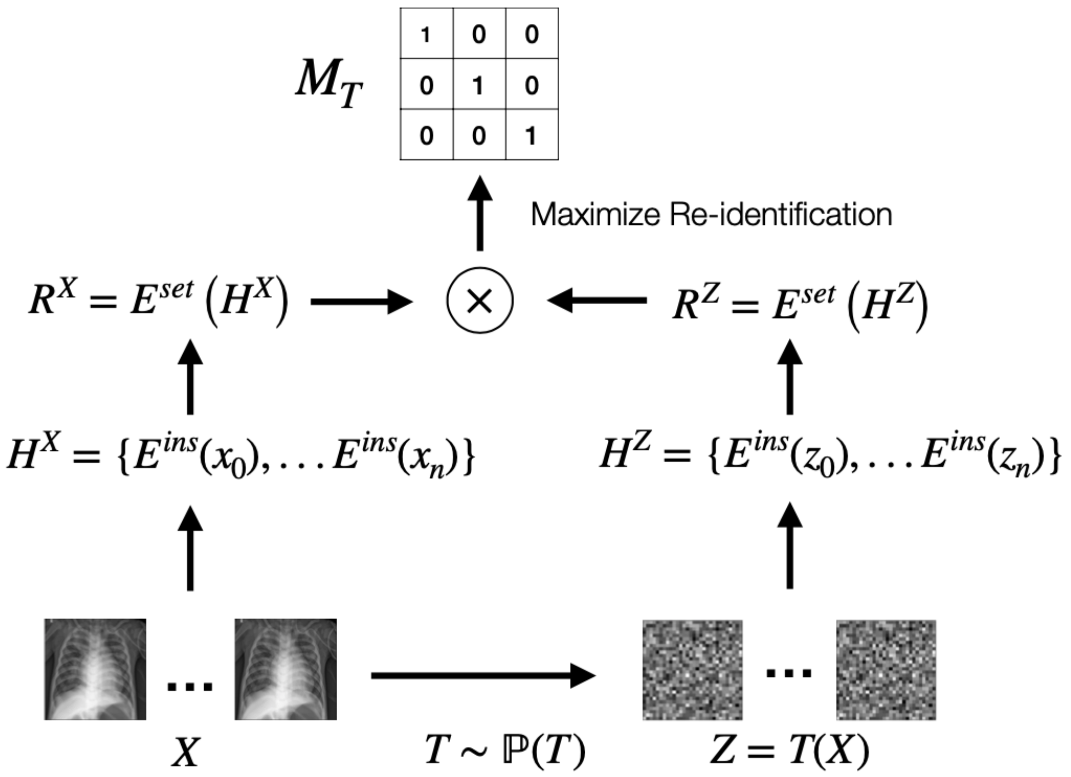

M_T
1 0 0
0 1 0
0 0 1
Maximize Re-identification
$R^X = E^{set}(H^X)$
$R^Z = E^{set}(H^Z)$
$H^X = \{E^{ins}(x_0), \dots E^{ins}(x_n)\}$
$H^Z = \{E^{ins}(z_0), \dots E^{ins}(z_n)\}$
X
$T \sim \mathbb{P}(T)$
$Z = T(X)$

Figure 4-4: Architecture of the model-based attacker. Given pairs of raw samples $(X, LF(X))$ and encoded samples (Z, Y), the attacker learns to recover matching pairs $(x, z) \in M_T$. In this figure, we omit label information for clarity.

Similarly, for the encoded data, we form $H^Z = (h_1^Z, \dots, h_b^Z)$ where $h_i^Z = E^{\text{ins}}(z_i, y_i)$ and $R^Z = (r_1^Z, ..., r_b^Z)$ where each $r_i^Z = E^{\text{set}}\left(h_i^Z, H^Z\right)$.

Following prior work on contrastive estimation [26], we use the cosine distance between hidden representations to measure similarity:

$$\text{sim}(r_i^X, r_j^Z) = \frac{\left(r_i^X\right)^\top r_j^Z}{\|r_i^X\| \|r_j^Z\|} \ .$$

Then, we estimate the quantity $\mathbb{P}((x_i, z_j) \in M_T)$ as proportional to $\hat{p}(x_i, z_j)$:

$$\hat{p}(x_i, z_j) = \frac{\exp(\text{sim}(r_i^X, r_j^Z))}{\sum_{k,l}^{b} \exp(\text{sim}(r_k^X, r_l^Z))} \ .$$

The weights φ^{ins} and φ^{set} of the attacker's model E are trained to minimize the negative log-likelihood of re-identification across unknown T:

$$\mathcal{L}_{\text{reid}} = -\sum_{(x,z)\in M_T} \log\left(\hat{p}(x,z)\right) \ .$$

4.5.2 *Syfer*

Architecture As illustrated in Figure 4-5, we propose a new encoding scheme by learning to shape the distribution $\mathbb{P}(\boldsymbol{T})$. Specifically, we parametrize a transformation T^X using a neural network that we decompose into blocks of learned *obfuscator* layers (weights θ_{Syfer}), and random layers (weights θ_{key}). The *obfuscator* layers are trained to leverage the randomness of the subsequent random layers and learn a distribution $\mathbb{P}(\boldsymbol{T})$ that achieves privacy. In this framework, Alice constructs T^X by randomly sampling the weights θ_{key} and composing them with pre-trained obfuscator weights θ_{Syfer} to encode the raw data X. Alice chooses the label encoding T^Y by randomly sampling a permutation of the label identities $\{1,\ldots,k\}$, which is applied to $LF(X)$. We note that our T^Y assumes that Alice's dataset is class-balanced. If Alice's data is not class-balanced, she should down-sample her dataset to a class-balanced subset before release.

Alice's random choices of θ_{key} and T^Y act as her private key, and she can publish the encoded data with diagnosis labels for model development while being protected from re-identification attacks. Given Bob's trained classifier to infer $T^Y(LF(x))$ from $T^X(x)$, Alice then uses $\left(T^Y\right)^{-1}$ to decode the predictions.

Training Data owners may not have the computational capacity to train their own obfuscator layers, so we train *Syfer* without direct knowledge of X_A or LF. Instead, we rely on a public dataset X^{public} and use the null labeling function $LF(x) = 0$. To be successful, *Syfer* needs to generalize to held-out datasets, prediction tasks and attackers.

As shown in Algorithm 1, we train *Syfer*'s obfuscator layers (parameters θ_{Syfer}) to maximize the loss of an attacker E (parameters $\varphi = (\varphi^{\text{ins}}, \varphi^{\text{set}})$) and to minimize

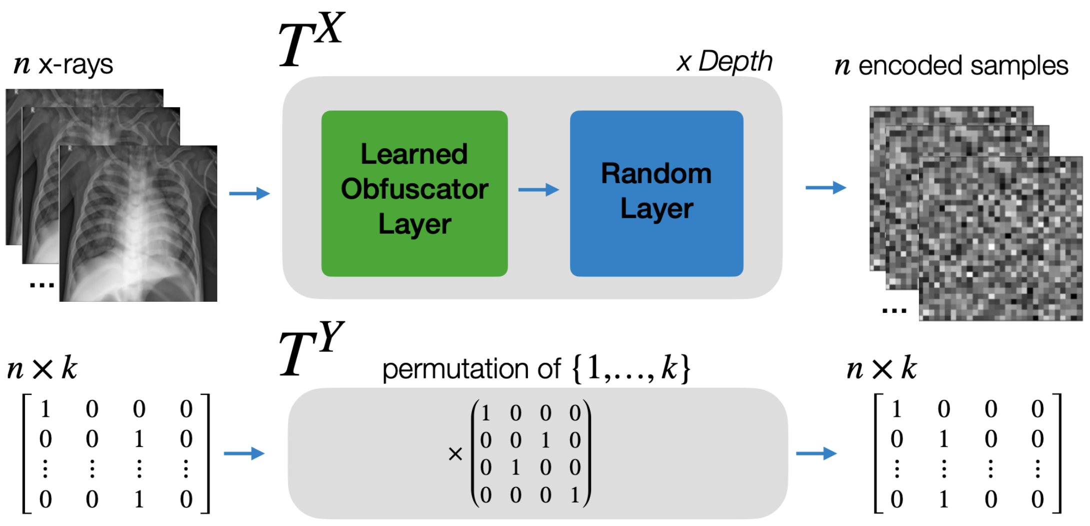


Figure 4-5: Proposed encoding scheme: *Syfer* uses repeating blocks of learned obfuscator layers and random neural network layers as T^X and samples a random permutation of $\{1, ..., k\}$ as T^Y.

the reconstruction loss of an ensemble of decoders $D_1, \ldots, D_s$ (parameters $\beta_1, \ldots \beta_s$).

At each step of training, we sample a transformation T^{batch} by choosing a new $\theta^{\text{batch}}_{key}$ to combine with the current θ_{Syfer} (Alg. 1, L.7-8). Using the current attacker weights φ, we then compute the re-identification loss (Alg. 1, L.9-10) as: $\mathcal{L}_{\text{reid}} = -\sum_{(x,z)\in M_T} \log\left(\hat{p}(x,z)\right)$

Next, we estimate the overall invertability of the encoding scheme by measuring the reconstruction loss of an ensemble of decoders $D_1, \ldots, D_s$. For each each decoder D_i, we randomly sample a private key θ^i_{key}, which is fixed throughout the training algorithm. Each decoder D_i is trained to reconstruct X from $Z = T^i(X)$ where T^i is constructed by composing the current θ_{Syfer} with θ^i_{key}. We update β_i to minimize the reconstruction loss (Alg. 1, L.12-17): $\mathcal{L}_{\text{rec}} = \sum_{i=1}^{s} \left(\mathbb{E}_X[||x - D_i \circ T^i(x)||^2]\right)$

We train our attacker and decoders in alternating fashion with *Syfer*'s obfuscator parameters. On even steps, *Syfer*'s weights θ_{Syfer} are updated to minimize the loss:

$$\mathcal{L}_{Syfer} = \lambda_{\text{rec}} \cdot \mathcal{L}_{\text{rec}} - \lambda_{\text{reid}} \cdot \mathcal{L}_{\text{reid}}$$

On odd steps, the attacker and decoders are updated to minimize $\mathcal{L}_{\text{reid}}$ and $\mathcal{L}_{\text{rec}}$

respectively (Alg. 1, L. 19-26).

In this optimization, the tasks of our attacker and decoders are asymmetric: the attacker is trained to generalize across transformations T (i.e. θ_{key}), while the decoders only need to generalize to unseen images, for a fixed key θ_{key}.

4.6 Experiments

Datasets For all experiments, we utilized two benchmark datasets of chest X-rays, NIH [156] and MIMIC-CXR [71], from the National Institutes of Health Clinical Center and Beth Israel Deaconess Medical Center respectively. Both datasets were randomly split into 60,20,20 for training, development and testing, and all images were downsampled to 64x64 pixels. We leveraged the NIH dataset to train all private encoding schemes (i.e. *Syfer* and baselines), and we evaluated the privacy and utility of all encoding schemes on the MIMIC-CXR dataset, for the binary classification tasks ($k = 2$) of predicting Edema, Consolidation, Cardiomegaly, and Atelectasis. This reflects the intended use of the tool, where a hospital leverages a pretrained *Syfer* for their heldout datasets.

