Advances in Generative Feature Learning

Zhilin Yang

CMU-LTI-19-010

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA 15213
www.lti.cs.cmu.edu

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.*

Abstract

It is crucial to use unlabeled data to improve learning because unlabeled data is more accessible and more abundant than labeled data. There are two major learning paradigms towards this goal—the unsupervised pretraining method pretrains a language model on unlabeled data and performs finetuning on downstream tasks, while semi-supervised learning jointly optimizes loss functions on labeled and unlabeled data. In this thesis, we study the problem of generative feature learning, which aims to use generative modeling to leverage unlabeled data for boosting target task performance.

We propose novel methods to address critical challenges in generative feature leanring. In the context of unsupervised learning, Transformer-XL is a novel architecture that enables modeling longer-range dependency. Built upon it, XLNet employs a permutation language modeling objective to bridge the gap between language modeling and unsupervised pretraining, which opens the possibility of applying language modeling progress to improve downstream tasks. In the setting of semi-supervised learning, we theoretically answer two open questions of GAN-based semi-supervised learning, which provides more fundamental understanding about why and how GANs work for semi-supervised learning. In addition, we pioneered the research of generative modeling based semi-supervised learning for tasks with complex data structures such as classification on graphs and question answering.

Empirically, our generative feature learning methods have played a leading role in the development of a variety of research topics. The proposed methods obtained the state-of-the-art results on more than 30 benchmarks at the time of their publication, including natural language inference, question answering, text classification, language modeling, semi-supervised learning, etc., demonstrating significant practical value.

Acknowledgments

It is the luckiest thing ever to have the opportunities to work with my advisors—Ruslan Salakhutdinov and William W. Cohen.

As a world-class expert in machine learning and deep learning, Russ has played an important role in my PhD research career. When I started my journey in deep learning, I did not have much experience or knowledge about deep learning. Russ guided me through the very first steps by pointing me to important papers and teaching me the core ideas of deep learning. As a result, I had the opportunities to work in the exciting area of developing deep learning methods to solve natural language understanding problems. I was thus lucky enough to witness and even contribute to the rapid progress of the field. Moreover, Russ is very open-minded about what research topics to work on. Because of this, I was able to explore different research topics in the first year such as image captioning, graph embedding learning, and transfer learning. I benefited a lot from such diversity of research topics in my first year of research. On one hand, it made me aware of the most challenging problems in each area, which essentially defines what are the most important problems in machine learning and gives me vision about the future. On the other, different approaches in these areas inspired my later research when I focused on the framework of generative feature learning. He also taught me important lessons in terms of research ideas, research methods, and paper writing.

William pays a lot of attention to the details including technical details and writing details. He usually gives very detailed comments about wording and grammar, and will carefully point out all the seemingly minor (but in fact very critical) issues about the organization of multiple sections and the correct usage of mathematical symbols. He also has a very rigorous attitude towards technical details. This is a very important merit I learned from him, and I always try to ensure that every step in our paper or experiments is well-motivated and logically solid. Our papers were seldom rejected by any conference with an acceptance rate over 90%, and I believe this is one of the most important reasons. Moreover, he made me believe that simplicity is among the most important things in research. We should never introduce unnecessary complexity or cosmetic maths into our methods if they do not lead to practical improvement or theoretical insight. He also taught me important research methods such as ablation studies. These lessons are valuable and pave the way for my research career.

Both Russ and William are very nice people to work with. And thus I was able to enjoy research as an important part of my life in the last four years, without feeling stress or discourage. We celebrated every successful breakthrough and tackled the biggest challenges in the field together. They were also extremely intelligent and knowledgeable, and it was my great pleasure and honor to work with two of the most wonderful minds in the world. In addition, they offered the most important advices for my career plan, for which I am very grateful.

I worked with Jason Weston at Facebook AI Research. I admired a lot of Jason's work because I believe they fundamentally shaped the field of NLP, with "NLP from scratch" and memory networks being the most prominent examples. Therefore I was thrilled to have the opportunity to work with him. The most important thing I learned from Jason is the methodology of objective-driven research. The main idea is to think about an ultimate goal such as solving dialog, identify the most critical challenges to reach the goal, and then decouple the big challenges into smaller, reachable key results. I believe this leads to a good trade-off between the significance of ultimate goals and the technical feasibility of a single paper.

I was also honored to work with Quoc V. Le at Google Brain. Quoc is a strong believer of big data and large computation. In fact, his groundbreaking work on AutoML is the best example. The philosophy is that it is important to scale with big data and large computation so that we focus on solving the most challenging problems that are not solvable even when we reach the limits of data and computation. I very much agree with this philosophy and have been deeply influenced. Another important lesson I learned from Quoc is optimism about the future of deep learning or AI. The golden era of AI has not passed, and the best is always yet to come. Recent evidence seems to suggest that this might be true—when we looked at the success of LSTMs and wondered if there is innovative work left to be done, the wave of Transformers sweeps through the entire field. Quoc also advocates for the generality of methods instead of being limited to specific applications. This is also an idea that I have been following in my research career.

I would like to thank Graham Neubig for being on my thesis committee. I very much appreciate his valuable advices on improving the quality of the thesis. These advices also lead to deeper thinking about the relationship between our previous work and the future of the field.

Zihang Dai has so far been my most important coauthor, excluding my advisors and mentors. Together we created XLNet, Transformer-XL, complement GAN, and the high-rank language models. I believe these are among the best papers I have written so far. I am surprised and deeply impressed by his strong, full-stack research skills, including implementation, hyperparameter tuning, brainstorming about new ideas, paper writing, understanding and analyzing results, deriving equations and theorems, and visualization. Our collaboration has led to numerous interesting ideas and significant insights.

I would like to thank my best friends at CMU. I have been lucky to meet Zihang Dai, Jiateng Xie, Qizhe Xie, Dylan Du, Guokun Lai, and Jingzhou Liu. I don't think I need to say much here, and you guys all understand.

Edward Chen, Yutao Zhang, and Jeff Jie are valuable friends that give me guidance in the startup world. Their passion, leadership and vision have greatly inspired me.

I would like to thank Hanxiao Liu for giving me numerous useful advices and suggestions both in terms of research and career plans. His opinions and experiences have been very inspiring. I would like to thank Ye Yuan for going to Chengdu Gourmet with me every week :), and of course the great collaborations and discussions we had. I also had a lot of discussions and wrote a couple of papers with Bhuwan Dhingra. He usually gives me new perspectives to a problem, and I really enjoyed working and brainstorming with him. Adams Yu has also been one of my good friends and I learned a lot from his experiences and insights. I would like to thank Peng Qi, Jake

Zhao, and Saizheng Zhang for our great collaborations on HotpotQA and GLoMo, from which I have learned a lot. I am grateful for the suggestions and ideas Thang Luong gave me during my internship at Google, and it was my pleasure to work with him. I would also like to thank Guoqing Zheng, Fan Yang, Diyi Yang, Wei-Cheng Chang, Junjie Hu, Yuexin Wu, Yichong Xu, Simon Du, Han Zhao, Zhiting Hu, Pengtao Xie, Yao-Hung Tsai, Max Ma, Ruochen Xu, Pengcheng Yin, Xinyu Wang, Hector Liu, Shikun Zhang, Keyang Xu, Xingyu Lin, Di Wang and many others for their support.

Jie Tang and Yuchun Ma gave me the opportunities to start my research career and explore the magical world of machine learning when I was at Tsinghua University. They continued to give me important and helpful advices after I graduated. I would like to say thank you to them.

Thank you, Ning, for going through the ups and downs with me. Thank you for the support, advices, understanding, and everything.

I thank my parents for everything they gave me, including the opportunity to be educated and the freedom to pursue my passion. Thank you so very much.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation: Learning from Unlabeled Data

For most machine learning applications and tasks, unlabeled data is more accessible and more abundant than labeled data. For example, major question answering datasets [84, 128, 196] nowadays typically have around 100K training examples in the form of question-answer pairs, which are tens of millions of words. In comparison, it is relatively easy to obtain 1,000 times more unlabeled English text data [15, 25, 118, 206]. In addition, there is a large number of other natural language understanding datasets that are significantly smaller, with only a few thousand training examples or less [170]. Furthermore, given a specific task, while it is possible to collect more labeled data, the procedure of labeled data collection is expensive and time-consuming. Therefore, it is unlikely to create a labeled dataset that matches the scale of unlabeled data. As a result, apart from the standard paradigm of learning on labeled data with supervised training signals, it is crucial to devise methods that use unlabeled data to aid the learning procedure.

In this thesis, we are interested in the problem of learning from unlabeled data, which can be generically formulated as follows. Given a set of labeled data and a (much larger) set of unlabeled data, we aim to design machine learning algorithms and models to fully leverage the unlabeled data and improve over systems that only have access to the labeled data.

There are two directions of research in the literature that fit into the above problem definition, namely *unsupervised pretraining* and *semi-supervised learning*.[1] Although both directions have the same generic goal and motivation, they mainly differ in two aspects.

- **Paradigm of knowledge transfer**. The knowledge transfer procedure of unsupervised pretraining usually consists of two phases—pretraining and finetuning. Specifically, a model is first pretrained on unlabeled data and then finetuned on labeled data.[2] On the contrary, in the setting of semi-supervised learning, a model is often jointly trained on both labeled and unlabeled data in one single training phase. As a consequence, unsupervised

[1]In the most general definition of semi-supervised learning, unsupervised pretraining is a special case of semi-supervised learning. However, in this thesis, we use the term "semi-supervised learning" in the narrow sense.

[2]Throughout the thesis, we use "unsupervised pretraining" to refer to either the sole pretraining phase or both phases combined, interchangeably depending on the context.

pretraining targets multiple tasks while semi-supervised learning mainly focuses on one task.

- **Data scale and distribution**. The scale of unlabeled data in the setting of unsupervised pretraining is usually much larger than that of semi-supervised learning, which is both the cause and effect of the knowledge transfer paradigm. For example, in the field of natural language understanding, unsupervised pretraining typically uses billions of tokens or more during the pretraining phase. In comparison, in a normal setting of semi-supervised learning, the scale of unlabeled data can be 10 to 1000 times smaller. The difference in data scale also results in a difference in data distribution. Typical unsupervised pretraining covers a wider range of unlabeled data distribution such as the entire Web, while semi-supervised learning uses unlabeled data that is relatively more domain-specific with a distribution closer to labeled data.

1.2 Background

Among other approaches, generative modeling has been an important method for leveraging unlabeled data, in the settings of both unsupervised pretraining and semi-supervised learning. In the most generic form, given some data x, a generative model aims to estimate the log likelihood $\log p(x)$. The log likelihood might be defined explicitly by a model that maps data to probabilities, or implicitly modeled as a transformation from a prior distribution to data as in generative adversarial networks (GANs) [47]. The benefits of using generative modeling objectives are their wide applicability. In principle, it is possible to estimate the log likelihood of data in any format, including text, images, and complex structures such as graphs.

For unsupervised pretraining, generative modeling has played an important role. Generative models such as Boltzmann machines and autoencoders were employed to learn unsupervised features [135, 168]. The framework was later extended by using new objectives such as puzzle-based objectives [114] and adversarial objectives [38]. Most of these approaches target image classification. In the field of NLP, [27] proposed to optimize a language modeling (i.e., a form of generative modeling on text) loss on unsupervised text corpora, and the learned model parameters along with the architecture are used as initialization for finetuning on target supervised tasks. The learning paradigm obtained increasing attention since 2017 with the success of a series of models, including ELMo [122], CoVe [100], GPT [126], and BERT [34]. These methods differ in terms of training data sizes, model sizes, amounts of computation, model architectures, and objective functions. From the perspectives of models and algorithms, there are three major advances.

1. **Architecture**. The field of unsupervised pretraining has witnessed a transition from recurrent neural networks (RNNs) [106], especially long short-term memories (LSTMs) [60], to Transformers [166] (see Section 2.1.2 for technical details). Due to the architectural design that involves direct connections between any two input units, Transformers have been shown to obtain better performance in machine translation [166] compared to RNNs. The most substantial improvement of Transformers over RNNs is in language modeling. Recent work [1, 31] shows Transformers and their variants substantially improve the ability of modeling long-range dependency and are able to scale without much optimization difficulty. These

advantages of Transformers pave the road for the success of applications in pretraining.

2. **Finetuning**. While some of the earlier methods [100, 122] use the pretrained models as feature extractors, GPT [126] reinvented the idea of finetuning on downstream tasks, which was originally proposed by [27]. The finetuning approach increases the percentage of pretrained parameters among all trainable parameters. Critically, when the input has multiple sequences, such as a question and a context for the task of question answering, GPT uses special symbols to join multiple sequences into one, so that finetuning can be performed on the concatenated sequence. As a result, this allows modeling the dependency between multiple input sequences with multiple pretrained layers. For some tasks such as question answering, this design proved important [126].
3. **Bidirectional context**. Earlier approaches are based on the standard language modeling objective. As a result, these approaches suffer from not being able to model bidirectional contexts with a multi-layer, deep pretrained network. BERT [34] addressed this issue by using a denoising autoencoding [168] based method, which yields improvement over previous methods.

Generative modeling has also been applied to semi-supervised learning. Variational autoencoders [74] were adapted to the semi-supervised learning setting in [75] by treating the classification label as an additional latent variable in a directed generative model. [98] adds auxiliary variables to the deep VAE structure to make variational distribution more expressive. Later, GANs started to be applied to semi-supervised learning as well. One of the earliest examples was CatGAN [149], which substitutes the binary discriminator in a standard GAN with a multi-class classifier, and trains both the generator and the discriminator using a information theoretical criteria on unlabeled data. This multi-class formulation has been a successful framework for semi-supervised learning. Under this framework, the feature matching GAN [136] further improves performance by considering a feature matching objective for the the generator.

1.3 Challenges

Although generative modeling based methods have been successful in the setting of learning from unlabeled data, there remain a few critical challenges.

- In the setting of unsupervised pretraining, although BERT allows modeling bidirectional contexts, it introduces a new problem—BERT is not able to model the dependencies between the target tokens for prediction due to the assumption that all target tokens are independent. On the contrary, standard language modeling is based on the product rule and thus does not suffer from this problem, but it is only able to model unidirectional contexts. It is not clear how to combine the advantages of different approaches while avoiding their disadvantages.
- Language modeling has been a long-standing, rapidly-developing area of research. However, not being able to model bidirectional contexts poses a question regarding whether there are meaningful applications of language modeling besides text generation.
- Conventional language modeling approaches rely on RNNs, which are not the best models for capturing long-range dependency due to the problem of gradient explosion and vanishing

[119]. Transformers demonstrate a large potential in this aspect because of the direct connections between all input units. However, previous Transformer language models employ a fixed-length context, which prohibits capturing any longer-range dependency beyond the predefined context length. In addition, the fixed-length segments are created by selecting a consecutive chunk of symbols without respecting the sentence or any other semantic boundary. Hence, the models lacks necessary contextual information to well predict the first few symbols, leading to inefficient optimization and inferior performance.

- In the setting of semi-supervised learning, although GAN-based approaches have shown superior performance, little has been understood about why it works. Prominently, there are two questions. First, why does the performance of classification seem contradictory to the quality of generated samples in empirical studies? Second, how does the classifier benefit from joint training with a generator?
- Despite the success of generative modeling based semi-supervised learning, most efforts focus on the standard setting of classification. However, these methods are not directly applicable to tasks with more complex data structures, such as classification on graphs and question answering.

1.4 Contributions

This thesis aims to tackle the above challenges using generative modeling. We first use a generic language to describe the idea of our learning framework called generative feature learning and then introduce technical advances under this framework.

The framework of generative feature learning can be described as follows. Given unlabeled data $\mathcal{D}_u = \{x\}$ and labeled data $\mathcal{D}_l = \{(x, y)\}$, we optimize the following two losses:

$$\mathcal{L}_g = \mathbb{E}_{x\sim\mathcal{D}_u} l_g(x, G_{\theta,\phi})$$
$$\mathcal{L}_t = \mathbb{E}_{x,y\sim\mathcal{D}_l} l_t(y, T_{\theta,\psi}(x))$$

where l_g and l_t are loss functions defined on each data point, $G_{\theta,\phi}$ is a generative model with two parameters θ and ϕ, and $T_{\theta,\psi}$ is a target task model with parameters θ and ψ, where θ is shared between the two models. The target task model maps some input x to output $\hat{y}$, which could be predictions of classification labels, question answers, etc.

The above formulation is kept generic to allow flexible design choices:

- The generative model G can be an implicit model or an explicit model. An implicit model transforms a noise vector z sampled from some prior distribution into a data point $x = G(z)$. An explicit model predicts the log probability $\log p_G(x) = G(x)$ of some data point x.
- The loss function l_g can be instantiated in different ways. For explicit generative models, l_g is usually defined as the negative log likelihood of x, written as $-G(x)$ or $-\log p_G(x)$. For implicit generative models, the definition of l_g is more flexible and more closely integrated with the target loss function l_t.
- The target loss function l_t could be instantiated based on specific target tasks, such as a standard classification cross entropy loss.

- The optimization paradigm could vary. It is possible to employ a multi-phase training procedure where each phase optimizes either $\mathcal{L}_g$ or $\mathcal{L}_t$. In other cases, the two losses might be jointly optimized.

In Chapter 2, we will introduce generative feature learning methods in the context of unsupervised pretraining:

- The neural architecture Transformer-XL advances the state of the art in language modeling. Transformer-XL enables capturing longer-range dependency and also resolves the context fragmentation problem. This makes it possible to truly unleash the potential of Transformers and improves language modeling performance for both long and short sequences.
- XLNet is an unsupervised pretraining framework that bridges the gap between language modeling and unsupervised pretraining. XLNet employs a permutation language modeling objective that provides more effective training signals. Moreover, with XLNet, it is now possible to translate language modeling progress into improvement on target tasks via unsupervised pretraining. For example, we show that Transformer-XL, the latest advance in language modeling, improves unsupervised pretraining performance. This also opens the possibility of applying language modeling progress to improve downstream tasks.

In Chapter 3, generative semi-supervised learning methods will be discussed:

- For GAN-based semi-supervised learning, we prove that under a widely-used discriminator objective in GAN-based semi-supervised learning, a preferred generator should instead generate complement data. We provide a theoretical framework to illustrate two points: (1) how a discriminator benefits from joint training with a GAN; (2) it is contradictory to have a good generative model and a good classifier. This framework provides more fundamental understanding about why and how GANs work for semi-supervised learning and improves performance empirically.
- Going beyond the standard semi-supervised learning formulation, we pioneered the research of generative feature learning for tasks with complex data structures. For semi-supervised question answering, we train a generative model to generate questions based on unlabeled text, and combine model-generated questions with human-generated questions for training question answering models. Domain adaptation techniques based on reinforcement learning are integrated into the training procedure to alleviate data distribution discrepancy. For semi-supervised classification on graphs, we define the unsupervised loss function as generating random walk paths on the graphs.

Overall, the approaches proposed in the thesis have played a leading role in the development of a variety of research topics. Specifically, we hold or used to gold the state-of-the-art results on more than 30 benchmarks, including natural language inference, question answering, text classification, language modeling, semi-supervised learning, etc.

Chapter 2

Unsupervised Pretraining with Generative Modeling

This chapter introduces our generative feature learning methods in the setting of unsupervised pretraining. We attempt to answer the following questions:

- Is it possible to truly unleash the potential of Transformers at capturing long-range dependency?
- Is it possible to combine the advantages of autoregressive and autoencoding approaches while avoiding their disadvantages?
- Is it possible to bridge the gap between language modeling and pretraining, making it possible to leverage language modeling progress to improve target tasks?

Our approach decouples the problem of unsupervised pretraining into two subproblems: (1) improving language modeling; (2) building a learning paradigm with which progress in language modeling can be translated into benefits in unsupervised pretraining. Following this idea, in Section 2.1, we will first present our neural architecture Transformer-XL that substantially improves language modeling by capturing longer-range dependency and resolving the context fragmentation problem. In Section 2.2, a learning paradigm called XLNet will be introduced to bridge the gap between language modeling and unsupervised pretraining.

2.1 Attentive Language Models Beyond a Fixed-Length Context

This section introduces the work originally published at ACL 2019 [31].

2.1.1 Motivation

Language modeling is among the important problems that require modeling long-term dependency, with successful applications such as unsupervised pretraining [27, 34, 122, 126]. However, it has been a challenge to equip neural networks with the capability to model long-term dependency in sequential data. Recurrent neural networks (RNNs), in particular Long Short-Term Memory (LSTM) networks [60], have been a standard solution to language modeling and obtained strong results on multiple benchmarks. Despite the wide adaption, RNNs are difficult to optimize due to gradient vanishing and explosion [61], and the introduction of gating in LSTMs and the gradient clipping technique [51] might not be sufficient to fully address this issue. Empirically, previous work has found that LSTM language models use 200 context words on average [73], indicating room for further improvement.

On the other hand, the direct connections between long-distance word pairs baked in attention mechanisms might ease optimization and enable the learning of long-term dependency [5, 166]. Recently, Al-Rfou et al. [1] designed a set of auxiliary losses to train deep Transformer networks for character-level language modeling, which outperform LSTMs by a large margin. Despite the success, the LM training in Al-Rfou et al. [1] is performed on separated fixed-length segments of a few hundred characters, without any information flow across segments. As a consequence of the fixed context length, the model cannot capture any longer-term dependency beyond the predefined context length. In addition, the fixed-length segments are created by selecting a consecutive chunk of symbols without respecting the sentence or any other semantic boundary. Hence, the model lacks necessary contextual information needed to well predict the first few symbols, leading to inefficient optimization and inferior performance. We refer to this problem as *context fragmentation*.

To address the aforementioned limitations of fixed-length contexts, we propose a new architecture called Transformer-XL (meaning extra long). We introduce the notion of recurrence into our deep self-attention network. In particular, instead of computing the hidden states from scratch for each new segment, we reuse the hidden states obtained in previous segments. The reused hidden states serve as memory for the current segment, which builds up a recurrent connection between the segments. As a result, modeling very long-term dependency becomes possible because information can be propagated through the recurrent connections. Meanwhile, passing information from the previous segment can also resolve the problem of context fragmentation. More importantly, we show the necessity of using relative positional encodings rather than absolute ones, in order to enable state reuse without causing temporal confusion. Hence, as an additional technical contribution, we introduce a simple but more effective relative positional encoding formulation that generalizes to attention lengths longer than the one observed during training.

2.1.2 Background

In the last few years, the field of language modeling has witnessed many significant advances, including but not limited to devising novel architectures to better encode the context [1, 10, 102, 106], improving regularization and optimization algorithms [42] , speeding up the Softmax computation [49] , and enriching the output distribution family [195].

To capture the long-range context in language modeling, a line of work directly feeds a representation of the wider context into the network as an additional input. Existing works range from ones where context representations are manually defined [66, 105, 173] to others that rely

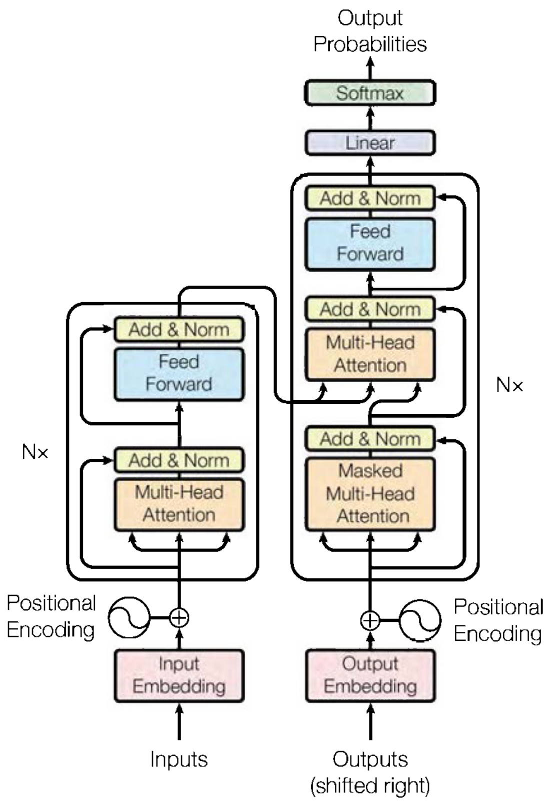


Figure 2.1: The Transformer architecture.

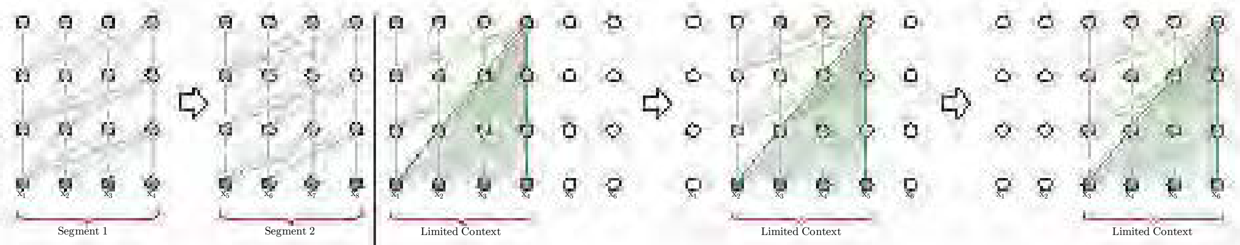
on document-level topics learned from data [37, 174].

More broadly, in generic sequence modeling, how to capture long-term dependency has been a long-standing research problem. From this perspective, since the ubiquitous adaption of LSTM, many efforts have been spent on relieving the vanishing gradient problem, including better initialization [86], additional loss signal [159], augmented memory structure [72] and others that modify the internal architecture of RNNs to ease the optimization [91, 182]. Different from them, our work is based on the Transformer architecture and shows that language modeling as a real-world task benefits from the ability to learn longer-term dependency.

Transformers

Transformers are a deep self-attention network proposed by [166]. The model architecture is illustrated in Figure 2.1. Each layer consists of a self-attention sublayer and a feed-forward sublayer. A positional encoding is added to the word embedding so that the model is aware of the input sequence order. Formally, Transformer with a single attention head can be summarized as follows. For $n = 1, \ldots, N$:

$$\begin{aligned}\mathbf{q}^n, \mathbf{k}^n, \mathbf{v}^n &= \mathbf{h}^{n-1}\mathbf{W}_q^{n\top}, \mathbf{h}^{n-1}\mathbf{W}_k^{n\top}, \mathbf{h}^{n-1}\mathbf{W}_v^{n\top} \\ \mathbf{A}_{i,j}^n &= \mathbf{q}_i^{n\top}\mathbf{k}_j^n\end{aligned}$$



(a) Train phase. (b) Evaluation phase.

Figure 2.2: Illustration of the vanilla model with a segment length 4.

$$\begin{aligned}
\mathbf{a}^n &= \text{Masked-Softmax}(\mathbf{A}^n)\mathbf{v}^n \\
\mathbf{o}^n &= \text{LayerNorm}(\text{Linear}(\mathbf{a}^n) + \mathbf{h}^{n-1}) \\
\mathbf{h}^n &= \text{Positionwise-Feed-Forward}(\mathbf{o}^n)
\end{aligned}$$

with $\mathbf{h}^0$ defined as the sum of the word embedding sequence and the positional encoding sequence. The amount of computation is quadratic with respect to the sequence length, while the number of parameters is constant to the sequence length. Because of the self-attention operation, Transformer directly connects any pairs of input units, which has a potential of better optimization and modeling longer-range dependency.

2.1.3 Approach

Given a corpus of tokens $\mathbf{x} = (x_1, \ldots, x_T)$, the task of language modeling is to estimate the joint probability $P(\mathbf{x})$, which is often auto-regressively factorized as $P(\mathbf{x}) = \prod_t P(x_t \mid \mathbf{x}_{<t})$, where $\mathbf{x}_{<t}$ represents $(x_1, \ldots, x_{t-1})$. With the factorization, the problem reduces to estimating each conditional factor. In this work, we stick to the standard neural approach to modeling the conditional probability. Specifically, a trainable neural network is used to encode the context $\mathbf{x}_{<t}$ into a fixed size hidden state, which is multiplied with the word embeddings to obtain the logits. The logits are then fed into the Softmax function, yielding a categorical probability distribution over the next token.

Vanilla Transformer Language Models

In order to apply Transformer or self-attention to language modeling, the central problem is how to train a Transformer to effectively encode an arbitrarily long context into a fixed size representation. Given infinite memory and computation, a simple solution would be to process the entire context sequence using an unconditional Transformer decoder, similar to a feed-forward neural network. However, this is infeasible because resources are finite in practice.

One feasible but crude approximation is to split the entire corpus into shorter segments of manageable sizes, and only train the model within each segment, ignoring all contextual information from previous segments. This is the idea adopted by Al-Rfou et al. [1]. We call it the *vanilla model* and visualize it in Fig. 2.2a. Under this training paradigm, information never flows across segments in either the forward or backward pass. There are two critical limitations of using a fixed-length context. First, the largest possible dependency length is upper bounded by the segment length, which is a few hundred on character-level language modeling [1]. Therefore, although the self-attention mechanism is less affected by the vanishing gradient problem compared to RNNs, the vanilla model is not able to fully exploit this optimization advantage. Second, though it is possible to use padding to respect the sentence or other semantic boundaries, in practice it has been standard practice to simply chunk long text into fixed-length segments due to improved efficiency [1, 34, 122]. However, simply chunking a sequence into fixed-length segments will lead to the context fragmentation problem as discussed in Section 2.1.1.

During evaluation, at each step, the vanilla model also consumes a segment of the same length as in training, but only makes one prediction at the last position. Then, at the next step, the segment is shifted to the right by only one position, and the new segment has to be processed all from scratch. As shown in Fig. 2.2b, this procedure ensures that each prediction utilizes the longest possible context exposed during training, and also relieves context fragmentation issue encountered in training. However, this evaluation procedure is extremely expensive. We will show that our proposed architecture is able to substantially improve the evaluation speed.

Segment-Level Recurrence with State Reuse

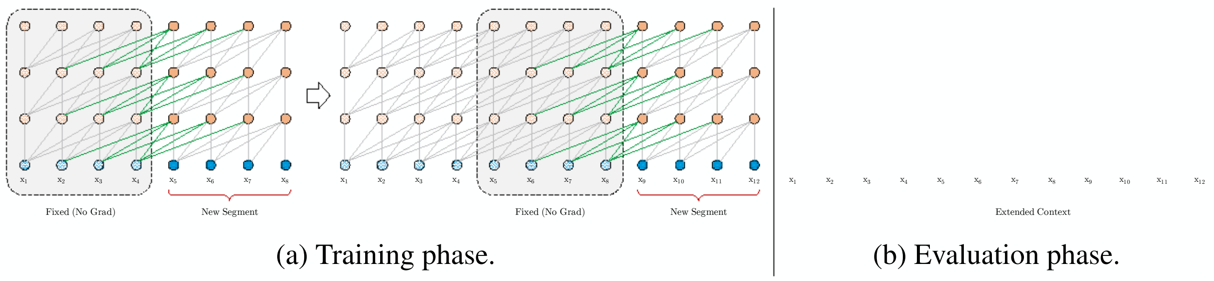

x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
Fixed (No Grad)
New Segment
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12}
Fixed (No Grad)
New Segment
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12}
Extended Context
(a) Training phase.
(b) Evaluation phase.

Figure 2.3: Illustration of the Transformer-XL model with a segment length 4.

To address the limitations of using a fixed-length context, we propose to introduce a recurrence mechanism to the Transformer architecture. During training, the hidden state sequence computed for the previous segment is *fixed* and *cached* to be reused as an extended context when the model processes the next new segment, as shown in Fig. 2.3a. Although the gradient still remains within a segment, this additional input allows the network to exploit information in the history, leading to an ability of modeling longer-term dependency and avoiding context fragmentation. Formally, let the two consecutive segments of length L be $\mathbf{s}_\tau = [x_{\tau,1}, \cdots, x_{\tau,L}]$ and $\mathbf{s}_{\tau+1} = [x_{\tau+1,1}, \cdots, x_{\tau+1,L}]$ respectively. Denoting the n-th layer hidden state sequence produced for the τ-th segment $\mathbf{s}_\tau$ by $\mathbf{h}_\tau^n \in \mathbb{R}^{L\times d}$, where d is the hidden dimension. Then, the n-th layer hidden state for segment $\mathbf{s}_{\tau+1}$ is produced (schematically) as follows,

$$\begin{aligned}
&\widetilde{\mathbf{h}}_{\tau+1}^{n-1} = \left[\text{SG}(\mathbf{h}_\tau^{n-1}) \circ \mathbf{h}_{\tau+1}^{n-1}\right], \\
&\mathbf{q}_{\tau+1}^n, \mathbf{k}_{\tau+1}^n, \mathbf{v}_{\tau+1}^n = \mathbf{h}_{\tau+1}^{n-1}\mathbf{W}_q^\top, \widetilde{\mathbf{h}}_{\tau+1}^{n-1}\mathbf{W}_k^\top, \widetilde{\mathbf{h}}_{\tau+1}^{n-1}\mathbf{W}_v^\top, \\
&\mathbf{h}_{\tau+1}^n = \text{Transformer-Layer}\left(\mathbf{q}_{\tau+1}^n, \mathbf{k}_{\tau+1}^n, \mathbf{v}_{\tau+1}^n\right).
\end{aligned}$$

where the function SG$(\cdot)$ stands for stop-gradient, the notation $[\mathbf{h}_u \circ \mathbf{h}_v]$ indicates the concatenation of two hidden sequences along the length dimension, and $\mathbf{W}_\cdot$ denotes model parameters. Compared to the standard Transformer, the critical difference lies in that the key $\mathbf{k}_{\tau+1}^n$ and value $\mathbf{v}_{\tau+1}^n$ are conditioned on the extended context $\widetilde{\mathbf{h}}_{\tau+1}^{n-1}$ and hence $\mathbf{h}_\tau^{n-1}$ cached from the previous segment. We emphasize this particular design by the green paths in Fig. 2.3a.

With this recurrence mechanism applied to every two consecutive segments of a corpus, it essentially creates a segment-level recurrence in the hidden states. As a result, the effective context being utilized can go beyond just two segments. However, notice that the recurrent dependency between $\mathbf{h}_{\tau+1}^n$ and $\mathbf{h}_\tau^{n-1}$ shifts one layer downwards per-segment, which differs from the same-layer recurrence in conventional RNN-LMs. Consequently, the largest possible dependency length grows linearly w.r.t. the number of layers as well as the segment length, i.e., $O(N \times L)$, as visualized by the shaded area in Fig. 2.3b. This is analogous to truncated BPTT [106], a technique developed for training RNN-LMs. However, different from truncated BPTT, our method caches a sequence of hidden states instead of the last one, and should be applied together with the relative positional encoding technique described in Section 2.1.3.

Besides achieving extra long context and resolving fragmentation, another benefit that comes with the recurrence scheme is significantly faster evaluation. Specifically, during evaluation, the representations from the previous segments can be reused instead of being computed from scratch as in the case of the vanilla model. In theory, the recurrence mechanism improves the evaluation speed by $O(l)$, where l is the attention length during evaluation time. In our experiments on enwik8, Transformer-XL is up to 1,800+ times faster than the vanilla model during evaluation (see Section 2.1.4).

Finally, notice that the recurrence scheme does not need to be restricted to only the previous segment. In theory, we can cache as many previous segments as the GPU memory allows, and reuse all of them as the extra context when processing the current segment. Thus, we can cache a predefined length-M old hidden states spanning (possibly) multiple segments, and refer to them as the memory $\mathbf{m}_\tau^n \in \mathbb{R}^{M \times d}$, due to a clear connection to the memory augmented neural networks [52, 179]. In our experiments, we set M equal to the segment length during training, and increase it by multiple times during evaluation.

Relative Positional Encodings

While we found the idea presented in the previous subsection very appealing, there is a crucial technical challenge we haven't solved in order to reuse the hidden states. That is, how can we keep the positional information coherent when we reuse the states? Recall that, in the standard Transformer, the information of sequence order is provided by a set of positional encodings, denoted as $\mathbf{U} \in \mathbb{R}^{L_{\text{max}} \times d}$, where the i-th row $\mathbf{U}_i$ corresponds to the i-th *absolute* position within a segment and L_{max} prescribes the maximum possible length to be modeled. Then, the actual input to the Transformer is the element-wise addition of the word embeddings and the positional encodings. If we simply adapt this positional encoding to our recurrence mechanism, the hidden state sequence would be computed schematically by

$$\begin{aligned}\mathbf{h}_{\tau+1} &= f(\mathbf{h}_\tau, \mathbf{E}_{\mathbf{s}_{\tau+1}} + \mathbf{U}_{1:L}) \\ \mathbf{h}_\tau &= f(\mathbf{h}_{\tau-1}, \mathbf{E}_{\mathbf{s}_\tau} + \mathbf{U}_{1:L}),\end{aligned}$$

where $\mathbf{E}_{\mathbf{s}_\tau} \in \mathbb{R}^{L\times d}$ is the word embedding sequence of $\mathbf{s}_\tau$, and f represents a transformation function. Notice that, both $\mathbf{E}_{\mathbf{s}_\tau}$ and $\mathbf{E}_{\mathbf{s}_{\tau+1}}$ are associated with the same positional encoding $\mathbf{U}_{1:L}$. As a result, the model has no information to distinguish the positional difference between $x_{\tau,j}$ and $x_{\tau+1,j}$ for any $j = 1, \ldots, L$, resulting in a sheer performance loss.

