Université de Montréal

On Representation Learning for Generative Models of Text

par

Sandeep Subramanian

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée en vue de l'obtention du grade de
Philosophiæ Doctor (Ph.D.)
en Discipline

August 25, 2021

Université de Montréal

Faculté des arts et des sciences

Cette thèse intitulée

On Representation Learning for Generative Models of Text

présentée par

Sandeep Subramanian

a été évaluée par un jury composé des personnes suivantes :

Philippe Langlais
(président-rapporteur)

Christopher Pal
(directeur de recherche)

Yoshua Bengio
(codirecteur)

Aaron Courville
(membre du jury)

Samuel R. Bowman
(examinateur externe)

Ian Charest
(représentant du doyen de la FESP)

Sommaire

Cette thèse fait des petits pas dans la construction et la compréhension des systèmes d'apprentissage des représentations neuronales et des modèles génératifs pour le traitement du langage naturel. Il est présenté comme une thèse par article qui contient quatre travaux.

Dans le premier article, nous montrons que l'apprentissage multi-tâches peut être utilisé pour combiner les biais inductifs de plusieurs tâches d'apprentissage auto-supervisées et supervisées pour apprendre des représentations de phrases distribuées de longueur fixe à usage général qui obtiennent des résultats solides sur les tâches d'apprentissage par transfert en aval sans tout modèle de réglage fin.

Le deuxième article s'appuie sur le premier et présente un modèle génératif en deux étapes pour le texte qui modélise la distribution des représentations de phrases pour produire de nouveaux plongements de phrases qui servent de " contour neuronal " de haut niveau qui est reconstruit en mots avec un récurrent neuronal autorégressif conditionnel décodeur.

Le troisième article étudie la nécessité de représentations démêlées pour la génération de texte contrôlable. Une grande partie des systèmes de génération de texte contrôlables reposent sur l'idée que le contrôle d'un attribut (ou d'un style) particulier nécessite la construction de représentations dissociées qui séparent le contenu et le style (Hu et al., 2017; Shen et al., 2017b; Fu et al., 2017). Nous démontrons que les représentations produites dans des travaux antérieurs qui utilisent la formation contradictoire du domaine ne sont pas dissociées dans la pratique. Nous présentons ensuite une approche qui ne vise pas à apprendre des représentations démêlées et montrons qu'elle permet d'obtenir des résultats nettement meilleurs que les travaux antérieurs.

Dans le quatrième article, nous concevons des modèles de langage de transformateur qui apprennent les représentations à plusieurs échelles de temps et montrent que ceux-ci peuvent aider à réduire l'empreinte mémoire importante de ces modèles. Il présente trois architectures multi-échelles différentes qui présentent des compromis favorables entre la perplexité et l'empreinte mémoire.

Mots-clés: Apprentissage profond, traitement du langage naturel, apprentissage de la représentation, modèles génératifs, modélisation du langage

Summary

This thesis takes baby steps in building and understanding neural representation learning systems and generative models for natural language processing. It is presented as a thesis by article that contains four pieces of work.

In the first article, we show that multi-task learning can be used to combine the inductive biases of several self-supervised and supervised learning tasks to learn general-purpose fixed-length distributed sentence representations that achieve strong results on downstream transfer learning tasks without any model fine-tuning.

The second article builds on the first and presents a two-step generative model for text that models the distribution of sentence representations to produce novel sentence embeddings that serves as a high level "neural outline" that is reconstructed to words with a conditional autoregressive RNN decoder.

The third article studies the necessity of disentangled representations for controllable text generation. A large fraction of controllable text generation systems rely on the idea that control over a particular attribute (or style) requires building disentangled representations that separate content and style (Hu et al., 2017; Shen et al., 2017b; Fu et al., 2017). We demonstrate that representations produced in previous work that use domain adversarial training are not disentangled in practice. We then present an approach that does not aim to learn disentangled representations and show that it achieves significantly better results than prior work.

In the fourth article, we design transformer language models that learn representations at multiple time scales and show that these can help address the large memory footprint these models typically have. It presents three different multi-scale architectures that exhibit favorable perplexity vs memory footprint trade-offs.

Keywords: Deep Learning, Natural Language Processing, Representation Learning, Generative Models, Language Modeling

Contents

List of tables

List of figures

List of Acronyms and Abbreviations

BPTT	Backpropagation Through Time
CNN	Convolutional Neural Network
ERM	Empirical Risk Minimization
GAN	Generative Adversarial Network
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
MLE	Maximum Likelihood Estimation
MLP	Multi-Layer Perceptron
MTL	Multi-task Learning
NLP	Natural Language Processing
NLU	Natural Language Understanding

NMT	Neural Machine Translation
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
TBPTT	Truncated Backpropagation Through Time
VAE	Variational Autoencoder
WGAN	Wasserstein Generative Adversarial Network
WGAN-GP	Wasserstein Generative Adversarial Network with Gradient Penalty

Acknowledgements

I feel extremely privileged to have had the chance to pursue a PhD at Mila. I've learned an immense amount from amazing peers and mentors and I will always been grateful to everyone at this incredible lab.