For experiments considering a specific diagnosis task, we followed common practice [68] and excluded exams with an uncertain disease label i.e., the clinical diagnosis did not explicitly rule out or confirm the disease. Then, we selected one random negative control case for each positive case in order to create a balanced dataset. Our dataset statistics are shown in Appendix C.1.

***Syfer* Implementation Details** As shown in Figure 4-5, *Syfer* consists of repeated blocks of trained obfuscator layers and random neural network layers. Following prior work in vision transformers [185], *Syfer* operates at the level of patches of images. We implemented our trained obfuscator layers as Simple Attention Units (SAU), a gated multi-head self attention module. The full architecture details are listed in Appendix C.2. The SAU module is detailed in Appendix C.3.

We implemented the instance encoder E^{ins} and set encoder E^{set} of our adversary model as a depth 3 and depth 1 SAU respectively. We utilized separate E^{ins} networks to encode the raw data $(X, LF(X))$ and encoded data (Z, Y). We use a single decoder D_1 (i.e. $s = 1$) and implement it as a depth 3 SAU. While using an ensemble of decoders should improve the invertibility of the encodings, we saw that using $s = 5$ did not significantly improve downstream utility. The full training details are described in Appendix C.2 and training *Syfer* is fully reproducible in our code release.

Privacy Estimators To evaluate the ability of *Syfer* to defend against re-identification attacks, we trained attackers to re-identify raw images from *Syfer* encodings on the MIMIC-CXR dataset. Since we cannot bound the prior knowledge the attacker may have over X_A, we consider the extreme case and train our attackers on their evaluation set, i.e. we only use MIMIC-CXR's training set for privacy evaluation. As a result, the attacker does not have to generalize to held-out images, but only to held-out private encoders T.

As described in Section 4.5.1, the attacker is trained to re-identify raw images from encoded images across new unobserved private keys using an image encoder E^{ins} and a set encoder E^{set}. This attacker estimates $\mathbb{P}((x, z) \in M)$ for an encoding scheme Γ on a dataset X. Across our experiments, we implemented E^{ins} as either a ResNet-18 [59], a ViT [185], or a SAU. We implemented E^{set} as a depth 1 SAU. All attackers were trained for 500 epochs.

We computed the guesswork of each attacker by sorting the scores $\hat{p}(x, z)$ and identifying the index of the first correct correspondence. To measure the attackers average performance, we also evaluated the ROC AUC of the attacker attempting to predict an (x, z) matching as a binary classification task. A higher guesswork and lower re-identification AUC (ReID AUC) reflect a more private encoding scheme.

Generalized Privacy We first evaluated the guesswork and re-identification AUC (ReID AUC) of attackers trained using only encoded images (i.e. without labels) on the entire unfiltered MIMIC-CXR training set. For *Syfer*, this only requires using the neural encoder T^X. We compared *Syfer* to prior lightweight encoding schemes, including InstaHide [67] and Dauntless [169, 168]; and differential privacy methods, like DP-Image [93]. We now detail our baseline implementations.

- To assess the value of training *Syfer*'s obfuscator layers, we compared *Syfer* to an ablation with randomly initialized obfuscator layers, *Syfer*-Random.
- InstaHide randomly mixes each private image with 2 other private images (i.e with MixUp [181]) and then randomly flips each pixel sign.
- Dauntless [168] applies a separate random linear layer to each 16x16 pixel patch of the images, with each random weight initialized as according to a standard Gaussian distribution.
- DP-Image [93] adds independent laplacian noise to the latent space of an auto-encoder to produce differentially private encodings. We trained our auto-encoder on the NIH dataset and measure its l_1-sensitivity on MIMIC-CXR dataset as $\Delta f = 137.8$. We then applied laplacian noise parametrized by $b = \Delta f / \varepsilon$ for varying values of ε to achieve ε-LDP.

We report the expected guesswork and AUC for each attack as well as 95% confidence intervals (CI). To compute confidence intervals, we sampled 100 bootstrap samples of 10,000 images (all encoded by a single T) from the MIMIC-CXR training set. Our 100 bootstraps consisted of 10 random data samples (of 10,000) across 10 random T.

Privacy with Real Labeling Functions In practice, the encoded images are released with encoded labels to enable model development on tasks of interest. Using this additional knowledge, attackers may be able to better re-identify private data. To evaluate the privacy of *Syfer* encodings when released with public labels, we trained the attackers to re-identify raw images given access to (raw image, raw label) pairs and (obfuscated image, obfuscated label) pairs. To highlight the importance of *Syfer*'s label encoding scheme T^Y in this scenario, we also train attackers on an ablation of *Syfer* which does not encode the labels and releases (obfuscated image, raw label). This corresponds to using only *Syfer*'s neural encoder T^X. In particular, this characterizes the risks of using *Syfer* without first balancing disease labels, as that would negate the effectiveness of T^Y.

We performed this attack independently per diagnosis. We implemented the instance encoder E^{ins} of our attacker as an SAU, our self-attention module, and represented the disease label an additional learned 256 dimensional input token for E^{ins}. As before, our attackers were trained for 500 epochs, and evaluated on the MIMIC-CXR training set. We report the expected guesswork and AUC for each attack as well as 95% confidence intervals. To compute confidence intervals, we sampled 100 random T and encoded the whole class-balanced MIMIC-CXR training set for each sampled T.

Utility Evaluation We evaluated the utility of an encoding scheme on the MIMIC-CXR dataset by measuring the ROC-AUC of diagnosis models trained using its encodings. We compared the utility of *Syfer* to a plaintext baseline (i.e. using raw data), which provides us with a utility upper bound. To isolate the impact of training *Syfer*'s obfuscator layers on utility, we also compared the utility of *Syfer* to *Syfer*-Random. We also computed the utility of our differential privacy baseline, DP-Image with a ε values of ∞, 128 and 32 (lower ε values lead to an AUC of 0.50). For each encoding scheme, we experimented with different classifier architectures (e.g. SAU vs ResNet-18), dropout rates and weight decay, and reported the test AUC for the architecture that achieved the best validation AUC.

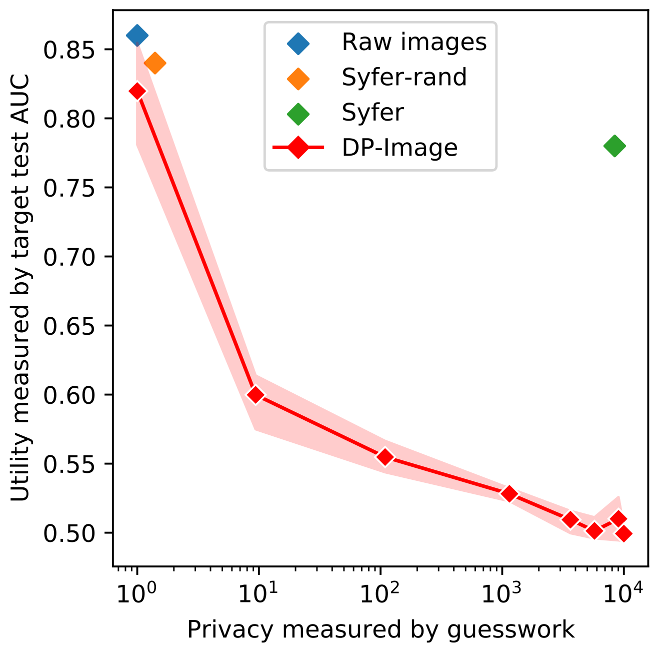
4.7 Main privacy-utility results

Generalized Privacy We report our generalized privacy results, which consider re-identification attacks on the unlabeled MIMIC-CXR dataset, in Table 4.1, with higher guesswork and lower ReID AUC denoting increased privacy. While *Syfer* was trained to maintain privacy against an SAU-based attacker on the NIH training set, we found that its privacy generalized to a held-out dataset, MIMIC-CXR, and held-out attack architectures (e.g. ResNet-18 and ViT). *Syfer* obtains a guesswork of 8411 (95% CI

Figure 4-6: Utility for chest X-ray prediction tasks across different encoding schemes.

	E	C	C	A	*Avg*
Raw data	0.91	0.78	0.89	0.85	0.86
Encoded data					
∞-DP	0.87	0.76	0.84	0.81	0.82
128-DP	0.62	0.56	0.61	0.61	0.60
32-DP	0.53	0.55	0.51	0.53	0.53
Syfer-Rand	0.89	0.75	0.86	0.84	0.84
Syfer	0.82	0.69	0.81	0.78	0.78

(a) All metrics are ROC AUCs across the MIMIC-CXR test set. Guides of abbreviations for medical diagnosis: (E)dema, (Co)nsolidation, (Ca)rdiomegaly and (A)telectasis.



(b) Privacy-utility tradeoff

Table 4.1: Privacy evaluation of different encoding schemes against an SAU based attacker (unless specified) on the unlabeled MIMIC-CXR dataset. For *Syfer*, only T^X is used. Our DP baseline is DP-Image, using laplacian noise parametrized by $b = \Delta f/\varepsilon$, with $\Delta f = 137.772$. Metrics are averages over 100 trials using 10,000 samples each, followed by 95% confidence intervals (CI). AUC 95% confidence intervals are all within +/- 0.01 and were omitted for brievety.

Encoding	Guesswork	ReId AUC
Dauntless	$1.0\ (1, 1)$	1.00
InstaHide	$1.0\ (1, 1)$	1.00
Syfer-Random	$1.7\ (1, 4)$	0.99
Syfer (T^X only)		
vs SAU	$8476\ (1971, 20225)$	0.50
vs ViT	$8411\ (5219, 12033)$	0.50
vs Resnet-18	$10070\ (9871, 10300)$	0.50

Encoding	Guesswork	ReId AUC
∞-DP	$1\ (1, 1)$	1.00
128-DP	$9.40\ (1, 31)$	0.87
64-DP	$109\ (3.9, 348)$	0.71
32-DP	$1146\ (86, 3403)$	0.60
16-DP	$3636\ (36, 10605)$	0.55
8-DP	$5732\ (443, 15257)$	0.51
4-DP	$9037\ (453, 28882)$	0.50

Table 4.2: Privacy evaluation of *Syfer* when released with different diagnosis labels in MIMIC-CXR dataset, using label encoding or not (ablation). Metrics are averages over 100 trials using 10,000 samples each, followed by 95% CI.