In order to avoid this failure mode, the fundamental idea is to only encode the *relative* positional information in the hidden states. Conceptually, the positional encoding gives the model a temporal clue or "bias" about how information should be gathered, i.e., where to attend. For the same purpose, instead of incorporating bias statically into the initial embedding, one can inject the same information into the attention score of each layer. More importantly, it is more intuitive and generalizable to define the temporal bias in a relative manner. For instance, when a query vector $q_{\tau,i}$ attends on the key vectors $\mathbf{k}_{\tau,\leq i}$, it does not need to know the absolute position of each key vector to identify the temporal order of the segment. Instead, it suffices to know the relative distance between each key vector $k_{\tau,j}$ and itself $q_{\tau,i}$, i.e. $i - j$. Practically, one can create a set of relative positional encodings $\mathbf{R} \in \mathbb{R}^{L_{\text{max}}\times d}$, where the i-th row $\mathbf{R}_i$ indicates a relative distance of i between two positions. By injecting the relative distance dynamically into the attention score, the query vector can easily distinguish the representations of $x_{\tau,j}$ and $x_{\tau+1,j}$ from their different distances, making the state reuse mechanism feasible. Meanwhile, we won't lose any temporal information, as the absolute position can be recovered recursively from relative distances.

Previously, the idea of relative positional encodings has been explored in the context of machine translation [141] and music generation [63]. Here, we offer a different derivation, arriving at a new form of relative positional encodings, which not only has a one-to-one correspondence to its absolute counterpart but also enjoys much better generalization empirically (see Section 2.1.4). Firstly, in the standard Transformer [166], the attention score between query q_i and key vector k_j within the same segment can be decomposed as

$$\begin{aligned}\mathbf{A}^{\text{abs}}_{i,j} &= \mathbf{q}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{k}_j \\ &= (\mathbf{E}_{x_i} + \mathbf{U}_i)^\top \mathbf{W}_q^\top \mathbf{W}_k (\mathbf{E}_{x_j} + \mathbf{U}_j) \\ &= \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(b)} \\ &+ \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(c)} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(d)}.\end{aligned}$$

Following the idea of only relying on relative positional information, we propose to re-parameterize the four terms as follows

$$\begin{aligned}\mathbf{A}^{\text{rel}}_{i,j} &= \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(b)} \\ &+ \underbrace{u^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(c)} + \underbrace{v^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(d)}.\end{aligned}$$

- The first change we make is to replace all appearances of the absolute positional embedding $\mathbf{U}_j$ for computing key vectors in term (b) and (d) with its relative counterpart $\mathbf{R}_{i-j}$. This essentially reflects the prior that only the relative distance matters for where to attend. Note that $\mathbf{R}$ is a sinusoid encoding matrix [166] without learnable parameters.
- Secondly, we introduce a trainable parameter $u \in \mathbb{R}^d$ to replace the query $\mathbf{U}_i^\top \mathbf{W}_q^\top$ in term (c). In this case, since the query vector is the same for all query positions, it suggests that the attentive bias towards different words should remain the same regardless of the query position. With a similar reasoning, a trainable parameter $v \in \mathbb{R}^d$ is added to substitute $\mathbf{U}_i^\top \mathbf{W}_q^\top$ in term (d).
- Finally, we deliberately separate the two weight matrices $\mathbf{W}_{k,E}$ and $\mathbf{W}_{k,R}$ for producing the content-based key vectors and location-based key vectors respectively.

Under the new parameterization, each term has an intuitive meaning: term (a) represents content-based addressing, term (b) captures a content-dependent positional bias, term (c) governs a global content bias, and (d) encodes a global positional bias.

In comparison, the formulation in Shaw et al. [141] only has terms (a) and (b), dropping the two bias terms (c) and (d). Moreover, Shaw et al. [141] merge the multiplication $\mathbf{W}_k\mathbf{R}$ into a single trainable matrix $\hat{\mathbf{R}}$, which abandons the inductive bias built into the original sinusoid positional encoding [166]. In contrast, our relative positional embedding $\mathbf{R}$ adapts the sinusoid formulation. As a benefit of the inductive bias, a model trained on a memory of some certain length can automatically generalize to a memory several times longer during evaluation.

Equipping the recurrence mechanism with our proposed relative positional embedding, we finally arrive at the Transformer-XL architecture. For completeness, we summarize the computational procedure for a N-layer Transformer-XL with a single attention head here. For $n = 1, \ldots, N$:

$$\begin{aligned}\widetilde{\mathbf{h}}_\tau^{n-1} &= \left[\text{SG}(\mathbf{m}_\tau^{n-1}) \circ \mathbf{h}_\tau^{n-1}\right] \\ \mathbf{q}_\tau^n, \mathbf{k}_\tau^n, \mathbf{v}_\tau^n &= \mathbf{h}_\tau^{n-1}{\mathbf{W}_q^n}^\top, \widetilde{\mathbf{h}}_\tau^{n-1}{\mathbf{W}_{k,E}^n}^\top, \widetilde{\mathbf{h}}_\tau^{n-1}{\mathbf{W}_v^n}^\top \\ \mathbf{A}_{\tau,i,j}^n &= {\mathbf{q}_{\tau,i}^n}^\top \mathbf{k}_{\tau,j}^n + {\mathbf{q}_{\tau,i}^n}^\top \mathbf{W}_{k,R}^n \mathbf{R}_{i-j} \\ &\quad + u^\top \mathbf{k}_{\tau,j} + v^\top \mathbf{W}_{k,R}^n \mathbf{R}_{i-j} \\ \mathbf{a}_\tau^n &= \text{Masked-Softmax}(\mathbf{A}_\tau^n)\mathbf{v}_\tau^n \\ \mathbf{o}_\tau^n &= \text{LayerNorm}(\text{Linear}(\mathbf{a}_\tau^n) + \mathbf{h}_\tau^{n-1}) \\ \mathbf{h}_\tau^n &= \text{Positionwise-Feed-Forward}(\mathbf{o}_\tau^n)\end{aligned}$$

with $\mathbf{h}_\tau^0 := \mathbf{E}_{\mathbf{s}_\tau}$ defined as the word embedding sequence. In addition, it is worth mentioning that a naive way to compute $\mathbf{A}$ requires computing $\mathbf{W}_{k,R}^n \mathbf{R}_{i-j}$ for all pairs (i, j), whose cost is quadratic w.r.t. the sequence length. However, noticing that the value of $i - j$ only ranges from zero to the sequence length, we show a simple computation procedure in Section 2.1.3, which reduces the cost to be linear w.r.t. the sequence length.

Efficient Computation of the Attention with Relative Positional Embedding

As we discussed in Section 2.1.3, the naive way of computing the $\mathbf{W}_{k,R}\mathbf{R}_{i-j}$ for all pairs (i, j) is subject to a quadratic cost. Here, we present a simple method with only a linear cost. Firstly, notice that the relative distance $i - j$ can only be integer from 0 to $M + L - 1$, where M and L are the memory length and segment length respectively. Hence, the rows of the matrix

$$\mathbf{Q} := \begin{bmatrix} \mathbf{R}_{M+L-1}^\top \\ \mathbf{R}_{M+L-2}^\top \\ \vdots \\ \mathbf{R}_1^\top \\ \mathbf{R}_0^\top \end{bmatrix} \mathbf{W}_{k,R}{}^\top = \begin{bmatrix} [\mathbf{W}_{k,R}\mathbf{R}_{M+L-1}]^\top \\ [\mathbf{W}_{k,R}\mathbf{R}_{M+L-2}]^\top \\ \vdots \\ [\mathbf{W}_{k,R}\mathbf{R}_1]^\top \\ [\mathbf{W}_{k,R}\mathbf{R}_0]^\top \end{bmatrix} \in \mathbb{R}^{(M+L)\times d}$$

consist of all possible vector outputs of $\mathbf{W}_{k,R}\mathbf{R}_{i-j}$ for any (i, j). Note that we have defined $\mathbf{Q}$ in a reversed order, i.e., $\mathbf{Q}_k = \mathbf{W}_{k,R}\mathbf{R}_{M+L-1-k}$, to make further discussion easier.

Next, we collect the term (b) for all possible i, j into the following $L \times (M + L)$ matrix,

$$\begin{aligned} \mathbf{B} &= \begin{bmatrix} q_0^\top \mathbf{W}_{k,R}\mathbf{R}_M & \cdots & q_0^\top \mathbf{W}_{k,R}\mathbf{R}_0 & 0 & \cdots & 0 \\ q_1^\top \mathbf{W}_{k,R}\mathbf{R}_{M+1} & \cdots & q_1^\top \mathbf{W}_{k,R}\mathbf{R}_1 & q_1^\top \mathbf{W}_{k,R}\mathbf{R}_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ q_{L-1}^\top \mathbf{W}_{k,R}\mathbf{R}_{M+L-1} & \cdots & q_{L-1}^\top \mathbf{W}_{k,R}\mathbf{R}_{M+L-1} & q_{L-1}^\top \mathbf{W}_{k,R}\mathbf{R}_{L-1} & \cdots & q_{L-1}^\top \mathbf{W}_{k,R}\mathbf{R}_0 \end{bmatrix} \\ &= \begin{bmatrix} q_0^\top \mathbf{Q}_{L-1} & \cdots & q_0^\top \mathbf{Q}_{M+L-1} & 0 & \cdots & 0 \\ q_1^\top \mathbf{Q}_{L-2} & \cdots & q_1^\top \mathbf{Q}_{M+L-2} & q_1^\top \mathbf{Q}_{M+L-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ q_{L-1}^\top \mathbf{Q}_0 & \cdots & q_{L-1}^\top \mathbf{Q}_M & q_{L-1}^\top \mathbf{Q}_{M+1} & \cdots & q_{L-1}^\top \mathbf{Q}_{M+L-1} \end{bmatrix} \end{aligned}$$

Then, we further define

$$\widetilde{\mathbf{B}} = \mathbf{q}\mathbf{Q}^\top = \begin{bmatrix} q_0^\top \mathbf{Q}_0 & \cdots & q_0^\top \mathbf{Q}_M & q_0^\top \mathbf{Q}_{M+1} & \cdots & q_0^\top \mathbf{Q}_{M+L-1} \\ q_1^\top \mathbf{Q}_0 & \cdots & q_1^\top \mathbf{Q}_M & q_1^\top \mathbf{Q}_{M+1} & \cdots & q_1^\top \mathbf{Q}_{M+L-1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ q_{L-1}^\top \mathbf{Q}_0 & \cdots & q_{L-1}^\top \mathbf{Q}_M & q_{L-1}^\top \mathbf{Q}_{M+1} & \cdots & q_{L-1}^\top \mathbf{Q}_{M+L-1} \end{bmatrix}.$$

Now, it is easy to see an immediate relationship between $\mathbf{B}$ and $\widetilde{\mathbf{B}}$, where the i-th row of $\mathbf{B}$ is simply a left-shifted version of i-th row of $\widetilde{\mathbf{B}}$. Hence, the computation of $\mathbf{B}$ only requires a matrix multiplication $\mathbf{q}\mathbf{Q}^\top$ to compute $\widetilde{\mathbf{B}}$ and then a set of left-shifts.

Model	#Param	PPL
Grave et al. [50] - LSTM	-	48.7
Bai et al. [6] - TCN	-	45.2
Dauphin et al. [33] - GCNN-8	-	44.9
Grave et al. [50] - Neural cache	-	40.8
Dauphin et al. [33] - GCNN-14	-	37.2
Merity et al. [104] - QRNN	151M	33.0
Rae et al. [127] - Hebbian + Cache	-	29.9
Ours - Transformer-XL Standard	151M	**24.0**
Baevski and Auli [4] - Adaptive Input$^\diamond$	247M	20.5
Ours - Transformer-XL Large	257M	**18.3**

Table 2.1: Comparison with state-of-the-art results on WikiText-103. $^\diamond$ indicates contemporary work.

Similarly, we can collect all term (d) for all possible i, j into another $L \times (M + L)$ matrix $\mathbf{D}$,

$$\mathbf{D} = \begin{bmatrix} v^\top \mathbf{Q}_{L-1} & \cdots & v^\top \mathbf{Q}_{M+L-1} & 0 & \cdots & 0 \\ v^\top \mathbf{Q}_{L-2} & \cdots & v^\top \mathbf{Q}_{M+L-2} & v^\top \mathbf{Q}_{M+L-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ v^\top \mathbf{Q}_{0} & \cdots & v^\top \mathbf{Q}_{M} & v^\top \mathbf{Q}_{M+1} & \cdots & v^\top \mathbf{Q}_{M+L-1} \end{bmatrix}.$$

Then, we can follow the same procedure to define

$$\widetilde{\mathbf{d}} = [\mathbf{Q}v]^\top = \begin{bmatrix} v^\top \mathbf{Q}_0 & \cdots & v^\top \mathbf{Q}_M & v^\top \mathbf{Q}_{M+1} & \cdots & v^\top \mathbf{Q}_{M+L-1} \end{bmatrix}.$$

Again, each row of $\mathbf{D}$ is simply a left-shifted version of $\widetilde{\mathbf{d}}$. Hence, the main computation cost comes from the matrix-vector multiplication $\widetilde{\mathbf{d}} = [\mathbf{Q}v]^\top$, which is not expensive any more.

2.1.4 Experiments

Main Results

We apply Transformer-XL to a variety of datasets on both word-level and character-level language modeling to have a comparison with state-of-the-art systems, including WikiText-103 [102], enwik8 [99], text8 [99], Billion Word [17], and Penn Treebank [105].

WikiText-103 is the largest available word-level language modeling benchmark with long-term dependency. It contains 103M training tokens from 28K articles, with an average length of 3.6K tokens per article, which allows testing the ability of long-term dependency modeling. We set the attention length to 384 during training and 1600 during evaluation. We adopted adaptive softmax and input representations [4, 49]. As shown in Table 2.1, Transformer-XL reduces the previous state-of-the-art (SoTA) perplexity from 20.5 to 18.3, which demonstrates the superiority of the Transformer-XL architecture.

Model	**#Param**	**bpc**
Ha et al. [57] - LN HyperNetworks	27M	1.34
Chung et al. [21] - LN HM-LSTM	35M	1.32
Zilly et al. [207] - RHN	46M	1.27
Mujika et al. [112] - FS-LSTM-4	47M	1.25
Krause et al. [80] - Large mLSTM	46M	1.24
Knol [78] - cmix v13	-	1.23
Al-Rfou et al. [1] - 12L Transformer	44M	1.11
Ours - 12L Transformer-XL	41M	**1.06**
Al-Rfou et al. [1] - 64L Transformer	235M	1.06
Ours - 18L Transformer-XL	88M	1.03
Ours - 24L Transformer-XL	277M	**0.99**

Table 2.2: Comparison with state-of-the-art results on enwik8.

Model	**#Param**	**bpc**
Cooijmans et al. [24] - BN-LSTM	-	1.36
Chung et al. [21] - LN HM-LSTM	35M	1.29
Zilly et al. [207] - RHN	45M	1.27
Krause et al. [80] - Large mLSTM	45M	1.27
Al-Rfou et al. [1] - 12L Transformer	44M	1.18
Al-Rfou et al. [1] - 64L Transformer	235M	1.13
Ours - 24L Transformer-XL	277M	**1.08**

Table 2.3: Comparison with state-of-the-art results on text8.

The dataset enwik8 contains 100M bytes of unprocessed Wikipedia text. We compare our architecture with the previous results in Table 2.2. Under the model size constraint, the 12-layer Transformer-XL achieves a new SoTA result, outperforming the 12-layer vanilla Transformer from Al-Rfou et al. [1] by 0.05, while both Transformer variants have a large margin over conventional RNN-based models. Notably, our 12-layer architecture achieves the same result as the 64-layer network from Al-Rfou et al. [1], using only 17% of the parameter budget. In order to see whether better performances can be obtained by increasing the model size, we train 18-layer and 24-layer Transformer-XLs with increased model sizes. With the attention length 784 during training and 3,800 during evaluation, we obtained a new SoTA result and our method is the first to break through 1.0 on widely-studied character-level benchmarks. Different from Al-Rfou et al. [1], Transformer-XL does not need any auxiliary losses, and thus all benefits are due to a better architecture.

Similar to but different from enwik8, text8 contains 100M processed Wikipedia characters created by lowercasing the text and removing any character other than the 26 letters `a` through `z`, and space. Due to the similarity to enwik8, we simply adapt the best model and the same hyper-parameters on enwik8 to text8 without further tuning. The comparison with previous

Model	#Param	PPL
Shazeer et al. [142] - Sparse Non-Negative	33B	52.9
Chelba et al. [17] - RNN-1024 + 9 Gram	20B	51.3
Kuchaiev and Ginsburg [81] - G-LSTM-2	-	36.0
Dauphin et al. [33] - GCNN-14 bottleneck	-	31.9
Jozefowicz et al. [70] - LSTM	1.8B	30.6
Jozefowicz et al. [70] - LSTM + CNN	1.04B	30.0
Shazeer et al. [143] - Low-Budget MoE	$\sim$5B	34.1
Shazeer et al. [143] - High-Budget MoE	$\sim$5B	28.0
Shazeer et al. [144] - Mesh Tensorflow	4.9B	24.0
Baevski and Auli [4] - Adaptive Input$^\diamond$	0.46B	24.1
Baevski and Auli [4] - Adaptive Input$^\diamond$	1.0B	23.7
Ours - Transformer-XL Base	0.46B	23.5
Ours - Transformer-XL Large	0.8B	**21.8**

Table 2.4: Comparison with state-of-the-art results on Billion Word. $^\diamond$ indicates contemporary work.

methods is summarized in Table 2.3. Again, Transformer-XL achieves the new SoTA result with a clear margin.

The Billion Word dataset does not preserve any long-term dependencies because sentences have been shuffled. Consequently, this dataset tests the ability of modeling only short-term dependency. The comparison between Transformer-XL and the other methods is shown in Table 2.4. Although Transformer-XL is mainly designed to better capture longer-term dependency, it dramatically improves the single-model SoTA from 23.7 to 21.8. Specifically, Transformer-XL significantly outperforms a contemporary method using vanilla Transformers [4], suggesting the advantage of Transformer-XL is generalizable to modeling short sequences. We conjecture that this is because we resolve the segment fragmentation problem as discussed in Section 2.1.3.

We also report the results on word-level Penn Treebank in Table 2.5. Similar to AWD-LSTM [103], we apply variational dropout and weight average to Transformer-XL. With proper regularization, Transformer-XL achieves a new SoTA result among models without two-step finetuning. Penn Treebank has only 1M training tokens, which implies that Transformer-XL also generalizes well even on small datasets.

Ablation Study

We conduct two sets of ablation studies to examine the effects of two proposed techniques used in Transformer-XL: the recurrence mechanism and the new positional encoding scheme.

The first study is performed on WikiText-103, which requires modeling long-term dependency. The results are reported in Table 2.6. Among the compared encoding schemes, Shaw et al. [141] is relative, while Vaswani et al. [166] and Al-Rfou et al. [1] are absolute. "Full" and "half" losses refer to applying a cross entropy loss to all or the recent half positions in the segment. We

Model	#Param	PPL
Inan et al. [64] - Tied Variational LSTM	24M	73.2
Zilly et al. [207] - Variational RHN	23M	65.4
Zoph and Le [208] - NAS Cell	25M	64.0
Merity et al. [103] - AWD-LSTM	24M	58.8
Pham et al. [123] - Efficient NAS	24M	58.6
Liu et al. [94] - Differentiable NAS	23M	56.1
Yang et al. [195] - AWD-LSTM-MoS	22M	55.97
Melis et al. [101] - Dropout tuning	24M	55.3
Ours - Transformer-XL	24M	**54.52**
Merity et al. [103] - AWD-LSTM+Finetune†	24M	57.3
Yang et al. [195] - MoS+Finetune†	22M	**54.44**

Table 2.5: Comparison with state-of-the-art results on Penn Treebank. † indicates use of two-step finetuning.

experimented with the “half” setting because we found some positional encodings are only able to work with the “half” setting due to the context fragmentation problem (see Section 2.1.3). We found that absolute encodings only work well with half losses because half losses exclude positions with very short attention lengths during training for better generalization. Table 2.6 shows that both the recurrence mechanism and our encoding scheme are necessary to achieve the best performance, as well as generalizing to longer attention sequences during evaluation time. Although the backpropagation length during training is only 128, with the two techniques the attention length can be increased to 640 at test time. In the standard setting with 151M parameters, the perplexity decreases as the attention length increases.

Since the recurrence mechanism costs additional memory, we also compare Transformer-XL with baselines under the same GPU memory constraints. As shown in Table 2.8, despite using a shorter backpropagation length, Transformer-XL remains superior to the baselines.

The second study targets at isolating the effects of resolving the context fragmentation problem from the benefit of capturing longer context length. In order to achieve this goal, we deliberately choose a dataset that does not require long-term dependency, so that any improvement from establishing the recurrence can be attributed to solving the context fragmentation. Specifically, we perform this controlled experiment on the Billion Word dataset, which can only benefit from removing the context fragmentation. We train a 20-layer Transformer-XL with $\sim$0.3B parameters for 400K steps. As shown in Table 2.7, using segment-level recurrence substantially improves performance even when long-term dependency is not needed, which is consistent with our previous discussion that the recurrence mechanism resolves the context fragmentation problem. Moreover, our relative positional encodings is also superior to Shaw et al. [141] on short sequences.

Relative Effective Context Length

Khandelwal et al. [73] proposed a method to evaluate the *Effective Context Length* (ECL) of a

Remark	**Recurrence**	**Encoding**	**Loss**	**PPL init**	**PPL best**	**Attn Len**
Transformer-XL (128M)	✓	Ours	Full	**27.02**	**26.77**	**500**
-	✓	Shaw et al. [141]	Full	27.94	27.94	256
-	✓	Ours	Half	28.69	28.33	460
-	✗	Ours	Full	29.59	29.02	260
-	✗	Ours	Half	30.10	30.10	120
-	✗	Shaw et al. [141]	Full	29.75	29.75	120
-	✗	Shaw et al. [141]	Half	30.50	30.50	120
-	✗	Vaswani et al. [166]	Half	30.97	30.97	120
Transformer (128M)†	✗	Al-Rfou et al. [1]	Half	31.16	31.16	120
Transformer-XL (151M)	✓	Ours	Full	23.43	**23.09**	**640**
					23.16	450
					23.35	300

Table 2.6: Ablation study on WikiText-103. For the first two blocks, we use a slightly smaller model (128M parameters). † indicates that the corresponding row is reduced to the same setting as the Transformer network in [1], except that two auxiliary losses are not implemented in our experiments. "PPL init" refers to using the same length as training. "PPL best" indicates the perplexity obtained by using the optimal length. "Attn Len" is the shortest possible attention length during evaluation to achieve the corresponding result (PPL best). Increasing the attention length during evaluation improves performance only when our positional encoding is used. The "Transformer-XL (151M)" setting uses a standard parameter budget as previous work [104], where we observe a similar effect when increasing the attention length during evaluation.

sequence model. ECL is the longest length to which increasing the context span would lead to a gain more than a threshold. However, ECL ignores the fact that it is harder to get improvement when a model already achieves a lower perplexity using only a shorter context, and thus it is not suitable for fair comparison among multiple models. We instead propose a new metric called *Relative Effective Context Length* (RECL). RECL is defined on a model group instead of a single model, and the gain of a long context is measured by the relative improvement over the *best* short-context model. As such, the model group shares the same baseline to enable fair comparison. RECL also has a parameter r, which means constraining the comparison to the top-r hard examples. See Appedix 2.1.4 for more details about RECL. As shown in Table 2.9, Transformer-XL manages to model dependency of 900 words long on average with $r = 0.1$. The RECL of Transformer-XL is 80% and 450% longer than recurrent networks and Transformer respectively. Both the recurrence mechanism and our positional encodings contribute to a longer RECL. This further substantiates our argument that Transformer-XL is able to model longer-term dependency.

Method	PPL
Ours	**25.2**
With Shaw et al. [141] encodings	25.7
Without recurrence	27.1

Table 2.7: Ablation study on Billion Word, a dataset without long-term dependency.

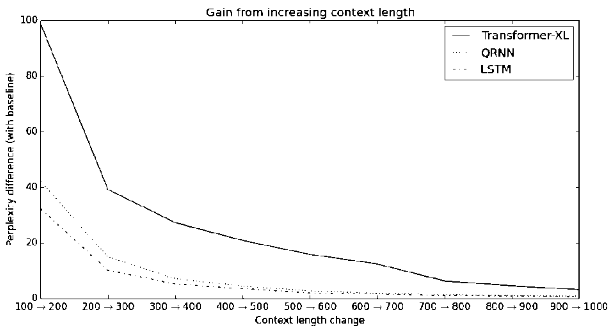
Backprop Len	Recurrence	Encoding	Loss	pplx best	pplx init	Attn Len
128	✓	Ours	Full	**26.77**	**27.02**	**500**
128	✓	Ours	Partial	28.33	28.69	460
176	✗	Ours	Full	27.98	28.43	400
172	✗	Ours	Partial	28.83	28.83	120

Table 2.8: Ablation study on WikiText-103 with the same GPU memory constraints.

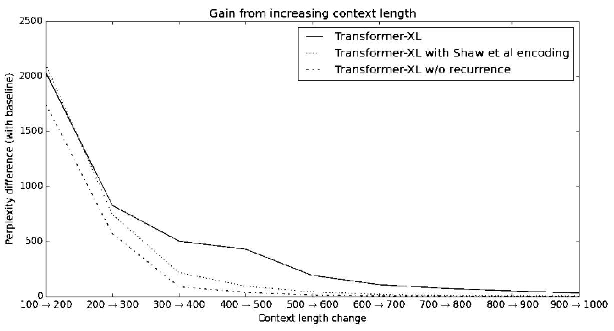
Evaluation Speed

Finally, we compare the evaluation speed of our model with the vanilla Transformer model [1]. As shown in Table 2.10, due to the state reuse scheme, Transformer-XL achieves an up to 1,874 times speedup during evaluation.

Details About RECL



(a) Transformer-XL vs RNNs



(b) Transformer-XL vs Baseline

Figure 2.4: Visualizing unnormalized relative perplexity gains with $r = 0.1$.

In this section, we describe the details of the metric RECL. Let $\mathcal{M} = \{m_1, m_2, \cdots, m_N\}$ be a model group consisting of N models. Let $l_i(c, t)$ denote the loss of model m_i on the t-th token in the corpus with a context length c. Concretely, the loss can be written as

$$l_i(c, t) = -\log P_{m_i}(x_t | x_{t-1}, \cdots, x_{t-c})$$

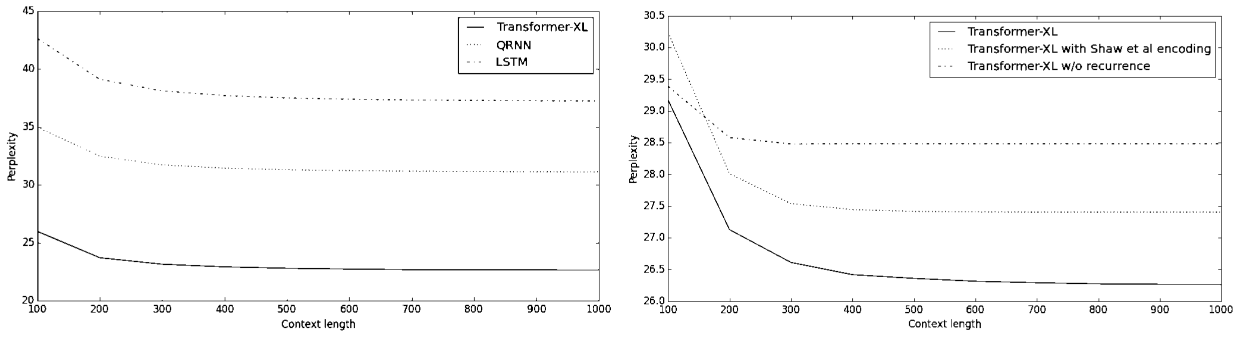
where P_{m_i} is the probability distribution given by model m_i, and x_t is the t-th token in the corpus. Given a short context length c and a long context length c' such that $c' \geq c$, we can further define

Model	$r = 0.1$	$r = 0.5$	$r = 1.0$
Transformer-XL 151M	**900**	**800**	**700**
QRNN	500	400	300
LSTM	400	300	200
Transformer-XL 128M	**700**	**600**	**500**
- use Shaw et al. [141] encoding	400	400	300
- remove recurrence	300	300	300
Transformer	128	128	128

Table 2.9: Relative effective context length (RECL) comparison. See text for the definition of RECL and r. The first three models and the last four models are compared as two *model groups* when we calculate RECL (RECL is computed on a model group rather than a single model). Each group has the same parameter budget.

Attn Len	**How much Al-Rfou et al. [1] is slower**
3,800	1,874x
2,800	1,409x
1,800	773x
800	363x

Table 2.10: Slowdown in terms of running time during evaluation. Evaluation is based on per-token time on one GPU.



(a) Transformer-XL vs RNNs

(b) Transformer-XL vs Baseline

Figure 2.5: Perplexity vs context length.

a baseline for each position t,

$$b(c, t) = \min_{i=1}^{N} l_i(c, t)$$

The *relative loss* of m_i w.r.t. the model group $\mathcal{M}$ is written as

$$f_i(c, c') = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \min\left(b(c, t), l_i(c', t)\right)$$

The above equation uses the minimum loss of all models on the short length c as a baseline, and only losses smaller than the baseline will be effectively counted towards the relative loss. This enables fair comparison between multiple models because all models with a long context length c' need to improve over the same baseline. Sometimes we only care about those positions where the baseline performs poorly (which means short-term dependency with context length c is not sufficient), so given a ratio parameter r, we define the set $\mathcal{T}$ is the above equation as

$$\mathcal{T} = \text{top-}r \text{ positions } t \text{ with largest } b(c,t)$$

The *relative gain* is subsequently defined as the relative perplexity reduction:

$$g_i(c,c') = \frac{\exp f_i(c,c) - \exp f_i(c,c')}{\exp f_i(c,c)}$$

Given a step size Δ, we then use an algorithm to find the RECL by thresholding the relative gain:

1. Set initial short context length c, and long context length $c' = c + \Delta$
2. Compute $g_i(c,c')$. If $g_i(c,c') < 0.01$, return RECL $= c$. If $g_i(c,c') \geq 0.01$, set $c = c', c' = c + \Delta$ and go to step 1.

In Figure 2.4, we visualize the unnormalized relative perplexity gains $(\exp f_i(c,c) - \exp f_i(c,c'))$ with various pairs of (c,c') when $r = 0.1$. It is clear that Transformer-XL has a longer RECL compared to RNNs and other baselines because the relative gains are substantially larger.

For reference, we plot the perplexities with varying context lengths in Figure 2.5. The y-axis denotes the "normal" perplexity (not calibrated by baselines).

Generated Text

In this section, we present some generated text from our best model trained the Wikitext-103 dataset. Since it is difficult to quantitatively evaluate unconditional text generation due to the lack of effective evaluation metrics, we focus on qualitative case studies. Trained only on WikiText-103 which is medium-sized, Transformer-XL is already able to generate relatively coherent articles with thousands of tokens without manual cherry picking, despite minor flaws. We seed our Transformer-XL with a context of at most 512 consecutive tokens randomly sampled from the test set of Wikitext-103. Then, we run Transformer-XL to generate a *pre-defined* number of tokens (500 or 1,000 in our case). For each generation step, we first find the top-40 probabilities of the next-step distribution and sample from top-40 tokens based on the re-normalized distribution. To help reading, we detokenize the context, the generated text and the reference text. Three generated examples are shown in Tables 2.11, 2.12, and 2.13. Note that we do not perform any cherry picking and present the first three examples we generate. In the text, "= text =", "= = text = =" and "= = = text = = =" denote the Wikipedia page tile, section title and subsection title, respectively, due to the original data preprocessing procedure of Wikitext-103 [102].

As we can see, though only trained on 100M tokens, Transformer-XL is a strong model at generating long text articles, particularly in the following aspects:

- Transformer-XL is able to structurally maintain the sectional arrangement of Wikipedia.
- Transformer-XL manages to semantically stay on the same topic throughout the course of generation.
- Long-range references are common in the generated text.
- Transformer-XL often generates novel content that is not present in the training data.

For more detailed explanation of the interesting observations in each example, please refer to the corresponding caption.

Despite the overall excellence of the generation quality, the model can only perceive the seed context and hallucinate what to generate based on the limited knowledge (100M tokens only) it is trained on. As a result, the generated text sometimes looks clearly relevant but not close enough or to the point compared to what human writer would do. That said, we believe this issue is mostly a problem of limited training data size and could be alleviated by using a larger training set.

Context:
Kershaw started the 2010 season by posting a 3.07 ERA in April, but did so by walking 22 batters in 29 innings. On May 4, he had his worst start of his career against the Milwaukee Brewers at Dodger Stadium, throwing just 57 pitches in 11 / 3 innings, while retiring only four of the 13 batters he faced — including the pitcher. He was booed loudly upon being pulled from the game. Kershaw said after the game, " I didn't give our team any kind of chance. It's just not a good feeling to let your teammates down, let everybody down. It stings, it hurts. I 've got to figure things out. " Kershaw rebounded his next start by pitching an 8 inning two-hitter and out-dueling the then undefeated Ubaldo Jiménez. He credited his control of the slider being the major turning point for him. Later in the season, he was suspended for five games after hitting Aaron Rowand of the Giants with a pitch in a game on July 20. The incident occurred after both teams were given a warning following Giants ace Tim Lincecum hitting Matt Kemp earlier in the game. He threw his first career complete game shutout on September 14, 2010 also against San Francisco and finished the season with a record of 13 – 10 and a 2.91 ERA in 32 starts, pitching 2041 / 3 innings and recording 212 strikeouts.
= = = = 2011 season: 1st Cy Young Award = = = =
After finishing the 2010 season strong, the Dodgers named Kershaw as the Opening Day Starter for the 2011 season. On May 29, he pitched the second complete-game shutout of his career, striking out 10 while winning a two-hitter against the Florida Marlins, 8 – 0; he also had two singles and an RBI, scoring twice in the game. He produced his third career shutout on June 20, a two-hit, 11-strikeout effort against the Detroit Tigers. Kershaw became the first Dodgers starter to strike out the side in the 9th inning since Sandy Koufax's perfect game. In his next start, on June 26, Kershaw pitched another complete game (against the Los Angeles Angels of Anaheim). He became the first Dodger starter to have back-to-back complete game victories since Jeff Weaver in the 2005 season and the first Dodger to have double-digit strikeouts in consecutive starts since Chan-Ho Park in the 2000 season. He was awarded the National League Player of the Week award for the week of June 20 – 26 as a result of those two starts. Midway through June, Kershaw had amassed 32 career victories

XL Generation:
, tying him with Luis Castillo for the third best start total in the league and trailing only Johnny Bench and Eddie Mathews (37). The Dodgers signed Kershaw to a one-year deal on July 14, 2011, with a player option for 2012, in a deal worth just under $ 300,000. The Dodgers entered the 2011 season as the major league leaders, and Kershaw had the best record in the National League in saves (39), games started (78) and walks (120); he also had the lowest win / loss ratio in the league (1.87), and he had the worst on-base percentage (.349). He had the best ERA (2.00), most wins (16), or most shutouts (29) in his career (25), while leading the league in losses per nine innings pitched (4.01). Following the 2011 season, the Dodgers traded Kershaw to the Kansas City Royals on July 29 for pitcher Javier Torres.
= = = Kansas City Royals = = =
= = = = 2012 season = = = =
During spring training, Kershaw played very well. He was selected to spring training as a relief pitcher for the Royals for the 2012 season. After an injury to closer Javier Vázquez, he was activated on April 29 to replace Matt Holliday in the Royals ' starting rotation. In his only start with the Royals, on August 6, 2012, Kershaw struck out five batters in seven innings pitched to help the Royals to their first victory in franchise history. On September 27, 2012, it appeared Kershaw was going to pitch a complete game shutout against the Detroit Tigers, but did not manage to do so since the Tigers won 3 – 1. At the conclusion of the season, Kershaw was named Major League Baseball's Most Valuable Player, was chosen to the All-Star Game at Busch Stadium and was named to the All-Star Game as the starting pitcher at shortstop. The Royals announced on February 4, 2013 that Kershaw would spend 2013 as starting pitcher, though he was expected to miss the season as a result of a shoulder injury.
= = = = 2013 season = = = =
On May 17, 2013, Kershaw sustained another back injury and did not start in August and October 2013. He appeared in 22 starts, all starts, finishing with a strikeout-to-walk ratio of 1.50 and a 2.91 ERA. He also had the third most strikeouts in the league: 10. On May 20, 2013, he

Reference:
, a 3.15 ERA and 593 career strikeouts in 568.2 innings. According to the Elias Sports Bureau, Kershaw was the first 23-year-old pitcher to have that many victories, an ERA that low and an average of more than one strikeout per inning since ERA became an official statistic in 1910. Kershaw was selected to the National League team for the 2011 Major League Baseball All-Star Game, his first All-Star selection. In the month of July, Kershaw was 4 – 1 with a 2.02 ERA and NL-leading 45 strikeouts, earning him the National League Pitcher of the Month Award. On August 23, he struck out Matt Holliday of the St. Louis Cardinals for his 200th strikeout of the season and became the 10th Dodger pitcher to record back-to-back 200 strikeout seasons and the first since Chan-Ho Park did it in the 2001 season. Kershaw finished the 2011 season by leading the NL with 21 wins, 248 strikeouts and a 2.28 ERA, winning the NL pitching Triple Crown, the first Triple Crown winner since Jake Peavy of the 2007 San Diego Padres and the first Dodger since Sandy Koufax won it in the 1966 season. Justin Verlander of the Detroit Tigers won the American League Triple Crown the same season, marking the first major-league season since 1924 to feature Triple Crown-winning pitchers in both leagues. Kershaw's 21 wins were the most by a Dodger pitcher since Orel Hershiser won 23 during the 1988 season. His ERA was the lowest by a Dodger since Hershiser's 2.03 in the 1985 season, his strikeouts were the most by a Dodger since Koufax's 317 in 1966 and his 233 1 / 3 innings pitched were the most since Chan Ho Park pitched 234 in 2001. Since 1965 when Koufax did it, Peavy and Kershaw are only two pitchers in the National League have led the league in wins, strikeouts, ERA, and WHIP (walks plus hits per inning pitched). Kershaw also became just the second <unk> to have a 240-plus strikeouts in a season before the age of 24, joining Vida Blue. After the season, Kershaw was awarded the Warren Spahn Award as the best left-handed pitcher in 2011, the Players Choice Award for Most Outstanding National League pitcher, the Gold Glove Award as the top fielding pitcher in the NL and the Sporting News (TSN) National League Pitcher of the Year. He was additionally selected as the starting pitcher for the TSN NL All-Star Team. On November 17, he was honored with the National League Cy Young Award, making him the youngest Cy Young winner since Dwight Gooden

Table 2.11: Example 1 – 500 tokens generated by XL using a snippet from the Wikitext-103 test set as initial context. The sample is randomly generated without any cherry picking.