	Syfer (using T^X and T^Y)		*Syfer* no label encoding (T^X only)	
Diagnosis	Guesswork	ReId AUC	Guesswork	ReId AUC
Edema	$3617\ (94, 11544)$	$0.50\ (0.49, 0.51)$	$47\ (12, 83)$	$0.76\ (0.76, 0.76)$
Consolidation	$1697\ (83, 5297)$	$0.55\ (0.53, 0.57)$	$36\ (2, 104)$	$0.76\ (0.76, 0.76)$
Cardiomegaly	$9834\ (2072, 15766)$	$0.51\ (0.49, 0.53)$	$42\ (17, 57)$	$0.75\ (0.75, 0.75)$
Atelectasis	$13189\ (2511, 28171)$	$0.50\ (0.48, 0.52)$	$80\ (65, 98)$	$0.75\ (0.75, 0.75)$

5219, 12033) and a ReId AUC of 0.50 (95% CI 0.49, 0.51), and achieves empirical privacy comparable to an ε-DP baseline with $\varepsilon = 4$ which obtains a guesswork of 9037 (95% CI 453, 28882) and a ReId AUC of 0.50 (95% CI 0.49, 0.51). We recall that a guesswork of 9,999 corresponds to guessing randomly in this evaluation, as described in Appendix 4.4. In contrast, the InstaHide and Dauntless baselines could not defend against re-identification attacks obtaining both a guesswork of 1.0 (95% CI 1, 1).

Privacy with Real Labeling Functions We evaluated the privacy of releasing *Syfer* encodings with different public labels in Table 4.2. Releasing *raw labels* resulted in significant privacy leakage with guessworks ranging from 36 (95% CI 2, 104) to 80 (95% CI 65, 98) for Consolidation and Atelectasis respectively. In contrast, when labels are protected using *Syfer*'s label encoding scheme and released alongside the image encodings, *Syfer* maintains privacy across all diagnoses tasks, with guessworks ranging from 1697 (95% CI 83, 5297) to 13189 (95% CI 2511, 28171) for Consolidation and Atelectasis respectively.

Utility Evaluation We report our results in predicting various medical diagnosis from X-rays in Table 4-6a. Models built on *Syfer* obtained an average AUC of 0.78, compared to 0.86 by the plaintext baseline and 0.84 by the *Syfer*-Random baseline. In contrast, Image-DP obtained average AUCs of 0.60, and 0.53 when using a ε values of 128 and 32 respectively. *Syfer* obtained a 25 point average AUC improvement over DP-Image while obtaining better privacy.

Additional Experiments To validate the applicability of our method beyond medical imaging for multi-channel images, we evaluate *Syfer* on CIFAR10 [82]. The results in 4.9 show that *Syfer* outperforms DP-Image, improving the utility AUC by 30 points, while being more private.

4.8 Additional utility analysis

In Figure 4-7, we plot the learning curves of *Syfer*, *Syfer*-Random and our plaintext baselines when training on fractions $\frac{1}{32}$, $\frac{1}{16}$, $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$ and 1 of the data.

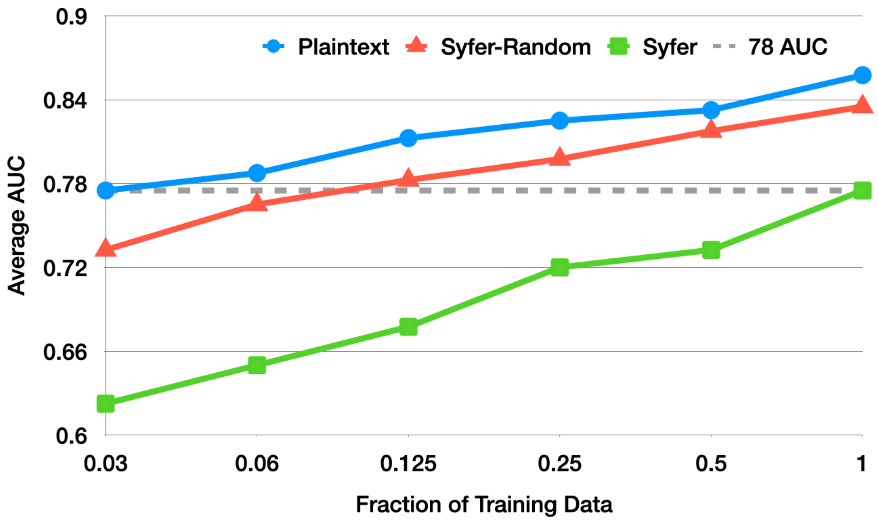


Figure 4-7: Average AUC on MIMIC-Test set when training with different fractions of the data when using *Syfer*, *Syfer*-Random, and Plaintext encodings.

We find that it takes plaintext models $\frac{1}{2}$ of the training data to reach the full performance of *Syfer*-Random, indicating that using a random *Syfer* architecture harms sample complexity. *Syfer*, which achieves strong privacy, requires more data to achieve the same utility, with *Syfer*-Random achieving the same average AUC when using less than $\frac{1}{8}$ of the data and Plaintext achieving the same performance when using $\frac{1}{32}$.

4.9 Imagenette and CIFAR10 experiments

We report additional experiments using Imagenette [66] (a subset of ImageNet [33]) and CIFAR10 [82]. For pretraining, we use the whole set of 13394 images from Imagenette. We resize the images so that the shortest dimension is 32 pixels, and crop the other dimension around the center to get a 32×32 image. To evaluate privacy and downstream utility of Syfer, we split the CIFAR10 dataset into 40000 train images, 10000 validation images and 10000 test images.

We follow the experimental procedure described in Section 4.6. In particular, for privacy experiments, we use the whole CIFAR train set to evaluate an attacker's guesswork. For utility experiments, we consider each label separately in a one-vs-rest binary classification and downsample the dataset such that classes are balanced.

We report measure privacy and utility for raw images, DP baselines, Syfer and Syfer-random in Tables 4.3, 4.4 and Figure 4-8. The experimental results follow the same trend as the X-ray experiments: *Syfer* achieves better privacy than a 2-DP-Image baseline while maintaining significantly better utility (30 AUC points on average).

Table 4.3: Privacy evaluation of different encoding schemes against an SAU-based attacker (unless specified) on the unlabeled CIFAR dataset. For *Syfer*, only T^X is used. Our DP baseline is DP-Image, using laplacian noise parametrized by $b = \Delta f/\varepsilon$, with $\Delta f = 815.1442$. Metrics are averages over 100 trials using 10,000 samples each, followed by 95% confidence intervals (CI).

Encoding	Guesswork	ReId AUC
Raw data	1.0 $(1, 1)$	1.00
∞-DP	1.0 $(1, 1)$	1.00
32-DP	300 $(13, 795)$	0.69
16-DP	3013 $(194, 8137)$	0.58
8-DP	6055 $(422, 19630)$	0.53
4-DP	8786 $(350, 28920)$	0.51
2-DP	6527 $(38, 17599)$	0.50
1-DP	9943 $(910, 26124)$	0.50
Syfer-Random	3.2 $(1, 10)$	0.99
Syfer		
vs SAU	12284 $(840, 43894)$	0.44
vs Resnet-18	11752 $(599, 36665)$	0.53
vs ViT	9999 $(9999, 9999)$	0.51

4.10 Qualitative results

In Figure 4-9, we visualize the impact of *Syfer* and DP-Image encodings when using different amounts of noise (parametrized by the diversity parameter b). Each row represents a different image. The *raw_x* column are raw images. The *DP-image no noise* column is the reconstructed image obtained with the trained auto-encoder that is used for the DP-Image baseline. We visualize the reconstructed images when varying amounts of noise are added.

Table 4.4: Utility on CIFAR10 as one-vs-rest binary classification tasks. All metrics are ROC AUCs across the CIFAR10 test set.

Encoding	0	1	2	3	4	5	6	7	8	9	*Avg*
Raw data	0.917	0.929	0.931	0.923	0.926	0.926	0.925	0.927	0.928	0.931	0.926
∞-DP	0.875	0.899	0.895	0.893	0.896	0.896	0.901	0.902	0.888	0.891	0.894
32.0-DP	0.514	0.508	0.514	0.507	0.513	0.508	0.520	0.519	0.494	0.513	0.511
16.0-DP	0.510	0.509	0.497	0.504	0.490	0.523	0.517	0.519	0.506	0.515	0.509
8.0-DP	0.483	0.497	0.511	0.491	0.506	0.502	0.524	0.498	0.501	0.497	0.501
4.0-DP	0.505	0.497	0.513	0.507	0.489	0.488	0.516	0.502	0.517	0.497	0.503
2.0-DP	0.515	0.509	0.484	0.524	0.511	0.501	0.486	0.499	0.495	0.509	0.503
1.0-DP	0.502	0.506	0.502	0.522	0.493	0.509	0.499	0.488	0.499	0.493	0.501
Syfer-Random	0.870	0.878	0.874	0.879	0.874	0.876	0.876	0.871	0.870	0.870	0.874
Syfer	0.801	0.810	0.806	0.799	0.813	0.815	0.802	0.790	0.790	0.797	0.802

The *Syfer* column shows encodings obtained when applying *Syfer*'s neural encoder for a specific choice of private weights θ_{key}: those are representative of the released images. We then train a decoder D_T for a specific choice of private weights θ_{key}. During training, the decoder has access to parallel data (raw image, encoded *Syfer* image). We visualize the decoded images in the *Syfer decoded* column. Note that this scenario would require Alice to have stored the correspondences between raw images and encoded images. We emphasize that storing these parallel datasets are never needed in our scheme: once Alice has encoded her training data samples with her private key, she can shuffle the resulting encoded dataset and forget the actual correspondences. All Alice needs for inference on new data is access to her private key (which she should keep private). In the event of a data breach, the attacker would only have access to nonparallel datasets, which is the scenario covered by our threat model. We note that if the private key is compromised, then the attacker can re-identify the private data. This is similar to traditional cryptographic systems where identifying the private key leads to total failure.

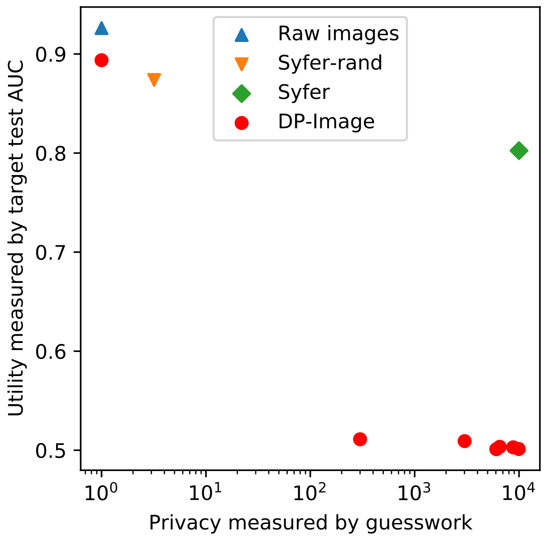


Figure 4-8: Privacy-utility tradeoff on CIFAR10. Privacy is measured as the expected guesswork. Utility is the average across 10 tasks of the test AUC achieved for the best dev AUC.