Original Wikipedia page: `https://en.wikipedia.org/wiki/Clayton_Kershaw`

There are many interesting observations from this example:

- Firstly, Kershaw never went to Royals in real life. Despite that, Transformer-XL stays on the fully imagined topic and keeps hallucinating the experience of Kershaw in Royals across the generated text.
- Secondly, notice that XL correctly tracks the chronological order from 2011 to 2012 and to the finally 2013 season in the section titles.
- In addition, notice that Transformer-XL accurately uses the the phrase "another back injury" in the 2013 season paragraph, since it has talked about one earlier injure in the 2012 season. This shows again Transformer-XL's ability of capturing long-term dependency.

Context:
= = Distribution = =
Species range across the Neotropics from Mexico in the north to Bolivia, Paraguay, and southern Brazil in the south. According to <unk> and coauthors, three species are found in Mexico, four in Central America, and 62 in South America. Three species are present in the Caribbean — two in Trinidad and Tobago, along the southern edge of the region, and one in Haiti.
= = Habitat and ecology = =
<unk> includes both large trees and small acaulescent palms which occupy a number of different ecological niches. Dense stands of some of the larger species are conspicuous elements on the landscape, while smaller species are found in both in the forest understorey and in savannas. Disturbance has been implicated in the formation of vegetation dominated by large <unk> species. In seasonally dry Amazonian forests the density of large adult A. <unk> palms was correlated with canopy openness; the species also dominates savannas formed by repeated forest fires in Trinidad and Tobago. <unk> speciosa forms pure stands in many parts of Brazil where natural forest vegetation has been cleared. Similarly, stands of A. <unk> in Bahia, Brazil (which are cultivated for <unk> fibre) are managed using fire — the seedlings survive cutting and burning, and are able to dominate burned forest patches. The fruit are dispersed by animals; fruit which are not dispersed frequently suffer seed predation by <unk> beetles. Certain species of <unk> have been mentioned as examples of " anachronistic " species which are adapted for dispersal by now-extinct Pleistocene megafauna. On <unk> Island, <unk>, in the Brazilian Amazon, <unk> <unk> fruit were consumed by tapirs, collared peccaries, deer and primates. Rodents, including agoutis, fed upon the fruit and, as the fruit availability declined, they fed on the seeds. Other dispersers of <unk> fruit include Crested <unk> which consume the fruit and disperse the seeds of A. <unk> in the Brazilian Pantanal.
= = Uses = =
<unk> species have a long history of human utilisation. <unk> <unk> <unk> seeds have been found in archaeological sites in Colombia dating back to 9000 BP. A variety of species remain important sources of edible oil, thatch, edible seeds and fibre. The leaves of <unk> <unk> and A. <unk> are used extensively for thatching. Several species are oil palms, with A. speciosa among the most important economically. Products extracted from A. speciosa were reported to support over 300,000 households in the Brazilian state of Maranhão in 2005, and in 1985 it was estimated to support over 450,000 households throughout the Brazil. <unk> fibres, extracted from the leaf bases of A. <unk>, are commercially important, and generated about US $

XL Generation:	**Reference:**
700,000 in 2013 on the black market. In 2004 the plant was reported to be responsible for a net profit of around US $ 3 million in Brazil. A 2012 study found A. speciosa to cause agricultural production at least 2 – 3 times greater than those achieved without prior knowledge of land management. = The Tale of Mrs. Tittlemouse = The Tale of Mrs. Tittlemouse is a children's book written and illustrated by H. M. Potter. It was first published in February 1910 by Frederick Warne & Co. and included illustrations by Frederick Warne. Mrs. Tittlemouse was intended to be a children's book, rather than a musical, and the tale describes her attempts to find her own home. Mrs. Tittlemouse is an average kitten and the tale follows a mischievous rabbit named Mr. Tittlemouse who tries to kill her in an attempt to get a ride on a horse. Potter later wrote of the rabbit and her attempts to kill her, " If [she] were a fox, I should like to have been able to show her the way. " Potter's work was inspired by The Tale of Little Black Sambo and her two nieces, Petunia and Petunia. It was the first book in the Potter family to be published in a hardback format, and was also the first book in the Peter Rabbit series. The tale was released on 13 November 1910 and was accompanied by a greeting card by Potter's brother-in-law, Frederick Warne. The tale was well received by critics and children, and sold over 600,000 copies in its first few years. In the first year of its publication, Potter's The Tale of Mrs. Tittlemouse won children's book awards in England, America and Australia. It was selected as a " recommended reading " by children in the US and Canada. = = Plot summary = = The tale opens with the pet rabbit Mrs. Tittlemouse wandering through a forest in a small litter of four. He is shot and badly injured by a deer after she attempts to escape. Mrs. Tittlemouse decides that she wants to find her own home, because she is eager to go on her own. She goes alone to the farm where she makes a little money by selling a few seeds and building a small cabin in the woods. She is approached by a wealthy hunter named Mr. Tittlemouse, who tries to kill her but Mrs. Tittlemouse kills him by stuffing a rope into his nose and killing him. She is rescued by Mr. Tittlemouse's wife Ruth, but Mrs. Tittlemouse then leaves the woodland with the baby. When she is spotted by	20 million in annual income to Brazilian farmers in 1996. = The Heart of Ezra Greer = The Heart of Ezra Greer is a 1917 American silent drama film produced by the Thanhouser Company and directed by Emile <unk>. The film focuses on Ezra Greer, a successful middle-aged man who searches for his college age daughter, Mary. The wayward Mary was romanced and abandoned by Jack <unk>, later bearing his child. Once Ezra becomes broke he finds employment as the valet for Jack <unk>. After Jack's engagement to a cabaret girl, Mary becomes upset and leaves her child at Jack's home. Contrary to Jack's wishes, Ezra keeps the child and Jack ultimately reveals that the child is his own. Ezra convinces Jack to make things right and Ezra convinces the cabaret girl to leave Jack. After a carriage accident in which the baby is injured, Ezra and Jack rush to the hospital and find Mary as a nurse crying over the child. The film ends with the marriage of Jack and Mary. The film was released by Pathé on October 7, 1917. The film was the final release from Thanhouser and was deemed to be an average film by most reviewers. Criticism for the film hinged on far-fetched coincidences to drive the plot. The film is presumed lost. = = Plot = = The film follows Ezra Greer, a middle-aged man who has worked hard since his youth. He cares deeply for his motherless daughter, Mary, but was unable to attend the annual commencement at her co-educational college. He awaits for her to return from college, but Mary leaves with her romantic interest, Jack <unk>. On promise of marriage and wealth, Mary is romanced and gives birth to a fatherless child. Without word from his daughter, Ezra resigns from his job and attempts to seek her out and finds a poor motherless child, Marie. With Ezra's money exhausted he seeks employment and finds it as the valet of Jack. One day, Mary seeks an announcement of Jack's engagement to a cabaret girl known as " The Baby Vamp ". Bitter over the prospect of her child's future, she leaves the child at Jack's home during his absence with a note. Jack orders Ezra to take the baby to an orphanage, but Marie begs Ezra to keep him. After continually seeing the child, Jack is overcome with remorse and explains to Ezra and seeks his advice. Not knowing he was making the case for his own daughter, Ezra convinces Jack to seek out Mary and forget the Baby Vamp. The Baby

Table 2.12: Example 2 – 500 tokens generated by XL using a snippet from the Wikitext-103 test set as initial context. The sample is randomly generated without any cherry picking.

Original Wikipedia page: `https://en.wikipedia.org/wiki/The_Tale_of_Mrs._Tittlemouse`.

This example exhibit some additional interesting properties of Transformer-XL:

- After finishing the last paragraph of the seed context, both the reference and generated text start a new topic (i.e., Wikipedia page), as marked by the single "= title =" line. This suggests the model has the ability of identifying the end of a topic / page, and randomly starting with a new topic.
- Even more interestingly, a newly-started page is on a book called "The Tale of Mrs. Tittlemouse". Transformer-XL manages to copy the same book title and some related information from the training set, but hallucinates *novel* content of the book. This demonstrates a degree of generalization instead of memorization. Please refer to the original book content at the Wikipedia page.

Context:
= Battle of Dürenstein =
The Battle of Dürenstein (also known as the Battle of <unk>, Battle of <unk> and Battle of <unk>; German: <unk> bei <unk>), on 11 November 1805 was an engagement in the Napoleonic Wars during the War of the Third Coalition. Dürenstein (modern <unk>) is located in the <unk> Valley, on the River Danube, 73 kilometers (45 mi) upstream from Vienna, Austria. The river makes a crescent-shaped curve between <unk> and nearby Krems an der Donau and the battle was fought in the flood plain between the river and the mountains. At Dürenstein a combined force of Russian and Austrian troops trapped a French division commanded by Théodore Maxime Gazan. The French division was part of the newly created VIII Corps, the so-called Corps Mortier, under command of Édouard Mortier. In pursuing the Austrian retreat from Bavaria, Mortier had over-extended his three divisions along the north bank of the Danube. Mikhail <unk> Kutuzov, commander of the Coalition force, enticed Mortier to send Gazan's division into a trap and French troops were caught in a valley between two Russian columns. They were rescued by the timely arrival of a second division, under command of Pierre Dupont de l 'Étang. The battle extended well into the night. Both sides claimed victory. The French lost more than a third of their participants, and Gazan's division experienced over 40 percent losses. The Austrians and Russians also had heavy <unk> to 16 <unk> perhaps the most significant was the death in action of Johann Heinrich von Schmitt, one of Austria's most capable chiefs of staff. The battle was fought three weeks after the Austrian capitulation at Ulm and three weeks before the Russo-Austrian defeat at the Battle of Austerlitz. After Austerlitz Austria withdrew from the war. The French demanded a high indemnity and Francis II abdicated as Holy Roman Emperor, releasing the German states from their allegiance to the Holy Roman Empire.
= = Background = =
In a series of conflicts from 1803-15 known as the Napoleonic Wars, various European powers formed five coalitions against the First French Empire. Like the wars sparked by the French Revolution (1789), these further revolutionized the formation, organization and training of European armies and led to an unprecedented militarization, mainly due to mass conscription. Under the leadership of Napoleon, French power rose quickly as the Grande Armée conquered most of Europe, and collapsed rapidly

XL Generation:
after the French conquest of Italy. This victory marked the beginning of the Second Coalition. Napoleon's rapid advance caused Austria – Russia, Britain and Russia to make peace negotiations. The Russian army under Prince Mikhail Mikhailovich Mikhailovich Chaffee, commander of the Imperial Russian Army in Switzerland, was able to drive off the Austrians in the Battle of Stockach (1805) and to take Pressburg (modern Ö3 Austria) from the Austrians. At an early stage the Austro-Hungarian fleet had been damaged in a storm off Cape Matapan; this allowed the Allies to launch a full-scale invasion of Italy in February 1805. The Battle of Austerlitz was the decisive French victory against Napoleon and the largest naval battle in history, involving more modern-day European navies. The French military situation worsened as Napoleon faced several threats related to his newly formed Army of Europe, particularly Russia itself, which was now on the verge of collapse. The Russian general and friend of Napoleon, Alexander, had been dissatisfied with the conduct of the Austrians; he was still in conflict with Emperor Napoleon, the French Republic's king, who had declared war on Austria during the summer of 1804. With the war over, Napoleon decided to launch a second invasion of Italy in October 1805.
= = Prelude = =
In July 1805, the French 1st Army entered southern Italy. The army, under the command of Marshal Marmont, were reinforced by a few battalions of infantry under Claude General Auguste de Marmont at the town of Philippsburg and another battalion at Belluno. On 17 September 1805, the army marched from Belluno towards Krems. By 29 September, they had reached Belluno and conducted its advance against a small Austrian force. By 31 September, the whole force had been reinforced by a brigade from the Army of Tyrol under the command of Pierre Augereau. The Austrians were now under the command of Marshal Jean Victor Marie Moreau, a member of the Directory. Moreau had taken command of the Austrian invasion force in the spring of 1805. His command included the VI Corps commanded by Jean Baptiste Drouet de Ney and the VI Corps commanded by Generals Jean Victor Marie Moreau and Joseph Souham. Ney's corps consisted of the III. Corps and VI. Corps, which consisted of the III Corps and VI. Corps, located in the Austrian Netherlands, was commanded by Friedrich Joseph, Count Baillet de Latour. Moreau's army consisted of six divisions and several associated brigades.

Reference:
after the disastrous invasion of Russia in 1812. Napoleon's empire ultimately suffered complete military defeat in the 1813 – 14 campaigns, resulting in the restoration of the Bourbon monarchy in France. Although Napoleon made a spectacular return in 1815, known as the Hundred Days, his defeat at the Battle of Waterloo, the pursuit of his army and himself, his abdication and banishment to the Island of Saint Helena concluded the Napoleonic Wars.
= = Danube campaign = =
From 1803-06 the Third Coalition fought the First French Empire and its client states (see table at right). Although several naval battles determined control of the seas, the outcome of the war was decided on the continent, predominantly in two major land operations in the Danube valley: the Ulm campaign in the upper Danube and the Vienna campaign, in the middle Danube valley. Political conflicts in Vienna delayed Austria's entry into the Third Coalition until 1805. After hostilities of the War of the Second Coalition ended in 1801, Archduke <unk> emperor's <unk> advantage of the subsequent years of peace to develop a military restructuring plan. He carefully put this plan into effect beginning in 1803 – 04, but implementation was incomplete in 1805 when Karl Mack, Lieutenant Field Marshal and Quartermaster-General of the Army, implemented his own restructuring. Mack bypassed Charles ' methodical approach. Occurring in the field, Mack's plan also undermined the overall command and organizational structure. Regardless, Mack sent an enthusiastic report to Vienna on the military's readiness. Furthermore, after misreading Napoleon's maneuvers in Württemberg, Mack also reported to Vienna on the weakness of French dispositions. His reports convinced the war party advising the emperor, Francis II, to enter the conflict against France, despite Charles ' own advice to the contrary. Responding to the report and rampant anti-French fever in Vienna, Francis dismissed Charles from his post as generalissimo and appointed his <unk> brother-in-law, Archduke Ferdinand, as commander. The inexperienced Ferdinand was a poor choice of replacement for the capable Charles, having neither maturity nor aptitude for the assignment. Although Ferdinand retained nominal command, day-to-day decisions were placed in the hands of Mack, equally ill-suited for such an important assignment. When Mack was wounded early in the campaign, he was unable to take full charge of the army. Consequently, command further devolved to Lieutenant Field Marshal Karl Philipp, Prince of Schwarzenberg, an able cavalry officer but inexperienced in the command of such a large army.

= = Aftermath = = = = = First Coalition forces = = = On 9 October 1805 the French Army of the Danube was attacked by an Austrian army under Archduke Charles at the Battle of Austerlitz. Although Charles and Charles had not had much time to regroup, on 10 October, he launched his attack on the Polish forces under Friedrich Joseph, Count of Lauenburg. After three days, Charles' army captured Lauenburg. The French forces pursued the Austrians to the Silesian border, where they encountered strong Austrian resistance. These conflicts forced the Austrians to retreat into Tyrol and Austria agreed to a truce. The Austrian army, commanded by Wenzel Anton Karl, Count of Merveldt, was reduced to around 10,000 men. It was initially planned that Archduke Charles would launch a counter-attack against the French army on the same day, as Napoleon had hoped, but this was not carried out. On 25 October, Merveldt left Styria for Tyrol. On the same day, Austria launched its new offensive against the French at Ulm. Charles withdrew his army from the region for a third time at the Battle of Elchingen, under the overall command of the Austrian generals, Ferdinand and Friedrich Wilhelm of Jülich-Cleves-Berg. To prevent Archduke Charles from escaping from the battlefield, the commander of the Habsburg army, Archduke Charles, planned to occupy the fortress Linz; instead, he decided to force Franz von Hipper to surrender the city. However, as Charles moved to the south, Moreau arrived on the scene with additional soldiers – including the entire Imperial Guard – and defeated the Austrians at the Battle of Hohenlinden on 28 October. The loss of Linz resulted in Austria's complete defeat at Hohenlinden. In the meantime, the French Army of Observation and Preparedness was reorganized into the Army of the Danube under Feldzeugmeister (Colonel-General) Friedrich Freiherr von Hotze. The army was composed of the I, IV, VI, VI, VII, VIII and IX Corps. With reinforcements from Italy and France, it formed new battalions, companies, and squadrons in the Austrian army. On 17 November 1804, at the Battle of Jena-Auerstadt the Army of Silesia and the Army of Silesia joined forces, but by the time that the French approached Vienna, the Prussians had already surrendered. As the Austrians did not want to allow the war to continue, they decided to abandon their territories in the north and move their army to the north and west, cutting off Charles from Vienna. The Battle of Warsaw was fought on 23 November 1805 between the French Army of the Danube and the Austrian Army of Styria in the vicinity of Warsaw and Pressburg (modern Trnava, Slovakia). At that time Habsburg forces	= = = Road to Ulm = = = The campaign in the upper Danube valley began in October, with several clashes in Swabia. Near the Bavarian town of Wertingen, 40 kilometers (25 mi) northwest of Augsburg, on 8 October the 1st Regiment of dragoons, part of Murat's Reserve Cavalry Corps, and grenadiers of Lannes ' V Corps surprised an Austrian force half its size. The Austrians were arrayed in a line and unable to form their defensive squares quickly enough to protect themselves from the 4,000 dragoons and 8,000 grenadiers. Nearly 3,000 Austrians were captured and over 400 were killed or wounded. A day later, at another small town, <unk> south of the Danube <unk> French 59th Regiment of the Line stormed a bridge over the Danube and, humiliatingly, chased two large Austrian columns toward Ulm. The campaign was not entirely bad news for Vienna. At Haslach, Johann von Klenau arranged his 25,000 infantry and cavalry in a prime defensive position and, on 11 October, the overly confident General of Division Pierre Dupont de l'Étang attacked Klenau's force with fewer than 8,000 men. The French lost 1,500 men killed and wounded. Aside from taking the Imperial Eagles and <unk> of the 15th and 17th Dragoons, Klenau's force also captured 900 men, 11 guns and 18 ammunition wagons. Klenau's victory was a singular success. On 14 October Mack sent two columns out of Ulm in preparation for a breakout to the north: one under Johann Sigismund Riesch headed toward Elchingen to secure the bridge there, and the other under Franz von Werneck went north with most of the heavy artillery. Recognizing the opportunity, Marshal Michel Ney hurried the rest of his VI Corps forward to re-establish contact with Dupont, who was still north of the Danube. In a two-pronged attack Ney sent one division to the south of Elchingen on the right bank of the Danube. This division began the assault at Elchingen. At the same time another division crossed the river to the east and moved west against Riesch's position. After clearing Austrian pickets from a bridge, the French attacked and captured a strategically located abbey at the top of the hill at bayonet point. The Austrian cavalry unsuccessfully tried to fend off the French, but the Austrian infantry broke and ran. In this engagement alone, the Austrians lost more than half their reserve artillery park, 6,000 (out of 8,000 total participants) dead, wounded or captured and four colors. Reisch's column also failed to destroy the bridges across the Danube. Napoleon's lightning campaign exposed the Austrian indecisive command structure and poor supply apparatus. Mack

Table 2.13: Example 3 – 1,000 tokens generated by XL using a snippet from the Wikitext-103 test set as initial context. The sample is randomly generated without any cherry picking.

Original Wikipedia page: `https://en.wikipedia.org/wiki/Battle_of_D%C3%BCrenstein`.

- Although this example is significantly longer, we can see that Transformer-XL is still able to stay on the same topic and makes up non-existing stories about the Napoleon wars.
- Notably, from the second section on, the generated text correctly follows a fine-grained chronological order *on the level of month and day* to narrate events in 1805, except a mistake (1804 instead of 1805) near the end of the paragraph. To ease reading which we have highlighted all the date related phrases by magenta in the generation.

2.2 XLNet: Generalized Autoregressive Pretraining

2.2.1 Motivation

Unsupervised representation learning has been highly successful in the domain of natural language processing [27, 34, 100, 122, 126]. Typically, these methods first pretrain neural networks on large-scale unlabeled text corpora, and then finetune the models or representations on downstream tasks. Under this shared high-level idea, different unsupervised pretraining objectives have been explored in literature. Among them, autoregressive (AR) language modeling and autoencoding (AE) have been the two most successful pretraining objectives.

AR language modeling seeks to estimate the probability distribution of a text corpus with an autoregressive model [27, 122, 126]. Specifically, given a text sequence $\mathbf{x} = (x_1, \cdots, x_T)$, AR language modeling factorizes the likelihood into a forward product $p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t \mid \mathbf{x}_{<t})$ or a backward one $p(\mathbf{x}) = \prod_{t=T}^{1} p(x_t \mid \mathbf{x}_{>t})$. A parametric model (e.g. a neural network) is trained to model each conditional distribution. Since an AR language model is only trained to encode a uni-directional context (either forward or backward), it is not effective at modeling deep bidirectional contexts. On the contrary, downstream language understanding tasks often require bidirectional context information. This results in a gap between AR language modeling and effective pretraining.

In comparison, AE based pretraining does not perform explicit density estimation but instead aims to reconstruct the original data from corrupted input. A notable example is BERT [34], which has been the state-of-the-art pretraining approach. Given the input token sequence, a certain portion of tokens are replaced by a special symbol `[MASK]`, and the model is trained to recover the original tokens from the corrupted version. Since density estimation is not part of the objective, BERT is allowed to utilize bidirectional contexts for reconstruction. As an immediate benefit, this closes the aforementioned bidirectional information gap in AR language modeling, leading to improved performance. However, the artificial symbols like `[MASK]` used by BERT during pretraining are absent from real data at finetuning time, resulting in a pretrain-finetune discrepancy. Moreover, since the predicted tokens are masked in the input, BERT is not able to model the joint probability using the product rule as in AR language modeling. In other words, BERT assumes the predicted tokens are independent of each other given the unmasked tokens, which is oversimplified as high-order, long-range dependency is prevalent in natural language [31].

Faced with the pros and cons of existing language pretraining objectives, in this work, we propose XLNet, a generalized autoregressive method that leverages the best of both AR language modeling and AE while avoiding their limitations.

Related Work The idea of permutation-based AR modeling has been explored in [44, 164], but there are several key differences. Previous models are orderless, while XLNet is essentially order-aware with positional encodings. This is important for language understanding because an orderless model is degenerated to bag-of-words, lacking basic expressivity. The above difference results from the fundamental difference in motivation—previous models aim to improve density estimation by baking an "orderless" inductive bias into the model while XLNet is motivated by enabling AR language models to learn bidirectional contexts.

2.2.2 Approach

Background

In this section, we first review and compare the conventional AR language modeling and BERT for language pretraining. Given a text sequence $\mathbf{x} = [x_1, \cdots, x_T]$, AR language modeling performs pretraining by maximizing the likelihood under the forward autoregressive factorization:

$$\max_\theta \quad \log p_\theta(\mathbf{x}) = \sum_{t=1}^{T} \log p_\theta(x_t \mid \mathbf{x}_{<t}) = \sum_{t=1}^{T} \log \frac{\exp\left(h_\theta(\mathbf{x}_{1:t-1})^\top e(x_t)\right)}{\sum_{x'} \exp\left(h_\theta(\mathbf{x}_{1:t-1})^\top e(x')\right)}, \tag{2.1}$$

where $h_\theta(\mathbf{x}_{1:t-1})$ is a context representation produced by neural models, such as RNNs or Transformers, and $e(x)$ denotes the embedding of x. In comparison, BERT is based on denoising auto-encoding. Specifically, for a text sequence $\mathbf{x}$, BERT first constructs a corrupted version $\hat{\mathbf{x}}$ by randomly setting a portion (e.g. 15%) of tokens in $\mathbf{x}$ to a special symbol `[MASK]`. Let the masked tokens be $\bar{\mathbf{x}}$. The training objective is to reconstruct $\bar{\mathbf{x}}$ from $\hat{\mathbf{x}}$:

$$\max_\theta \quad \log p_\theta(\bar{\mathbf{x}} \mid \hat{\mathbf{x}}) \approx \sum_{t=1}^{T} m_t \log p_\theta(x_t \mid \hat{\mathbf{x}}) = \sum_{t=1}^{T} m_t \log \frac{\exp\left(H_\theta(\hat{\mathbf{x}})_t^\top e(x_t)\right)}{\sum_{x'} \exp\left(H_\theta(\hat{\mathbf{x}})_t^\top e(x')\right)}, \tag{2.2}$$

where $m_t = 1$ indicates x_t is masked, and H_θ is a Transformer that maps a length-T text sequence $\mathbf{x}$ into a sequence of hidden vectors $H_\theta(\mathbf{x}) = [H_\theta(\mathbf{x})_1, H_\theta(\mathbf{x})_2, \cdots, H_\theta(\mathbf{x})_T]$. The pros and cons of the two pretraining objectives are compared in the following aspects:

- **Independence Assumption**: As emphasized by the $\approx$ sign in Eq. (2.2), BERT factorizes the joint conditional probability $p(\bar{\mathbf{x}} \mid \hat{\mathbf{x}})$ based on an independence assumption that all masked tokens $\bar{\mathbf{x}}$ are separately reconstructed. In comparison, the AR language modeling objective equation 2.1 factorizes $p_\theta(\mathbf{x})$ using the product rule that holds universally without such an independence assumption.
- **Input noise**: The input to BERT contains artificial symbols like `[MASK]` that never occur in downstream tasks, which creates a pretrain-finetune discrepancy. Replacing `[MASK]` with original tokens as in [34] does not solve the problem because original tokens can be only used with a small probability — otherwise Eq. (2.2) will be trivial to optimize. In comparison, AR language modeling does not rely on any input corruption and does not suffer from this issue.
- **Context dependency**: The AR representation $h_\theta(\mathbf{x}_{1:t-1})$ is only conditioned on the tokens up to position t (i.e. tokens to the left), while the BERT representation $H_\theta(\mathbf{x})_t$ has access to the contextual information on both sides. As a result, the BERT objective allows the model to be pretrained to better capture bidirectional context.

Objective: Permutation Language Modeling

According to the comparison above, AR language modeling and BERT possess their unique advantages over the other. A natural question to ask is whether there exists a pretraining objective that brings the advantages of both while avoiding their weaknesses.

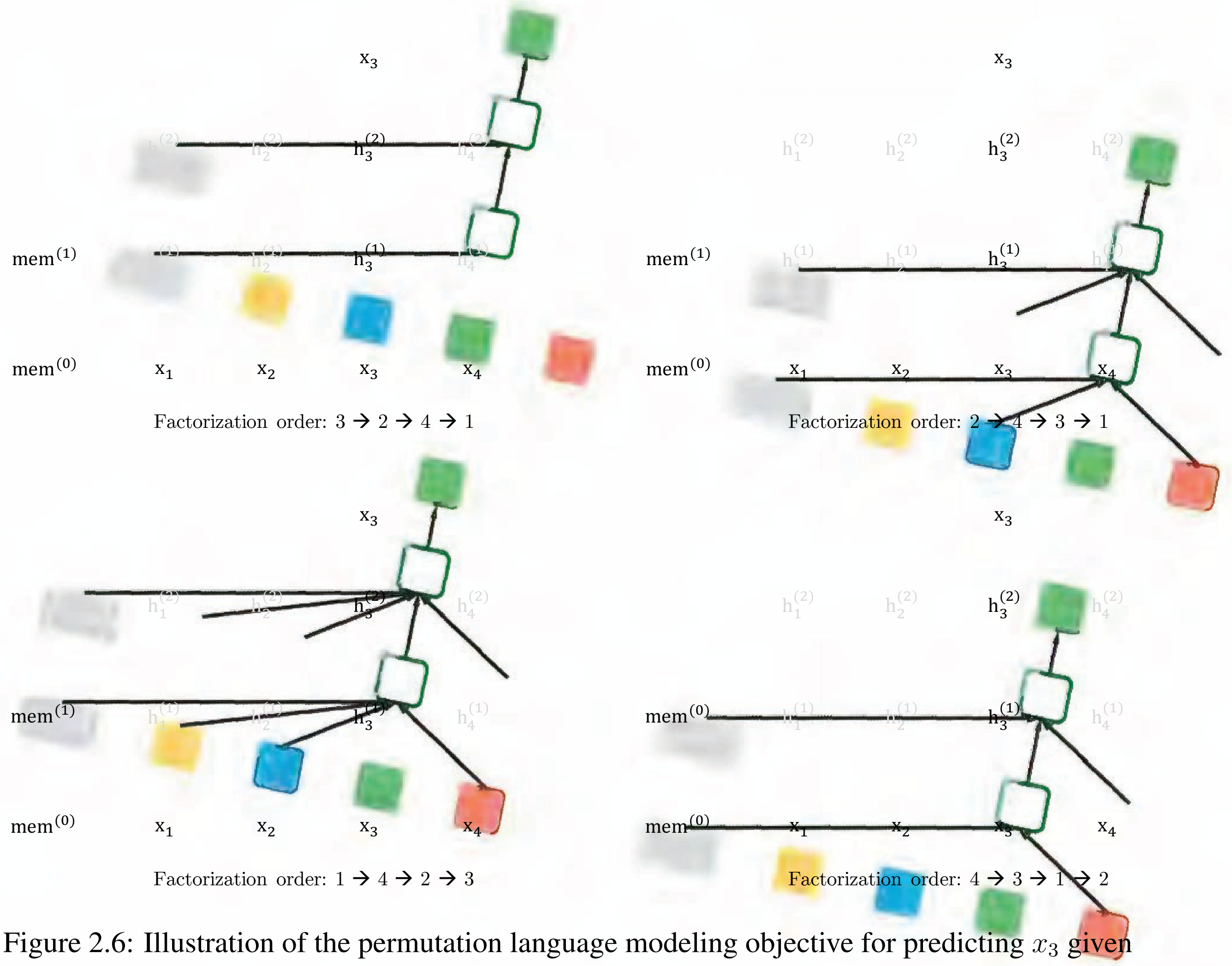

x3
mem(1)
mem(0)
x1
x2
x3
x4
Factorization order: 3 → 2 → 4 → 1
Factorization order: 2 → 4 → 3 → 1
Factorization order: 1 → 4 → 2 → 3
Factorization order: 4 → 3 → 1 → 2

Figure 2.6: Illustration of the permutation language modeling objective for predicting x_3 given the same input sequence $\mathbf{x}$ but with different factorization orders.

Borrowing ideas from orderless NADE [164], we propose the permutation language modeling objective that not only retains the benefits of AR models but also allows models to capture bidirectional contexts. Specifically, for a sequence $\mathbf{x}$ of length T, there are $T!$ different orders to perform a valid autoregressive factorization. Intuitively, if model parameters are shared across all factorization orders, in expectation, the model will learn to gather information from all positions on both sides.

To formalize the idea, let $\mathcal{Z}_T$ be the set of all possible permutations of the length-T index sequence $[1, 2, \ldots, T]$. We use z_t and $\mathbf{z}_{<t}$ to denote the t-th element and the first $t-1$ elements of a permutation $\mathbf{z} \in \mathcal{Z}_T$. Then, our proposed permutation language modeling objective can be expressed as follows:

$$\max_{\theta} \quad \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T}\left[\sum_{t=1}^{T} \log p_\theta(x_{z_t} \mid \mathbf{x}_{\mathbf{z}<t})\right]. \tag{2.3}$$

Essentially, for a text sequence $\mathbf{x}$, we sample a factorization order $\mathbf{z}$ at a time and decompose the likelihood $p_\theta(\mathbf{x})$ according to factorization order. Since the same model parameter θ is shared across all factorization orders during training, in expectation, x_t has seen every possible element $x_i \neq x_t$ in the sequence, hence being able to capture the bidirectional context. Moreover, as this objective fits into the AR framework, it naturally avoids the independence assumption and the pretrain-finetune discrepancy discussed in Section 2.2.2.

Remark on Permutation The proposed objective only permutes the factorization order, not the sequence order. In other words, we keep the original sequence order, use the positional encodings corresponding to the original sequence, and rely on a proper attention mask in Transformers to achieve permutation of the factorization order. Note that this choice is necessary, since the model will only encounter text sequences with the natural order during finetuning.

To provide an overall picture, we show an example of predicting the token x_3 given the same input sequence $\mathbf{x}$ but under different factorization orders in Figure 2.6.

Architecture: Two-Stream Self-Attention for Target-Aware Representations

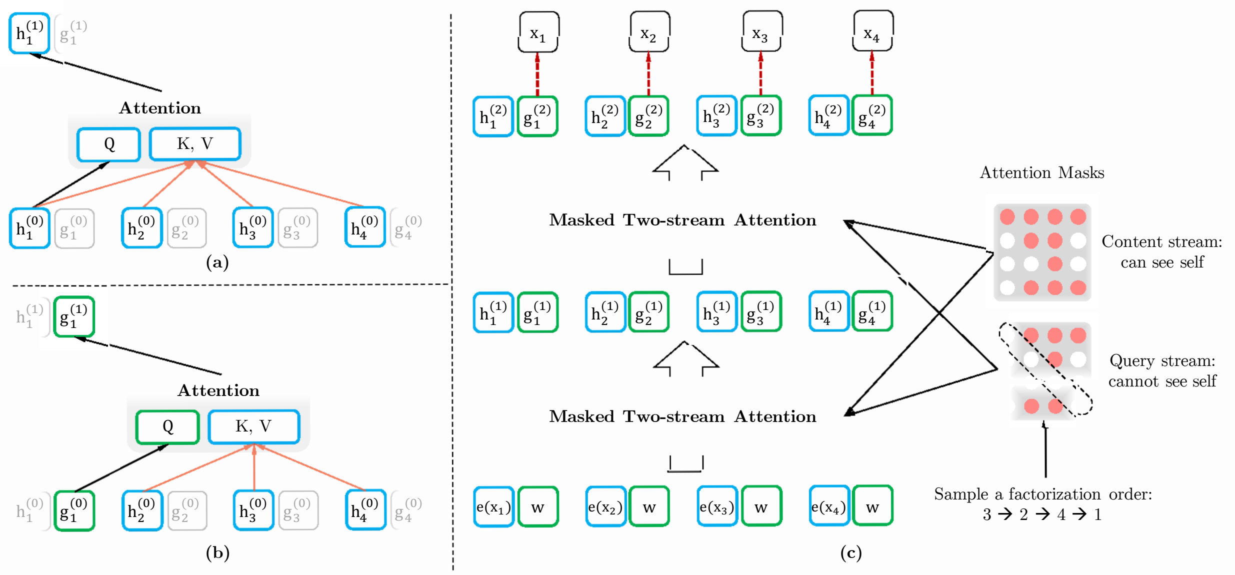

Attention
Q
K, V
(a)
(b)
(c)
Masked Two-stream Attention
Attention Masks
Content stream: can see self
Query stream: cannot see self
Sample a factorization order: 3 → 2 → 4 → 1
e(x₁)
w

Figure 2.7: (a): Content stream attention, which is the same as the standard self-attention. (b): Query stream attention, which does not have access information about the content x_{z_t}. (c): Overview of the permutation language modeling training with two-stream attention.