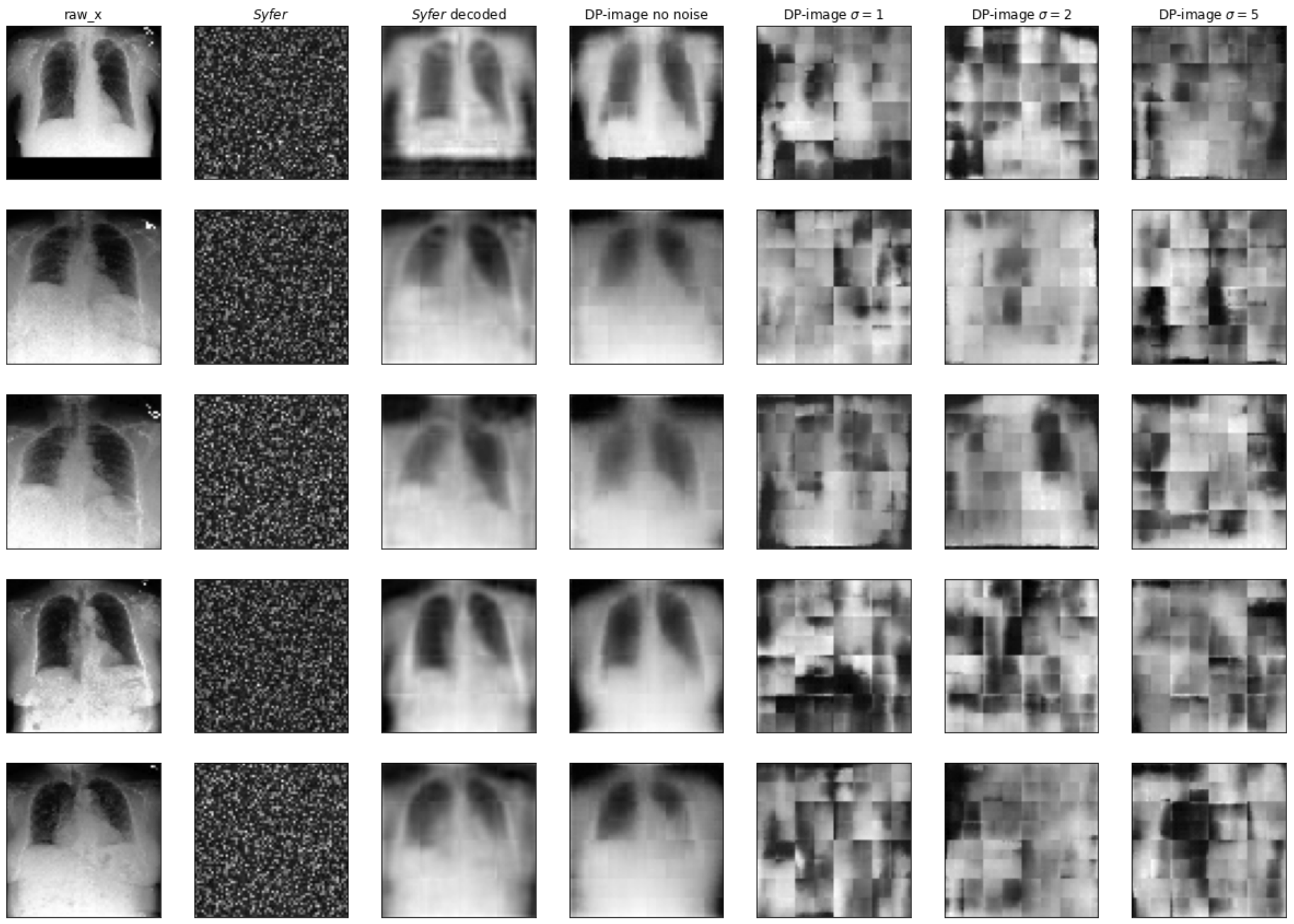


Figure 4-9: Vizualisations of raw images, *Syfer* encodings, decoded *Syfer* encodings, and DP-Image encodings. *Syfer* encodings were obtained after applying the T^X part of *Syfer* to raw images. Decoded *Syfer* encodings are obtained by a model D_T trained on a set of parallel training data (plaintext attack). DP-Image encodings are shown with varying amount of noise.

4.11 Conclusion

4.11.1 Summary of contributions

We propose *Syfer*, an encoding scheme for releasing private data for machine learning model development while preventing raw data re-identification. *Syfer* uses trained obfuscator layers and random neural networks to minimize the likelihood of re-identification, while encouraging the (almost) invertibility of the overall transformation. To evaluate private encoding schemes, we define a guesswork-based evaluation framework, where the privacy of an encoding scheme is the number of attacker guesses to re-identify a single raw image. We introduce an efficient contrastive algorithm to estimate the privacy of arbitrary encoding schemes on given datasets. In experiments on MIMIC-CXR, a large chest X-ray benchmark, we show that *Syfer* obtains strong privacy across held-out attackers, obtaining an average guesswork of 8411, whereas prior encoding schemes like Dauntless [169], InstaHide [67] did not meet our privacy standard, obtaining guessworks of 1. which rivals an ε-LDP baseline with $\varepsilon = 4$. While differential privacy baselines can achieve privacy with enough noise, we found this came with a massive loss of utility, with DP-Image obtaining an average AUC of 0.53 for a guesswork of 1146 at $\varepsilon = 32$. In contrast, models built on *Syfer* encodings approached the utility of our plaintext baseline, obtaining an average AUC of 0.78 compared to 0.86 by the plaintext model.

4.11.2 Limitations and future work

While this chapter demonstrates the potential of our method for de-identification using a radiology dataset, there are certain assumptions and limitations that need to be considered for real-world applications.

Threat model Our method assumes that the attacker has knowledge of whether a target individual is present in the released dataset, which is often referred to as *prosecutor risk.* However, in some scenarios, captured by *journalistic risk,* it may be safe to assume that the attacker does not have such knowledge. In such cases, sophisticated methods like *Syfer* may not be necessary.

Additionally, our threat model assumes a computationally unbounded adversary. However, in practice, we rely on model-based attackers for both the development and evaluation of our method. In the implementation, we assume that the compute power of the model builder (Bob) is not worse than that of the attacker (Eve). It is important to consider that more powerful models may lead to more successful attacks on our *Syfer*.

Formal privacy guarantees While we demonstrate the effectiveness of *Syfer* in achieving strong privacy, we do not provide formal privacy guarantees in this work. Future research can explore deriving theoretical bounds on privacy based on specific assumptions about the data distribution, attacker capabilities, and attack strategies. Developing provable privacy guarantees for *Syfer* will contribute to its adoption and deployment in privacy-sensitive domains.

Generalizability and robustness Although we demonstrate that our method generalizes to unseen datasets, multiple unseen attackers, and various diagnosis tasks, this does not guarantee its generalizability to arbitrary datasets. Further research is needed to study the privacy impact of domain shifts, and it is crucial to evaluate the privacy of our method on specific datasets before releasing them. However, it should be noted that delaying data release until privacy evaluation is conducted may itself result in privacy leakage. Real-world datasets are subject to domain shifts over time, and attackers' capabilities may evolve. Future work should investigate the privacy robustness of *Syfer* under domain shifts and consider mechanisms to adapt the encoding scheme to evolving attack strategies.

Scalability and efficiency Our experiments primarily focus on the MIMIC-CXR dataset, which is a large chest X-ray benchmark of grayscale images. Appendix 4.9 demonstrates the applicability of our method to multi-channel images. However, the scalability and efficiency of *Syfer* on larger and more diverse datasets require further investigation. In our work, we downsampled images to a resolution of 64x64, as it struck a good balance between utility and privacy. However, the performance and resource requirements of *Syfer* may vary when applied to datasets with higher resolutions or different modalities.

Development challenges The use of privately encoded training data may introduce challenges for model development, as it limits certain standard practices such as manual error inspection or traditional data augmentation. Addressing these challenges would likely require the development of new techniques specifically tailored to privacy-preserving model training.

Overall, continued innovation in the architecture design and training of our method can lead to further improvements in the trade-off between privacy and utility. However, it is crucial for data owners to carefully evaluate the privacy implications of our method on their own datasets before releasing them to ensure compliance with privacy regulations.

Chapter 5

Conclusion

5.1 Summary of contributions

In this thesis, we proposed methods to enhance the reliability and user control over the outputs of generative models, empowering users to have greater confidence in the generated predictions and ensuring adherence to predefined constraints.

In Chapter 2, we introduced the Blank Language Model (BLM) as a flexible approach for text generation. BLM allows for the generation of a variable number of tokens consistent with the given context by filling in partially specified text with one or more blanks. Through extensive experiments on various text rewriting tasks, including ancient text restoration, and style transfer, we have demonstrated the effectiveness of our model for text infilling tasks.

In Chapter 3, we proposed a novel approach to conformal prediction for generative language models. By leveraging the idea of prediction sets, we developed a calibration-based stopping rule for sampling different outputs from language models. This allowed us to construct a set of candidate responses with statistical guarantees of containing at least one acceptable answer. Simultaneously, we introduced a rejection rule to set size by removing low-quality candidates from the output set. Through theoretical analysis and empirical evaluations on tasks such as open-domain question answering, text summarization, and radiology report generation, we demonstrated the effectiveness of our approach.

In Chapter 4, we utilized image generation techniques to address the challenge of sharing private data for machine learning development while preserving data privacy. We presented *Syfer*, a method that trains obfuscator layers to balance privacy and invertibility constraints. When combined with random neural networks, *Syfer* enables a computationally-limited data owner to outsource machine learning training to untrusted third parties while preventing the shared data from being re-identifiable. We have showcased the applicability of our method on a radiology dataset.

5.2 Limitations and future work

Throughout this thesis, we have made several contributions, each with its own limitations. These limitations should be taken into consideration for future research and practical applications. To gain a better understanding of these limitations, we encourage readers to refer to the respective discussions in Section 2.6.2 (for Chapter 2), Section 3.8.2 (for Chapter 3), and Section 4.11.2 (for Chapter 4). We provide a summary of these limitations below for clarity.

In Chapter 2, our focus was on BLM (Blank Language Model) for text generation. However, there are still areas that need to be explored in future work. For instance, it would be valuable to investigate the potential of our method for representation learning by comparing it to similar pretrained methods through training BLM at scale. Additionally, it would be interesting to extend BLM to the sequence-to-sequence setting and adapt the framework to other sequence modeling tasks.

Chapter 3 delved into conformal prediction for language generation tasks, with a specific emphasis on controlling miscoverage and optimizing for prediction set size. To enhance our framework and showcase its potential in various settings, further exploration of alternative objectives, admission functions, quality functions, and aggregation functions is warranted. Another promising avenue for research would involve experimenting with modifying the underlying language model to directly enforce desirable properties. For example, guiding the decoding process to promote diversity or applying our framework to BLM to influence the generative process with more control.