While the permutation language modeling objective has the desired properties, a naive implementation with standard Transformer parameterization may not work. To see the problem, assume we parameterize the next-token distribution $p_\theta(X_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}})$ using the standard Softmax formulation, i.e., $p_\theta(X_{z_t} = x \mid \mathbf{x}_{\mathbf{z}_{<t}}) = \frac{\exp\left(e(x)^\top h_\theta(\mathbf{x}_{\mathbf{z}_{<t}})\right)}{\sum_{x'} \exp\left(e(x')^\top h_\theta(\mathbf{x}_{\mathbf{z}_{<t}})\right)}$, where $h_\theta(\mathbf{x}_{\mathbf{z}_{<t}})$ denotes the hidden representation of $\mathbf{x}_{\mathbf{z}_{<t}}$ produced by the shared Transformer network after proper masking. Now notice that the representation $h_\theta(\mathbf{x}_{\mathbf{z}_{<t}})$ does not depend on which position it will predict, i.e., the value of z_t. Consequently, the same distribution is predicted regardless of the target position, which is not able to learn useful representations. Specifically, let's consider two different

permutations $\mathbf{z}^{(1)}$ and $\mathbf{z}^{(2)}$ satisfying the following relationship

$$\mathbf{z}^{(1)}_{<t} = \mathbf{z}^{(2)}_{<t} = \mathbf{z}_{<t} \quad \text{but} \quad z^{(1)}_t = i \neq j = z^{(2)}_t.$$

Then, substituting the two permutations respectively into the naive parameterization, we have

$$\underbrace{p_\theta(X_i = x \mid \mathbf{x}_{\mathbf{z}_{<t}})}_{z^{(1)}_t = i,\, \mathbf{z}^{(1)}_{<t} = \mathbf{z}_{<t}} = \underbrace{p_\theta(X_j = x \mid \mathbf{x}_{\mathbf{z}_{<t}})}_{z^{(1)}_t = j,\, \mathbf{z}^{(2)}_{<t} = \mathbf{z}_{<t}} = \frac{\exp\left(e(x)^\top h(\mathbf{x}_{\mathbf{z}_{<t}})\right)}{\sum_{x'} \exp\left(e(x')^\top h(\mathbf{x}_{\mathbf{z}_{<t}})\right)}.$$

Effectively, two different target positions i and j share exactly the same model prediction. However, the ground-truth distribution of two positions should certainly be different.

To avoid this problem, we propose to re-parameterize the next-token distribution to be target position aware:

$$p_\theta(X_{z_t} = x \mid \mathbf{x}_{z_{<t}}) = \frac{\exp\left(e(x)^\top g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)\right)}{\sum_{x'} \exp\left(e(x')^\top g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)\right)}, \tag{2.4}$$

where $g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)$ denotes a new type of representations which additionally take the target position z_t as input.

Two-Stream Self-Attention While the idea of target-aware representations removes the ambiguity in target prediction, how to formulate $g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)$ remains a non-trivial problem. Among other possibilities, we propose to "stand" at the target position z_t and rely on the position z_t to gather information from the context $\mathbf{x}_{\mathbf{z}_{<t}}$ through attention. For this parameterization to work, there are two requirements that are contradictory in a standard Transformer architecture:

1. to predict the token x_{z_t}, $g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)$ should only use the *position* z_t and not the *content* x_{z_t}, otherwise the objective becomes trivial;
2. to predict the other tokens x_{z_j} with $j > t$, $g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)$ should also encode the content x_{z_t} to provide full contextual information.

To resolve such a contradiction, we propose to use two sets of hidden representations instead of one:

- The content representation $h_\theta(\mathbf{x}_{\mathbf{z}_{\leq t}})$, or abbreviated as h_{z_t}, which serves a similar role to the standard hidden states in Transformer. This representation encodes *both* the context and x_{z_t} itself.
- The query representation $g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)$, or abbreviated as g_{z_t}, which only has access to the contextual information $\mathbf{x}_{\mathbf{z}_{<t}}$ and the position z_t, but not the content x_{z_t}, as discussed above.

Computationally, the first layer query stream is initialized with a trainable vector, i.e. $g_i^{(0)} = w$, while the content stream is set to the corresponding word embedding, i.e. $h_i^{(0)} = e(x_i)$. For each self-attention layer $m = 1, \dots, M$, the two streams of representations are *schematically*[1] updated with a shared set of parameters as follows (illustrated in Figures 2.7 (a) and (b)):

$$g_{z_t}^{(m)} \leftarrow \text{Attention}(\text{Q} = g_{z_t}^{(m-1)}, \text{KV} = \mathbf{h}^{(m-1)}_{\mathbf{z}_{<t}}; \theta), \quad \text{(query stream: use } z_t \text{ but cannot see } x_{z_t}\text{)}$$
$$h_{z_t}^{(m)} \leftarrow \text{Attention}(\text{Q} = h_{z_t}^{(m-1)}, \text{KV} = \mathbf{h}^{(m-1)}_{\mathbf{z}_{\leq t}}; \theta), \quad \text{(content stream: use both } z_t \text{ and } x_{z_t}\text{).}$$

[1]To avoid clutter, we omit the implementation details including multi-head attention, residual connection, layer normalization and position-wise feed-forward as used in Transformer(-XL).

where Q, K, V denote the query, key, and value in an attention operation [166]. The update rule of the content representations is exactly the same as the standard self-attention, so during finetuning, we can simply drop the query stream and use the content stream as a normal Transformer(-XL). Finally, we can use the last-layer query representation $g_{z_t}^{(M)}$ to compute Eq. (2.4).

Partial Prediction While the permutation language modeling objective equation 2.3 has several benefits, it is a much more challenging optimization problem due to the permutation and causes slow convergence in preliminary experiments. To reduce the optimization difficulty, we choose to only predict the last tokens in a factorization order. Formally, we split $\mathbf{z}$ into a non-target subsequence $\mathbf{z}_{\leq c}$ and a target subsequence $\mathbf{z}_{>c}$, where c is the cutting point. The objective is to maximize the log-likelihood of the target subsequence conditioned on the non-target subsequence, i.e.,

$$\max_{\theta} \quad \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T}\left[\log p_\theta(\mathbf{x}_{\mathbf{z}_{>c}} \mid \mathbf{x}_{\mathbf{z}_{\leq c}})\right] = \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T}\left[\sum_{t=c+1}^{|\mathbf{z}|} \log p_\theta(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}})\right]. \tag{2.5}$$

Note that $\mathbf{z}_{>c}$ is chosen as the target because it possesses the longest context in the sequence given the current factorization order $\mathbf{z}$. A hyperparameter K is used such that about $1/K$ tokens are selected for predictions; i.e., $|\mathbf{z}|/(|\mathbf{z}|-c) \approx K$. For unselected tokens, their query representations need not be computed, which saves speed and memory.

Incorporating Ideas from Transformer-XL

Since our objective function fits in the AR framework, we incorporate the state-of-the-art AR language model, Transformer-XL [31], into our pretraining framework, and name our method after it. We integrate two important techniques in Transformer-XL, namely the relative positional encoding scheme and the segment recurrence mechanism. We apply relative positional encodings based on the original sequence as discussed earlier, which is straightforward. Now we discuss how to integrate the recurrence mechanism into the proposed permutation setting and enable the model to reuse hidden states from previous segments. Without loss of generality, suppose we have two segments taken from a long sequence $\mathbf{s}$; i.e., $\tilde{\mathbf{x}} = \mathbf{s}_{1:T}$ and $\mathbf{x} = \mathbf{s}_{T+1:2T}$. Let $\tilde{\mathbf{z}}$ and $\mathbf{z}$ be permutations of $[1 \cdots T]$ and $[T+1 \cdots 2T]$ respectively. Then, based on the permutation $\tilde{\mathbf{z}}$, we process the first segment, and then cache the obtained content representations $\tilde{\mathbf{h}}^{(m)}$ for each layer m. Then, for the next segment $\mathbf{x}$, the attention update with memory can be written as

$$h_{z_t}^{(m)} \leftarrow \text{Attention}(\text{Q} = h_{z_t}^{(m-1)}, \text{KV} = \left[\tilde{\mathbf{h}}^{(m-1)}, \mathbf{h}_{\mathbf{z}_{\leq t}}^{(m-1)}\right]; \theta)$$

where $[.,.]$ denotes concatenation along the sequence dimension. Notice that positional encodings only depend on the actual positions in the original sequence. Thus, the above attention update is independent of $\tilde{\mathbf{z}}$ once the representations $\tilde{\mathbf{h}}^{(m)}$ are obtained. This allows caching and reusing the memory without knowing the factorization order of the previous segment. In expectation, the model learns to utilize the memory over all factorization orders of the last segment. The query stream can be computed in the same way. Finally, Figure 2.7 (c) presents an overview of the proposed permutation language modeling with two-stream attention.

Bidirectional Input Pipeline. Because Transformer-XL relies on reusing the hidden states from the previous batch as memory, our approach has to enforce an order between batches. The order can be either forward or backward, where each batch is the next or previous token sequence of the last batch. Our implementation allocates half of the batch size for each order. Note that autoencoding approaches like BERT do not have an order between batches, so the batches can be loaded uniformly random without dependency.

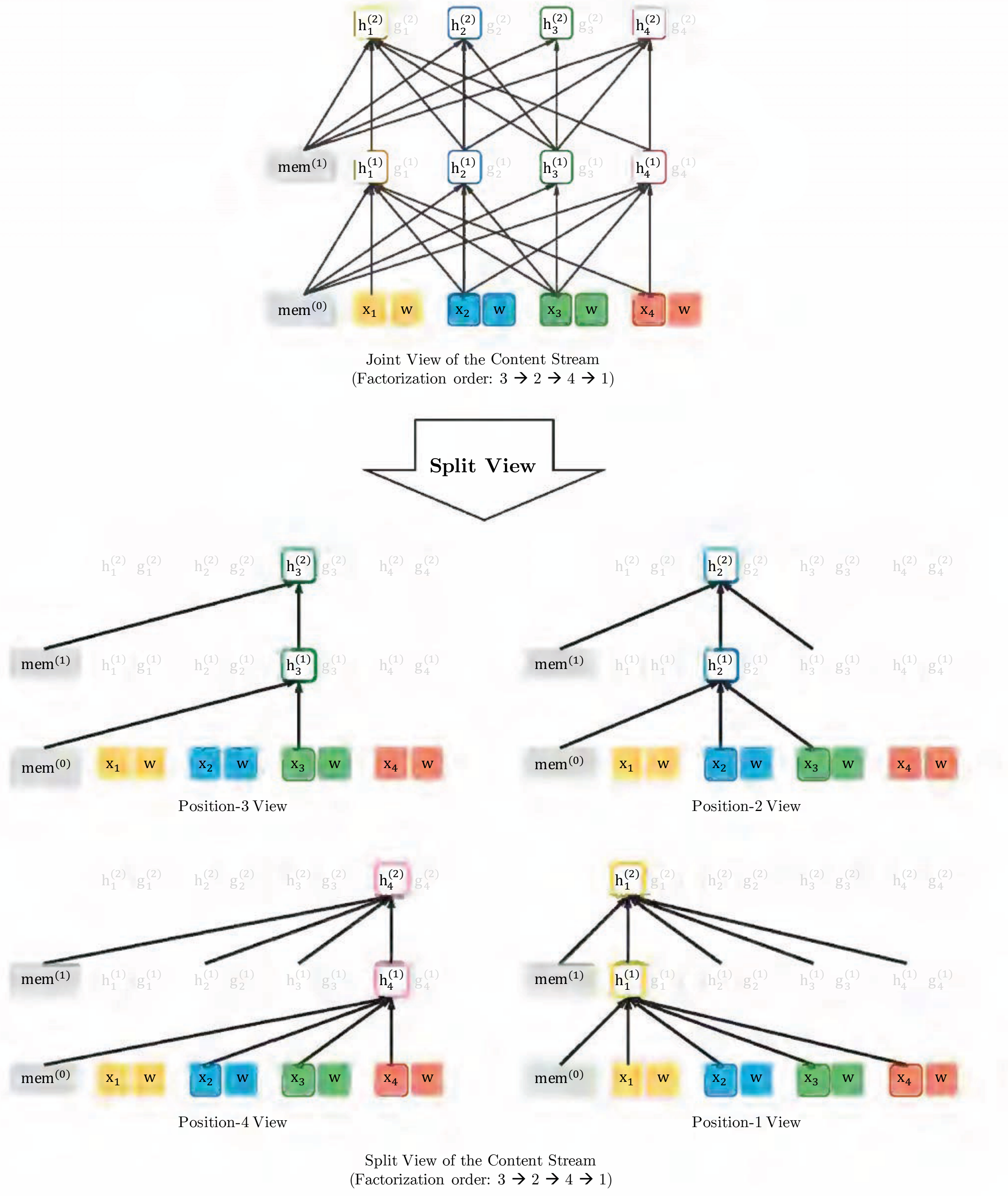
Visualizing Memory and Permutation

In this section, we provide a detailed visualization of the proposed permutation language modeling objective, including the mechanism of reusing memory (aka the recurrence mechanism), how we use attention masks to permute the factorization order, and the difference of the two attention streams. As shown in Figure 2.8 and 2.9, given the current position z_t, the attention mask is decided by the permutation (or factorization order) $\mathbf{z}$ such that only tokens the occur before z_t in the permutation can be attended; i.e., positions z_i with $i < t$. Moreover, comparing Figure 2.8 and 2.9, we can see how the query stream and the content stream work differently with a specific permutation through attention masks. The main difference is that the query stream cannot do self-attention and does not have access to the token at the position, while the content stream performs normal self-attention.

Modeling Multiple Segments

Many downstream tasks have multiple input segments, e.g., a question and a context paragraph in question answering. We now discuss how we pretrain XLNet to model multiple segments in the autoregressive framework. During the pretraining phase, following BERT, we randomly sample two segments (either from the same context or not) and treat the concatenation of two segments as one sequence to perform permutation language modeling. We only reuse the memory that belongs to the same context. Specifically, the input to our model is similar to BERT: [A, SEP, B, SEP, CLS], where "SEP" and "CLS" are two special symbols and "A" and "B" are the two segments. Although we follow the two-segment data format, XLNet-Large does not use the objective of next sentence prediction [34] as it does not show consistent improvement in our ablation study (see Section 2.2.3).

Relative Segment Encodings Architecturally, different from BERT that adds an absolute segment embedding to the word embedding at each position, we extend the idea of relative encodings from Transformer-XL to also encode the segments. Given a pair of positions i and j in the sequence, if i and j are from the same segment, we use a segment encoding $\mathbf{s}_{ij} = \mathbf{s}_+$ or otherwise $\mathbf{s}_{ij} = \mathbf{s}_-$, where $\mathbf{s}_+$ and $\mathbf{s}_-$ are learnable model parameters for each attention head. In other words, we only consider whether the two positions are *within the same segment*, as opposed to considering *which specific segments they are from*. This is consistent with the core idea of relative encodings; i.e., only modeling the relationships between positions. When i attends to j, the segment encoding $\mathbf{s}_{ij}$ is used to compute an attention weight $a_{ij} = (\mathbf{q}_i + \mathbf{b})^\top \mathbf{s}_{ij}$, where $\mathbf{q}_i$ is the query vector as in a standard attention operation and $\mathbf{b}$ is a learnable head-specific bias vector. Finally, the value a_{ij} is added to the normal attention weight. There are two benefits of using relative segment encodings. First, the inductive bias of relative encodings improves generalization [31]. Second, it opens the possibility of finetuning on tasks that have more than



Joint View of the Content Stream
(Factorization order: 3 → 2 → 4 → 1)
Split View
Position-3 View
Position-2 View
Position-4 View
Position-1 View
Split View of the Content Stream
(Factorization order: 3 → 2 → 4 → 1)

Figure 2.8: A detailed illustration of the **content stream** of the proposed objective with both the joint view and split views based on a length-4 sequence under the factorization order [3, 2, 4, 1]. Note that if we ignore the query representation, the computation in this figure is simply the standard self-attention, though with a particular attention mask.

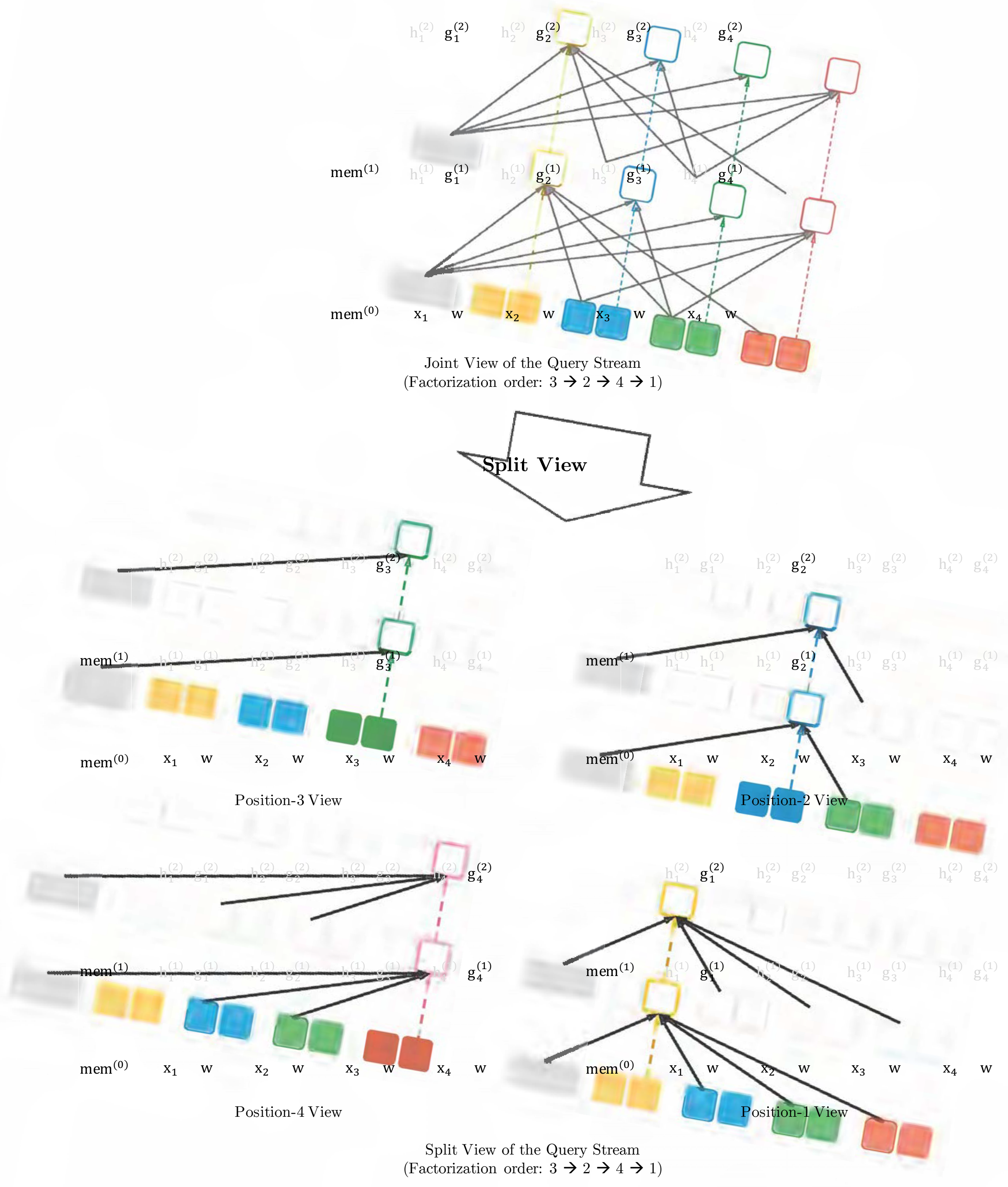

mem$^{(1)}$
mem$^{(0)}$
$h_1^{(2)}$ $g_1^{(2)}$ $h_2^{(2)}$ $g_2^{(2)}$ $h_3^{(2)}$ $g_3^{(2)}$ $h_4^{(2)}$ $g_4^{(2)}$
$h_1^{(1)}$ $g_1^{(1)}$ $h_2^{(1)}$ $g_2^{(1)}$ $h_3^{(1)}$ $g_3^{(1)}$ $h_4^{(1)}$ $g_4^{(1)}$
x_1 w x_2 w x_3 w x_4 w
Joint View of the Query Stream
(Factorization order: 3 → 2 → 4 → 1)
Split View
Position-3 View
Position-2 View
Position-4 View
Position-1 View
Split View of the Query Stream
(Factorization order: 3 → 2 → 4 → 1)

Figure 2.9: A detailed illustration of the **query stream** of the proposed objective with both the joint view and split views based on a length-4 sequence under the factorization order [3, 2, 4, 1]. The dash arrows indicate that the query stream cannot access the token (content) at the same position, but only the location information.

two input segments, which is not possible using absolute segment encodings.

Discussion and Analysis

Comparison with BERT Comparing Eq. (2.2) and (2.5), we observe that both BERT and XLNet perform partial prediction, i.e., only predicting a subset of tokens in the sequence. This is a necessary choice for BERT because if all tokens are masked, it is impossible to make any meaningful predictions. In addition, for both BERT and XLNet, partial prediction plays a role of reducing optimization difficulty by only predicting tokens with sufficient context. However, the independence assumption discussed in Section 2.2.2 prevents BERT from modeling the dependency between targets.

To better understand the difference, let's consider a concrete example [New, York, is, a, city]. Suppose both BERT and XLNet select the two tokens [New, York] as the prediction targets and maximize $\log p(\text{New York} \mid \text{is a city})$. Also suppose that XLNet samples the factorization order [is, a, city, New, York]. In this case, BERT and XLNet respectively reduce to the following objectives:

$$\mathcal{J}_{\text{BERT}} = \log p(\text{New} \mid \text{is a city}) + \log p(\text{York} \mid \text{is a city}),$$

$$\mathcal{J}_{\text{XLNet}} = \log p(\text{New} \mid \text{is a city}) + \log p(\text{York} \mid \text{New}, \text{is a city}).$$

Notice that XLNet is able to capture the dependency between the pair (New, York), which is omitted by BERT. Although in this example, BERT learns some dependency pairs such as (New, city) and (York, city), it is obvious that XLNet always learns **more** dependency pairs given the same target and contains "denser" effective training signals.

We now turn to a more general discussion with formal expressions. Inspired by previous work [195], given a sequence $\mathbf{x} = [x_1, \cdots, x_T]$, we define a set of target-context pairs of interest, $\mathcal{I} = \{(x, \mathcal{U})\}$, where $\mathcal{U}$ is a set of tokens in $\mathbf{x}$ that form a context of x. Intuitively, we want the model to learn the dependency of x on $\mathcal{U}$ through a pretraining loss term $\log p(x \mid \mathcal{U})$. For example, given the above sentence, the pairs of interest $\mathcal{I}$ could be instantiated as:

$$\mathcal{I} = \Big\{ \big(x = \text{York}, \mathcal{U} = \{\text{New}\}\big),\ \big(x = \text{York}, \mathcal{U} = \{\text{city}\}\big),\ \big(x = \text{York}, \mathcal{U} = \{\text{New, city}\}\big),\ \cdots \Big\}.$$

Note that $\mathcal{I}$ is merely a virtual notion without unique ground truth, and our analysis will hold regardless of how $\mathcal{I}$ is instantiated.

Given a set of target tokens $\mathcal{T}$ and a set of non-target tokens $\mathcal{N} = \mathbf{x} \backslash \mathcal{T}$, BERT and XLNet both maximize $\log p(\mathcal{T} \mid \mathcal{N})$ but with different formulations:

$$\mathcal{J}_{\text{BERT}} = \sum_{x \in \mathcal{T}} \log p(x \mid \mathcal{N}); \quad \mathcal{J}_{\text{XLNet}} = \sum_{x \in \mathcal{T}} \log p(x \mid \mathcal{N} \cup \mathcal{T}_{<x})$$

where $\mathcal{T}_{<x}$ denote tokens in $\mathcal{T}$ that have a factorization order prior to x. Both objectives consist of multiple *loss terms* in the form of $\log p(x \mid \mathcal{V}_x)$. Intuitively, if there exists a target-context pair $(x, \mathcal{U}) \in \mathcal{I}$ such that $\mathcal{U} \subseteq \mathcal{V}_x$, then the loss term $\log p(x \mid \mathcal{V}_x)$ provides a training signal to the dependency between x and $\mathcal{U}$. For convenience, we say a target-context pair $(x, \mathcal{U}) \in \mathcal{I}$ is *covered* by a model (objective) if $\mathcal{U} \subseteq \mathcal{V}_x$.

Given the definition, let's consider two cases:

- If $\mathcal{U} \subseteq \mathcal{N}$, the dependency $(x, \mathcal{U})$ is covered by both BERT and XLNet.
- If $\mathcal{U} \subseteq \mathcal{N} \cup \mathcal{T}_{<x}$ and $\mathcal{U} \cap \mathcal{T}_{<x} \neq \emptyset$, the dependency can only be covered by XLNet but not BERT. As a result, XLNet is able to cover more dependencies than BERT. In other words, the XLNet objective contains more effective training signals, which empirically leads to better performance in Section 2.2.3.

Note that it might be possible to predict only one token for each input sequence with all the other tokens as the context. In this case, a BERT objective does not suffer from the independence assumption. However, this modified objective will lead to extremely low data efficiency as only one training signal is provided, i.e., 85 times less training signals compared to XLNet because XLNet predicts 85 tokens for each sequence of length 512. From this perspective, XLNet can be viewed as achieving a trade-off between dependency modeling and data efficiency.

Comparison with Language Modeling Borrowing the above examples and notations, a standard AR language model like GPT [126] is only able to cover the dependency $(x = \text{York}, \mathcal{U} = \{\text{New}\})$ but not $(x = \text{New}, \mathcal{U} = \{\text{York}\})$. XLNet, on the other hand, is able to cover both in expectation over all factorization orders. Such a limitation of AR language modeling can be critical in real-world applications. For example, consider a span extraction question answering task with the context "Thom Yorke is the singer of Radiohead" and the question "Who is the singer of Radiohead". The representations of "Thom Yorke" are not dependent on "Radiohead" with AR language modeling and thus they will not be chosen as the answer by the standard approach that employs softmax over all token representations. More formally, consider a context-target pair $(x, \mathcal{U})$:

- If $\mathcal{U} \cap \mathcal{T}_{<x} \neq \emptyset$, where $\mathcal{T}_{<x}$ denotes the tokens prior to x in the original sequence, AR language modeling is not able to cover the dependency.
- In comparison, XLNet is able to cover all dependencies in expectation.

Approaches like ELMo [122] concatenate forward and backward language models in a shallow manner, which is not sufficient for modeling deep interactions between the two directions.

Bridging the Gap Between Language Modeling and Pretraining With a deep root in density estimation[2] [9, 115, 164], language modeling has been a rapidly-developing research area [1, 4, 31]. However, there has been a gap between language modeling and pretraining due to the lack of the capability of bidirectional context modeling, as analyzed above. It has even been challenged by some machine learning practitioners whether language modeling is a meaningful pursuit if it does not directly improve downstream tasks [3]. XLNet generalizes language modeling and bridges such a gap. As a result, it further "justifies" language modeling research. Moreover, it becomes possible to leverage the rapid progress of language modeling research for pretraining. As an example, we integrate Transformer-XL into XLNet to demonstrate the usefulness of the latest language modeling progress.

Connection with the Generative Feature Learning Framework

XLNet instantiated the generative feature learning framework in Section 1.4 as follows:

[2]The problem of language modeling is essentially density estimation for text data.

[3]`https://openreview.net/forum?id=HJePno0cYm`

- The generative model G is a Transformer-XL model with the two-stream attention mechanism.
- The generative loss function l_g is defined as the permutation language modeling loss function, which optimizes the expectation of the negative log-likelihood w.r.t. all permutations of the factorization order.
- The target loss function l_t is defined as downstream task loss. There are in fact multiple target loss functions as a pretrained model can be finetuned on multiple downstream tasks.
- The training paradigm is two-phase. First the model is pretrained on unlabeled data using l_g and finetuned on each downstream target task respectively using the target loss l_t.

2.2.3 Experiments

Pretraining and Implementation

Following BERT [34], we use the BooksCorpus [206] and English Wikipedia as part of our pretraining data, which have 13GB plain text combined. In addition, we include Giga5 (16GB text) [118], ClueWeb 2012-B (extended from [15]), and Common Crawl [25] for pretraining. We use heuristics to aggressively filter out short or low-quality articles for ClueWeb 2012-B and Common Crawl, which results in 19GB and 78GB text respectively. After tokenization with SentencePiece [82], we obtain 2.78B, 1.09B, 4.75B, 4.30B, and 19.97B subword pieces for Wikipedia, BooksCorpus, Giga5, ClueWeb, and Common Crawl respectively, which are 32.89B in total.

Our largest model XLNet-Large has the same architecture hyperparameters as BERT-Large, which results in a similar model size. The sequence length and memory length are set to 512 and 384 respectively. We train XLNet-Large on 512 TPU v3 chips for 500K steps with an Adam optimizer, linear learning rate decay and a batch size of 2048, which takes about 2.5 days. It was observed that the model still underfits the data at the end of training but continuing training did not help downstream tasks, which indicates that given the optimization algorithm, the model does not have enough capacity to fully leverage the data scale. However, in this work, we refrain from training a larger model as its practical usage for finetuning might be limited. Further, we train an XLNet-Base, analogous to BERT-Base, on BooksCorpus and Wikipedia only, for ablation study and fair comparison with BERT. Related results are presented in Section 2.2.3.

Since the recurrence mechanism is introduced, we use a bidirectional data input pipeline where each of the forward and backward directions takes half of the batch size. For training XLNet-Large, we set the partial prediction constant K as 6 (see Section 2.2.2). Our finetuning procedure follows BERT [34] except otherwise specified. We employ an idea of *span-based prediction*, where we first sample a length $L \in [1, \cdots, 5]$, and then randomly select a consecutive span of L tokens as prediction targets within a context of (KL) tokens.

RACE	Accuracy	Middle	High
GPT [126]	59.0	62.9	57.4
BERT [117]	72.0	76.6	70.1
BERT+OCN* [130]	73.5	78.4	71.5
BERT+DCMN* [199]	74.1	79.5	71.8
XLNet	**81.75**	**85.45**	**80.21**

Table 2.14: Comparison with state-of-the-art results on the test set of RACE, a reading comprehension task. $*$ indicates using ensembles. "Middle" and "High" in RACE are two subsets representing middle and high school difficulty levels. All BERT and XLNet results are obtained with a 24-layer architecture with similar model sizes (aka BERT-Large). Our single model outperforms the best ensemble by 7.6 points in accuracy.

SQuAD1.1	EM	F1	SQuAD2.0	EM	F1
Dev set results without data augmentation					
BERT [34]	84.1	90.9	BERT† [34]	78.98	81.77
XLNet	**88.95**	**94.52**	XLNet	**86.12**	**88.79**
Test set results on leaderboard, with data augmentation (as of June 19, 2019)					
Human [128]	82.30	91.22	BERT+N-Gram+Self-Training [34]	85.15	87.72
ATB	86.94	92.64	SG-Net	85.23	87.93
BERT* [34]	87.43	93.16	BERT+DAE+AoA	85.88	88.62
XLNet	**89.90**	**95.08**	XLNet	**86.35**	**89.13**

Table 2.15: A single model XLNet outperforms human and the best ensemble by 7.6 EM and 2.5 EM on SQuAD1.1. $*$ means ensembles, † marks our runs with the official code.

RACE Dataset

The RACE dataset [83] contains near 100K questions taken from the English exams for middle and high school Chinese students in the age range between 12 to 18, with the answers generated by human experts. This is one of the most difficult reading comprehension datasets that involve challenging reasoning questions. Moreover, the average length of the passages in RACE are longer than 300, which is significantly longer than other popular reading comprehension datasets such as SQuAD [129]. As a result, this dataset serves as a challenging benchmark for long text understanding. We use a sequence length of 640 during finetuning. As shown in Table 2.14, a single model XLNet outperforms the best ensemble by 7.6 points in accuracy. It is also clear that XLNet substantially outperforms other pretrained models such as BERT and GPT. Since RACE contains relatively long passages, we believe one of the reasons why XLNet obtains substantial gains on this dataset is that the integration of the Transformer-XL architecture improves the capability of modeling long text, besides the AR objective. More analysis on the sequence length is presented in Section 2.2.3.

Model	IMDB	Yelp-2	Yelp-5	DBpedia	AG	Amazon-2	Amazon-5
CNN [69]	-	2.90	32.39	0.84	6.57	3.79	36.24
DPCNN [69]	-	2.64	30.58	0.88	6.87	3.32	34.81
Mixed VAT [108, 134]	4.32	-	-	0.70	4.95	-	-
ULMFiT [62]	4.6	2.16	29.98	0.80	5.01	-	-
BERT [184]	4.51	1.89	29.32	0.64	-	2.63	34.17
XLNet	**3.79**	**1.55**	**27.80**	**0.62**	**4.49**	**2.40**	**32.26**

Table 2.16: Comparison with state-of-the-art error rates on the test sets of several text classification datasets. All BERT and XLNet results are obtained with a 24-layer architecture with similar model sizes (aka BERT-Large).

SQuAD Dataset

SQuAD is a large-scale reading comprehension dataset with two tasks. SQuAD1.1 [128] contains questions that always have a corresponding answer in the given passages, while SQuAD2.0 [129] introduces unanswerable questions. To finetune an XLNet on SQuAD2.0, we jointly apply a logistic regression loss for answerability prediction similar to classification tasks and a standard span extraction loss for question answering [34]. Since v1.1 and v2.0 share the same answerable questions in the training set, we simply remove the answerability prediction part from the model finetuned on v2.0 for evaluation on v1.1. As the top leaderboard entries all employ some form of data augmentation, we jointly train an XLNet on SQuAD2.0 and NewsQA [160] for our leaderboard submission. As shown in Table 2.15, XLNet obtains the state-of-the-art single model results on the leaderboard, outperforming a series of BERT-based methods. Notably, on v1.1, an XLNet single model outperforms human and the best ensemble by 7.6 and 2.5 points in EM. Finally, for direct comparison with BERT to eliminate the effects of additional tricks in leaderboard submissions, we compare XLNet against BERT on the dev set. XLNet substantially outperforms BERT by 3.6 and 7.0 points in F1 for v1.1 and v2.0.

Text Classification

Following previous work on text classification [108, 200], we evaluate XLNet on the following benchmarks: IMDB, Yelp-2, Yelp-5, DBpedia, AG, Amazon-2, and Amazon-5. According to Table 2.16, XLNet achieves new state-of-the-art results on all the considered datasets, reducing the error rate by 16%, 18%, 5%, 9% and 5% on IMDB, Yelp-2, Yelp-5, Amazon-2, and Amazon-5 respectively compared to BERT.

GLUE Dataset

The GLUE dataset [171] is a collection of 9 natural language understanding tasks. The test set labels are removed from the publicly released version, and all the practitioners must submit their predictions on the evaluation server to obtain test set results. In Table 2.17, we present results of multiple settings, including single-task and multi-task, as well as single models and ensembles.

Model	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	WNLI
Single-task single models on dev									
BERT [2]	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-
XLNet	**89.8/-**	**93.9**	**91.8**	**83.8**	**95.6**	**89.2**	**63.6**	**91.8**	-
Single-task single models on test									
BERT [34]	86.7/85.9	91.1	89.3	70.1	94.9	89.3	60.5	87.6	65.1
Multi-task ensembles on test (from leaderboard as of June 19, 2019)									
Snorkel* [132]	87.6/87.2	93.9	89.9	80.9	96.2	91.5	63.8	90.1	65.1
ALICE*	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8
MT-DNN* [95]	87.9/87.4	96.0	89.9	**86.3**	96.5	92.7	68.4	91.1	89.0
XLNet*	**90.2/89.7**†	**98.6**†	90.3†	**86.3**	**96.8**†	**93.0**	67.8	**91.6**	**90.4**

Table 2.17: Results on GLUE. $*$ indicates using ensembles, and $\dagger$ denotes single-task results in a multi-task row. All results are based on a 24-layer architecture with similar model sizes (aka BERT-Large). See the upper-most rows for direct comparison with BERT and the lower-most rows for comparison with state-of-the-art results on the public leaderboard.

Model	NDCG@20	ERR@20
DRMM [56]	24.3	13.8
KNRM [28]	26.9	14.9
Conv [28]	28.7	18.1
BERT†	30.53	18.67
XLNet	**31.10**	**20.28**

Table 2.18: Comparison with state-of-the-art results on the test set of ClueWeb09-B, a document ranking task. $\dagger$ indicates our implementations.

In the multi-task setting, we jointly train an XLNet on the four largest datasets—MNLI, SST-2, QNLI, and QQP—and finetune the network on the other datasets. Only single-task training is employed for the four large datasets. For QNLI, we employed a pairwise relevance ranking scheme as in [95] for our test set submission. However, for fair comparison with BERT, our result on the QNLI dev set is based on a standard classification paradigm. For WNLI, we use the loss described in [79]. A multi-task ensemble XLNet achieves the state-of-the-art results on 7 out of 9 tasks on the public leaderboard. On the most widely-benchmarked task MNLI, XLNet improves the "matched" and "mismatched" settings by 2.0 and 1.8 points respectively. Note that the leaderboard competitors employ improved techniques over BERT such as distillation, modified multi-task losses, or meta learning, but still underperform XLNet which does not employ additional tricks besides using a standard multi-task learning method. Since the leaderboard is not intended for ablation study or hyperparameter tuning, we only evaluated our best multi-task models on the test set. To obtain a direct comparison with BERT, we run a single-task XLNet on the dev set. As shown in the upper-most rows of Table 2.17, XLNet consistently outperforms BERT, with an improvement of 13.4 points, 3.2 points, 3.0 points, 2.4 points, 1.8 points on RTE, MNLI, CoLA, SST-2, and STS-B respectively.

ClueWeb09-B Dataset

Following the setting in previous work [28], we use the ClueWeb09-B dataset to evaluate the performance on document ranking. The queries were created by the TREC 2009-2012 Web Tracks based on 50M documents and the task is to rerank the top 100 documents retrieved using a standard retrieval method. Since document ranking, or ad-hoc retrieval, mainly concerns the low-level representations instead of high-level semantics, this dataset serves as a testbed for evaluating the quality of word embeddings. We use a pretrained XLNet to extract word embeddings for the documents and queries without finetuning, and employ a kernel pooling network [187] to rank the documents. According to Table 2.18, XLNet substantially outperforms the other methods, including a BERT model that uses the same training procedure as ours. This illustrates that XLNet learns better low-level word embeddings than BERT. Note that for fair comparison we exclude the results (19.55 in ERR@20, slightly worse than ours) in [186] as it uses additional entity-related data.

Ablation Study

We perform an ablation study to understand the importance of each design choice based on four datasets with diverse characteristics. Specifically, there are three main aspects we hope to study:

- The effectiveness of the permutation language modeling objective, especially compared to the denoising auto-encoding objective used by BERT.
- The importance of using Transformer-XL as the backbone neural architecture and employing segment-level recurrence (i.e. using memory).
- The necessity of some implementation details including span-based prediction, the bidirectional input pipeline, and next-sentence prediction.

With these purposes in mind, in Table 2.19, we compare 6 XLNet-Base variants with different implementation details (rows 3 - 8), the original BERT-Base model (row 1), and an additional Transformer-XL baseline trained with the denoising auto-encoding (DAE) objective used in BERT but with the bidirectional input pipeline (row 2). For fair comparison, all models are based on a 12-layer architecture with the same model hyper-parameters as BERT-Base and are trained on only Wikipedia and the BooksCorpus. All results reported are the median of 5 runs.

Examining rows 1 - 4 of Table 2.19, we see the two full XLNet-Base models trained with different values of K significantly outperform both BERT and the DAE trained Transformer-XL across tasks, showing the superiority of the permutation language modeling objective. Meanwhile, it is also interesting to see that the DAE trained Transformer-XL achieves better performance than BERT on tasks with long text such as RACE and SQuAD, suggesting the excellence of Transformer-XL in language modeling also benefits pretraining. Next, if we remove the memory caching mechanism (row 5), the performance clearly drops, especially for RACE which involves the longest context among the 4 tasks. In addition, rows 6 - 7 show that both span-based prediction and the bidirectional input pipeline play important roles in XLNet. Finally, we unexpectedly find the the next-sentence prediction objective proposed in the original BERT does not necessarily lead to an improvement in our setting. Instead, it tends to harm the performance except for the RACE dataset. Hence, when we train XLNet-Large, we exclude the next-sentence prediction objective.

#	Model	RACE	SQuAD2.0		MNLI	SST-2
			F1	EM	m/mm	
1	BERT-Base	64.3	76.30	73.66	84.34/84.65	92.78
2	DAE + Transformer-XL	65.03	79.56	76.80	84.88/84.45	92.60
3	XLNet-Base ($K = 7$)	66.05	**81.33**	**78.46**	**85.84/85.43**	92.66
4	XLNet-Base ($K = 6$)	66.66	80.98	78.18	85.63/85.12	**93.35**
5	- memory	65.55	80.15	77.27	85.32/85.05	92.78
6	- span-based pred	65.95	80.61	77.91	85.49/85.02	93.12
7	- bidirectional data	66.34	80.65	77.87	85.31/84.99	92.66
8	+ next-sent pred	**66.76**	79.83	76.94	85.32/85.09	92.89

Table 2.19: Ablation study. The results of BERT on RACE are taken from [199]. We run BERT on the other datasets using the official implementation and the same hyperparameter search space as XLNet. K is a hyperparameter to control the optimization difficulty (see Section 2.2.2). All models are pretrained on the same data.

Chapter 3

Semi-Supervised Learning with Generative Modeling

This chapter focuses on using generative modeling to improve downstream task performance in a semi-supervised learning setting. Semi-supervised learning has been extensively studied in literature [67, 202, 205]. A batch of novel models have been recently proposed for semi-supervised learning based on representation learning techniques, such as generative models [75] and ladder networks [131].

Despite the progress, the following questions remain:

- Can we understand GAN-based semi-supervised learning in a more fundamental way? How does the classifier benefit from joint training with the generator?
- Can we apply generative modeling based approaches to semi-supervised tasks with more complex data structures, such as classification on graphs and question answering?

We consider three scenarios: 1) semi-supervised classification by generating low-density adversarial samples, 2) semi-supervised question answering by generating natural language questions given context, and 3) semi-supervised learning on graphs by modeling the generation of random walk.

3.1 Semi-Supervised Learning with GANs

This section introduces the work originally published at NeurIPS 2017 [30].

3.1.1 Motivation

Deep neural networks are usually trained on a large amount of labeled data, and it has been a challenge to apply deep models to datasets with limited labels. Semi-supervised learning (SSL) aims to leverage the large amount of unlabeled data to boost the model performance, particularly focusing on the setting where the amount of available labeled data is limited. Traditional graph-based methods [8, 205] were extended to deep neural networks [76, 178, 190], which involves applying convolutional neural networks [88] and feature learning techniques to graphs so that the underlying manifold structure can be exploited. [131] employs a Ladder network to minimize the layerwise reconstruction loss in addition to the standard classification loss. Variational auto-encoders have also been used for semi-supervised learning [75, 98] by maximizing the variational lower bound of the unlabeled data log-likelihood.

Recently, generative adversarial networks (GANs) [47] were demonstrated to be able to generate visually realistic images. GANs set up an adversarial game between a discriminator and a generator. The goal of the discriminator is to tell whether a sample is drawn from true data or generated by the generator, while the generator is optimized to generate samples that are not distinguishable by the discriminator. Feature matching (FM) GANs [136] apply GANs to semi-supervised learning on K-class classification. The objective of the generator is to match the first-order feature statistics between the generator distribution and the true distribution. Instead of binary classification, the discriminator employs a $(K+1)$-class objective, where true samples are classified into the first K classes and generated samples are classified into the $(K+1)$-th class. This $(K+1)$-class discriminator objective leads to strong empirical results, and was later widely used to evaluate the effectiveness of generative models [40, 163].

Though empirically feature matching improves semi-supervised classification performance, the following questions still remain open. First, it is not clear why the formulation of the discriminator can improve the performance when combined with a generator. Second, it seems that good semi-supervised learning and a good generator cannot be obtained at the same time. For example, [136] observed that mini-batch discrimination generates better images than feature matching, but feature matching obtains a much better semi-supervised learning performance. The same phenomenon was also observed in [163], where the model generated better images but failed to improve the performance on semi-supervised learning.

In this work, we take a step towards addressing these questions. First, we show that given the current $(K+1)$-class discriminator formulation of GAN-based SSL, good semi-supervised learning requires a "bad" generator. Here by *bad* we mean the generator distribution should not match the true data distribution. Then, we give the definition of a preferred generator, which is to generate complement samples in the feature space (see Section 3.1.3). Theoretically, under mild assumptions, we show that a properly optimized discriminator obtains correct decision boundaries in high-density areas in the feature space if the generator is a *complement generator*.

Based on our theoretical insights, we analyze why feature matching works on 2-dimensional toy datasets. It turns out that our practical observations align well with our theory. However, we also find that the feature matching objective has several drawbacks. Therefore, we develop a novel formulation of the discriminator and generator objectives to address these drawbacks. In our approach, the generator minimizes the KL divergence between the generator distribution and a target distribution that assigns high densities for data points with low densities in the true distribution, which corresponds to the idea of a complement generator. Furthermore, to enforce our assumptions in the theoretical analysis, we add the conditional entropy term to the discriminator objective.

3.1.2 Related Prior Work

Besides the adversarial feature matching approach [136], several previous works have incorporated the idea of adversarial training in semi-supervised learning. Notably, [149] proposes categorical generative adversarial networks (CatGAN), which substitutes the binary discriminator in standard GAN with a multi-class classifier, and trains both the generator and the discriminator using information theoretical criteria on unlabeled data. From the perspective of regularization, [109, 110] propose virtual adversarial training (VAT), which effectively smooths the output distribution of the classifier by seeking virtually adversarial samples. It is worth noting that VAT bears a similar merit to our approach, which is to learn from auxiliary non-realistic samples rather than realistic data samples. Despite the similarity, the principles of VAT and our approach are orthogonal, where VAT aims to enforce a smooth function while we aim to leverage a generator to better detect the low-density boundaries. Different from aforementioned approaches, [193] proposes to train conditional generators with adversarial training to obtain complete sample pairs, which can be directly used as additional training cases. Recently, Triple GAN [90] also employs the idea of conditional generator, but uses adversarial cost to match the two model-defined factorizations of the joint distribution with the one defined by paired data.

Apart from adversarial training, there has been other efforts in semi-supervised learning using deep generative models recently. As an early work, [75] adapts the original Variational Auto-Encoder (VAE) to a semi-supervised learning setting by treating the classification label as an additional latent variable in the directed generative model. [98] adds auxiliary variables to the deep VAE structure to make variational distribution more expressive. With the boosted model expressiveness, auxiliary deep generative models (ADGM) improve the semi-supervised learning performance upon the semi-supervised VAE. Different from the explicit usage of deep generative models, the Ladder networks [131] take advantage of the local (layerwise) denoising auto-encoding criterion, and create a more informative unsupervised signal through lateral connection.

The idea of using samples to define the domain distribution has been explored in [177]. The main difference is that we consider using generative models to generate complement samples instead of using existing data.

3.1.3 Theoretical Analysis

Given a labeled set $\mathcal{L} = \{(x, y)\}$, let $\{1, 2, \cdots, K\}$ be the label space for classification. Let D and G denote the discriminator and generator, and P_D and p_G denote the corresponding distributions. Consider the discriminator objective function of GAN-based semi-supervised learning [136]:

$$\max_D \mathbb{E}_{x,y\sim\mathcal{L}} \log P_D(y|x, y \leq K) + \mathbb{E}_{x\sim p} \log P_D(y \leq K|x) + \mathbb{E}_{x\sim p_G} \log P_D(K+1|x), \quad (3.1)$$

where p is the true data distribution. The probability distribution P_D is over $K+1$ classes where the first K classes are true classes and the $(K+1)$-th class is the fake class. The objective function consists of three terms. The first term is to maximize the log conditional probability for labeled data, which is the standard cost as in supervised learning setting. The second term is to maximize the log probability of the first K classes for unlabeled data. The third term is to maximize the log probability of the $(K+1)$-th class for generated data. Note that the above objective function bears a similar merit to the original GAN formulation if we treat $P(K+1|x)$ to be the probability of fake samples, while the only difference is that we split the probability of true samples into K sub-classes.

Let $h(x)$ be a nonlinear vector-valued function, and w_k be the weight vector for class k. As a standard setting in previous work [40, 136], the discriminator D is defined as

$$P_D(k|x) = \frac{\exp(w_k^\top h(x))}{\sum_{k'=1}^{K+1} \exp(w_{k'}^\top h(x))}.$$

Since this is a form of over-parameterization, w_{K+1} is fixed as a zero vector [136]. We next discuss the choices of different possible G's.

Perfect Generator

Here, by perfect generator we mean that the generator distribution p_G exactly matches the true data distribution p, i.e., $p_G = p$. We now show that when the generator is perfect, it does not improve the generalization over the supervised learning setting.

Proposition 1. *If $p_G = p$, and D has infinite capacity, then for any optimal solution $D = (w, h)$ of the following supervised objective,*

$$\max_D \mathbb{E}_{x,y\sim\mathcal{L}} \log P_D(y|x, y \le K), \tag{3.2}$$

there exists $D^* = (w^*, h^*)$ such that D^* maximizes Eq. (3.1) and that for all x, $P_D(y|x, y \le K) = P_{D^*}(y|x, y \le K)$.

Proof. Given an optimal solution $D = (w, h)$ for the supervised objective, due to the infinite capacity of the discriminator, there exists $D^* = (w^*, h^*)$ such that for all x and $k \le K$,

$$\exp(w_k^{*\top} h^*(x)) = \frac{\exp(w_k^\top h(x))}{\sum_{k'} \exp(w_{k'}^\top h(x))} \tag{3.3}$$

For all x,

$$P_{D^*}(y|x, y \le K) = \frac{\exp(w_k^{*\top} h^*(x))}{\sum_{k'} \exp(w_{k'}^{*\top} h^*(x))} = \frac{\exp(w_k^\top h(x))}{\sum_{k'} \exp(w_{k'}^\top h(x))} = P_D(y|x, y \le K)$$

Let L_D be the supervised objective in Eq. (3.1). Since $p = p_G$, the objective in Eq. (3.1) can be written as

$$J_D = L_D + \mathbb{E}_{x\sim p}\left[\log P_D(K+1|x) + \log(1 - P_D(K+1|x))\right]$$

Given Eq. (3.3), we have

$$P_{D^*}(K+1|x) = \frac{1}{1 + \sum_k \exp w_k^{*\top} h^*(x)} = \frac{1}{2}$$

Therefore, D^* maximizes the second term of J_D. Because D maximizes L_D, D^* also maximizes L_D. It follows that D^* maximizes J_D. □

Proposition 1 states that for any optimal solution D of the supervised objective, there exists an optimal solution D^* of the $(K+1)$-class objective such that D and D^* share the same generalization error. In other words, using the $(K+1)$-class objective does not prevent the model from experiencing any arbitrarily high generalization error that it could suffer from under the supervised objective. Moreover, since all the optimal solutions are equivalent w.r.t. the $(K+1)$-class objective, it is the optimization algorithm that really decides which specific solution the model will reach, and thus what generalization performance it will achieve. This implies that when the generator is perfect, the $(K+1)$-class objective by itself is not able to improve the generalization performance. In fact, in many applications, an almost infinite amount of unlabeled data is available, so learning a perfect generator for purely sampling purposes should not be useful. In this case, our theory suggests that not only the generator does not help, but also unlabeled data is not effectively utilized when the generator is perfect.

Complement Generator

The function h maps data points in the input space to the feature space. We use $f = h(x)$ to denote feature space data points, as opposed to x that denotes input space data points. Let $p_k(f)$ be the density of the data points of class k in the feature space. Given a threshold ϵ_k, let F_k be a subset of the data support where $p_k(f) > \epsilon_k$, i.e., $F_k = \{f : p_k(f) > \epsilon_k\}$. We assume that given $\{\epsilon_k\}_{k=1}^K$, the F_k's are disjoint with a margin. More formally, for any $f_j \in F_j$, $f_k \in F_k$, and $j \neq k$, we assume that there exists a real number $0 < \alpha < 1$ such that $\alpha f_j + (1-\alpha) f_k \notin F_j \cup F_k$. As long as the probability densities of different classes do not share any mode, i.e., $\forall i \neq j, \mathrm{argmax}_f p_i(f) \cap \mathrm{argmax}_f p_j(f) = \emptyset$, this assumption can always be satisfied by tuning the thresholds ϵ_k's. With the assumption held, we will show that the model performance would be better if the thresholds could be set to smaller values (ideally zero). We also assume that each F_k contains at least one labeled data point.

Suppose $\cup_{k=1}^K F_k$ is bounded by a convex set $\mathcal{B}$. If the support F_G of a generator G in the feature space is a relative complement set in $\mathcal{B}$, i.e., $F_G = \mathcal{B} - \cup_{k=1}^K F_k$, we call G a complement generator. The reason why we utilize a bounded $\mathcal{B}$ to define the complement is presented in Section 3.1.3. Note that the definition of complement generator implies that G is a function of h. By treating G as a function of h, theoretically D can optimize the original objective function in Eq. (3.1).

Now we present the assumption on the convergence conditions of the discriminator. Let $\mathcal{U}$ and $\mathcal{G}$ be the sets of unlabeled data and generated data.

Assumption 1. ***Convergence conditions.*** *When D converges on a finite training set $\{\mathcal{L}, \mathcal{U}, \mathcal{G}\}$, D learns a (strongly) correct decision boundary for all training data points. More specifically,*

- *(A1) for any $(x, y) \in \mathcal{L}$, we have $w_y^\top h(x) > w_k^\top h(x)$ for any other class $k \neq y$;*
- *(A2) for any $x \in \mathcal{G}$, we have $0 > \max_{k=1}^K w_k^\top h(x)$;*
- *(A3) for any $x \in \mathcal{U}$, we have $\max_{k=1}^K w_k^\top h(x) > 0$.*

In Assumption 1, conditions (A1) and (A2) assume classification correctness on labeled data and true-fake correctness on generated data respectively, which is directly induced by the objective function. Likewise, it is also reasonable to assume true-fake correctness on unlabeled data, i.e., $\log \sum_k \exp w_k^\top h(x) > 0$ for $x \in \mathcal{U}$. However, condition (A3) goes beyond this and assumes $\max_k w_k^\top h(x) > 0$. We discuss this issue in detail in Section 3.1.3 and argue that these assumptions are reasonable. Moreover, in Section 3.1.5, our approach addresses this issue explicitly by adding a conditional entropy term to the discriminator objective to enforce condition (A3).

Lemma 1. *Suppose for all k, the L2-norms of weights w_k are bounded by $\|w_k\|_2 \leq C$. Suppose that there exists $\epsilon > 0$ such that for any $f_G \in F_G$, there exists $f'_G \in \mathcal{G}$ such that $\|f_G - f'_G\|_2 \leq \epsilon$. With the conditions in Assumption 1, for all $k \leq K$, we have $w_k^\top f_G < C\epsilon$.*

Proof. Let $\Delta f = f_G - f'_G$, then we have $\|\Delta f\|_2 \leq \epsilon$. Because $w_k^\top f'_G < 0$ by assumption, it follows

$$w_k^\top f_G = w_k^\top (f'_G + \Delta f) = w_k^\top f'_G + w_k^\top \Delta f < w_k^\top \Delta f \leq C\epsilon$$

□

Corollary 1. *When unlimited generated data samples are available, with the conditions in Lemma 1, we have $\lim_{|\mathcal{G}| \to \infty} w_k^\top f_G \leq 0$.*

Proposition 2. *Given the conditions in Corollary 1, for all class $k \leq K$, for all feature space points $f_k \in F_k$, we have $w_k^\top f_k > w_j^\top f_k$ for any $j \neq k$.*

Proof. Without loss of generality, suppose $j = \arg\max_{j \neq k} w_j^\top f_k$. Now we prove it by contradiction. Suppose $w_k^\top f_k \leq w_j^\top f_k$. Since F_k's are disjoint with a margin, $\mathcal{B}$ is a convex set and $F_G = \mathcal{B} - \cup_k F_k$, there exists $0 < \alpha < 1$ such that $f_G = \alpha f_k + (1-\alpha) f_j$ with $f_G \in F_G$ and f_j being the feature of a labeled data point in F_j. By Corollary 1, it follows that $w_j^\top f_G \leq 0$. Thus, $w_j^\top f_G = \alpha w_j^\top f_k + (1-\alpha) w_j^\top f_j \leq 0$. By Assumption 1, $w_j^\top f_k > 0$ and $w_j^\top f_j > 0$, leading to contradiction. It follows that $w_k^\top f_k > w_j^\top f_k$ for any $j \neq k$. □

Proposition 2 guarantees that when G is a complement generator, under mild assumptions, a near-optimal D learns correct decision boundaries in each high-density subset F_k (defined by ϵ_k) of the data support in the feature space. Intuitively, the generator generates complement samples so the logits of the true classes are forced to be low in the complement. As a result, the discriminator obtains class boundaries in low-density areas. This builds a connection between our approach with manifold-based methods [8, 205] which also leverage the low-density boundary assumption.

With our theoretical analysis, we can now answer the questions raised in Section 3.1.1. First, the $(K+1)$-class formulation is effective because the generated complement samples encourage the discriminator to place the class boundaries in low-density areas (Proposition 2). Second, good semi-supervised learning indeed requires a bad generator because a perfect generator is not able to improve the generalization performance (Proposition 1).

On the Feature Space Bound Assumption

To obtain our theoretical results, we assume that $\cup_{k=1}^K F_k$ is bounded by a convex set $\mathcal{B}$. And the definition of complement generator requires that $F_G = \mathcal{B} - \cup_{k=1}^K F_k$. Now we justify the necessity of the introduction of $\mathcal{B}$.

The bounded $\mathcal{B}$ is introduced to ensure that Assumption 1 is realizable. We first show that for Assumption 1 to hold, F_G must be a convex set.

We define $S = \{f : \max_{k=1}^K w_k^\top f < 0\}$.

Lemma 2. *S is a convex set.*

Proof. We prove it by contradiction. Suppose S is a non-convex set, then there exists $f_1, f_2 \in S$, and $0 < \alpha < 1$, such that $f = \alpha f_1 + (1-\alpha) f_2 \notin S$. For all k, we have $w_k^\top f_1 < 0$ and $w_k^\top f_2 < 0$, and thus it follows

$$w_k^\top f = \alpha w_k^\top f_1 + (1-\alpha) w_k^\top f_2 < 0$$

Therefore, $\max_{k=1}^K w_k^\top f < 0$, and we have $f \in S$, leading to contradiction.

We conclude that S is a convex set. □

If the feature space is unbounded and F_G is defined as $\mathbb{R}^d - \cup_{k=1}^K F_k$, where d is the feature space dimension, then by Assumption 1, we have $S = F_G$. Since F_G is the complement set of $\cup_{k=1}^K F_k$ and F_k's are disjoint, F_G is a non-convex set, if $K \geq 2$. However, by Lemma 2, F_G is convex, leading to contradiction. We therefore define the complement generator using a bound $\mathcal{B}$.

The Reasonableness of Assumption 1

Here, we justify the proposed Assumption 1.

Classification correctness on $\mathcal{L}$. For (A1), it assumes the correctness of classification on labeled data $\mathcal{L}$. This only requires the transformation $h(x)$ to have high enough capacity, such that the *limited amount* of labeled data points are linearly separable in the feature space. Under the setting of semi-supervised learning, where $|\mathcal{L}|$ is quite limited, this assumption is usually reasonable.

True-Fake correctness on $\mathcal{G}$. For (A2), it assumes that on generated data, the classifier can correctly distinguish between true and generated data. This can be seen by noticing that $w_{K+1}^\top f = 0$, and the assumption thus reduces to $w_{K+1}^\top h(x) > \max_{k=1}^K w_k^\top h(x)$. For this part to hold, again we essentially require a transformation $h(x)$ with high enough capacity to distinguish true and fake data, which is a standard assumption made in GAN literature.

Strong true-fake belief on $\mathcal{U}$. Finally, (A3) of the assumption is a little bit trickier than the other two.

- Firstly, note that (A3) is related to the true-fake correctness, because $\max_{k=1}^K w_k^\top h(x) > 0 = w_{K+1}^\top h(x)$ is a *sufficient* (but not necessary) condition for x being classified as a true data point. Instead, the actual necessary condition is that $\log \sum_{k=1}^K \exp(w_k^\top h(x)) \geq w_{K+1}^\top h(x) = 0$. Thus, it means the condition (A3) might be violated.

- However, using the relationship $\log \sum_{k=1}^{K} \exp(w_k^\top h(x)) \leq \log K \max_{k=1}^{K} \exp(w_k^\top h(x))$, to guarantee the necessary condition $\log \sum_{k=1}^{K} \exp(w_k^\top h(x)) \geq 0$, we must have

$$\log K \max_{k=1}^{K} \exp(w_k^\top h(x)) \geq 0$$
$$\implies \max_{k=1}^{K} w_k^\top h(x) \geq \log 1/K$$

Hence, if the condition (A3) is violated, it means

$$\log 1/K \leq \max_{k=1}^{K} w_k^\top h(x) \leq 0$$

Note that this is a very small interval for the logit $w_k^\top h(x)$, whose possible range expands the entire real line $(-\infty, \infty)$. Thus, the region where such violation happens should be limited in size, making the assumption reasonable in practice. Moreover, even there exists a limited violation region, as long as part (A1) and part (A2) in Assumption 1 hold, Proposition 2 always holds for regions inside $\mathcal{U}$ where $\max_{k=1}^{K} w_k^\top h(x) > 0$. This can be viewed as a further Corollary.

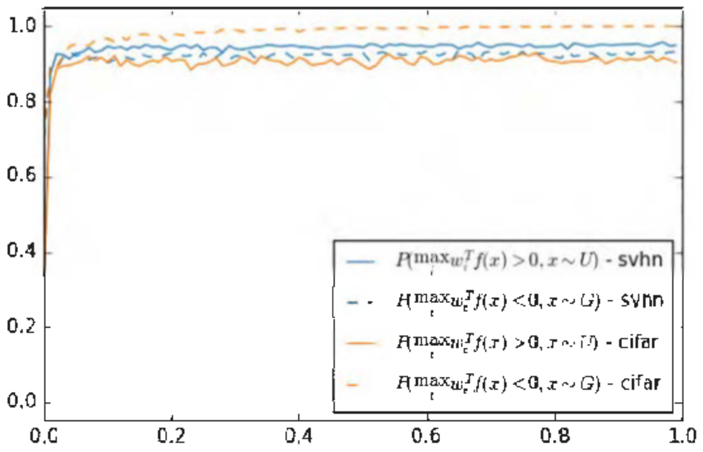


Figure 3.1: Percentage of the test samples that satisfy the assumption under our best model.

Empirically, we find that it is easy for the model to satisfy the correctness assumption on labeled data perfectly. To verify the other two assumptions, we keep track of the percentage of test samples that the two assumptions hold under our best models. More specifically, to verify the true-fake correctness on $\mathcal{G}$, we calculate the ratio after each epoch

$$\frac{\sum_{x \sim \mathcal{T}} \mathbb{I}[\max_{i=1}^{K} w_i^\top h(x) > 0]}{|\mathcal{T}|},$$

where $\mathcal{T}$ denotes the test set and $|\mathcal{T}|$ is number of sample in it. Similarly, for the strong true-fake belief on $\mathcal{U}$, we generate the same number of samples as $|\mathcal{T}|$ and calculate

$$\frac{\sum_{x \sim p_G} \mathbb{I}[\max_i w_i^T h(x) < 0]}{|\mathcal{T}|}$$

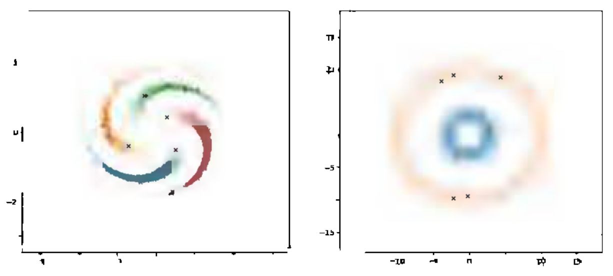


Figure 3.2: Labeled and unlabeled data are denoted by cross and point respectively, and different colors indicate classes.

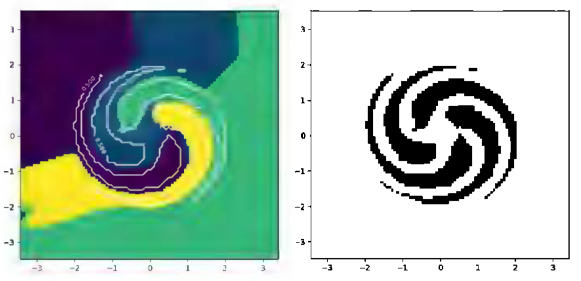


Figure 3.3: Left: Classification decision boundary, where the white line indicates true-fake boundary; Right: True-Fake decision boundary

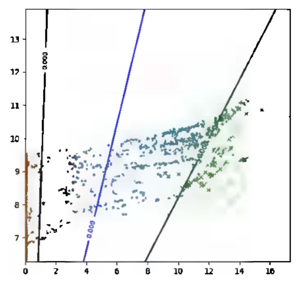


Figure 3.4: Feature space at convergence

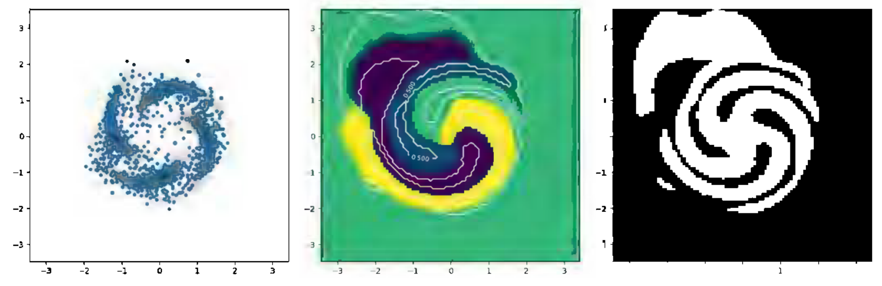


Figure 3.5: Left: Blue points are generated data, and the black shadow indicates unlabeled data. Middle and right can be interpreted as above.

The plot is presented in Fig. 3.1. As we can see, the two ratios are both above 0.9 for both SVHN and CIFAR-10, which suggests our assumptions are reasonable in practice.

3.1.4 Case Study on Synthetic Data

In the previous section, we have established the fact a complement generator, instead of a perfect generator, is what makes a good semi-supervised learning algorithm. Now, to get a more intuitive understanding, we conduct a case study based on two 2D synthetic datasets, where we can easily verify our theoretical analysis by visualizing the model behaviors. In addition, by analyzing how feature matching (FM) [136] works in 2D space, we identify some potential problems of it, which motivates our approach to be introduced in the next section. Specifically, two synthetic datasets are four spins and two circles, as shown in Fig. 3.2.

Soundness of complement generator. Firstly, to verify that the complement generator is a preferred choice, we construct the complement generator by uniformly sampling from the a bounded 2D box that contains all unlabeled data, and removing those on the manifold. Based on the complement generator, the result on four spins is visualized in Fig. 3.3. As expected, both the classification and true-fake decision boundaries are almost perfect. More importantly, the classification decision boundary always lies in the fake data area (left panel), which well matches our theoretical analysis.

Visualization of feature space. Next, to verify our analysis about the feature space, we choose the feature dimension to be 2, apply the FM to the simpler dataset of two circles, and visualize the feature space in Fig. 3.4. As we can see, most of the generated features (blue points) resides in between the features of two classes (green and orange crosses), although there exists some overlap. As a result, the discriminator can almost perfectly distinguish between true and generated samples as indicated by the black decision boundary, satisfying the our required Assumption 1. Meanwhile, the model obtains a perfect classification boundary (blue line) as our analysis suggests.

Pros and cons of feature matching. Finally, to further understand the strength and weakness of FM, we analyze the solution FM reaches on four spins shown in Fig. 3.5. From the left panel, we can see many of the generated samples actually fall into the data manifold, while the rest scatters around in the nearby surroundings of data manifold. It suggests that by matching the first-order moment by SGD, FM is performing some kind of distribution matching, though in a rather *weak* manner. Loosely speaking, FM has the effect of generating samples close to the manifold. But due to its weak power in distribution matching, FM will inevitably generate samples outside of the manifold, especially when the data complexity increases. Consequently, the generator density p_G is usually lower than the true data density p within the manifold and higher outside. Hence, an optimal discriminator $P_{D^*}(K+1 \mid x) = p(x)/(p(x)+p_G(x))$ could still distinguish between true and generated samples in many cases. However, there are two types of mistakes the discriminator can still make.

1. Higher density mistake inside manifold: Since the FM generator still assigns a significant amount of probability mass inside the support, wherever $p_G > p > 0$, an optimal discriminator will incorrectly predict samples in that region as "fake". Actually, this problem has already shown up when we examine the feature space (Fig. 3.4).
2. Collapsing with missing coverage outside manifold: As the feature matching objective for the generator only requires matching the first-order statistics, there exists many trivial solutions the generator can end up with. For example, it can simply collapse to mean of unlabeled features, or a few surrounding modes as along as the feature mean matches. Actually, we do see such collapsing phenomenon in high-dimensional experiments when FM is used (see Fig. 3.6a and Fig. 3.6c) As a result, a collapsed generator will fail to cover some gap areas between manifolds. Since the discriminator is only well-defined on the union of the data supports of p and p_G, the prediction result in such missing area is under-determined and fully relies on the smoothness of the parametric model. In this case, significant mistakes can also occur.

3.1.5 Approach

As discussed in previous sections, feature matching GANs suffer from the following drawbacks: 1) the first-order moment matching objective does not prevent the generator from collapsing (missing coverage); 2) feature matching can generate high-density samples inside manifold; 3) the discriminator objective does not encourage realization of condition (A3) in Assumption 1 as discussed in Section 3.1.3. Our approach aims to explicitly address the above drawbacks.

Following prior work [47, 136], we employ a GAN-like implicit generator. We first sample a latent variable z from a uniform distribution $\mathcal{U}(0,1)$ for each dimension, and then apply a deep convolutional network to transform z to a sample x.

Generator Entropy

Fundamentally, the first drawback concerns the entropy of the distribution of generated features, $\mathcal{H}(p_G(f))$. This connection is rather intuitive, as the collapsing issue is a clear sign of low entropy. Therefore, to avoid collapsing and increase coverage, we consider explicitly increasing the entropy.

Although the idea sounds simple and straightforward, there are two practical challenges. Firstly, as implicit generative models, GANs only provide samples rather than an analytic density form. As a result, we cannot evaluate the entropy exactly, which rules out the possibility of naive optimization. More problematically, the entropy is defined in a high-dimensional feature space, which is changing dynamically throughout the training process. Consequently, it is difficult to estimate and optimize the generator entropy in the feature space in a stable and reliable way. Faced with these difficulties, we consider two practical solutions.

The first method is inspired by the fact that input space is essentially static, where estimating and optimizing the counterpart quantities would be much more feasible. Hence, we instead increase the generator entropy in the *input space*, i.e., $\mathcal{H}(p_G(x))$, using a technique derived from an information theoretical perspective and relies on variational inference (VI). Specially, let $\mathcal{Z}$ be the latent variable space, and $\mathcal{X}$ be the input space. We introduce an additional encoder, $q : \mathcal{X} \mapsto \mathcal{Z}$, to define a variational upper bound of the negative entropy [29], $-\mathcal{H}(p_G(x)) \leq -\mathbb{E}_{x,z\sim p_G} \log q(z|x) = L_{\text{VI}}$. Hence, minimizing the upper bound L_{VI} effectively increases the generator entropy. In our implementation, we formulate q as a diagonal Gaussian with bounded variance, i.e. $q(z|x) = \mathcal{N}(\mu(x), \sigma^2(x))$, with $0 < \sigma(x) < \theta$, where $\mu(\cdot)$ and $\sigma(\cdot)$ are neural networks, and θ is the threshold to prevent arbitrarily large variance.

Alternatively, the second method aims at increasing the generator entropy in the feature space by optimizing an auxiliary objective. Concretely, we adapt the pull-away term (PT) [201] as the auxiliary cost, $L_{\text{PT}} = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j\neq i} \left(\frac{h(x_i)^\top h(x_j)}{\|h(x_i)\| \|h(x_j)\|} \right)^2$, where N is the size of a mini-batch and x are samples. Intuitively, the pull-away term tries to orthogonalize the features in each mini-batch by minimizing the squared cosine similarity. Hence, it has the effect of increasing the diversity of generated features and thus the generator entropy.