In Chapter 4, we introduced *Syfer* and demonstrated its value for de-identification on a radiology dataset. However, it is important to highlight certain assumptions and limitations that should be considered for real-world applications. The achieved privacy of *Syfer* is not theoretically guaranteed, and additional research is required to evaluate the privacy impact on specific datasets. It is crucial to assess the generalizability of *Syfer* to arbitrary datasets and investigate the privacy implications of domain shifts.

By addressing these limitations and exploring the avenues for future work outlined in each chapter, we can further advance the research and practical applications of methods for reliable and flexible generation with guarantees.

Appendices

Appendix A

Blank Language Model

A.1 Implementation Details for Text Infilling Baselines

A.1.1 Insertion Transformer

We implement the Insertion Transformer in our own framework, using the same Transformer encoder module as for BLM and replacing the prediction layers by Insertion Transformer's mechanism. The canvas is also generated according to the training procedure of Insertion Transformer. See Figure 2-5.

A.1.2 Masked Language Model

We use the `RobertaForMaskedLM` architecture in the Transformers library for MLM [162, 98].

At test time, the model is given an easier version of the text infilling task where blanks are expanded into sequences of $\langle \text{mask} \rangle$ tokens of the target length (or equivalently, the model uses an oracle to predict the length of the infilling).

We experiment with three decoding strategies: (1) one-shot: the model predicts all masks simultaneously (2) left-to-right: the model fills in the masks from left to right (3) confident-first: the model fills one mask at a time that has the highest score.

We report results for the confident-first strategy which has the best performance.

See Figure 2-6.

A.1.3 BERT+LM

We use the `bert-base-uncased` model as served by the Transformers library [162, 35]. The left-to-right language model is a Transformer decoder to predict tokens in a blank. Its input word embedding is concatenated with BERT's output in the blank position at each time step. See Figure 2-7.

A.1.4 Seq2seq-full and Seq2seq-fill

For both seq2seq baselines, we use Fairseq's `transformer_iwslt_de_en` architecture [114]. To generate training data, we apply the blanking procedure to the input dataset and generate k copies of each sentence with different masks. We experiment with $k \in \{1, 10, 100\}$ and report the best performance, obtained by $k = 10$.

A.2 Monte-Carlo Estimate of Perplexity

For a sentence x of length n, we estimate $p(x;\theta)$ in Eq. (2.5) with m samples:

$$X_m = \frac{n!}{m} \sum_{i=1}^{m} p(x, \sigma_i; \theta)$$

where σ_i's are randomly sampled orders.

Note that X_m is an unbiased estimate of $p(x;\theta)$:

$$\mathbb{E}[X_m] = p(x;\theta)$$

The estimated PPL is accordinly:

$$Y_m = X_m^{-1/n}$$

Since $z^{-1/n}$ is a convex function of z,

$$\mathbb{E}[Y_m] = \mathbb{E}[X_m^{-1/n}] \geq \mathbb{E}[X_m]^{-1/n} = p(x;\theta)^{-1/n}$$

i.e., the expectation of the estimated PPL $\geq$ the actual PPL. As m increases, the variance of X_m decreases, and the inequality becomes tighter.

Hence, we will observe that as m increases, the estimated PPL becomes smaller and converges to the real PPL.

Appendix B

Conformal Language Modeling

B.1 Pareto testing

We recall the Pareto Testing method introduced by [85].

B.2 Admission functions

Note that, as briefly discussed in §3.4.1, $\hat{\lambda}$ is also valid if a conservative, "approximate" admission function $\bar{A}_i$ is used in place of a "complete" A_i during calibration.

Corollary B.2.1 (Conservative sampling-based LTT)**.** Suppose that over $\mathcal{D}_{\text{cal}}$ we let $L_i(\lambda) = \mathbf{1}\{\nexists y \in \mathcal{C}_\lambda(X_i) \colon \bar{A}_i(y) = 1\}$ where $\bar{A}(y) \le A(y)$, $\forall y \in \mathcal{V}^*$. Then $\mathcal{C}_{\hat{\lambda}}(X_{\text{test}})$ still satisfies Eq. equation 3.1.

B.3 Additional experimental details

In this section, we provide additional details regarding the experiments conducted for the three tasks discussed in Section 3.5.

B.3.1 Radiology report generation

Dataset For the radiology report generation experiment, we utilized the labeled MIMIC-CXR and MIMIC-CXR-JPG datasets [71]. The MIMIC-CXR dataset can be accessed at https://physionet.org/content/mimic-cxr/2.0.0/ under the PhysioNet Credentialed Health Data License 1.5.0. Similarly, the MIMIC-CXR-JPG dataset is available at https://physionet.org/content/mimic-cxr-jpg/2.0.0/ under the same license.

We start with the standard splits prescribed in MIMIC-CXR-JPG. However, we further divide the training set into a train set and a dev set using a 0.9/0.1 ratio. The train set is used for training the model, using the validation set for early stopping. We then exclusively use the dev set for conformal prediction experiments. Subsequently, we filtered the dataset to include only anterior to posterior (AP) or posterior to anterior (PA) views and retained only one image per report. Furthermore, we removed examples where the report did not start with the phrase "FINAL REPORT" as these reports often contained a summary of the findings at the beginning, inadvertently leaking the answer we aimed to generate with the model. Table B.1 provides a statistical overview of the resulting dataset.

Table B.1: Dataset statistics for preprocessed MIMIC-CXR. The split indices and preprocessing scripts are available within our code release.

Split	**Train**	**Dev**	**Validate**	**Test**
Number of Images	176,078	19,658	1,594	2,799
Number of Studies	176,078	19,658	1,594	2,799
Number of Patients	54,482	6,053	463	286

Each image was resized and cropped to a resolution of 224x224. Following prior methodology [110], we split each report into a *prompt* part and a *findings* part (which may also contain the *impressions* section) by identifying one of the following phrases: "FINDINGS AND IMPRESSION", "FINDINGS" or "IMPRESSION".

Model The image encoder used in our experiment was a Vision Transformer (ViT) model pretrained on ImageNet-21k at a resolution of 224x224. Specifically, we utilized the `google/vit-base-patch16-224-in21k` model available in the Transformers library [162]. The text decoder was a GPT2-small model (`gpt2` on HuggingFace). We trained the model with a batch size of 128 distributed over 8 GPUs, resulting in a batch size of 16 per GPU. The AdamW optimizer was employed with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The learning rate was set to 5×10^{-5}. The training process consisted of 10 epochs, and the total training time on 8 RTX A6000 GPUs was approximately 11 hours.

Generations Candidate reports were sampled from the model using default arguments from the Transformers library, i.e. top_k = 50, top_p = 1.0 and temperature = 1. Each generated report is then evaluated using a trained CheXbert model [138]. The CheXbert model is available at https://stanfordmedicine.box.com/ under the Stanford Academic Software License. The CheXbert model labels each report for 14 conditions, assigning one of the following labels: "Blank," "Positive," "Negative," or "Uncertain."

To determine the admission of a candidate report, we compare it with a reference (human) report from the MIMIC dataset. If the candidate report matches all 14 labels of the reference report, the admission function returns 1; otherwise, it returns 0.

Components We define a component as a sentence delimited by a period. The component-level admission function is defined based on how well a sentence"almost matches" one of the reference sentences. Two sentences are considered to "almost match" if their `ROUGE-L` score is above 0.4. If a sentence almost matches a reference sentence, the component-level admission function returns 1; otherwise, it returns 0.

Component classifier For the MIMIC-CXR dataset, we also train a classifier to produce a conformity score for pairs (x, e), where x represents an image and e is a sentence. This classifier is used in §3.6 as a possible implementation of the component-level score function $\mathcal{F}^C$.

To train this classifier, we reserve a subset of the dev set. A sentence e is labeled as acceptable if its `ROUGE-L` score with any reference sentence exceeds a threshold of 0.4. The classifier is trained using these labeled pairs to learn the conformity score for image-sentence pairs.

B.3.2 Open-domain question answering

We use the TriviaQA [75] dataset available at https://nlp.cs.washington.edu/triviaqa/ under the Apache License Version 2.0.

To generate candidate responses, we used LLaMA-13B [147]. We considered the closed-book setting, where the model does not have access to supporting text for answering the questions. We performed experiments in the few-shot setting by providing 32 example question-answer pairs sampled from the training set.

A truncated prompt used for generating answers on the TriviaQA dev set is reproduced as an illustration in Figure B-1. Please note that the actual prompt used in the experiment contains 32 question-answer pairs.

For generating answers in the open-domain question answering task, we use the default Transformers parameters reported in the previous section. We extract an answer by considering the text until the first line break, comma, or period is encountered. We then normalize the answers: this involves converting the generated answers to lowercase, removing articles, punctuation, and duplicate whitespace. Generated answers are then compared using the exact match metric: an answer is considered correct only if it matches the provided answer exactly.

B.3.3 Length-normalization

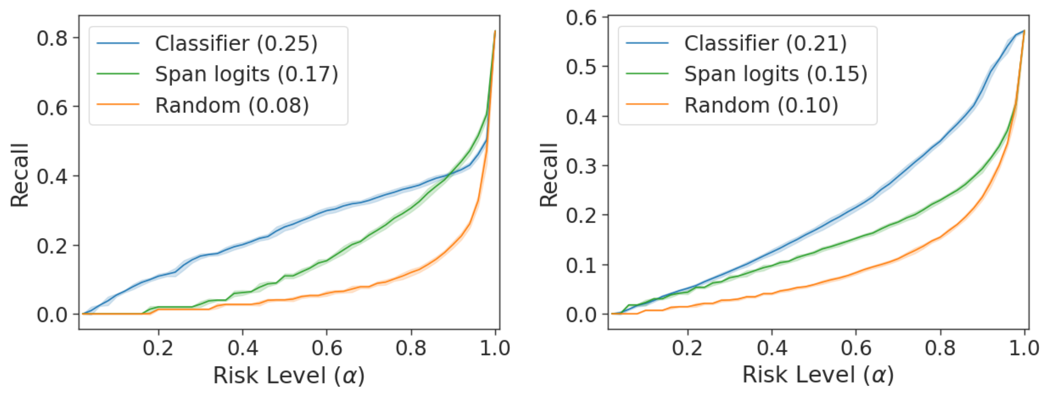
For all tasks, we apply length-normalization [165] to the model logits, i.e. we compute:

$$\mathcal{Q}(x, y_k) = \exp\left(\frac{\log p_\theta(y_k|x)}{lp(y_k)}\right)$$

where

$$lp(y) = \frac{(5+|y|)^{0.6}}{(5+1)^{0.6}}.$$

B.4 Additional results



(a) MIMIC-CXR (b) CNN/DM

Figure B-2: Conformal component selection results for $\mathcal{C}_{\gamma}^{\text{inner}}$ as a function of α. We report the recall achieved by $\mathcal{C}_{\gamma}^{\text{inner}}$, which we want to maximize. We also report the AUC over α.