Generating Low-Density Samples

The second drawback of feature matching GANs is that high-density samples can be generated in the feature space, which is not desirable according to our analysis. Similar to the argument in Section 3.1.5, it is infeasible to directly minimize the density of generated features. Instead, we enforce the generation of samples with low density in the input space. Specifically, given a threshold ϵ, we minimize the following term as part of our objective:

$$\mathbb{E}_{x\sim p_G} \log p(x) \mathbb{I}[p(x) > \epsilon] \tag{3.4}$$

where $\mathbb{I}[\cdot]$ is an indicator function. Using a threshold ϵ, we ensure that only high-density samples are penalized while low-density samples are unaffected. Intuitively, this objective pushes the generated samples to "move" towards low-density regions defined by $p(x)$. To model the probability distribution over images, we simply adapt the state-of-the-art density estimation model for natural images, namely the PixelCNN++ [137] model. The PixelCNN++ model is used to estimate the density $p(x)$ in Eq. (3.4). The model is pretrained on the training set, and fixed during semi-supervised training.

Generator Objective and Interpretation

Combining our solutions to the first two drawbacks of feature matching GANs, we have the following objective function of the generator:

$$\min_G \quad -\mathcal{H}(p_G) + \mathbb{E}_{x\sim p_G} \log p(x)\mathbb{I}[p(x) > \epsilon] + \|\mathbb{E}_{x\sim p_G} h(x) - \mathbb{E}_{x\sim\mathcal{U}} h(x)\|^2. \tag{3.5}$$

This objective is closely related to the idea of complement generator discussed in Section 3.1.3. To see that, let's first define a target complement distribution in the input space as follows

$$p^*(x) = \begin{cases} \frac{1}{Z}\frac{1}{p(x)} & \text{if } p(x) > \epsilon \text{ and } x \in \mathcal{B}_x \\ C & \text{if } p(x) \leq \epsilon \text{ and } x \in \mathcal{B}_x, \end{cases}$$

where Z is a normalizer, C is a constant, and $\mathcal{B}_x$ is the set defined by mapping $\mathcal{B}$ from the feature space to the input space. With the definition, the KL divergence (KLD) between $p_G(x)$ and $p^*(x)$ is

$$\text{KL}(p_G\|p^*) = -\mathcal{H}(p_G) + \mathbb{E}_{x\sim p_G} \log p(x)\mathbb{I}[p(x) > \epsilon] + \mathbb{E}_{x\sim p_G}\Big(\mathbb{I}[p(x) > \epsilon] \log Z - \mathbb{I}[p(x) \leq \epsilon] \log C\Big).$$

The form of the KLD immediately reveals the aforementioned connection. Firstly, the KLD shares two exactly the same terms with the generator objective equation 3.5. Secondly, while $p^*(x)$ is only defined in $\mathcal{B}_x$, there is not such a hard constraint on $p_G(x)$. However, the feature matching term in Eq. equation 3.5 can be seen as softly enforcing this constraint by bringing generated samples "close" to the true data (Cf. Section 3.1.4). Moreover, because the identity function $\mathbb{I}[\cdot]$ has zero gradient almost everywhere, the last term in KLD would not contribute any informative gradient to the generator. In summary, optimizing our proposed objective equation 3.5 can be understood as minimizing the KL divergence between the generator distribution and a desired complement distribution, which connects our practical solution to our theoretical analysis.

Conditional Entropy

In order for the complement generator to work, according to condition (A3) in Assumption 1, the discriminator needs to have strong true-fake belief on unlabeled data, i.e., $\max_{k=1}^K w_k^\top h(x) > 0$. However, the objective function of the discriminator in [136] does not enforce a dominant class. Instead, it only needs $\sum_{k=1}^K P_D(k|x) > P_D(K+1|x)$ to obtain a correct decision boundary, while the probabilities $P_D(k|x)$ for $k \leq K$ can possibly be uniformly distributed. To guarantee the strong true-fake belief in the optimal conditions, we add a conditional entropy term to the discriminator objective and it becomes,

$$\begin{aligned}\max_D \quad & \mathbb{E}_{x,y\sim\mathcal{L}} \log p_D(y|x, y \le K) + \mathbb{E}_{x\sim\mathcal{U}} \log p_D(y \le K|x) + \\ & \mathbb{E}_{x\sim p_G} \log p_D(K+1|x) + \mathbb{E}_{x\sim\mathcal{U}} \sum_{k=1}^{K} p_D(k|x) \log p_D(k|x).\end{aligned} \tag{3.6}$$

By optimizing Eq. (3.6), the discriminator is encouraged to satisfy condition (A3) in Assumption 1. Note that the same conditional entropy term has been used in other semi-supervised learning methods [110, 149] as well, but here we motivate the minimization of conditional entropy based on our theoretical analysis of GAN-based semi-supervised learning.

To train the networks, we alternatively update the generator and the discriminator to optimize Eq. (3.5) and Eq. (3.6) based on mini-batches. If an encoder is used to maximize $\mathcal{H}(p_G)$, the encoder and the generator are updated at the same time.

Connection with the Generative Feature Learning Framework

Our approach substantiated the generative feature learning framework in Section 1.4 as follows:

- The generative model G is a GAN.
- The generative loss function l_g is defined as in Eq. (3.5).
- The target loss function l_t is defined as in Eq. (3.6).
- The two loss functions are jointly trained.

3.1.6 Experiments

We mainly consider three widely used benchmark datasets, namely MNIST, SVHN, and CIFAR-10. As in previous work, we randomly sample 100, 1,000, and 4,000 labeled samples for MNIST, SVHN, and CIFAR-10 respectively during training, and use the standard data split for testing. We use the 10-quantile log probability to define the threshold ϵ in Eq. equation 3.5. We add instance noise to the input of the discriminator [3, 146], and use spatial dropout [157] to obtain faster convergence.

Main Results

We compare the the results of our best model with state-of-the-art methods on the benchmarks in Table 3.1. Our proposed methods consistently improve the performance upon feature matching. We achieve new state-of-the-art results on all the datasets when only small discriminator architecture is considered. Our results are also state-of-the-art on MNIST and SVHN among all single-model results, even when compared with methods using self-ensembling and large discriminator architectures. Finally, note that because our method is actually orthogonal to VAT [110], combining VAT with our presented approach should yield further performance improvement in practice.

Methods	MNIST (# errors)	SVHN (% errors)	CIFAR-10 (% errors)
CatGAN [149]	191 ± 10	-	19.58 ± 0.46
SDGM [98]	132 ± 7	16.61 ± 0.24	-
Ladder network [131]	106 ± 37	-	20.40 ± 0.47
ADGM [98]	96 ± 2	22.86	-
FM [136] *	93 ± 6.5	8.11 ± 1.3	18.63 ± 2.32
ALI [39]	-	7.42 ± 0.65	17.99 ± 1.62
VAT small [110] *	136	6.83	14.87
Our best model *	**79.5 ± 9.8**	**4.25 ± 0.03**	**14.41 ± 0.30**
Triple GAN [90] *‡	91 ± 58	5.77 ± 0.17	16.99 ± 0.36
Π model [85] †‡	-	5.43 ± 0.25	16.55 ± 0.29
VAT+EntMin+Large [110]†	-	4.28	13.15

Table 3.1: Comparison with state-of-the-art methods as of the time of our publication (i.e., May 2017) on three benchmark datasets. Only methods without data augmentation are included. $*$ indicates using the same (small) discriminator architecture, † indicates using a larger discriminator architecture, and ‡ means self-ensembling.

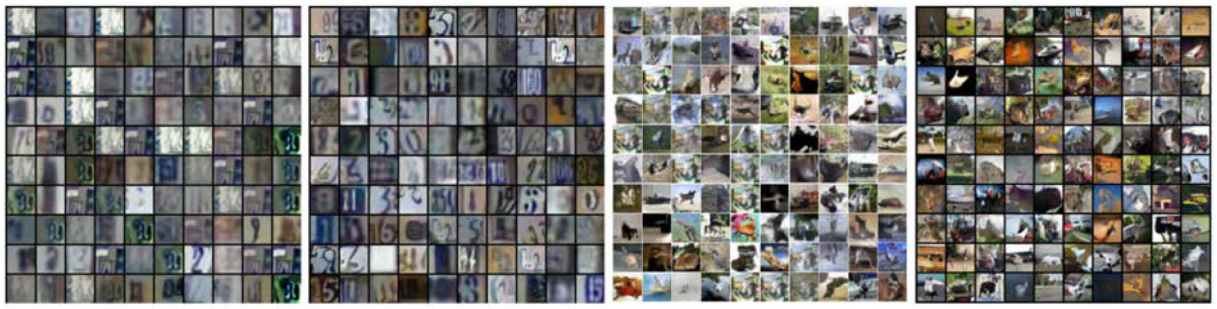

(a) FM on SVHN (b) Ours on SVHN (c) FM on CIFAR (d) Ours on CIFAR

Figure 3.6: Comparing images generated by FM and our model. FM generates collapsed samples, while our model generates diverse “bad” samples.

Ablation Study

We report the results of ablation study in Table 3.2. In the following, we analyze the effects of several components in our model, subject to the intrinsic features of different datasets.

First, the generator entropy terms (VI and PT) (Section 3.1.5) improve the performance on SVHN and CIFAR by up to 2.2 points in terms of error rate. Moreover, as shown in Fig 3.6, our model significantly reduces the collapsing effects present in the samples generated by FM, which also indicates that maximizing the generator entropy is beneficial. On MNIST, probably due to its simplicity, no collapsing phenomenon was observed with vanilla FM training [136] or in our setting. Under such circumstances, maximizing the generator entropy seems to be unnecessary, and the estimation bias introduced by approximation techniques can even hurt the performance.

Second, the low-density (LD) term is useful when FM indeed generates samples in high-

Setting	Error	Setting	Error
MNIST FM	85.0 ± 11.7	CIFAR FM	16.14
MNIST FM+VI	86.5 ± 10.6	CIFAR FM+VI	14.41
MNIST FM+LD	79.5 ± 9.8	CIFAR FM+VI+Ent	15.82
MNIST FM+LD+Ent	89.2 ± 10.5		

Setting	Error	Setting	Max log-p
SVHN FM	6.83	MNIST FM	-297
SVHN FM+VI	5.29	MNIST FM+LD	-659
SVHN FM+PT	4.63	SVHN FM+PT+Ent	-5809
SVHN FM+PT+Ent	4.25	SVHN FM+PT+LD+Ent	-5919
SVHN FM+PT+LD+Ent	4.19	SVHN 10-quant	-5622

Setting ϵ as q-th centile	$q = 2$	$q = 10$	$q = 20$	$q = 100$
Error on MNIST	77.7 ± 6.1	79.5 ± 9.8	80.1 ± 9.6	85.0 ± 11.7

Table 3.2: Ablation study. *FM* is feature matching. *LD* is the low-density enforcement term in Eq. (3.4). *VI* and *PT* are two entropy maximization methods described in Section 3.1.5. *Ent* means the conditional entropy term in Eq. (3.6). *Max log-p* is the maximum log probability of generated samples, evaluated by a PixelCNN++ model. *10-quant* shows the 10-quantile of true image log probability. *Error* means the number of misclassified examples on MNIST, and error rate (%) on others.

density areas. MNIST is a typical example in this case. When trained with FM, most of the generated hand written digits are highly realistic and have high log probabilities according to the density model (Cf. max log-p in Table 3.2). Hence, when applied to MNIST, LD improves the performance by a clear margin. By contrast, few of the generated SVHN images are realistic (Cf. Fig. 3.6a). Quantitatively, SVHN samples are assigned very low log probabilities (Cf. Table 3.2). As expected, LD has a negligible effect on the performance for SVHN. Moreover, the "max log-p" column in Table 3.2 shows that while LD can reduce the maximum log probability of the generated MNIST samples by a large margin, it does not yield noticeable difference on SVHN. This further justifies our analysis. Based on the above conclusion, we conjecture LD would not help on CIFAR where sample quality is even lower. Thus, we did not train a density model on CIFAR due to the limit of computational resources.

Third, adding the conditional entropy term has mixed effects on different datasets. While the conditional entropy (Ent) is an important factor of achieving the best performance on SVHN, it hurts the performance on MNIST and CIFAR. One possible explanation relates to the classic exploitation-exploration tradeoff, where minimizing conditional entropy favors exploitation and minimizing the classification loss favors exploration. During the initial phase of training, the discriminator is relatively *uncertain* and thus the gradient of the conditional entropy term might dominate. As a result, the discriminator learns to be more confident even on incorrect predictions, and thus gets trapped in local minima.

Lastly, we vary the values of the hyper-parameter ϵ in Eq. (3.5). As shown at the bottom of Table 3.2, reducing ϵ clearly leads to better performance, which further justifies our analysis in

Sections 3.1.4 and 3.1.3 that off-manifold samples are favorable.

Generated Samples

We compare the generated samples of FM and our approach in Fig. 3.6. The FM images in Fig. 3.6c are extracted from previous work [136]. While collapsing is widely observed in FM samples, our model generates diverse "bad" images, which is consistent with our analysis.

3.1.7 Conclusions

Contributions

For the first time, we provided theoretical answers to the following open questions in GAN-based semi-supervised learning:

- Why are good classification and good generation contradictory?
- How does the classifier benefit from joint training with the generator?

We showed that different from previous belief, we in fact need to generate complement data in order to achieve better classification performance. Essentially, we built a connection between GAN-based semi-supervised learning and the low-density separation principle.

Empirically, our approach substantially improves over vanilla feature matching GANs, and obtains state-of-the-art results at the time of publication on MNIST, SVHN, and CIFAR-10 when all methods are compared under the same discriminator architecture. Our results on MNIST and SVHN also represent the state of the art at the time of publication amongst all single-model results.

Subsequent Work

The idea of generating complement examples has inspired subsequent work on using generative models in the settings of semi-supervised learning and few-shot learning [77, 89, 175, 198]. Subsequent work has also explored different ideas in the field of semi-supervised learning, including using GANs to perform manifold regularization [87], using GANs to model local manifold [124], cross-view training [22], exploiting the similarities between data points with graph structures [97], GANs with a new Integral Probability Metric [111], deep co-training [125], and unsupervised data augmentation [184]. The unsupervised data augmentation achieves the current state-of-the-art results on SVHN and CIFAR-10, while it is possible that recent progress can be combined to yield better performance.

3.2 Semi-Supervised QA by Generating Questions

This section introduces the work originally published at ACL 2017 [193].

3.2.1 Motivation

Recently, various neural network models were proposed and successfully applied to the tasks of questions answering (QA) and/or reading comprehension [35, 185, 192]. While achieving state-of-the-art performance, these models rely on a large amount of labeled data. However, it is extremely difficult to collect large-scale question answering datasets. Historically, many of the question answering datasets have only thousands of question answering pairs, such as WebQuestions [11], MCTest [133], WikiQA [189], and TREC-QA [169]. Although larger question answering datasets with hundreds of thousands of question-answer pairs have been collected, including SQuAD [128], MSMARCO [113], and NewsQA [160], the data collection process is expensive and time-consuming in practice. This hinders real-world applications for domain-specific question answering.

Compared to obtaining labeled question answer pairs, it is trivial to obtain unlabeled text data. In this work, we study the following problem of semi-supervised question answering: is it possible to leverage unlabeled text to boost the performance of question answering models, especially when only a small amount of labeled data is available? The problem is challenging because conventional manifold-based semi-supervised learning algorithms [190, 204] cannot be straightforwardly applied. Moreover, since the main foci of most question answering tasks are extraction rather than generation, it is also not sensible to use unlabeled text to improve language modeling as in machine translation [54].

To better leverage the unlabeled text, we propose a novel neural framework called *Generative Domain-Adaptive Nets* (GDANs). The starting point of our framework is to use linguistic tags to extract possible answer chunks in the unlabeled text, and then train a generative model to generate questions given the answer chunks and their contexts. The model-generated question-answer pairs and the human-generated question-answer pairs can then be combined to train a question answering model, referred to as a *discriminative model* in the following text. However, there is discrepancy between the model-generated data distribution and the human-generated data distribution, which leads to suboptimal discriminative models. To address this issue, we further propose two domain adaptation techniques that treat the model-generated data distribution as a different domain. First, we use an additional *domain tag* to indicate whether a question-answer pair is model-generated or human-generated. We condition the discriminative model on the domain tags so that the discriminative model can learn to factor out domain-specific and domain-invariant representations. Second, we employ a reinforcement learning algorithm to fine-tune the generative model to minimize the loss of the discriminative model in an adversarial way.

In addition, we present a simple and effective baseline method for semi-supervised question answering. Although the baseline method performs worse than our GDAN approach, it is extremely easy to implement and can still lead to substantial improvement when only limited labeled data is available.

3.2.2 Related Prior Work

Semi-Supervised Learning. Semi-supervised learning has been extensively studied in literature [203]. A batch of novel models have been recently proposed for semi-supervised learning based on representation learning techniques, such as generative models [75], ladder networks [131] and graph embeddings [190]. However, most of the semi-supervised learning methods are based on combinations of the supervised loss $p(\mathbf{y}|\mathbf{x})$ and an unsupervised loss $p(\mathbf{x})$. In the context of reading comprehension, directly modeling the likelihood of a paragraph would not possibly improve the supervised task of question answering. Moreover, traditional graph-based semi-supervised learning [204] cannot be easily extended to modeling the unlabeled answer chunks.

Domain Adaptation. Domain adaptation has been successfully applied to various tasks, such as classification [43] and machine translation [19, 68]. Several techniques on domain adaptation [45] focus on learning distribution invariant features by sharing the intermediate representations for downstream tasks. Another line of research on domain adaptation attempt to match the distance between different domain distributions in a low dimensional space [7, 96]. There are also methods seeking a domain transition from the source domain to the target domain [46, 48, 116]. Our work gets inspiration from a practice in [68] and [19] based on appending domain tags. However, our method is different from the above methods in that we apply domain adaptation techniques to the outputs of a generative model rather than a natural data domain.

Question Answering. Various neural models based on attention mechanisms [18, 26, 35, 71, 140, 148, 161, 172, 176, 185] have been proposed to tackle the tasks of question answering and reading comprehension. However, the performance of these neural models largely relies on a large amount of labeled data available for training.

Learning with Multiple Models. GANs [47] formulated a adversarial game between a discriminative model and a generative model for generating realistic images. Ganin and Lempitsky [43] employed a similar idea to use two models for domain adaptation. Review networks [191] employ a discriminative model as a regularizer for training a generative model. In the context of machine translation, given a language pair, various recent work studied jointly training models to learn the mappings in both directions [162, 183].

3.2.3 Problem Definition

Let us first introduce the problem of *semi-supervised question answering*.

Let $L = \{q^{(i)}, a^{(i)}, p^{(i)}\}_{i=1}^{N}$ denote a question answering dataset of N instances, where $q^{(i)}$, $a^{(i)}$, and $p^{(i)}$ are the question, answer, and paragraph of the i-th instance respectively. The goal of question answering is to produce the answer $a^{(i)}$ given the question $q^{(i)}$ along with the paragraph $p^{(i)}$. We will drop the superscript $\cdot^{(i)}$ when the context is unambiguous. In our formulation, following the setting in SQuAD [128], we specifically focus on extractive question answering, where a is always a consecutive chunk of text in p. More formally, let $p = (p_1, p_2, \cdots, p_T)$ be a sequence of word tokens with T being the length, then a can always be represented as $a = (p_j, p_{j+1}, \cdots, p_{k-1}, p_k)$, where j and k are the start and end token indices respectively. The questions can also be represented as a sequence of word tokens $q = (q_1, q_2, \cdots, q_{T'})$ with length T'.

In addition to the labeled dataset L, in the semi-supervised setting, we are also given a set of unlabeled data, denoted as $U = \{a^{(i)}, p^{(i)}\}_{i=1}^{M}$, where M is the number of unlabeled instances.

Note that it is usually trivial to have access to an almost infinite number of paragraphs p from sources such as Wikipedia articles and other web pages. And since the answer a is always a consecutive chunk in p, we argue that it is also sensible to extract possible answer chunks from the unlabeled text using linguistic tags. We will discuss the technical details of answer chunk extraction in Section 3.2.5, and in the formulation of our framework, we assume that the answer chunks a are available.

Given both the labeled data L and the unlabeled data U, the goal of semi-supervised question answering is to learn a question answering model D that captures the probability distribution $\mathbb{P}(a|p, q)$. We refer to this question answering model D as the *discriminative model*, in contrast to the generative model that we will present in Section 3.2.4.

A Simple Baseline

We now present a simple baseline for semi-supervised question answering. Given a paragraph $p = (p_1, p_2, \cdots, p_T)$ and the answer $a = (p_j, p_{j+1}, \cdots, p_{k-1}, p_k)$, we extract

$$(p_{j-W}, p_{j-W+1}, \cdots, p_{j-1}, p_{k+1}, p_{k+2}, p_{k+W})$$

from the paragraph and treat it as the question. Here W is the window size and is set at 5 in our experiments so that the lengths of the questions are similar to human-generated questions. The context-based question-answer pairs on U are combined with human-generated pairs on L for training the discriminative model. Intuitively, this method extracts the contexts around the answer chunks to serve as hints for the question answering model. Surprisingly, this simple baseline method leads to substantial improvements when labeled data is limited.

3.2.4 Approach

Though the simple method described in Section 3.2.3 can lead to substantial improvement, we aim to design a learning-based model to move even further. In this section, we will describe the model architecture and the training algorithms for the GDANs. We will use a notation in the context of question answering following Section 3.2.3, but one should be able to extend the notion of GDANs to other applications as well.

The GDAN framework consists of two models, *a discriminative model* and a *generative model*. We will first discuss the two models in detail in the context of question answering, and then present an algorithm based on reinforcement learning to combine the two models.

Discriminative Model

The discriminative model learns the conditional probability of an answer chunk given the paragraph and the question, i.e., $\mathbb{P}(a|p, q)$. We employ a gated-attention (GA) reader [35] as our base model in this work, but our framework does not make any assumptions about the base models being used. The discriminative model is referred to as D.

The GA model consists of K layers with K being a hyper-parameter. Let $\mathbf{H}_p^k$ be the intermediate paragraph representation at layer k, and $\mathbf{H}_q$ be the question representation. The paragraph representation $\mathbf{H}_p^k$ is a $T \times d$ matrix, and the question representation $\mathbf{H}_q$ is a $T' \times d$ matrix, where d is the dimensionality of the representations. Given the paragraph p, we apply a bidirectional Gated Recurrent Unit (GRU) network [20] on top of the embeddings of the sequence $(p_1, p_2, \cdots, p_T)$, and obtain the initial paragraph representation $\mathbf{H}_p^0$. Given the question q, we also apply another bidirectional GRU to obtain the question representation $\mathbf{H}_q$.

The question and paragraph representations are combined with the gated-attention (GA) mechanism [35]. More specifically, for each paragraph token p_i, we compute

$$\alpha_j = \frac{\exp \mathbf{h}_{q,j}^T \mathbf{h}_{p,i}^{k-1}}{\sum_{j'=1}^{T'} \exp \mathbf{h}_{q,j'}^T \mathbf{h}_{p,i}^{k-1}}$$

$$\mathbf{h}_{p,i}^k = \sum_{j=1}^{T'} \alpha_j \mathbf{h}_{q,j} \odot \mathbf{h}_{p,i}^{k-1}$$

where $\mathbf{h}_{p,i}^k$ is the i-th row of $\mathbf{H}_p^k$ and $\mathbf{h}_{q,j}$ is the j-th row of $\mathbf{H}_q$.

Since the answer a is a sequence of consecutive word tokens in the paragraph p, we apply two softmax layers on top of $\mathbf{H}_p^K$ to predict the start and end indices of a, following Yang et al. [192].

Domain Adaptation with Tags

We will train our discriminative model on both model-generated question-answer pairs and human-generated pairs. However, even a well-trained generative model will produce questions somewhat different from human-generated ones. Learning from both human-generated data and model-generated data can thus lead to a biased model. To alleviate this issue, we propose to view the model-generated data distribution and the human-generated data distribution as two different data domains and explicitly incorporate domain adaptation into the discriminative model.

More specifically, we use a *domain tag* as an additional input to the discriminative model. We use the tag "d_true" to represent the domain of human-generated data (i.e., the *true* data), and "d_gen" for the domain of model-generated data. Following a practice in domain adaptation [19, 68], we append the domain tag to the end of both the questions and the paragraphs. By introducing the domain tags, we expect the discriminative model to factor out domain-specific and domain-invariant representations. At test time, the tag "d_true" is appended.

Generative Model

The generative model learns the conditional probability of generating a question given the paragraph and the answer, i.e., $\mathbb{P}(q|p, a)$. We implement the generative model as a sequence-to-sequence model [151] with a copy mechanism [53, 55].

The generative model consists of an encoder and a decoder. An encoder is a GRU that encodes the input paragraph into a sequence of hidden states $\mathbf{H}$. We inject the answer information by appending an additional zero/one feature to the word embeddings of the paragraph tokens; i.e., if a word token appears in the answer, the feature is set at one, otherwise zero.

Algorithm 1 Training Generative Domain-Adaptive Nets

```
**Input:** labeled data $L$, unlabeled data $U$, #iterations $T_G$ and $T_D$
Initialize $G$ by MLE training on $L$
Randomly initialize $D$
while not stopping do
  for $t \leftarrow 1$ to $T_D$ do
    Update $D$ to maximize $J(L, \text{d\_true}, D) + J(U_G, \text{d\_gen}, D)$ with SGD
  end for
  for $t \leftarrow 1$ to $T_G$ do
    Update $G$ to maximize $J(U_G, \text{d\_true}, D)$ with Reinforce and SGD
  end for
end while
return model $D$
```

The decoder is another GRU with an attention mechanism over the encoder hidden states $\mathbf{H}$. At each time step, the generation probabilities over all word types are defined with a copy mechanism:

$$\mathbf{p}_{\text{overall}} = g_t \mathbf{p}_{\text{vocab}} + (1 - g_t)\mathbf{p}_{\text{copy}} \tag{3.7}$$

where g_t is the probability of generating the token from the vocabulary, while $(1 - g_t)$ is the probability of copying a token from the paragraph. The probability g_t is computed based on the current hidden state $\mathbf{h}_t$:

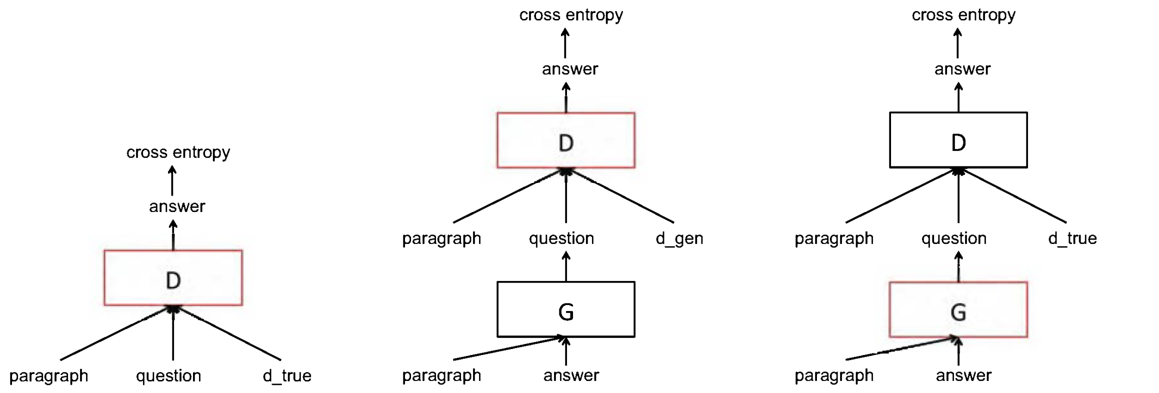
$$g_t = \sigma(\mathbf{w}_g^T \mathbf{h}_t)$$

where σ denotes the logistic function and $\mathbf{w}_g$ is a vector of model parameters. The generation probabilities $\mathbf{p}_{\text{vocab}}$ are defined as a softmax function over the word types in the vocabulary, and the copying probabilities $\mathbf{p}_{\text{copy}}$ are defined as a softmax function over the word types in the paragraph. Both $\mathbf{p}_{\text{vocab}}$ and $\mathbf{p}_{\text{copy}}$ are defined as a function of the current hidden state $\mathbf{h}_t$ and the attention results [53].

Training Algorithm

We first define the objective function of the GDANs, and then present an algorithm to optimize the given objective function. Similar to the Generative Adversarial Nets (GANs) [47] and adversarial domain adaptation [43], the discriminative model and the generative model have different objectives in our framework. However, rather than formulating the objective as an adversarial game between the two models [43, 47], in our framework, the discriminative model relies on the data generated by the generative model, while the generative model aims to match the model-generated data distribution with the human-generated data distribution using the signals from the discriminative model.

Given a labeled dataset $L = \{p^{(i)}, q^{(i)}, a^{(i)}\}_{i=1}^N$, the objective function of a discriminative model D for a supervised learning setting can be written as $\sum_{p^{(i)},q^{(i)},a^{(i)} \in L} \log \mathbb{P}_D(a^{(i)} | p^{(i)}, q^{(i)})$, where $\mathbb{P}_D$ is a probability distribution defined by the model D. Since we also incorporate domain tags into the model D, we denote the objective function as



(a) Training the discriminative model on labeled data. (b) Training the discriminative model on unlabeled data. (c) Training the generative model on unlabeled data.

Figure 3.7: Model architecture and training. Red boxes denote the modules being updated. “d_true” and “d_gen” are two domain tags. D is the discriminative model and G is the generative model. The objectives for the three cases are all to minimize the cross entropy loss of the answer chunks.

$$J(L, \text{tag}, D) = \frac{1}{|L|} \sum_{p^{(i)}, q^{(i)}, a^{(i)} \in L} \log \mathbb{P}_{D,\text{tag}}(a^{(i)} | p^{(i)}, q^{(i)})$$

meaning that the domain tag, “tag”, is appended to the dataset L. We use $|L| = N$ to denote the number of the instances in the dataset L. The objective function is averaged over all instances such that we can balance labeled and unlabeled data.

Let U_G denote the dataset obtained by generating questions on the unlabeled dataset U with the generative model G. The objective of the discriminative model is then to maximize J for both labeled and unlabeled data under the domain adaptation notions, i.e., $J(L, \text{d\_true}, D) + J(U_G, \text{d\_gen}, D)$.

Now we discuss the objective of the generative model. Similar to the dual learning [183] framework, one can define an auto-encoder objective. In this case, the generative model aims to generate questions that can be reconstructed by the discriminative model, i.e., maximizing $J(U_G, \text{d\_gen}, D)$. However, this objective function can lead to degenerate solutions because the questions can be thought of as an overcomplete representation of the answers [168]. For example, given p and a, the generative model might learn to generate trivial questions such as copying the answers, which does not contributed to learning a better D.

Instead, we leverage the discriminative model to better match the model-generated data distribution with the human-generated data distribution. We propose to define an adversarial training objective $J(U_G, \text{d\_true}, D)$. We append the tag “d_true” instead of “d_gen” for the model-generated data to “fool” the discriminative model. Intuitively, the goal of G is to generate "useful" questions where the usefulness is measured by the probability that the generated questions can be answered correctly by D.

The overall objective function now can be written as

$$\max_D J(L, \text{d\_true}, D) + J(U_G, \text{d\_gen}, D) \tag{3.8}$$

$$\max_G J(U_G, \text{d\_true}, D) \tag{3.9}$$

With the above objective function in mind, we present a training algorithm in Algorithm 1 to train a GDAN. We first pretrain the generative model on the labeled data L with maximum likelihood estimation (MLE):

$$\max_G \sum_{i=1}^{N} \sum_{t=1}^{T'} \log \mathbb{P}_G(q_t^{(i)} | q_{<t}^{(i)}, p^{(i)}, a^{(i)})$$

where $\mathbb{P}_G$ is the probability defined by Eq. 3.7.

We then alternatively update D and G based on their objectives. To update D, we sample one batch from the labeled data L and one batch from the unlabeled data U_G, and combine the two batches to perform a gradient update step. Since the output of G is discrete and non-differentiable, we use the Reinforce algorithm [181] to update G. The action space is all possible questions with length T' (possibly with padding) and the reward is the objective function $J(U_G, \text{d\_true}, D)$. Let θ_G be the parameters of G. The gradient can be written as

$$\begin{aligned} &\frac{\partial J(U_G, \text{d\_true}, D)}{\partial \theta_G} \\ &= \mathbb{E}_{\mathbb{P}_G(q|p,a)}(\log \mathbb{P}_{D,\text{d\_true}}(a|p,q) - b)\frac{\partial \log \mathbb{P}_G(q|p,a)}{\partial \theta_G} \end{aligned}$$

where we use an average reward from samples as the baseline b. We approximate the expectation $\mathbb{E}_{\mathbb{P}_G(q|p,a)}$ by sampling one instance at a time from $\mathbb{P}_G(q|p,a)$ and then do an update step. This training algorithm is referred to as reinforcement learning (RL) training in the following sections. The overall architecture and training algorithm are illustrated in Figure 3.7.

MLE vs RL. The generator G has two training phases–MLE training and RL training, which are different in that: 1) RL training does not require labels, so G can explore a broader data domain of p using unlabeled data, while MLE training requires labels; 2) MLE maximizes $\log P(q|p,a)$, while RL maximizes $\log P_D(a|q,p)$. Since $\log P(q|a,p)$ is the sum of $\log P(q|p)$ and $\log P(a|q,p)$ (plus a constant), maximizing $\log P(a|q,p)$ does not require modeling $\log P(q|p)$ that is irrelevant to QA, which makes optimization easier. Moreover, maximizing $\log P(a|q,p)$ is consistent with the goal of QA.

Connection to the Generative Feature Learning Framework

Our approach substantiated the generative feature learning framework in Section 1.4 as follows:

- The generative model G is an LSTM autoregressive language model.
- The generative loss function l_g is defined in Eq. (3.9).
- The target loss function l_t is defined in Eq. (3.8).
- The two loss functions are jointly trained.

3.2.5 Experiments

Answer Extraction

As discussed in Section 3.2.3, our model assumes that answers are available for unlabeled data. In this section, we introduce how we use linguistic tags and rules to extract answer chunks from unlabeled text.

To extract answers from massive unlabelled Wikipedia articles, we first sample 205,511 Wikipedia articles that are not used in the training, development and test sets in the SQuAD1.1 dataset. We extract the paragraphs from each article, and limit the length of each paragraph at the word level to be less than 850. In total, we obtain 950,612 paragraphs from unlabelled articles.

Answers in the SQuAD dataset can be categorized into ten types, i.e., "Date", "Other Numeric", "Person", "Location", "Other Entity", "Common Noun Phrase", "Adjective Phrase", "Verb Phrase", "Clause" and "Other" [128]. For each paragraph from the unlabeled articles, we utilize Stanford Part-Of-Speech (POS) tagger [158] to label each word with the corresponding POS tag, and implement a simple constituency parser to extract the noun phrase, verb phrase, adjective and clause based on a small set of constituency grammars. Next, we use Stanford Named Entity Recognizer (NER) [41] to assign each word with one of the seven labels, i.e., "Date", "Money", "Percent", "Location", "Organization" and "Time". We then categorize a span of consecutive words with the same NER tags of either "Money" or "Percent" as the answer of the type "Other Numeric". Similarly, we categorize a span of consecutive words with the same NER tags of "Organization" as the answer of the type "Other Entity". Finally, we subsample five answers from all the extracted answers for each paragraph according to the percentage of answer types in the SQuAD dataset. We obtain 4,753,060 answers in total, which is about 50 times larger than the number of answers in the SQuAD dataset.

Settings and Comparison Methods

The original SQuAD dataset consists of 87,636 training instances and 10,600 development instances. Since the test set is not published, we split 10% of the training set as the test set, and the remaining 90% serves as the actual training set. Instances are split based on articles; i.e., paragraphs in one article always appear in only one set. We tune the hyper-parameters and perform early stopping on the development set using the F1 scores, and the performance is evaluated on the test set using both F1 scores and exact matching (EM) scores [128].

We compare the following methods. **SL** is the supervised learning setting where we train the model D solely on the labeled data L. **Context** is the simple context-based method described in Section 3.2.3. **Context + domain** is the "Context" method with domain tags as described in Section 3.2.4. **Gen** is to train a generative model and use the generated questions as additional training data. **Gen + GAN** refers to the domain adaptation method using GANs [43]; in contrast to the original work, the generative model is updated using Reinforce. **Gen + dual** refers to the dual learning method [183]. **Gen + domain** is "Gen" with domain tags, while the generative model is trained with MLE and fixed. **Gen + domain + adv** is the approach we propose (Cf. Figure 3.7 and Algorithm 1), with "adv" meaning adversarial training based on Reinforce. We use our own implementation of "Gen + GAN" and "Gen + dual", since the GAN model [43] does not handle discrete features and the dual learning model [183] cannot be directly applied to question answering. When implementing these two baselines, we adopt the learning schedule introduced by Ganin and Lempitsky [43], i.e., gradually increasing the weights of the gradients for the generative model G.

Results and Analysis

We study the performance of different models with varying labeling rates and unlabeled dataset sizes. Labeling rates are the percentage of training instances that are used to train D. The results are reported in Table 3.4. Though the unlabeled dataset we collect consists of around 5 million instances, we also sample a subset of around 50,000 instances to evaluate the effects of the size of unlabeled data. The highest labeling rate in Table 3.4 is 0.9 because 10% of the training instances are used for testing. Since we do early stopping on the development set using the F1 scores, we also report the development F1. We report two metrics, the F1 scores and the exact matching (EM) scores [128], on the test set. All metrics are computed using the official evaluation scripts.

SL v.s. SSL. We observe that semi-supervised learning leads to consistent improvements over supervised learning in all cases. Such improvements are substantial when labeled data is limited. For example, the GDANs improve over supervised learning by 9.87 points in F1 and 7.26 points in EM when the labeling rate is 0.1. With our semi-supervised learning approach, we can use only 0.1 training instances to obtain even better performance than a supervised learning approach with 0.2 training instances, saving more than half of the labeling costs.

Comparison with Baselines. By comparing "Gen + domain + adv" with "Gen + GAN" and "Gen + Dual", it is clear that the GDANs perform substantially better than GANs and dual learning. With labeling rate 0.1, GDANs outperform dual learning and GANs by 2.47 and 4.29 points respectively in terms of F1.

Ablation Study. We also perform an ablation study by examining the effects of "domain" and "adv" when added to "gen". It can be seen that both the domain tags and the adversarial training contribute to the performance of the GDANs when the labeling rate is equal to or less than 0.5. With labeling rate 0.9, adding domain tags still leads to better performance but adversarial training does not seem to improve the performance by much.

Unlabeled Data Size. Moreover, we observe that the performance can be further improved when a larger unlabeled dataset is used, though the gain is relatively less significant compared to changing the model architecture. For example, increasing the unlabeled dataset size from 50K to 5M, the performance of GDANs increases by 0.38 points in F1 and 0.52 points in EM.

Context-Based Method. Surprisingly, the simple context-based method, though performing worse than GDANs, still leads to substantial gains; e.g., 7.00 points in F1 with labeling rate 0.1. Adding domain tags can improve the performance of the context-based method as well.

MLE vs RL. We plot the loss curve of $-J(U_G, \text{d\_gen}, D)$ for both the MLE-trained generator ("Gen + domain") and the RL-trained generator ("Gen + domain + adv") in Figure 3.8. We observe that the training loss for D on RL-generated questions is lower than MLE-generated questions, which confirms that RL training maximizes $\log P(a|p, q)$.

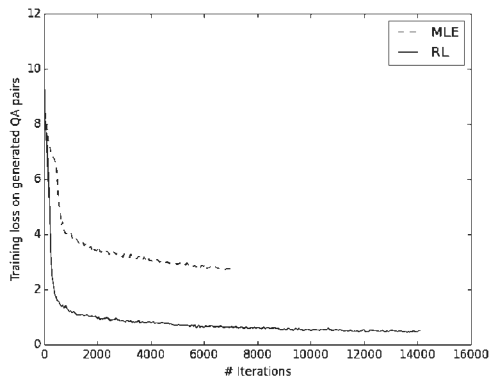


Figure 3.8: Comparison of discriminator training loss $-J(U_G, \text{d\_gen}, D)$ on generated QA pairs. The lower the better. MLE refers to questions generated by maximum likelihood training, and RL refers to questions generated by reinforcement learning.

Samples of Generated Questions. We present some questions generated by our model in Table 3.3. The generated questions are post-processed by removing repeated subs-sequences. Compared to MLE-generated questions, RL-generated questions are more informative (Cf., P1, P2, and P4), and contain less "UNK" (unknown) tokens (Cf., P1). Moreover, both semantically and syntactically, RL-generated questions are more accurate (Cf., P3 and P5).

Limitations

Despite the improvement shown in Table 3.4, there are a couple of limitations of our study. First, our approach was developed right after the release of the SQuAD dataset [128] based on an early question answering architecture GA Reader [35]. As a result, the results we obtained using semi-supervised learning are far behind the current state of the art based on pretrained models (see Section 2.2). Second, our approach does not yield much improvement when all of the labeled data is used. It will be interesting future work to investigate the gains based on using pretrained models as the baselines, as well as using pretraining to improve the generator.

3.2.6 Conclusions

Contributions

Our work was the first to explore joint training of a neural question generator with a question answering model to improve the performance of question answering. Moreover, we propose the Generative Domain-Adaptive Nets that employ domain adaptation techniques on generative models with reinforcement learning. In addition, we show that our framework leads to substantial improvements over the GA supervised learning baseline by a large margin when limited labeled data is available.

Subsequent Work

Subsequent work has built upon the idea of jointly training a question generation model and a question answering model [147, 150, 153, 154, 197]. Different from the approaches based on generative modeling, [36] proposed to create cloze-style questions based on unlabeled corpora and used these questions to pretrain a question answering model.

Table 3.3: Sampled generated questions given the paragraphs and the answers. *P* means paragraphs, *A* means answers, *GQ* means groundtruth questions, and *Q* means questions generated by our models. *MLE* refers to maximum likelihood training, and *RL* refers to reinforcement learning so as to maximize $J(U_G, \text{d\_true}, D)$. We truncate the paragraphs to only show tokens around the answer spans with a window size of 20.

P1: is mediated by ige , which triggers degranulation of mast cells and basophils when cross - linked by antigen . type ii hypersensitivity occurs when antibodies bind to antigens on the patient ' s own cells , marking them for destruction . this **A:** type ii hypersensitivity **GQ:** antibody - dependent hypersensitivity belongs to what class of hypersensitivity ? **Q (MLE):** what was the UNK of the patient ' s own cells ? **Q (RL):** what occurs when antibodies bind to antigens on the patient ' s own cells by antigen when cross
P2: an additional warming of the earth ' s surface . they calculate with confidence that co0 has been responsible for over half the enhanced greenhouse effect . they predict that under a " business as usual " (bau) scenario , **A:** over half **GQ:** how much of the greenhouse effect is due to carbon dioxide ? **Q (MLE):** what is the enhanced greenhouse effect ? **Q (RL):** what the enhanced greenhouse effect that co0 been responsible for
P3:) narrow gauge lines , which are the remnants of five formerly government - owned lines which were built in mountainous areas . **A:** mountainous areas **GQ:** where were the narrow gauge rail lines built in victoria ? **Q (MLE):** what is the government government government - owned lines built ? **Q (RL):** what were the remnants of government - owned lines built in
P4: but not both). in 0000 , bankamericard was renamed and spun off into a separate company known today as visa inc . **A:** visa inc . **GQ:** what present - day company did bankamericard turn into ? **Q (MLE):** what was the separate company bankamericard ? **Q (RL):** what today as bankamericard off into a separate company known today as spun off into a separate company known today
P5: legrande writes that " the formulation of a single all - encompassing definition of the term is extremely difficult , if **A:** legrande **GQ:** who wrote that it is difficult to produce an all inclusive definition of civil disobedience ? **Q (MLE):** what is the term of a single all all all all encompassing definition of a single all **Q (RL):** what writes " the formulation of a single all - encompassing definition of the term all encompassing encompassing encompassing encompassing

Table 3.4: Performance with various labeling rates, unlabeled data sizes $|U|$, and methods. "Dev" denotes the development set, and "test" denotes the test set. F1 and EM are two metrics.

Labeling rate	$\|U\|$	Method	Dev F1	Test F1	Test EM
0.1	50K	SL	0.4262	0.3815	0.2492
0.1	50K	Context	0.5046	0.4515	0.2966
0.1	50K	Context + domain	0.5139	0.4575	0.3036
0.1	50K	Gen	0.5049	0.4553	0.3018
0.1	50K	Gen + GAN	0.4897	0.4373	0.2885
0.1	50K	Gen + dual	0.5036	0.4555	0.3005
0.1	50K	Gen + domain	0.5234	0.4703	0.3145
0.1	50K	Gen + domain + adv	**0.5313**	**0.4802**	**0.3218**
0.2	50K	SL	0.5134	0.4674	0.3163
0.2	50K	Context	0.5652	0.5132	0.3573
0.2	50K	Context + domain	0.5672	0.5200	0.3581
0.2	50K	Gen	0.5643	0.5159	0.3618
0.2	50K	Gen + GAN	0.5525	0.5037	0.3470
0.2	50K	Gen + dual	0.5720	0.5192	0.3612
0.2	50K	Gen + domain	0.5749	0.5216	0.3658
0.2	50K	Gen + domain + adv	**0.5867**	**0.5394**	**0.3781**
0.5	50K	SL	0.6280	0.5722	0.4187
0.5	50K	Context	0.6300	0.5740	0.4195
0.5	50K	Context + domain	0.6307	0.5791	0.4237
0.5	50K	Gen	0.6237	0.5717	0.4155
0.5	50K	Gen + GAN	0.6110	0.5590	0.4044
0.5	50K	Gen + dual	0.6368	0.5746	0.4163
0.5	50K	Gen + domain	**0.6378**	0.5826	0.4261
0.5	50K	Gen + domain + adv	0.6375	**0.5831**	**0.4267**
0.9	50K	SL	**0.6611**	0.6070	0.4534
0.9	50K	Context	0.6560	0.6028	0.4507
0.9	50K	Context + domain	0.6553	**0.6105**	0.4557
0.9	50K	Gen	0.6464	0.5970	0.4445
0.9	50K	Gen + GAN	0.6396	0.5874	0.4317
0.9	50K	Gen + dual	0.6511	0.5892	0.4340
0.9	50K	Gen + domain	**0.6611**	0.6102	**0.4573**
0.9	50K	Gen + domain + adv	0.6585	0.6043	0.4497
0.1	5M	SL	0.4262	0.3815	0.2492
0.1	5M	Context	0.5140	0.4641	0.3014
0.1	5M	Context + domain	0.5166	0.4599	0.3083
0.1	5M	Gen	0.5099	0.4619	0.3103
0.1	5M	Gen + domain	0.5301	0.4703	0.3227
0.1	5M	Gen + domain + adv	**0.5442**	**0.4840**	**0.3270**
0.9	5M	SL	0.6611	0.6070	0.4534
0.9	5M	Context	0.6605	0.6026	0.4473
0.9	5M	Context + domain	0.6642	0.6066	0.4548
0.9	5M	Gen	0.6647	0.6065	**0.4600**
0.9	5M	Gen + domain	**0.6726**	0.6092	0.4599
0.9	5M	Gen + domain + adv	0.6670	**0.6102**	0.4531

3.3 Semi-Supervised Learning on Graphs

In this section, we consider semi-supervised learning on graphs. We define a generative process to generate the random walk paths on a graph, and this generative modeling objective is jointly trained with the supervised learning objective for classification on graphs. This work was originally published at ICML 2016 [190].

3.3.1 Motivation

A large number of semi-supervised learning algorithms jointly optimize two training objective functions: the supervised loss over labeled data and the unsupervised loss over both labeled and unlabeled data. *Graph-based semi-supervised learning* defines the loss function as a weighted sum of the supervised loss over labeled instances and a graph Laplacian regularization term [8, 178, 202, 205]. The graph Laplacian regularization is based on the assumption that nearby nodes in a graph are likely to have the same labels. Graph Laplacian regularization is effective because it constrains the labels to be consistent with the graph structure.

Recently developed unsupervised representation learning methods learn embeddings that predict a distributional *context*, e.g. a word embedding might predict nearby *context words* [107, 120], or a node embedding might predict nearby nodes in a graph [121, 155]. Embeddings trained with distributional context can be used to boost the performance of related tasks. For example, word embeddings trained from a language model can be applied to part-of-speech tagging, chunking and named entity recognition [23, 194].

In this paper we consider not word embeddings but graph embeddings. Existing results show that graph embeddings are effective at classifying the nodes in a graph, such as user behavior prediction in a social network [121, 155]. However, the graph embeddings are usually learned separately from the supervised task, and hence do not leverage the label information in a specific task. Hence graph embeddings are in some sense complementary to graph Laplacian regularization that does not produce useful features itself and might not be able to fully leverage the distributional information encoded in the graph structure.

The main highlight of our work is to incorporate embedding techniques into the graph-based semi-supervised learning setting. We propose a novel graph-based semi-supervised learning framework, *Planetoid* (Predicting Labels And Neighbors with Embeddings Transductively Or Inductively from Data).

3.3.2 Background and Related Prior Work

Graph-Based Semi-Supervised Learning

Let L and U be the number of labeled and unlabeled instances. Let $\mathbf{x}_{1:L}$ and $\mathbf{x}_{L+1:L+U}$ denote the feature vectors of labeled and unlabeled instances respectively. The labels $y_{1:L}$ are also given. Based on both labeled and unlabeled instances, the problem of semi-supervised learning is defined as learning a classifier $f : \mathbf{x} \rightarrow y$. There are two learning paradigms, transductive learning and inductive learning. Transductive learning [202, 205] only aims to apply the classifier f on the unlabeled instances observed at training time, and the classifier does not generalize to unobserved instances. For instance, transductive support vector machine (TSVM) [67] maximizes the "unlabeled data margin" based on the low-density separation assumption that a good decision hyperplane lies on a sparse area of the feature space. Inductive learning [8, 178], on the other hand, aims to learn a parameterized classifier f that is generalizable to unobserved instances.

In addition to labeled and unlabeled instances, a graph, denoted as a $(L+U) \times (L+U)$ matrix A, is also given to graph-based semi-supervised learning methods. Each entry a_{ij} indicates the similarity between instance i and j, which can be either labeled or unlabeled. The graph A can either be derived from distances between instances [205], or be explicitly derived from external data, such as a knowledge graph [180] or a citation network between documents [65]. In this paper, we mainly focus on the setting that a graph is explicitly given and represents additional information not present in the feature vectors (e.g., the graph edges correspond to hyperlinks between documents, rather than distances between the bag-of-words representation of a document).

Graph-based semi-supervised learning is based on the assumption that nearby nodes tend to have the same labels. Generally, the loss function of graph-based semi-supervised learning in the binary case can be written as

$$\begin{aligned} &\sum_{i=1}^{L} l(y_i, f(x_i)) + \lambda \sum_{i,j} a_{ij} \|f(x_i) - f(x_j)\|^2 \\ = &\sum_{i=1}^{L} l(y_i, f(x_i)) + \lambda \mathbf{f}^T \Delta \mathbf{f} \end{aligned} \tag{3.10}$$

In Eq. (3.10), the first term is the standard supervised loss function, where $l(\cdot,\cdot)$ can be log loss, squared loss or hinge loss. The second term is the graph Laplacian regularization, which incurs a large penalty when similar nodes with a large w_{ij} are predicted to have different labels $f(x_i) \neq f(x_j)$. The graph Laplacian matrix Δ is defined as $\Delta = A - D$, where D is a diagonal matrix with each entry defined as $d_{ii} = \sum_j a_{ij}$. λ is a constant weighting factor. (Note that we omit the parameter regularization terms for simplicity.) Various graph-based semi-supervised learning algorithms define the loss functions as variants of Eq. (3.10). Label propagation [205] forces f to agree with labeled instances $y_{1:L}$; f is a label lookup table for unlabeled instances in the graph, and can be obtained with a closed-form solution. Learning with local and global consistency [202] defines l as squared loss and f as a label lookup table; it does not force f to agree with labeled instances. Modified Adsorption (MAD) [152] is a variant of label propagation that allows prediction on labeled instances to vary and incorporates node uncertainty. Manifold regularization [8] parameterizes f in the Reproducing Kernel Hilbert Space (RKHS) with l being squared loss or hinge loss. Since f is a parameterized classifier, manifold regularization is inductive and can naturally handle unobserved instances.

Semi-supervised embedding [178] extends the regularization term in Eq. (3.10) to be

$$\sum_{i,j} a_{ij} \|\mathbf{g}(x_i) - \mathbf{g}(x_j)\|^2,$$

where g represents embeddings of instances, which can be the output labels, hidden layers or auxiliary embeddings in a neural network. By extending the regularization from f to g, this method imposes stronger constraints on a neural network. Iterative classification algorithm (ICA) [139] uses a local classifier that takes the labels of neighbor nodes as input, and employs an iterative process between estimating the local classifier and assigning new labels.

Learning Embeddings

Extensive prior research has been done on learning graph embeddings. A probabilistic generative model was proposed to learn node embeddings that generate the edges in a graph [145]. A clustering method [58] was proposed to learn latent social states in a social network to predict social ties.

More recently, a number of embedding learning methods are based on the Skipgram model, which is a variant of the softmax model. Given an instance and its context, the objective of Skipgram is usually formulated as minimizing the log loss of predicting the context using the embedding of an instance as input features. Formally, let $\{(i, c)\}$ be a set of pairs of instance i and context c, the loss function can be written as

$$-\sum_{(i,c)} \log p(c|i) = -\sum_{(i,c)} \left(\mathbf{w}_c^T \mathbf{e}_i - \log \sum_{c' \in \mathcal{C}} \exp(\mathbf{w}_{c'}^T \mathbf{e}_i) \right) \tag{3.11}$$

where $\mathcal{C}$ is the set of all possible context, $\mathbf{w}$'s are parameters of the Skipgram model, and $\mathbf{e}_i$ is the embedding of instance i. Skipgram was first introduced to learn representations of words, known as word2vec [107]. In word2vec, for each training pair (i, c), the instance i is the current word whose embedding is under estimation; the context c is each of the surrounding words of i within a fixed window size in a sentence; the context space $\mathcal{C}$ is the vocabulary of the corpus. Skipgram was later extended to learn graph embeddings. Deepwalk [121] uses the embedding of a node to predict the context in the graph, where the context is generated by random walk. More specifically, for each training pair (i, c), the instance i is the current node whose embedding is under estimation; the context c is each of the neighbor nodes within a fixed window size in a generated random walk sequence; the context space $\mathcal{C}$ is all the nodes in the graph. LINE [155] extends the model to have multiple context spaces $\mathcal{C}$ for modeling both first and second order proximity.

Although Skipgram-like models for graphs have received much attention, many other models exist. TransE [13] learns the embeddings of entities in a knowledge graph jointly with their relations. Autoencoders were used to learn graph embeddings for clustering on graphs [156].

3.3.3 Approach

Following the notations in the previous section, the input to our method includes labeled instances $\mathbf{x}_{1:L}$, $y_{1:L}$, unlabeled instances $\mathbf{x}_{L+1:L+U}$ and a graph denoted as a matrix A. Each instance i has an embedding denoted as $\mathbf{e}_i$.

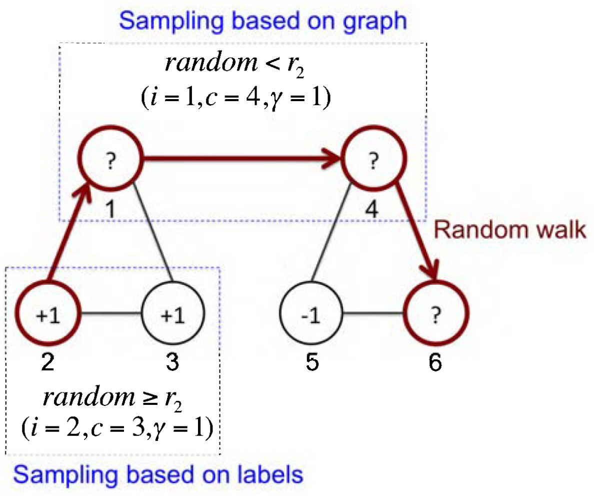


Figure 3.9: An example of sampling from context distribution $p(i, c, \gamma)$ when $\gamma = 1$ and $d = 2$. In circles, $+1$ denotes positive instances, -1 denotes negative instances, and $?$ denotes unlabeled instances. If $random < r_2$, we first sample a random walk $2 \to 1 \to 4 \to 6$, and then sample two nodes in the random walk within distance d. If $random \geq r_2$, we sample two instances with the same labels.

We formulate our framework based on feed-forward neural networks. Given the input feature vector $\mathbf{x}$, the k-th hidden layer of the network is denoted as $\mathbf{h}^k$, which is a nonlinear function of the previous hidden layer $\mathbf{h}^{k-1}$ defined as: $\mathbf{h}^k(\mathbf{x}) = \text{ReLU}(\mathbf{W}^k \mathbf{h}^{k-1}(\mathbf{x}) + b^k)$, where $\mathbf{W}^k$ and b^k are parameters of the k-th layer, and $\mathbf{h}^0(\mathbf{x}) = \mathbf{x}$. We adopt rectified linear unit $\text{ReLU}(x) = \max(0, x)$ as the nonlinear function in this work.

The loss function of our framework can be expressed as

$$\mathcal{L}_s + \lambda \mathcal{L}_u,$$

where $\mathcal{L}_s$ is a supervised loss of predicting the labels, and $\mathcal{L}_u$ is an unsupervised loss of predicting the graph context. In the following sections, we first formulate $\mathcal{L}_u$ by introducing how to sample context from the graph, and then formulate $\mathcal{L}_s$ to form our semi-supervised learning framework.

Sampling Context

We formulate the unsupervised loss $\mathcal{L}_u$ as a variant of Eq. (3.11). Given a graph A, the basic idea of our approach is to sample pairs of instance i and context c, and then formulate the loss $\mathcal{L}_u$ using the log loss $-\log p(c|i)$ as in Eq. (3.11). We first present the formulation of $\mathcal{L}_u$ by introducing negative sampling, and then discuss how to sample pairs of instance and context.

It is usually intractable to directly optimize Eq. (3.11) due to normalization over the whole context space $\mathcal{C}$. Negative sampling was introduced to address this issue [107], which samples negative examples to approximate the normalization term. In our case, we are sampling (i, c, γ) from a distribution, where i and c denote instance and context respectively, $\gamma = +1$ means (i, c)

is a positive pair and $\gamma = -1$ means negative. Given (i, c, γ), we minimize the cross entropy loss of classifying the pair (i, c) to a binary label γ:

$$-\mathbb{I}(\gamma = 1) \log \sigma(\mathbf{w}_c^T \mathbf{e}_i) - \mathbb{I}(\gamma = -1) \log \sigma(-\mathbf{w}_c^T \mathbf{e}_i),$$

where σ is the sigmoid function defined as $\sigma(x) = 1/(1 + e^{-x})$, and $\mathbb{I}(\cdot)$ is an indicator function that outputs 1 when the argument is true, otherwise 0. Therefore, the unsupervised loss with negative sampling can be written as

$$\mathcal{L}_u = -\mathbb{E}_{(i,c,\gamma)} \log \sigma(\gamma \mathbf{w}_c^T \mathbf{e}_i) \tag{3.12}$$

The distribution $p(i, c, \gamma)$ is conditioned on labels $y_{1:L}$ and the graph A. However, since they are the input to our algorithm and kept fixed, we drop the conditioning in our notation.

We now define the distribution $p(i, c, \gamma)$ directly using a sampling process, which is illustrated in Algorithm 2. There are two types of context that are sampled in this algorithm. The first type of context is based on the graph A, which encodes the structure (distributional) information, and the second type of context is based on the labels, which we use to inject label information into the embeddings. We use a parameter $r_1 \in (0, 1)$ to control the ratio of positive and negative samples, and use $r_2 \in (0, 1)$ to control the ratio of two types of context.

With probability r_2, we sample the context based on the graph A. We first uniformly sample a random walk sequence S. More specifically, we uniformly sample the first instance S_1 from the set $1 : L + U$. Given the previous instance $S_{k-1} = i$, the next instance $S_k = j$ is sampled with probability $a_{ij} / \sum_{j'=1}^{L+U} a_{ij'}$. With probability r_1, we sample a positive pair (i, c) from the set $\{(S_j, S_k) : |j - k| < d\}$, where d is another parameter determining the window size. With probability $(1 - r_1)$, we uniformly corrupt the context c to sample a negative pair.

With probability $(1 - r_2)$, we sample the context based on the class labels. Positive pairs have the same labels and negative pairs have different labels. Only labeled instances $1 : L$ are sampled.

Our random walk based sampling method is built upon Deepwalk [121]. In contrast to their method, our method handles real-valued A, incorporates negative sampling, and explicitly samples from labels with probability $(1 - r_2)$ to inject supervised information.

An example of sampling when $\gamma = 1$ is shown in Figure 3.9.

Transductive Formulation

In this section, we present a method that infers the labels of unlabeled instances $y_{L+1:L+U}$ without generalizing to unobserved instances. Transductive learning usually performs better than inductive learning because transductive learning can leverage the unlabeled test data when training the model [67].

We apply k layers on the input feature vector $\mathbf{x}$ to obtain $\mathbf{h}^k(\mathbf{x})$, and l layers on the embedding $\mathbf{e}$ to obtain $\mathbf{h}^l(\mathbf{e})$, as illustrated in Figure 3.10a. The two hidden layers are concatenated, and fed to a softmax layer to predict the class label of the instance. More specifically, the probability of predicting the label y is written as:

$$p(y|\mathbf{x}, \mathbf{e}) = \frac{\exp[\mathbf{h}^k(\mathbf{x})^T, \mathbf{h}^l(\mathbf{e})^T] \mathbf{w}_y}{\sum_{y'} \exp[\mathbf{h}^k(\mathbf{x})^T, \mathbf{h}^l(\mathbf{e})^T] \mathbf{w}_{y'}}, \tag{3.13}$$

Algorithm 2 Sampling Context Distribution $p(i, c, \gamma)$

```
**Input:** graph $A$, labels $y_{1:L}$, parameters $r_1, r_2, q, d$
Initialize triplet $(i, c, \gamma)$
if $random < r_1$ then $\gamma \leftarrow +1$ else $\gamma \leftarrow -1$
if $random < r_2$ then
    Uniformly sample a random walk $S$ of length $q$
    Uniformly sample $(S_j, S_k)$ with $|j - k| < d$
    $i \leftarrow S_j, c \leftarrow S_k$
    if $\gamma = -1$ then uniformly sample $c$ from $1 : L + U$
else
    if $\gamma = +1$ then
        Uniformly sample $(i, c)$ with $y_i = y_c$
    else
        Uniformly sample $(i, c)$ with $y_i \neq y_c$
    end if
end if
return $(i, c, \gamma)$
```

where $[\cdot, \cdot]$ denotes concatenation of two row vectors, the super script $\mathbf{h}^T$ denotes the transpose of vector $\mathbf{h}$, and $\mathbf{w}$ represents the model parameter.

Combined with Eq. (3.12), the loss function of transductive learning is defined as:

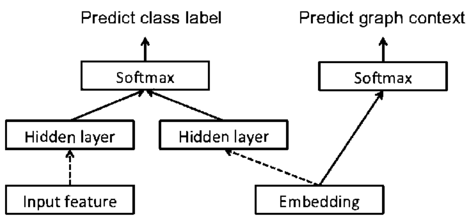
$$-\frac{1}{L}\sum_{i=1}^{L} \log p(y_i|\mathbf{x}_i, \mathbf{e}_i) - \lambda \mathbb{E}_{(i,c,\gamma)} \log \sigma(\gamma \mathbf{w}_c^T \mathbf{e}_i),$$

where the first term is defined by Eq. (3.13), and λ is a constant weighting factor. The first term is the loss function of class label prediction and the second term is the loss function of context prediction. This formulation is transductive because the prediction of label y depends on the embedding $\mathbf{e}$, which can only be learned for instances observed in the graph A during training time.

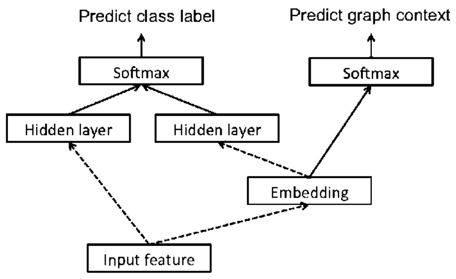
Inductive Formulation

While we consider transductive learning in the above formulation, in many cases, it is desirable to learn a classifier that can generalize to unobserved instances, especially for large-scale tasks. For example, machine reading systems [16] very frequently encounter novel entities on the Web and it is not practical to train a semi-supervised learning system on the entire Web. However, since learning graph embeddings is transductive in nature, it is not straightforward to do it in an inductive setting. Perozzi et al. [121] addressed this issue by retraining the embeddings incrementally, which is time consuming and does not scale (and not inductive essentially).

To make the method inductive, the prediction of label y should only depend on the input feature vector $\mathbf{x}$. Therefore, we define the embedding $\mathbf{e}$ as a parameterized function of feature



(a) Transductive Formulation



(b) Inductive Formulation

Figure 3.10: Network architecture: transductive v.s. inductive. Each dotted arrow represents a feed-forward network with an arbitrary number of layers (we use only one layer in our experiments). Solid arrows denote direct connections.

$\mathbf{x}$, as shown in Figure 3.11d. Similar to the transductive formulation, we apply k layers on the input feature vector $\mathbf{x}$ to obtain $\mathbf{h}^k(\mathbf{x})$. However, rather than using a "free" embedding, we apply l_1 layers on the input feature vector $\mathbf{x}$ and define it as the embedding $\mathbf{e} = \mathbf{h}^{l_1}(\mathbf{x})$. Then another l_2 layers are applied on the embedding $\mathbf{h}^{l_2}(\mathbf{e}) = \mathbf{h}^{l_2}(\mathbf{h}^{l_1}(\mathbf{x}))$, denoted as $\mathbf{h}^l(\mathbf{x})$ where $l = l_1 + l_2$. The embedding $\mathbf{e}$ in this formulation can be viewed as a hidden layer that is a parameterized function of the feature $\mathbf{x}$.

With the above formulation, the label y only depends on the feature $\mathbf{x}$. More specifically,

$$p(y|\mathbf{x}) = \frac{\exp[\mathbf{h}^k(\mathbf{x})^T, \mathbf{h}^l(\mathbf{x})^T]\mathbf{w}_y}{\sum_{y'} \exp[\mathbf{h}^k(\mathbf{x})^T, \mathbf{h}^l(\mathbf{x})^T]\mathbf{w}_{y'}} \tag{3.14}$$

Replacing $\mathbf{e}_i$ in Eq. (3.12) with $\mathbf{h}^{l_1}(\mathbf{x}_i)$, the loss function of inductive learning is

$$-\frac{1}{L}\sum_{i=1}^{L} \log p(y_i|\mathbf{x}_i) - \lambda \mathbb{E}_{(i,c,\gamma)} \log \sigma(\gamma \mathbf{w}_c^T \mathbf{h}^{l_1}(\mathbf{x}_i))$$

where the first term is defined by Eq. (3.14).

Training

We adopt stochastic gradient descent (SGD) [14] to train our model in mini-batch mode. We first sample a batch of labeled instances and take a gradient step to optimize the loss function of class label prediction. We then sample a batch of context (i, c, γ) and take another gradient step to optimize the loss function of context prediction. We repeat the above procedures for T_1 and T_2 iterations respectively to approximate the weighting factor λ. Algorithm 3 illustrates the SGD-based training algorithm for the transductive formulation. Similarly, we can replace $p(y_i|\mathbf{x}_i, \mathbf{e}_i)$ with $p(y_i|\mathbf{x}_i)$ in $\mathcal{L}_s$ to obtain the training algorithm for the inductive formulation. Let θ denote all model parameters. We update both embeddings $\mathbf{e}$ and parameters θ in transductive learning, and update only parameters θ in inductive learning. Before the joint training procedure, we apply a number of training iterations that optimize the unsupervised loss $\mathcal{L}_u$ alone and use the learned embeddings $\mathbf{e}$ as initialization for joint training.

Algorithm 3 Model Training (Transductive)

```
Input: $A$, $\mathbf{x}_{1:L+U}$, $y_{1:L}$, $\lambda$, batch iterations $T_1, T_2$ and sizes $N_1, N_2$
repeat
  for $t \leftarrow 1$ to $T_1$ do
    Sample a batch of labeled instances $i$ of size $N_1$
    $\mathcal{L}_s = -\frac{1}{N_1}\sum_i p(y_i|\mathbf{x}_i, \mathbf{e}_i)$
    Take a gradient step for $\mathcal{L}_s$
  end for
  for $t \leftarrow 1$ to $T_2$ do
    Sample a batch of context from $p(i, c, \gamma)$ of size $N_2$
    $\mathcal{L}_u = -\frac{1}{N_2}\sum_{(i,c,\gamma)} \log \sigma(\gamma \mathbf{w}_c^T \mathbf{e}_i)$
    Take a gradient step for $\mathcal{L}_u$
  end for
until stopping
```

Table 3.5: Dataset statistics.

DATASET	#CLASSES	#NODES	#EDGES
CITESEER	6	3,327	4,732
CORA	7	2,708	5,429
PUBMED	3	19,717	44,338
DIEL	4	4,373,008	4,464,261
NELL	210	65,755	266,144

Connection to the Generative Feature Learning Framework

Our approach substantiated the generative feature learning framework in Section 1.4 as follows:

- The generative model G is a multi-layer feed-forward network.
- The generative loss function l_g is to predict the neighbors in the random walk paths.
- The target loss function l_t is the standard classification loss.
- The two loss functions are jointly trained.

3.3.4 Experiments

In our experiments, *Planetoid-T* and *Planetoid-I* denote the transductive and inductive formulation of our approach. We compare our approach with label propagation (LP) [205], semi-supervised embedding (SemiEmb) [178], manifold regularization (ManiReg) [8], TSVM [67], and graph embeddings (GraphEmb) [121]. Another baseline method, denoted as *Feat*, is a linear softmax model that takes only the feature vectors $\mathbf{x}$ as input. We also derive a variant *Planetoid-G* that learns embeddings to jointly predict class labels and graph context without use of feature vectors. The architecture of Planetoid-G is similar to Figure 3.10a except that the *input feature*

Table 3.6: Accuracy on text classification. Upper rows are inductive methods and lower rows are transductive methods.

METHOD	CITESEER	CORA	PUBMED
FEAT	0.572	0.574	0.698
MANIREG	0.601	0.595	0.707
SEMIEMB	0.596	0.590	0.711
PLANETOID-I	**0.647**	0.612	**0.772**
TSVM	0.640	0.575	0.622
LP	0.453	0.680	0.630
GRAPHEMB	0.432	0.672	0.653
PLANETOID-G	0.493	0.691	0.664
PLANETOID-T	0.629	**0.757**	0.757

and the corresponding hidden layers are removed. Among the above methods, LP, GraphEmb and Planetoid-G do not use the features $\mathbf{x}$, while TSVM and Feat do not use the graph A. We include these methods into our experimental settings to better evaluate our approach. Our preliminary experiments on the text classification datasets show that the performance of our model is not very sensitive to specific choices of the network architecture[1]. We adapt the implementation of GraphEmb[2] to our Skipgram implementation. We use the Junto library [152] for label propagation, and SVMLight[3] for TSVM. We also use our own implementation of ManiReg and SemiEmb by modifying the symbolic objective function in Planetoid. In all of our experiments, we set the model hyper-parameters to $r_1 = 5/6$, $q = 10$, $d = 3$, $N_1 = 200$ and $N_2 = 200$ for Planetoid. We use the same r_1, q and d for GraphEmb, and the same N_1 and N_2 for ManiReg and SemiEmb. We tune r_2, T_1, T_2, the learning rate and hyper-parameters in other models based on an additional data split with a different random seed.

The statistics for five of our benchmark datasets are reported in Table 3.5. For each dataset, we split all instances into three parts, labeled data, unlabeled data, and test data. Inductive methods are trained on the labeled and unlabeled data, and tested on the test data. Transductive methods, on the other hand, are trained on the labeled, unlabeled data, and test data without labels.

Text Classification

We first considered three text classification datasets[4], Citeseer, Cora and Pubmed [139]. Each dataset contains bag-of-words representation of documents and citation links between the documents. We treat the bag-of-words as feature vectors $\mathbf{x}$. We construct the graph A based on the citation links; if document i cites j, then we set $a_{ij} = a_{ji} = 1$. The goal is to classify each

[1]We note that it is possible to develop other architectures for different applications, such as using a shared hidden layer for feature vectors and embeddings.

[2]`https://github.com/phanein/deepwalk`

[3]`http://svmlight.joachims.org/`

[4]`http://linqs.umiacs.umd.edu/projects//projects/lbc/`

Table 3.7: Recall@k on DIEL distantly-supervised entity extraction. Upper rows are inductive methods and lower rows are transductive methods. Results marked with $*$ are taken from the original DIEL paper [12] with the same data splits.

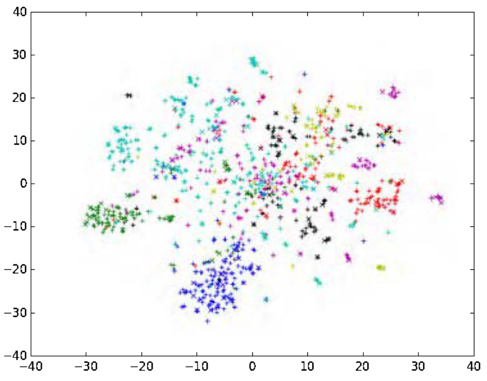
METHOD	RECALL@k
*FEAT	0.349
MANIREG	0.477
SEMIEMB	0.486
PLANETOID-I	**0.501**
*DIEL	0.405
*LP	0.162
GRAPHEMB	0.258
PLANETOID-G	0.394
PLANETOID-T	**0.500**
*UPPER BOUND	0.617

Table 3.8: Accuracy on NELL entity classification with labeling rates of 0.1, 0.01, and 0.001. Upper rows are inductive methods and lower rows are transductive methods.