We describe another metric useful to characterize the effectiveness of the components identified by our component selection method.

Given an input x and a component set $\mathcal{C}\gamma^{\text{inner}}(x)$, we compute the recall by counting the number of reference sentences that "almost match" at least one element in $\mathcal{C}\gamma^{\text{inner}}(x)$. We then divide this count by the total number of reference sentences for that particular example. This gives us a measure of how much of the human reference is covered by the selected components. To obtain the expected recall, we average the recall values over all examples. The expected recall is reported in Figure B-2.

In particular, we observe that component sets generated using scoring functions based on an auxiliary Classifier outperform uncertainty measures based solely on the span logits provided by the model.

```
Answer these questions

Q: Which American-born Sinclair won the Nobel Prize for Literature in 1930?
A: Sinclair Lewis
Q: Where in England was Dame Judi Dench born?
A: York
Q: In which decade did Billboard magazine first publish and American hit chart?
A: 30s
Q: From which country did Angola achieve independence in 1975?
A: Portugal
Q: Which city does David Soul come from?
A: Chicago
Q: Who won Super Bowl XX?
A: Chicago Bears
Q: Which was the first European country to abolish capital punishment?
A: Norway
Q: In which country did he widespread use of ISDN begin in 1988?
A: Japan
Q: What is Bruce Willis' real first name?
A: Walter
Q: Which William wrote the novel Lord Of The Flies?
A: Golding
Q: Which innovation for the car was developed by Prince Henry of Prussia in 1911?
A: Windshield wipers
Q: How is musician William Lee Conley better known?
A: Big Bill Broonzy
Q: How is Joan Molinsky better known?
A: Joan Rivers
...
```

Figure B-1: Truncated replication of the prompt used to generate answer on the TriviaQA dev set. The actual prompt contains 32 question-answer pairs.

Appendix C

Neural Obfuscation for De-identification

C.1 Dataset Statistics

We leveraged the NIH training set for training *Syfer*, and leveraged the unlabeled MIMIC-CXR training set for all generalized privacy evaluation. To evaluate utility and privacy with real labeling functions, we use the labeled subsets of the MIMIC-CXR dataset. The labeled MIMIC-CXR training and validation sets were filtered to be class balanced, by assigning random one negative control for each positive sample. The number of images per dataset is shown in Table C.1 The NIH dataset is available at https://nihcc.app.box.com/v/ChestXray-NIHCC. The MIMIC dataset is available at https://physionet.org/content/mimiciii-demo/1.4/ under the Open Data Commons Open Database License v1.0.

Table C.1: Dataset statistics for all datasets. The training and development sets of MIMIC CXR Edema, Consolidation, Cardiomegaly and Atelecatasis were filtered to contain one negative control for each positive sample. Guides of abbreviations for medical diagnosis: (E)dema, (Co)nsolidation, (Ca)rdiomegaly and (A)telectasis.

Dataset	Train	Dev	Test
Unlabeled			
NIH	40365	NA	NA
MIMIC-CXR	57696	NA	NA
Labeled			
MIMIC-CXR E	3660	1182	12125
MIMIC-CXR Co	1120	375	11031
MIMIC-CXR Ca	11724	3876	12791
MIMIC-CXR A	2164	3992	12129

C.2 *Syfer* Implementation Details

We experimented with multiple variants of the hyperparameters below and chose the combination that led to produced the best tradeoffs. In all experiments, we used a patch size of 16x16 pixels and 5 *Syfer* blocks. Earlier analysis showed that a larger patch size lead to better utility at the cost of worse privacy. Similarly, we saw that using 10 *Syfer* blocks increases privacy at the cost of worse utility.

We implemented our random neural networks as linear layers, followed by a SeLU nonlinearity and layer normalization. We saw that SeLU tends to outperform ReLU, Tanh and Sigmoid when used in random neural network-based schemes without an obfuscator. Layer normalization is used as a instance-level alternative to batch normalization. All random linear layers weights were sampled from a unit Gaussian (which does better than Xavier initialization and no worse than Orthogonal initialization), and we used separate random networks per patch. Our full *Syfer* architecture has 12.9M parameters, of which 6.6M are learned obfuscator parameters and 6.3M are random neural layer parameters.

We used a batch size of 128, the Adam optimizer and a learning rate of 0.001 for training *Syfer* and our estimators. We trained *Syfer* for 50,000 steps on the NIH training set to maximize the re-identification loss and minimize the reconstruction loss, with $\lambda_{\text{reid}} = 1$, $\lambda_{\text{rec}} = 10$. Our training regime alternates one step of obfuscator training with one step estimator training. We experimented with varying the number of estimator training in between steps of obfuscator training, as well as jointly upgrading the gradients of all modules and saw no substantial difference. The total training time is 2 hours and 50 minutes, using a single A100 GPU and is fully reproducible in our code release. We include additional timing benchmarks in Table C.2.

Table C.2: Timing benchmarks for using Syfer and our baselines.

Task	Runtime
Using 40,365 images of NIH Dataset	
(Pre)Training Syfer	2h 50min
(Pre)Training DP-Image's auto-encoder	28m
Using 57,696 images of MIMIC Dataset	
Reading raw images from disk	4s
Reading images and encoding them with Syfer on CPU	3m30s
Reading images and encoding them with DP-Image using a single GPU	12s
Reading images and encoding them with Syfer using a single GPU	13s
Using 11,724 images of downsampled MIMIC Dataset for Cardiomegaly	
Training a Resnet on raw images (single GPU)	11m37s +/- 37s
Training a Resnet on Syfer-encoded images (single GPU)	11m36s +/- 35s

C.3 SAU: Simple Attention Unit

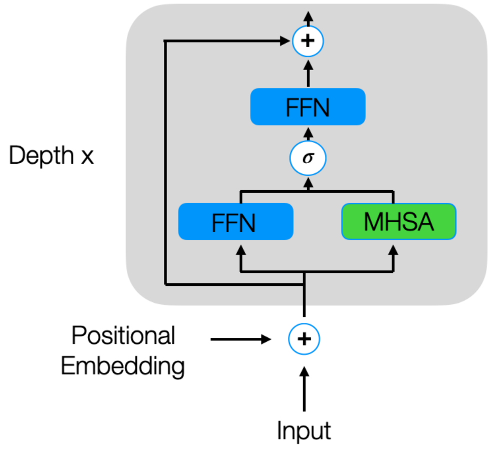


Figure C-1: Simple Attention Unit Architecture. The module uses a learnable gate at each layer to interpolate between leveraging behaving as a feed forward network (FFN) and a multi-headed self attention network (MHSA).

Our Simple Attention Unit (SAU), illustrated in Figure C-1, utilizes a learned gate, α, at each layer to interpolate between acting as a standard feed forward network (FFN) with no attention computation, and a multi-head self-attention (MHSA) network. We found that this allowed for faster and more stable training compared to ViTs[185, 38] in both privacy and utility experiments. To encode patch positions, we leverage a learned positional embedding for each location, following prior work [185, 38]. Each layer of the SAU is composed of the following operations:

$$x_{norm} = \text{BatchNorm}(x)$$

$$h_{ffn} = \text{SELU}(W_{in}x_{norm} + b_{in})$$

$$h_{attn} = \text{MHSA}(x_{norm})$$

$$h = \sigma(\alpha) \times h_{attn} + (1 - \sigma(\alpha)) \times h_{ffn}$$

$$o = \text{SELU}(W_o h + b_o) + x_{norm}$$

Where Multi-head self-attention (MHSA) is defined as:

$$K_i, Q_i, V_i = W_i x_{norm}$$

$$\text{head}_i = softmax(\frac{Q_i K_i^T}{\sqrt{d_k}})V^i$$

$$h_{attn} = \text{BatchNorm}(Concat(head_1, ..., head_h)W_{attn})$$

Where d_k is the dimension of each head, all W and b are learned parameters, and α is a learned gate. α is initialized at -2 for each layer.

C.4 DP-Image Baseline

DP-Image[93] is a differential privacy method based adding laplacian noise to the latent space of auto encoders to achieve differential privacy. We trained our auto-encoder on the NIH training set, with no noise, and apply it with noise on the MIMIC dataset. Our encoder, mapping each 64x64 pixel image x to a 256d latent code z, is composed of six convolutional layers, each followed by a leaky relu activation and batch normalization. Each convolutional layer had a kernel size of 3, a stride of 2. This was then reduced a single code z with global average pooling. Our decoder, which mapped z back to x, consisted of six transposed convolutional layers, each followed by a leaky relu activation and batch normalization. The auto encoder was trained to minimize the mean squared error between the decoded image and the original image.

Bibliography

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[2] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. CCS '16, page 308–318, New York, NY, USA, 2016. Association for Computing Machinery.

[3] Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification, 2022.

[4] Anastasios N. Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. Conformal risk control, 2023.

[5] Anastasios Nikolas Angelopoulos, Stephen Bates, Emmanuel J. Candès, Michael I. Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *ArXiv preprint: 2110.01052*, 2021.

[6] Anastasios Nikolas Angelopoulos, Stephen Bates, Jitendra Malik, and Michael I. Jordan. Uncertainty sets for image classifiers using conformal prediction. In *International Conference on Learning Representations (ICLR)*, 2021.

[7] E. Arikan. An inequality on guessing and its application to sequential decoding. In *IEEE International Symposium on Information Theory*, pages 322–, 1995.

[8] Yannis Assael, Thea Sommerschield, and Jonathan Prag. Restoring ancient text using deep learning: a case study on Greek epigraphy. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6368–6375, Hong Kong, China, November 2019. Association for Computational Linguistics.

[9] Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. *arXiv preprint arXiv:1809.10853*, 2018.

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

[11] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[12] Rina Foygel Barber, Emmanuel J Candes, Aaditya Ramdas, and Ryan J Tibshirani. Predictive inference with the jackknife+. *The Annals of Statistics*, 49(1):486–507, 2021.

[13] Stephen Bates, Anastasios Nikolas Angelopoulos, Lihua Lei, Jitendra Malik, and Michael I. Jordan. Distribution free, risk controlling prediction sets. *ArXiv preprint: 2101.02703*, 2020.

[14] Ahmad Beirami, Robert Calderbank, Mark M. Christiansen, Ken R. Duffy, and Muriel Médard. A characterization of guesswork on swiftly tilting curves. *IEEE Transactions on Information Theory*, 65(5):2850–2871, 2019.

[15] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10. ACM, 1988.

[16] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[17] Bernd Bohnet, Vinh Q. Tran, Pat Verga, Roee Aharoni, Daniel Andor, Livio Baldini Soares, Massimiliano Ciaramita, Jacob Eisenstein, Kuzman Ganchev, Jonathan Herzig, Kai Hui, Tom Kwiatkowski, Ji Ma, Jianmo Ni, Lierni Sestorain Saralegui, Tal Schuster, William W. Cohen, Michael Collins, Dipanjan Das, Donald Metzler, Slav Petrov, and Kellie Webster. Attributed question answering: Evaluation and modeling for attributed large language models, 2023.