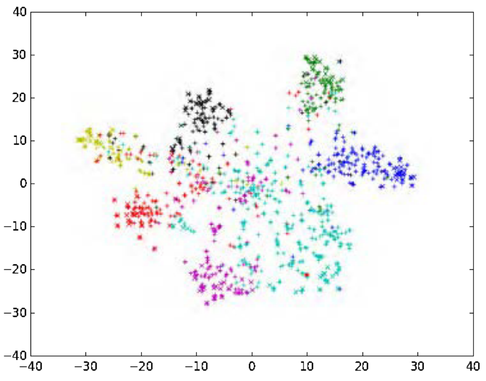
METHOD	0.1	0.01	0.001
FEAT	0.621	0.404	0.217
MANIREG	0.634	0.413	0.218
SEMIEMB	0.654	0.438	0.267
PLANETOID-I	0.702	0.598	0.454
LP	0.714	0.448	0.265
GRAPHEMB	0.795	0.725	0.581
PLANETOID-G/T	**0.845**	**0.757**	**0.619**

document into one class. We randomly sample 20 instances for each class as labeled data, $1,000$ instances as test data, and the rest are used as unlabeled data. The same data splits are used for different methods, and we compute the average accuracy for comparison.

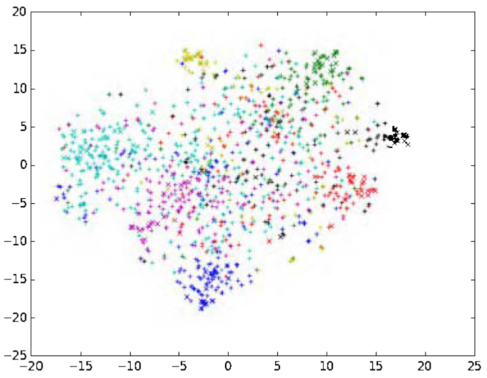
The experimental results are reported in Table 3.6. Among the inductive methods, Planetoid-I achieves the best performance on all the three datasets with the improvement of up to 6.1% on Pubmed, which indicates that our embedding techniques are more effective than graph Laplacian regularization. Among the transductive methods, Planetoid-T achieves the best performance on Cora and Pubmed, while TSVM performs the best on Citeseer. However, TSVM does not perform well on Cora and Pubmed. Planetoid-I slightly outperforms Planetoid-T on Citeseer and Pubmed, while Planetoid-T gets up to 14.5% improvement over Planetoid-I on Cora. We conjecture that in Planetoid-I, the feature vectors impose constraints on the learned embeddings, since they are represented by a parameterized function of the input feature vectors. If such constraints are appropriate, as is the case on Citeseer and Pubmed, it improves the non-convex optimization of



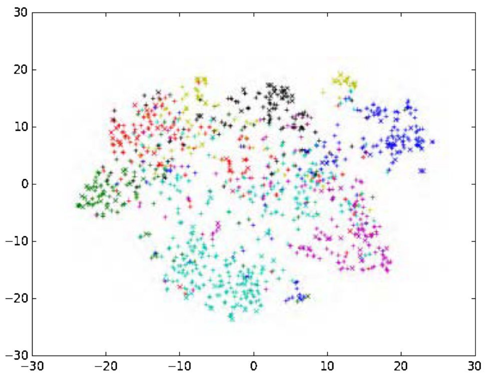
(a) GraphEmb



(b) Planetoid-T



(c) SemiEmb



(d) Planetoid-I

Figure 3.11: t-SNE Visualization of embedding spaces on the Cora dataset. Each color denotes a class.

embedding learning and leads to better performance. However, if such constraints rule out the optimal embeddings, the inductive model will suffer.

Planetoid-G consistently outperforms GraphEmb on all three datasets, which indicates that joint training with label information can improve the performance over training the supervised and unsupervised objectives separately. Figure 3.11 displays the 2-D embedding spaces on the Cora dataset using t-SNE [165]. Note that different classes are better separated in the embedding space of Planetoid-T than that of GraphEmb and SemiEmb, which is consistent with our empirical findings. We also observe similar results for the other two datasets.

Distantly-Supervised Entity Extraction

We next considered the DIEL (Distant Information Extraction using coordinate-term Lists) dataset [12]. The DIEL dataset contains pre-extracted features for each entity mention in text, and a graph that connects entity mentions to coordinate lists. The goal is to extract medical entities from text given feature vectors and the graph.

We follow the exact experimental setup as in the original DIEL paper [12], including data splits of different runs, preprocessing of entity mentions and coordinate lists, and evaluation. We treat the top-k entities given by a model as positive instances, and compute recall@k for evaluation (k is set to $240,000$ following the DIEL paper). We report the average result of 10 runs in Table 3.7, where *Feat* refers to a result obtained by SVM (referred to as DS-Baseline in the DIEL paper). The result of LP was also taken from [12]. *DIEL* in Table 3.7 refers to the method proposed by the original paper, which is an improved version of label propagation that trains classifiers on feature vectors based on the output of label propagation. We did not include TSVM into the comparison since it does not scale. Since we use Freebase as ground truth and some entities are not present in text, the upper bound of recall as shown in Table 3.7 is 0.617.

Both Planetoid-I and Planetoid-T significantly outperform all other methods. Each of Planetoid-I and Planetoid-T achieves the best performance in 5 out of 10 runs, and they give a similar recall on average, which indicates that there is no significant difference between these two methods on this dataset. Planetoid-G clearly outperforms GraphEmb, which again shows the benefit of joint training.

Entity Classification

We sorted out an entity classification dataset from the knowledge base of Never Ending Language Learning (NELL) [16] and a hierarchical entity classification dataset [32] that links NELL entities to text in ClueWeb09. We extracted the entities and the relations between entities from the NELL knowledge base, and then obtained text description by linking the entities to ClueWeb09. We use text bag-of-words representation as feature vectors of the entities.

We next describe how to construct the graph based on the knowledge base. We first remove relations that are not populated in NELL, including “generalizations”, “haswikipediaurl”, and “atdate”. In the knowledge base, each relation is denoted as a triplet (e_1, r, e_2), where e_1, r, e_2 denote head entity, relation, and tail entity respectively. We treat each entity e as a node in the graph, and each relation r is split as two nodes r_1 and r_2 in the graph. For each (e_1, r, e_2), we add two edges in the graph, (e_1, r_1) and (e_2, r_2).

We removed all classes with less than 10 entities. The goal is to classify the entities in the knowledge base into one of the 210 classes given the feature vectors and the graph. Let β be the labeling rate. We set β to 0.1, 0.01, and 0.001. $\max(\beta N, 1)$ instances are labeled for a class with N entities, so each class has at least one entity in the labeled data.

We report the results in Table 3.8. We did not include TSVM since it does not scale to such a large number of classes with the one-vs-rest scheme. Adding feature vectors does not improve the performance of Planetoid-T, so we set the feature vectors for Planetoid-T to be all empty, and therefore Planetoid-T is equivalent to Planetoid-G in this case.

Planetoid-I significantly outperforms the best of the other compared inductive methods—i.e., SemiEmb—by 4.8%, 16.0%, and 18.7% respectively with three labeling rates. As the labeling rate decreases, the improvement of Planetoid-I over SemiEmb becomes more significant.

Graph structure is more informative than features in this dataset, so inductive methods perform worse than transductive methods. Planetoid-G outperforms GraphEmb by 5.0%, 3.2% and 3.8%.

3.3.5 Conclusions

Contributions

For the first time, we proposed generative feature learning based methods in the setting of graph-based semi-supervised learning, in contrast to the conventional approaches that rely on graph Laplacian regularization. Since it is difficult to generalize graph embeddings to novel instances, we design a novel inductive approach that conditions embeddings on input features. Empirically, we show substantial improvement over existing methods.

Subsequent Work

Our method Planetoid has been extended and improved in subsequent work with wider applications such as recommendation systems and link prediction [59, 93, 188].

Subsequent to graph embedding based methods, graph neural networks [138] have also been popular for graph-based semi-supervised learning. These methods use the graph structure to define the computational graph, and as a result, the graph structure serves as passages of information. Notable examples include gated graph sequence neural networks [92], graph convolution networks [76], and graph attention networks [167]. Our Planetoid method has been used as a standard baseline for the development of a number of these methods and their variants.

Chapter 4

Conclusions

4.1 Summary and Comparison

Method	**Model G**	**Loss l_g**	**Loss l_t**	**Training**
Transformer-XL	AR	Likelihood	-	One-Phase
XLNet	AR	Likelihood	Target task loss	Pretraining & Finetuning
Comp GAN	GAN	Feature matching	$(K + 1)$-class loss	Joint training
Semi QA	AR	Adversarial	QA loss	Joint training
Planetoid	MLP	Likelihood	Classification loss	Joint training

Table 4.1: Summary and comparison of how different methods proposed in this work instantiate generative feature learning. AR means autoregressive modeling. "Comp GAN" refers to Complement GAN, "Semi QA" means semi-supervised QA, and "Planetoid" is our graph-based semi-supervised learning method.

We have presented the following generative feature learning methods:

1. **Transformer-XL.** Transformer-XL consists of a segment-level recurrence mechanism and a new relative positional encoding scheme. As a result, the architecture allows capturing longer-range dependency and resolves the context fragmentation problem. Transformer-XL achieves better language modeling performance on both long and short sequences, learns dependency that is 80% longer than RNNs and 450% longer than vanilla Transformers, and is up to 1,800 times faster than vanilla Transformers during language modeling evaluation.
2. **XLNet.** XLNet is a pretraining method based on the novel permutation language modeling objective. By using this objective, XLNet enables modeling bidirectional contexts using autoregressive models, avoiding making the independence assumption. XLNet also consists of a series of techniques to properly optimize the permutation language modeling objective, including a new way of parameterizing the conditional distribution and the two-stream attention mechanism. XLNet bridges the gap between language modeling and unsupervised pretraining and shows that progress in language modeling can be used to improve pretraining and thus downstream tasks.

3. **Complement GAN.** Theoretically we show that given the $(K+1)$-class classifier objective, good semi-supervised learning requires a complement generator that generates off-manifold samples. Our theoretical analysis answers the questions of how the classifier benefits from joint training with a generator and why good semi-supervised classification performance and a good generator cannot be obtained at the same time. Driven by theoretical understanding, we propose the definition of a preferred generator and use a new training objective to realize the goal.
4. **Generative Domain-Adaptive Nets.** Our semi-supervised question answering method trains a generative model to generate questions based on the unlabeled text, and combines model-generated questions with human-generated questions for training question answering models. We develop novel domain adaptation algorithms, based on reinforcement learning, to alleviate the discrepancy between the model-generated data distribution and the human-generated data distribution.
5. **Planetoid.** Given a graph between instances, Planetoid trains an embedding for each instance to jointly predict the class label and the neighborhood context in the graph. Planetoid has both transductive and inductive variants. In the transductive variant of our method, the class labels are determined by both the learned embeddings and input feature vectors, while in the inductive variant, the embeddings are defined as a parametric function of the feature vectors, so predictions can be made in instances not seed during training. Our method extends the concept of generative modeling to semi-supervised learning on graphs, which is essentially generating random walk paths on the graphs.

The above methods are summarized and compared in Table 4.1. We follow the notations in the definition of generative feature learning and describe how different methods instantiate the generative feature learning framework with custom loss functions, model formulation, and training paradigm.

Overall, our approaches obtained state-of-the-art results on over 30 benchmarks at the time of their publication, including natural language inference, question answering, text classification, language modeling, semi-supervised learning, etc., demonstrating the practical effectiveness.

4.2 Future Work

Although the generative feature learning methods proposed in this thesis have achieved remarkable success, there are still many questions remaining open.

- **Combining unsupervised pretraining and semi-supervised learning.** In the literature, unsupervised pretraining and semi-supervised learning have been developed to tackle different challenges. Unsupervised learning focuses on leveraging all-domain data, while semi-supervised learning is more about learning from unlabeled data with the same distribution. Clearly, there is a quality-quantity tradeoff. While unsupervised pretraining benefits from quantity, semi-supervised learning has the advantages of data quality (i.e., closer to the i.i.d assumption). Recently [184] attempted to combine the advantages of both by performing semi-supervised learning on top of pretrained models. However, the main focus was classification, and it will be interesting to study unifying or combining semi-supervised learning and unsupervised learning in a more effective way with applications to more tasks.

- **Efficient unsupervised pretraining.** Unsupervised pretraining methods heavily rely on large-scale computation and data, which prohibits most of the research community from working on meaningful related research and might eventually slow down the development of the area. There are two possible, interesting future directions to alleviate this problem. The first is to develop architectures that are more data efficient and parameter efficient. The second is to find a medium-scale *surrogate* setting where progress could be translated to benefits in pretraining. For example, since XLNet bridges the gap between language modeling and unsupervised pretraining, progress in language modeling might improve pretraining. At the same time, medium-scale language modeling benchmarks are prevalent, which forms a meaningful surrogate.
- **Pretraining for few-shot learning.** Unsupervised pretraining still does not solve all the NLP tasks, especially on tasks with little labeled data. For one thing, as mentioned above, combination with semi-supervised learning approaches might lead to improvements. For another, it is critical to develop unsupervised pretraining methods that are able to perform few-shot learning.
- **Effects of data scale/quality in unsupervised pretraining.** The effects of adding more data to unsupervised pretraining are not yet clear. Since curated corpora like Wikipedia and BooksCorpus are relatively scarce, additional data often come from the Web, which results in decrease of quality. It becomes a critical question of how to learn from relatively low quality but more abundant data. It will also be interesting to study how to jointly learn from curated text and Web text without being dominated by Web text.
- **XLNet for computer vision.** It will be intriguing to extend XLNet to computer vision, because Transformer-XL might have the potential of modeling longer-range dependency in images, which is very hard for CNNs.
- **GAN-based semi-supervised learning for text.** Applying our complement GANs approach to text requires generating discrete data, which results in the problem of non-differentiability. It will be interesting to explore ideas to tackle this challenge, such as methods that do not use explicit generators [109] or generation in the feature space.
- **Improving generator quality in semi-supervised QA.** The generator in semi-supervised QA was trained on limited data and thus was not able to generate realistic data. We conjecture that this is one of the reasons why semi-supervised QA was not able to improve performance when all training data was used. In the future, it will be interesting to also pretrain the generator using unlabeled data to address this problem.
- **More general intelligence.** It is one of our ultimate goals to achieve general intelligence that has the ability to reason, plan, and make decisions. For example, the ability of complex reasoning is one of the key signs of intelligence and has been targeted in a few benchmarks such as RACE [84] and HotpotQA [196]. The explainability issue has also been central for many settings [196]. We believe developing more effective, harder, more comprehensive datasets will provide proper testbeds for technical development and will be necessary for measuring the progress towards more general intelligence.

Specific to the field of NLP, we summarize the key goals and potential directions to explore for future work in Table 4.2

Goal	Direction
More general intelligence	Harder, more comprehensive datasets
Few-shot learning	Multi-task pretraining, long-range dependency modeling
Efficiency	Model compression, distillation, efficient architectures
Dialog and other seq2seq tasks	Unified encoder-decoder pretraining such as XLNet
Larger-scale model with more data	Improved optimization algorithms that scale better

Table 4.2: Summary of key goals and potential future directions.

Bibliography

[1] Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-level language modeling with deeper self-attention. *arXiv preprint arXiv:1808.04444*, 2018. xiii, 2, 8, 10, 11, 17, 18, 20, 21, 22, 41

[2] Anonymous. Bam! born-again multi-task networks for natural language understanding. anonymous preprint under review, 2018. 45

[3] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *NIPS 2016 Workshop on Adversarial Training*, volume 2016, 2017. 61

[4] Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. *arXiv preprint arXiv:1809.10853*, 2018. 16, 18, 41

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 8

[6] 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. 16

[7] Mahsa Baktashmotlagh, Mehrtash T Harandi, Brian C Lovell, and Mathieu Salzmann. Unsupervised domain adaptation by domain invariant projection. In *International Conference on Computer Vision*, pages 769–776, 2013. 66

[8] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006. 49, 54, 78, 79, 85

[9] Yoshua Bengio and Samy Bengio. Modeling high-dimensional discrete data with multi-layer neural networks. In *Advances in Neural Information Processing Systems*, pages 400–406, 2000. 41

[10] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003. 8

[11] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Conference on Empirical Methods in Natural Language Processing*, 2013. 65

[12] Lidong Bing, Sneha Chaudhari, Richard C Wang, and William W Cohen. Improving distant

supervision for information extraction using label propagation through lists. In *Conference on Empirical Methods in Natural Language Processing*, 2015. xiv, 87, 88, 89

[13] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795, 2013. 80

[14] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *International Conference on Computational Statistics*, pages 177–186. Springer, 2010. 84

[15] Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. Clueweb09 data set, 2009. 1, 42

[16] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. Toward an architecture for never-ending language learning. In *AAAI Conference on Artificial Intelligence*, volume 5, page 3, 2010. 83, 89

[17] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013. 16, 18

[18] Danqi Chen, Jason Bolton, and Christopher D Manning. A thorough examination of the cnn/daily mail reading comprehension task. In *Annual Meeting of the Association for Computational Linguistics*, 2016. 66

[19] Chenhui Chu, Raj Dabre, and Sadao Kurohashi. An empirical comparison of simple domain adaptation methods for neural machine translation. In *Annual Meeting of the Association for Computational Linguistics*, 2017. 66, 68

[20] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 68

[21] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*, 2016. 17

[22] Kevin Clark, Minh-Thang Luong, Christopher D Manning, and Quoc V Le. Semi-supervised sequence modeling with cross-view training. *arXiv preprint arXiv:1809.08370*, 2018. 64

[23] 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(Aug):2493–2537, 2011. 78

[24] Tim Cooijmans, Nicolas Ballas, César Laurent, Çağlar Gülçehre, and Aaron Courville. Recurrent batch normalization. *arXiv preprint arXiv:1603.09025*, 2016. 17

[25] Common Crawl. Common crawl. *URl: http://http://commoncrawl. org*, 2019. 1, 42

[26] Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. Attention-over-attention neural networks for reading comprehension. In *Annual Meeting of the Association for Computational Linguistics*, 2016. 66

[27] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087, 2015. 2, 3, 7, 31

[28] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 126–134. ACM, 2018. 45, 46

[29] Zihang Dai, Amjad Almahairi, Philip Bachman, Eduard Hovy, and Aaron Courville. Calibrating energy-based generative adversarial networks. In *International Conference on Learning Representations*, 2017. 59

[30] Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Ruslan R Salakhutdinov. Good semi-supervised learning that requires a bad gan. In *Advances in Neural Information Processing Systems*, pages 6510–6520, 2017. 49

[31] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019. 2, 7, 31, 36, 37, 41

[32] Bhavana Dalvi and William W Cohen. Hierarchical semi-supervised classification with incomplete class hierarchies. In *ACM International Conference on Web Search and Data Mining*, 2016. 89

[33] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. *arXiv preprint arXiv:1612.08083*, 2016. 16, 18

[34] 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. 2, 3, 7, 11, 31, 32, 37, 42, 43, 44, 45

[35] Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Gated-attention readers for text comprehension. In *Annual Meeting of the Association for Computational Linguistics*, 2017. 65, 66, 67, 68, 74

[36] Bhuwan Dhingra, Danish Pruthi, and Dheeraj Rajagopal. Simple and effective semi-supervised question answering. In *North American Chapter of the Association for Computational Linguistics Conference*, 2018. 75

[37] Adji B Dieng, Chong Wang, Jianfeng Gao, and John Paisley. Topicrnn: A recurrent neural network with long-range semantic dependency. *arXiv preprint arXiv:1611.01702*, 2016. 9

[38] Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. *arXiv preprint arXiv:1907.02544*, 2019. 2

[39] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *International Conference on Learning Representations*, 2016. 62

[40] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. In *International Conference on Learning Representations*, 2017. 50, 52

[41] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Annual Meeting of the Association for Computational Linguistics*, pages 363–370, 2005. 72

[42] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in

recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1019–1027, 2016. 8

[43] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, 2014. 66, 69, 72, 73

[44] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889, 2015. 31

[45] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *International Conference on Machine Learning*, pages 513–520, 2011. 66

[46] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073, 2012. 66

[47] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 2, 50, 58, 66, 69

[48] Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *International Conference on Computer Vision*, pages 999–1006, 2011. 66

[49] Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. Efficient softmax approximation for gpus. *arXiv preprint arXiv:1609.04309*, 2016. 8, 16

[50] Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache. In *International Conference on Learning Representations*, 2017. 16

[51] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013. 8

[52] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014. 12

[53] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Annual Meeting of the Association for Computational Linguistics*, 2016. 68, 69

[54] Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*, 2015. 65

[55] Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. Pointing the unknown words. In *Annual Meeting of the Association for Computational Linguistics*, 2016. 68

[56] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *ACM International on Conference on Information and Knowledge Management*, pages 55–64. ACM, 2016. 45

[57] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 17

[58] Mark S Handcock, Adrian E Raftery, and Jeremy M Tantrum. Model-based clustering for social networks. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 170(2):301–354, 2007. 80

[59] Ryohei Hisano. Semi-supervised graph embedding approach to dynamic link prediction. In *International Workshop on Complex Networks*, pages 109–121. Springer, 2018. 90

[60] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2, 8

[61] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001. 8

[62] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018. 44

[63] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, and Douglas Eck. An improved relative self-attention mechanism for transformer with application to music generation. *arXiv preprint arXiv:1809.04281*, 2018. 13

[64] Hakan Inan, Khashayar Khosravi, and Richard Socher. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462*, 2016. 19

[65] Ming Ji, Yizhou Sun, Marina Danilevsky, Jiawei Han, and Jing Gao. Graph regularized transductive classification on heterogeneous information networks. In *Machine Learning and Knowledge Discovery in Databases*, pages 570–586. Springer, 2010. 79

[66] Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein. Document context language models. *arXiv preprint arXiv:1511.03962*, 2015. 8

[67] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning*, volume 99, pages 200–209, 1999. 49, 79, 82, 85

[68] Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 2017. 66, 68

[69] Rie Johnson and Tong Zhang. Deep pyramid convolutional neural networks for text categorization. In *Annual Meeting of the Association for Computational Linguistics*, pages 562–570, 2017. 44

[70] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016. 18

[71] Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. Text understanding with the attention sum reader network. In *Annual Meeting of the Association for Computational Linguistics*, 2016. 66

[72] Nan Rosemary Ke, Anirudh Goyal ALIAS PARTH GOYAL, Olexa Bilaniuk, Jonathan Binas, Michael C Mozer, Chris Pal, and Yoshua Bengio. Sparse attentive backtracking: Temporal credit assignment through reminding. In *Advances in Neural Information Processing Systems*, pages 7650–7661, 2018. 9

[73] Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. Sharp nearby, fuzzy far away: How neural language models use context. *arXiv preprint arXiv:1805.04623*, 2018. 8, 19

[74] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 3

[75] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014. 3, 49, 50, 51, 66

[76] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. 49, 90

[77] Mark Kliger and Shachar Fleishman. Novelty detection with gan. *arXiv preprint arXiv:1802.10560*, 2018. 64

[78] Bryon Knol. cmix v13. `http://www.byronknoll.com/cmix.html`, 2017. 17

[79] Vid Kocijan, Ana-Maria Cretu, Oana-Maria Camburu, Yordan Yordanov, and Thomas Lukasiewicz. A surprisingly robust trick for winograd schema challenge. *arXiv preprint arXiv:1905.06290*, 2019. 45

[80] Ben Krause, Liang Lu, Iain Murray, and Steve Renals. Multiplicative lstm for sequence modelling. *arXiv preprint arXiv:1609.07959*, 2016. 17

[81] Oleksii Kuchaiev and Boris Ginsburg. Factorization tricks for lstm networks. *arXiv preprint arXiv:1703.10722*, 2017. 18

[82] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018. 42

[83] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*, 2017. 43

[84] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*, 2017. 1, 93

[85] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *International Conference on Learning Representations*, 2017. 62

[86] Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015. 9

[87] Bruno Lecouat, Chuan-Sheng Foo, Houssam Zenati, and Vijay R Chandrasekhar. Semi-supervised learning with gans: Revisiting manifold regularization. *arXiv preprint arXiv:1805.08957*, 2018. 64

[88] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 50

[89] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *arXiv preprint arXiv:1711.09325*, 2017. 64

[90] Chongxuan Li, Kun Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2017. 51, 62

[91] Shuai Li, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5457–5466, 2018. 9

[92] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015. 90

[93] Jiongqian Liang, Peter Jacobs, Jiankai Sun, and Srinivasan Parthasarathy. Semi-supervised embedding in attributed networks with outliers. In *SIAM International Conference on Data Mining*, pages 153–161. SIAM, 2018. 90

[94] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 19

[95] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*, 2019. 45

[96] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. Learning transferable features with deep adaptation networks. In *International Conference on Machine Learning*, pages 97–105, 2015. 66

[97] Yucen Luo, Jun Zhu, Mengxi Li, Yong Ren, and Bo Zhang. Smooth neighbors on teacher graphs for semi-supervised learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8896–8905, 2018. 64

[98] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. In *International Conference on Machine Learning*, 2016. 3, 50, 51, 62

[99] Matt Mahoney. Large text compression benchmark, 2011. 16

[100] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6297–6308, 2017. 2, 3, 31

[101] Gábor Melis, Charles Blundell, Tomáš Kočiskỳ, Karl Moritz Hermann, Chris Dyer, and Phil Blunsom. Pushing the bounds of dropout. *arXiv preprint arXiv:1805.09208*, 2018. 19

[102] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016. 8, 16, 23

[103] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017. 18, 19

[104] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. An analysis of neural language

modeling at multiple scales. *arXiv preprint arXiv:1803.08240*, 2018. xiii, 16, 20

[105] Tomas Mikolov and Geoffrey Zweig. Context dependent recurrent neural network language model. *SLT*, 12(234-239):8, 2012. 8, 16

[106] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010. 2, 8, 12

[107] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013. 78, 80, 81

[108] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*, 2016. 44

[109] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing with virtual adversarial training. In *International Conference on Learning Representations*, 2016. 51, 93

[110] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 51, 61, 62

[111] Youssef Mroueh, Chun-Liang Li, Tom Sercu, Anant Raj, and Yu Cheng. Sobolev gan. *arXiv preprint arXiv:1711.04894*, 2017. 64

[112] Asier Mujika, Florian Meier, and Angelika Steger. Fast-slow recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 5915–5924, 2017. 17

[113] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016. 65

[114] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016. 2

[115] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016. 41

[116] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011. 66

[117] Xiaoman Pan, Kai Sun, Dian Yu, Heng Ji, and Dong Yu. Improving question answering with external knowledge. *arXiv preprint arXiv:1902.00993*, 2019. 43

[118] Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. English gigaword fifth edition, linguistic data consortium. *Linguistic Data Consortium, Philadelphia.*, 2011. 1, 42

[119] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *CoRR, abs/1211.5063*, 2012. 4

[120] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. *Conference on Empirical Methods in Natural Language Processing*, 12:1532–1543, 2014. 78

[121] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 701–710, 2014. 78, 80, 82, 83, 85

[122] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018. 2, 3, 7, 11, 31, 41

[123] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018. 19

[124] Guo-Jun Qi, Liheng Zhang, Hao Hu, Marzieh Edraki, Jingdong Wang, and Xian-Sheng Hua. Global versus localized generative adversarial nets. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1517–1525, 2018. 64

[125] Siyuan Qiao, Wei Shen, Zhishuai Zhang, Bo Wang, and Alan Yuille. Deep co-training for semi-supervised image recognition. In *European Conference on Computer Vision*, pages 135–152, 2018. 64

[126] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*, 2018. 2, 3, 7, 31, 41, 43

[127] Jack W Rae, Chris Dyer, Peter Dayan, and Timothy P Lillicrap. Fast parametric learning with activation memorization. *arXiv preprint arXiv:1803.10049*, 2018. 16

[128] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016. 1, 43, 44, 65, 66, 72, 73, 74

[129] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018. 43, 44

[130] Qiu Ran, Peng Li, Weiwei Hu, and Jie Zhou. Option comparison network for multiple-choice reading comprehension. *arXiv preprint arXiv:1903.03033*, 2019. 43

[131] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554, 2015. 49, 50, 51, 62, 66

[132] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment*, 11(3):269–282, 2017. 45

[133] Matthew Richardson, Christopher JC Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Conference on Empirical Methods in Natural Language Processing*, 2013. 65

[134] Devendra Singh Sachan, Manzil Zaheer, and Ruslan Salakhutdinov. Revisiting lstm

networks for semi-supervised text classification via mixed objective function. 2018. 44

[135] Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Artificial Intelligence and Statistics*, pages 448–455, 2009. 2

[136] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2016. 3, 50, 51, 52, 57, 58, 60, 62, 64

[137] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *Annual Meeting of the Association for Computational Linguistics*, 2017. 60

[138] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20 (1):61–80, 2008. 90

[139] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93, 2008. 80, 86

[140] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations*, 2017. 66

[141] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018. 13, 14, 18, 19, 20, 21, 22

[142] Noam Shazeer, Joris Pelemans, and Ciprian Chelba. Skip-gram language modeling using sparse non-negative matrix probability estimation. *arXiv preprint arXiv:1412.1454*, 2014. 18

[143] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017. 18

[144] Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, et al. Mesh-tensorflow: Deep learning for supercomputers. In *Advances in Neural Information Processing Systems*, pages 10434–10443, 2018. 18

[145] Tom AB Snijders and Krzysztof Nowicki. Estimation and prediction for stochastic block-models for graphs with latent block structure. *Journal of Classification*, 14(1):75–100, 1997. 80

[146] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised map inference for image super-resolution. In *International Conference on Learning Representations*, 2017. 61

[147] Linfeng Song, Zhiguo Wang, and Wael Hamza. A unified query-based generative model for question generation and question answering. *arXiv preprint arXiv:1709.01058*, 2017. 75

[148] Alessandro Sordoni, Philip Bachman, Adam Trischler, and Yoshua Bengio. Iterative alternating neural attention for machine reading. *arXiv preprint arXiv:1606.02245*, 2016.

66

[149] Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. In *International Conference on Learning Representations*, 2016. 3, 51, 61, 62

[150] Yibo Sun, Duyu Tang, Nan Duan, Tao Qin, Shujie Liu, Zhao Yan, Ming Zhou, Yuanhua Lv, Wenpeng Yin, Xiaocheng Feng, Bing Qin, and Ting Liu. Joint learning of question answering and question generation. *IEEE Transactions on Knowledge and Data Engineering*, 2019. 75

[151] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014. 68

[152] Partha Pratim Talukdar and Koby Crammer. New regularized algorithms for transductive learning. In *Machine Learning and Knowledge Discovery in Databases*, pages 442–457. Springer, 2009. 79, 86

[153] Duyu Tang, Nan Duan, Tao Qin, Zhao Yan, and Ming Zhou. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*, 2017. 75

[154] Duyu Tang, Nan Duan, Zhao Yan, Zhirui Zhang, Yibo Sun, Shujie Liu, Yuanhua Lv, and Ming Zhou. Learning to collaborate for question answering and asking. In *North American Chapter of the Association for Computational Linguistics Conference*, 2018. 75

[155] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *The Web Conference*, pages 1067–1077, 2015. 78, 80

[156] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. Learning deep representations for graph clustering. In *AAAI Conference on Artificial Intelligence*, pages 1293–1299, 2014. 80

[157] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 648–656, 2015. 61

[158] Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 173–180, 2003. 72

[159] Trieu H Trinh, Andrew M Dai, Thang Luong, and Quoc V Le. Learning longer-term dependencies in rnns with auxiliary losses. *arXiv preprint arXiv:1803.00144*, 2018. 9

[160] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*, 2016. 44, 65

[161] Adam Trischler, Zheng Ye, Xingdi Yuan, and Kaheer Suleman. Natural language comprehension with the epireader. In *Annual Meeting of the Association for Computational Linguistics*, 2016. 66

[162] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. In *Annual Meeting of the Association for Computational Linguistics*, 2016. 66

[163] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Adversarial generator-encoder networks. *arXiv preprint arXiv:1704.02304*, 2017. 50

[164] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 17(1):7184–7220, 2016. 31, 33, 41

[165] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008. 88

[166] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010, 2017. 2, 8, 9, 13, 14, 18, 20, 36

[167] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 90

[168] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec): 3371–3408, 2010. 2, 3, 70

[169] Ellen M Voorhees and Dawn M Tice. Building a question answering test collection. In *International ACM SIGIR Conference*, 2000. 65

[170] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018. 1

[171] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019. 44

[172] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. In *International Conference on Learning Representations*, 2016. 66

[173] Tian Wang and Kyunghyun Cho. Larger-context language modelling. *arXiv preprint arXiv:1511.03729*, 2015. 8

[174] Wenlin Wang, Zhe Gan, Wenqi Wang, Dinghan Shen, Jiaji Huang, Wei Ping, Sanjeev Satheesh, and Lawrence Carin. Topic compositional neural language model. *arXiv preprint arXiv:1712.09783*, 2017. 9

[175] Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7278–7286, 2018. 64

[176] Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211*, 2016. 66

[177] Jason Weston, Ronan Collobert, Fabian Sinz, Léon Bottou, and Vladimir Vapnik. Inference with the universum. In *International Conference on Machine Learning*, pages 1009–1016. ACM, 2006. 51

[178] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer, 2012. 49, 78, 79, 85

[179] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014. 12

[180] Derry Wijaya, Partha Pratim Talukdar, and Tom Mitchell. Pidgin: ontology alignment using web text as interlingua. In *ACM International Conference on Information and Knowledge Management*, pages 589–598, 2013. 79

[181] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. 71

[182] Yuhuai Wu, Saizheng Zhang, Ying Zhang, Yoshua Bengio, and Ruslan R Salakhutdinov. On multiplicative integration with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 2856–2864, 2016. 9

[183] Yingce Xia, Di He, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, 2016. 66, 70, 72, 73

[184] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised data augmentation. *arXiv preprint arXiv:1904.12848*, 2019. 44, 64, 92

[185] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. In *International Conference on Learning Representations*, 2017. 65, 66

[186] Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. Word-entity duet representations for document ranking. In *International ACM SIGIR Conference*, pages 763–772. ACM, 2017. 46

[187] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. End-to-end neural ad-hoc ranking with kernel pooling. In *International ACM SIGIR Conference*, pages 55–64. ACM, 2017. 46

[188] Carl Yang, Lanxiao Bai, Chao Zhang, Quan Yuan, and Jiawei Han. Bridging collaborative filtering and semi-supervised learning: a neural approach for poi recommendation. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1245–1254. ACM, 2017. 90

[189] Yi Yang, Wen-tau Yih, and Christopher Meek. Wikiqa: A challenge dataset for open-domain question answering. In *Conference on Empirical Methods in Natural Language Processing*, 2015. 65

[190] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on Machine learning*, 2016. 49, 65, 66, 78

[191] Zhilin Yang, Ye Yuan, Yuexin Wu, William W Cohen, and Ruslan R Salakhutdinov. Review networks for caption generation. In *Advances in Neural Information Processing Systems*, pages 2361–2369, 2016. 66

[192] Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W Cohen, and Ruslan Salakhutdinov. Words or characters? fine-grained gating for reading comprehension. In *International Conference on Learning Representations*, 2017. 65, 68

[193] Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William W Cohen. Semi-supervised qa with generative domain-adaptive nets. In *Annual Meeting of the Association for Computational Linguistics*, 2017. 51, 64

[194] Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. Multi-task cross-lingual sequence tagging from scratch. In *International Conference on Learning Representations*, 2017. 78

[195] Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. Breaking the softmax bottleneck: A high-rank rnn language model. In *International Conference on Learning Representations*, 2018. 8, 19, 40

[196] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing*, 2018. 1, 93

[197] Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Sandeep Subramanian, Saizheng Zhang, and Adam Trischler. Machine comprehension by text-to-text neural question generation. *arXiv preprint arXiv:1705.02012*, 2017. 75

[198] Ruixiang Zhang, Tong Che, Zoubin Ghahramani, Yoshua Bengio, and Yangqiu Song. Metagan: An adversarial approach to few-shot learning. In *Advances in Neural Information Processing Systems*, pages 2365–2374, 2018. 64

[199] Shuailiang Zhang, Hai Zhao, Yuwei Wu, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. Dual co-matching network for multi-choice reading comprehension. *arXiv preprint arXiv:1901.09381*, 2019. xiv, 43, 47

[200] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657, 2015. 44

[201] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. In *International Conference on Learning Representations*, 2017. 59

[202] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in Neural Information Processing Systems*, 16(16):321–328, 2004. 49, 78, 79

[203] Xiaojin Zhu. Semi-supervised learning literature survey. 2005. 65

[204] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. *Citeseer*, 2002. 65, 66

[205] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *International conference on Machine learning*,

pages 912–919, 2003. 49, 54, 78, 79, 85

[206] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *International Conference on Computer Vision*, pages 19–27, 2015. 1, 42

[207] Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. Recurrent highway networks. *arXiv preprint arXiv:1607.03474*, 2016. 17, 19

[208] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 19