[18] Florian Bourse, Michele Minelli, Matthias Minihold, and Pascal Paillier. Fast homomorphic evaluation of deep discretized neural networks. In *Advances in Cryptology*, volume 10993, pages 483–512. Springer, 2018.

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

[20] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. *SIAM J. Comput.*, 43(2):831–871, 2014.

[21] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[22] Maxime Cauchois, Suyash Gupta, Alnur Ali, and John Duchi. Predictive inference with weak supervision, 2022.

[23] Maxime Cauchois, Suyash Gupta, and John C. Duchi. Knowing what you know: valid and validated confidence sets in multiclass and multilabel prediction. *Journal of Machine Learning Research (JMLR)*, 2021.

[24] William Chan, Nikita Kitaev, Kelvin Guu, Mitchell Stern, and Jakob Uszkoreit. Kermit: Generative insertion-based modeling for sequences. *arXiv preprint arXiv:1906.01604*, 2019.

[25] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 11–19. ACM, 1988.

[26] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[27] Victoria Cheng, Vinith M Suriyakumar, Natalie Dullerud, Shalmali Joshi, and Marzyeh Ghassemi. Can you fake it until you make it? impacts of differentially private synthetic data on downstream classification fairness. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 149–160, 2021.

[28] Hyunghoon Cho, David J Wu, and Bonnie Berger. Secure genome-wide association analysis using multiparty computation. *Nature biotechnology*, 36(6):547–551, 2018.

[29] Noam Chomsky. *Syntactic structures.* Mouton & Co., 1957.

[30] Mark M. Christiansen, Ken R. Duffy, Flávio du Pin Calmon, and Muriel Médard. Brute force searching, the typical set and guesswork. In *IEEE International Symposium on Information Theory*, pages 1257–1261, 2013.

[31] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.

[32] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019.

[33] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[34] Shrey Desai and Greg Durrett. Calibration of pre-trained transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 295–302, Online, November 2020. Association for Computational Linguistics.

[35] 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.

[36] Neil Dey, Jing Ding, Jack Ferrell, Carolina Kapper, Maxwell Lovig, Emiliano Planchon, and Jonathan P. Williams. Conformal prediction for text infilling and part-of-speech prediction. *The New England Journal of Statistics in Data Science*, 1(1):69–83, 2022.

[37] Chris Donahue, Mina Lee, and Percy Liang. Enabling language models to fill in the blanks. *arXiv preprint arXiv:2005.05339*, 2020.

[38] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[39] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[40] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. Glm: General language model pretraining with autoregressive blank infilling, 2022.

[41] Flávio du Pin Calmon, Muriel Médard, Linda M. Zeger, João Barros, Mark M. Christiansen, and Ken R. Duffy. Lists that are smaller than their parts: A coding approach to tunable secrecy. In *Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1387–1394, 2012.

[42] C. Dwork. Differential privacy: A survey of results. pages 1–19, 2008.

[43] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[44] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. Recurrent neural network grammars. *arXiv preprint arXiv:1602.07776*, 2016.

[45] Alexander Fabbri, Chien-Sheng Wu, Wenhao Liu, and Caiming Xiong. QAFactEval: Improved QA-based factual consistency evaluation for summarization. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2587–2601, Seattle, United States, July 2022. Association for Computational Linguistics.

[46] William Fedus, Ian Goodfellow, and Andrew M Dai. Maskgan: better text generation via filling in the _. *arXiv preprint arXiv:1801.07736*, 2018.

[47] Adam Fisch, Tal Schuster, Tommi Jaakkola, and Regina Barzilay. Efficient conformal prediction via cascaded inference with expanded admission. In *International Conference on Learning Representations (ICLR)*, 2021.

[48] Adam Fisch, Tal Schuster, Tommi Jaakkola, and Regina Barzilay. Few-shot conformal prediction with auxiliary tasks. In *International Conference on Machine Learning (ICML)*, 2021.

[49] Adam Fisch, Tal Schuster, Tommi Jaakkola, and Regina Barzilay. Conformal prediction sets with limited false positives. In *International Conference on Machine Learning (ICML)*, 2022.

[50] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, Online, November 2020. Association for Computational Linguistics.

[51] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178. ACM, 2009.

[52] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Constant-time machine translation with conditional masked language models. *arXiv preprint arXiv:1904.09324*, 2019.

[53] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In

Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229. ACM, 1987.

[54] Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*, 2016.

[55] Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*, 2017.

[56] Jiatao Gu, Qi Liu, and Kyunghyun Cho. Insertion-based decoding with automatically inferred generation order. *Transactions of the Association for Computational Linguistics*, 7:661–676, 2019.

[57] Jiatao Gu, Changhan Wang, and Junbo Zhao. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, pages 11179–11189, 2019.

[58] Chirag Gupta, Aleksandr Podkopaev, and Aaditya Ramdas. Distribution-free binary classification: prediction sets, confidence intervals and calibration. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[59] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[60] HIPAA. *Health Insurance Portability and Accountability Act of 1996*, 1996.

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

[62] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979.

[63] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

[64] Or Honovich, Roee Aharoni, Jonathan Herzig, Hagai Taitelbaum, Doron Kukliansy, Vered Cohen, Thomas Scialom, Idan Szpektor, Avinatan Hassidim, and Yossi Matias. TRUE: Re-evaluating factual consistency evaluation. In *Proceedings of the Second DialDoc Workshop on Document-grounded Dialogue and Conversational Question Answering*, pages 161–175, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[65] Eliahu Horwitz and Yedid Hoshen. Conffusion: Confidence intervals for diffusion models. 2022.

[66] J. Howard. Imagenette. `https://github.com/fastai/imagenette`, 2019 (accessed July 31, 2022).

[67] Yangsibo Huang, Zhao Song, Kai Li, and Sanjeev Arora. InstaHide: Instance-hiding schemes for private distributed learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4507–4518. PMLR, 13–18 Jul 2020.

[68] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, et al. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 590–597, 2019.

[69] Dongfu Jiang, Bill Yuchen Lin, and Xiang Ren. Pairreranker: Pairwise reranking for natural language generation, 2022.

[70] Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977, 2021.

[71] Alistair EW Johnson, Tom J Pollard, Nathaniel R Greenbaum, Matthew P Lungren, Chih-ying Deng, Yifan Peng, Zhiyong Lu, Roger G Mark, Seth J Berkowitz, and Steven Horng. Mimic-cxr-jpg, a large publicly available database of labeled chest radiographs. *arXiv preprint arXiv:1901.07042*, 2019.

[72] Erik Jones and Jacob Steinhardt. Capturing failures of large language models via human cognitive biases. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[73] James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. Pate-gan: Generating synthetic data with differential privacy guarantees. In *International conference on learning representations*, 2018.

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

[75] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension, 2017.

[76] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. Gazelle: A low latency framework for secure neural network inference. In *Proceedings of the 27th USENIX Conference on Security Symposium*, SEC'18, page 1651–1668, USA, 2018. USENIX Association.

[77] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know. 2022.

[78] Tushar Khot, Ashish Sabharwal, and Peter Clark. Scitail: A textual entailment dataset from science question answering. In *AAAI Conference on Artificial Intelligence*, 2018.

[79] Daniel Kifer and Ashwin Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 193–204, 2011.

[80] D. Ko, S. Choi, J. Shin, P. Liu, and Y. Choi. Structural image De-Identification for privacy-Preserving deep learning. *IEEE Access*, 8:119848–119862, 2020.

[81] Kalpesh Krishna, Aurko Roy, and Mohit Iyyer. Hurdles to progress in long-form question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4940–4957, Online, June 2021. Association for Computational Linguistics.

[82] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[83] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*, 2023.

[84] Philippe Laban, Tobias Schnabel, Paul N. Bennett, and Marti A. Hearst. SummaC: Re-visiting NLI-based models for inconsistency detection in summarization. *Transactions of the Association for Computational Linguistics*, 10:163–177, 2022.

[85] Bracha Laufer-Goldshtein, Adam Fisch, Regina Barzilay, and Tommi S. Jaakkola. Efficiently controlling multiple risks with pareto testing. In *The Eleventh International Conference on Learning Representations*, 2023.

[86] Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J. Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523):1094–1111, 2018.

[87] Jing Lei, James Robins, and Larry Wasserman. Distribution-free prediction sets. *Journal of the American Statistical Association*, 108(501):278–287, 2013.

[88] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[89] Juncen Li, Robin Jia, He He, and Percy Liang. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1865–1874, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[90] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[91] Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[92] Stephanie C. Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *ArXiv*, abs/2205.14334, 2022.

[93] Bo Liu, Ming Ding, Hanyu Xue, Tianqing Zhu, Dayong Ye, Li Song, and Wanlei Zhou. Dp-image: Differential privacy for image data in feature space. *arXiv preprint arXiv:2103.07073*, 2021.

[94] Dayiheng Liu, Jie Fu, Pengfei Liu, and Jiancheng Lv. TIGS: An inference algorithm for text infilling with gradient search. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4146–4156, 2019.

[95] Guanxiong Liu, Tzu-Ming Harry Hsu, Matthew McDermott, Willie Boag, Wei-Hung Weng, Peter Szolovits, and Marzyeh Ghassemi. Clinically accurate chest x-ray report generation. In *Machine Learning for Healthcare Conference*, pages 249–269. PMLR, 2019.

[96] Jian Liu, Mika Juuti, Yao Lu, and N. Asokan. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 619–631, New York, NY, USA, 2017. Association for Computing Machinery.

[97] Nelson F. Liu, Tianyi Zhang, and Percy Liang. Evaluating verifiability in generative search engines, 2023.

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

[99] Guillermo Lloret-Talavera, Marc Jorda, Harald Servat, Fabian Boemer, Chetan Chauhan, Shigeki Tomishima, Nilesh N Shah, and Antonio J Pena. Enabling homomorphically encrypted inference for large dnn models. *IEEE Transactions on Computers*, 2021.

[100] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Hannaneh Hajishirzi, and Daniel Khashabi. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. *arXiv preprint*, 2022.

[101] J.L. Massey. Guessing and entropy. In *IEEE International Symposium on Information Theory*, page 204, 1994.

[102] Neri Merhav and Asaf Cohen. Universal randomized guessing with application to asynchronous decentralized brute—force attacks. In *IEEE International Symposium on Information Theory (ISIT)*, pages 485–489, 2019.

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

[104] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[105] Mengqi Miao, Fandong Meng, Yijin Liu, Xiao-Hua Zhou, and Jie Zhou. Prevent the language model from being overconfident in neural machine translation. 2021.

[106] Sabrina J. Mielke, Arthur Szlam, Emily Dinan, and Y-Lan Boureau. Reducing conversational agents' overconfidence through linguistic calibration. *Transactions of the Association for Computational Linguistics*, 10:857–872, 2022.

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

[108] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[109] Eric Mitchell, Joseph Noh, Siyan Li, Will Armstrong, Ananth Agarwal, Patrick Liu, Chelsea Finn, and Christopher Manning. Enhancing self-consistency and performance of pre-trained language models through natural language inference. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1754–1768, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[110] Yasuhide Miura, Yuhao Zhang, Emily Bao Tsai, Curtis P. Langlotz, and Dan Jurafsky. Improving factual completeness and consistency of image-to-text radiology report generation, 2021.

[111] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 19–38. IEEE Computer Society, 2017.

[112] Aaron Nicolson, Jason Dowling, and Bevan Koopman. Improving chest x-ray report generation by leveraging warm-starting, 2022.

[113] OpenAI. Gpt-4 technical report. 2023.

[114] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[115] Harris Papadopoulos. Inductive conformal prediction: Theory and application to neural networks. In *Tools in Artificial Intelligence*, chapter 18. IntechOpen, Rijeka, 2008.

[116] Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alex Gammerman. Inductive confidence machines for regression. In *European Conference on Machine Learning*, pages 345–356. Springer, 2002.

[117] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

[118] Fernando Pereira. Formal grammar and information theory: together again? *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 358(1769):1239–1253, 2000.

[119] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.

[120] C.E. Pfister and W.G. Sullivan. Renyi entropy, guesswork moments, and large deviations. *IEEE Transactions on Information Theory*, 50(11):2794–2800, 2004.

[121] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[122] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. 2020.

[123] Maribeth Rauh, John Mellor, Jonathan Uesato, Po-Sen Huang, Johannes Welbl, Laura Weidinger, Sumanth Dathathri, Amelia Glaese, Geoffrey Irving, Iason Gabriel, William Isaac, and Lisa Anne Hendricks. Characteristics of harmful text: Towards rigorous benchmarking of language models, 2022.

[124] Nisarg Raval, Ashwin Machanavajjhala, and Jerry Pan. Olympus: Sensor privacy through utility aware obfuscation. *Proc. Priv. Enhancing Technol.*, 2019(1):5–25, 2019.

[125] Shauli Ravfogel, Yoav Goldberg, and Jacob Goldberger. Conformal nucleus sampling. 2023.

[126] Yaniv Romano, Evan Patterson, and Emmanuel Candès. Conformalized quantile regression. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[127] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

[128] Tal Schuster, Sihao Chen, Senaka Buthpitiya, Alex Fabrikant, and Donald Metzler. Stretching sentence-pair NLI models to reason over long documents and clusters. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 394–412, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[129] Tal Schuster, Adam Fisch, and Regina Barzilay. Get your vitamin C! robust fact verification with contrastive evidence. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 624–643, Online, June 2021. Association for Computational Linguistics.

[130] Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[131] Tal Schuster, Adam Fisch, Tommi Jaakkola, and Regina Barzilay. Consistent accelerated inference via confident adaptive transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4962–4979, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[132] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. *CoRR*, abs/1704.04368, 2017.

[133] Darsh J Shah, Lili Yu, Tao Lei, and Regina Barzilay. Nutri-bullets: Summarizing health studies by composing segments, 2021.

[134] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. In *Advances in neural information processing systems*, pages 6830–6841, 2017.

[135] Tianxiao Shen, Victor Quach, Regina Barzilay, and Tommi Jaakkola. Blank language models. *arXiv preprint arXiv:2002.03079*, 2020.

[136] Yong-Siang Shih, Wei-Cheng Chang, and Yiming Yang. Xl-editor: Post-editing sentences with xlnet. *arXiv preprint arXiv:1910.10479*, 2019.

[137] W. Sirichotedumrong, T. Maekawa, Y. Kinoshita, and H. Kiya. Privacy-preserving deep neural networks with pixel-based image encryption considering data augmentation in the encrypted domain. In *IEEE International Conference on Image Processing (ICIP)*, pages 674–678, 2019.

[138] Akshay Smit, Saahil Jain, Pranav Rajpurkar, Anuj Pareek, Andrew Y. Ng, and Matthew P. Lungren. Chexbert: Combining automatic labelers and expert annotations for accurate radiology report labeling using bert, 2020.

[139] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, and et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models, 2022.

[140] Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. Insertion transformer: Flexible sequence generation via insertion operations. *arXiv preprint arXiv:1902.03249*, 2019.

[141] Qing Sun, Stefan Lee, and Dhruv Batra. Bidirectional beam search: Forward-backward inference in neural sequence models for fill-in-the-blank image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6961–6969, 2017.

[142] Latanya Sweeney, Ji Su Yoo, Laura Perovich, Katherine E Boronow, Phil Brown, and Julia Green Brody. Re-identification risks in hipaa safe harbor data: A study of data from one environmental health study. *Technology science*, 2017, 2017.

[143] M. Tanaka. Learnable image encryption. In *IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, pages 1–2, 2018.

[144] Jacopo Teneggi, Matthew Tivnan, J. Webster Stayman, and Jeremias Sulam. How to trust your diffusion model: A convex optimization approach to conformal risk control. *ArXiv*, abs/2302.03791, 2023.

[145] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume*

1 (Long Papers), pages 809–819, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[146] Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. Dp-cgan: Differentially private synthetic data and label generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[147] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[148] Michael Carl Tschantz, Shayak Sen, and Anupam Datta. Sok: differential privacy as a causal property. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 354–371. IEEE, 2020.

[149] Helena Vasconcelos, Gagan Bansal, Adam Fourney, Q. Vera Liao, and Jennifer Wortman Vaughan. Generation probabilities are not enough: Exploring the effectiveness of uncertainty highlighting in ai-powered code completions. 2023.

[150] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[151] Vladimir Vovk. On-line confidence machines are well-calibrated. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science.*, 2002.

[152] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer-Verlag, Berlin, Heidelberg, 2005.

[153] Vladimir Vovk, Ivan Petej, and Valentina Fedorova. Large-scale probabilistic predictors with and without guarantees of validity. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[154] Vladimir Vovk, Jieli Shen, Valery Manokhin, and Min-ge Xie. Nonparametric predictive distributions based on conformal prediction. In *Proceedings of the Sixth Workshop on Conformal and Probabilistic Prediction and Applications*, 2017.

[155] Alex Wang and Kyunghyun Cho. Bert has a mouth, and it must speak: Bert as a markov random field language model. *arXiv preprint arXiv:1902.04094*, 2019.

[156] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common

thorax diseases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2097–2106, 2017.

[157] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[158] Lai Wei, Nihang Fu, Yuqi Song, Qian Wang, and Jianjun Hu. Probabilistic generative transformer language models for generative design of molecules, 2022.

[159] Lai Wei, Qinyang Li, Yuqi Song, Stanislav Stefanov, Edirisuriya M. D. Siriwardane, Fanglin Chen, and Jianjun Hu. Crystal transformer: Self-learning neural language model for generative and tinkering design of materials, 2022.

[160] Johannes Welbl, Amelia Glaese, Jonathan Uesato, Sumanth Dathathri, John F. J. Mellor, Lisa Anne Hendricks, Kirsty Anderson, Pushmeet Kohli, Ben Coppin, and Po-Sen Huang. Challenges in detoxifying language models. *ArXiv*, abs/2109.07445, 2021.

[161] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[162] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

[163] Chen Wu, Xuancheng Ren, Fuli Luo, and Xu Sun. A hierarchical reinforced sequence operation method for unsupervised text style transfer. *arXiv preprint arXiv:1906.01833*, 2019.

[164] Xing Wu, Tao Zhang, Liangjun Zang, Jizhong Han, and Songlin Hu. Mask and infill: Applying masked language model for sentiment transfer. In *IJCAI*, 2019.

[165] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[166] Zhenyu Wu, Haotao Wang, Zhaowen Wang, Hailin Jin, and Zhangyang Wang. Privacy-preserving deep action recognition: An adversarial learning framework and a new dataset. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[167] Zhenyu Wu, Zhangyang Wang, Zhaowen Wang, and Hailin Jin. Towards privacy-preserving visual recognition via adversarial training: A pilot study. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 606–624, 2018.

[168] Hanshen Xiao and Srinivas Devadas. The art of labeling: Task augmentation for private (collaborative) learning on transformed data. *Cryptology ePrint Archive*, 2021.

[169] Hanshen Xiao and Srinivas Devadas. Dauntless: Data augmentation and uniform transformation for learning with scalability and security. Cryptology ePrint Archive, Report 2021/201, 2021. `https://eprint.iacr.org/2021/201`.

[170] Taihong Xiao, Yi-Hsuan Tsai, Kihyuk Sohn, Manmohan Chandraker, and Ming-Hsuan Yang. Adversarial learning of privacy-preserving and task-oriented representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12434–12441, 2020.

[171] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739*, 2018.

[172] Jingjing Xu, Xu Sun, Qi Zeng, Xiaodong Zhang, Xuancheng Ren, Houfeng Wang, and Wenjie Li. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 979–988, 2018.

[173] Adam Yala, Victor Quach, Homa Esfahanizadeh, Rafael G. L. D'Oliveira, Ken R. Duffy, Muriel Médard, Tommi S. Jaakkola, and Regina Barzilay. Syfer: Neural obfuscation for private data release, 2022.

[174] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764, 2019.

[175] Zichao Yang, Zhiting Hu, Chris Dyer, Eric P Xing, and Taylor Berg-Kirkpatrick. Unsupervised text style transfer using language models as discriminators. In *Advances in Neural Information Processing Systems*, pages 7287–7298, 2018.

[176] Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3881–3890. JMLR. org, 2017.

[177] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167. IEEE Computer Society, 1986.

[178] Xiang Yue, Boshi Wang, Kai Zhang, Ziru Chen, Yu Su, and Huan Sun. Automatic evaluation of attribution by large language models. *arXiv preprint arXiv:2305.06311*, 2023.

[179] Polina Zablotskaia, Du Phan, Joshua Maynez, Shashi Narayan, Jie Ren, and Jeremiah Liu. On uncertainty calibration and selective generation in probabilistic neural summarization: A benchmark study. 2023.

[180] Najam Zaidi, Trevor Cohn, and Gholamreza Haffari. Decoding as dynamic programming for recurrent autoregressive models. In *International Conference on Learning Representations*, 2020.

[181] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[182] Yuan Zhang, Jason Baldridge, and Luheng He. PAWS: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1298–1308, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[183] Zhirui Zhang, Shuo Ren, Shujie Liu, Jianyong Wang, Peng Chen, Mu Li, Ming Zhou, and Enhong Chen. Style transfer as unsupervised machine translation. *arXiv preprint arXiv:1808.07894*, 2018.

[184] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

[185] Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*, 2021.

[186] Kaitlyn Zhou, Dan Jurafsky, and Tatsunori Hashimoto. Navigating the grey area: Expressions of overconfidence and uncertainty in language models. *ArXiv*, abs/2302.13439, 2023.

[187] Wanrong Zhu, Zhiting Hu, and Eric Xing. Text infilling. *arXiv preprint arXiv:1901.00158*, 2019.