Most importantly, I want to thank my advisor Chris for being the most supportive advisor one could ask for. His enthusiasm, positivity and optimism right from when I met him as a prospective student is something I hope to inculcate. I am also grateful to Yoshua Bengio, my co-advisor, for his early guidance and running a lab where one is surrounded by incredibly smart people, and providing us an environment where ideas are exchanged freely. Thank you Linda and Celine, without you, we'd be lost in an administrative maze.

I've made many wonderful friends over the course of my PhD. Dishank, Akilesh, Ankit, Vighnesh, David, Tegan, Olexa, Vinayak, Ankesh, Akram and Meghana - you have all been great company! Thank you Rithesh, Disha and Gunshi for all of the memorable conversations and outings at Beaver Lake. Chiheb for having a huge heart and always encouraging me with his kind words. Sarath, Prasanna, Chinna and Gaurav for making the winters eventful with board games. Rosemary, Anirudh and Alex for the Aunt Dai and Nos Thes dinner plans. I also want to thank my friends from undergrad and school that I've known since we were little kids - Krishna, Maddy, Suraj, Kanna, Aarthi, Deepthi, Dayn and Vivek. They have had such a big influence on me and I continue to hold a very special place for them in my heart. I'd also like to thank Juneki, Guillaume, Shrimai, Chirag, Dheeraj and Kazuya for an amazing time at CMU and for playing a big part in my decision to pursue a PhD.

I've also been fortunate to have been mentored by several amazing researchers during my internships. I'd like to thank Adam Trischler at Maluuba and Microsoft Research for being a great mentor and giving me a lot of freedom to explore many different and often silly ideas. Y-Lan Boureau and Marc'Aurelio Ranzato and for their guidance in not just one, but two internships at Facebook AI Research. Their wealth of knowledge and humility is something I've benefited from significantly. Ronan Collobert for hosting me at FAIR in Menlo Park and giving me the freedom to formulate and work on my ideas. Chris Pal at ElementAI for giving me the resources I needed to do large-scale NLP. Kundan Kumar at Descript, from whom I learned a great deal on how to do applied research at a company with a product that relies significantly on deep learning.

There were also several co-authors who have contributed substantially to the articles in this thesis and I haven't already thanked - Eric Smith, Ludovic Denoyer, Alessandro Sordoni and Aaron Courville. Thank you for giving me the opportunity to work with you. I've learned a lot from our interactions.

I would also like to thank Borealis AI for granting me a PhD fellowship and Microsoft Research for awarding my colleagues and I a research grant, giving me the resources I need to do research and write this thesis.

Finally, I am indebted to my family, My father and mother for giving me all the resources I needed to be a successful student right from my childhood. Thank you for giving me the right push whenever I needed it. My wife, Varsha, for constant support and understanding. Thank you for putting up with me and always volunteering to proof read my papers and this thesis.

Chapter 1

Introduction

Building computer systems that "understand" human language and can engage in meaningful conversations with them (Winograd, 1972) has been the goal for a large fraction of natural language processing (NLP) and AI researchers since the late 1950s. Initial attempts at building such systems involved designing rules and specifying by hand, the sequence in which to apply them to produce an appropriate response. This paradigm, despite having a few successes (Weizenbaum, 1966), needs to handle the vast complexity of human language with a set of hand design rules and heuristics, which has proven to be futile as building even very narrow domain systems require an extremely large number of rules.

Humans, in contrast, have the ability to continually learn from experiences and data. Attempts to impart such capabilities in NLP systems and several other application domains have been the focus of machine learning (ML) and statistics. These are largely data-driven approaches that have strong synergies with the vast amount of spoken and written language available on the internet. These techniques may either complement rule-based approaches in the field such as the induction and use of probabilistic context-free grammars (Baker, 1979) for parsing natural language or information extraction etc or be used "tabula rasa" without significant amounts of human specification such as n-gram language models[1] (Good, 1953) or IBM word alignment models (Brown et al., 1993).

This thesis focuses almost entirely on the latter class of systems where an entirely learning-based approach is adopted to take baby steps towards solving open problems in NLP. We focus on a particular class of learners - Artificial Neural Networks that in recent years have enabled significant advances across many fields. A few reasons for their success is that they can learn highly complex non-linear functions which allow them to model complex phenomena that we know and observe in language. They are easy to train on modern massively parallel computer hardware, making it easy to leverage the vast amounts of data available to us today. Neural networks have seen a rise and fall in popularity over the decades

[1]In practice, there tend to be human specified inductive biases such as the order of the language model and ways to smooth probabilities, but these are minimal when compared to rule-based methods

starting with the McCulloch and Pitt's (MCP) model (McCulloch & Pitts, 1943) and the Rosenblatt perceptron (Rosenblatt, 1958) which Minsky and Papert (Minsky & Papert, 1969) would later show couldn't learn non-linear functions such as XOR. While deep multi-layered networks could in theory model the XOR function, it wasn't clear at the time how one could train such a network. The backpropagation algorithm proposed later, a learning rule for deep differentiable multi-layered networks with non-linearities (Werbos, 1974; Rumelhart et al., 1986) made it possible to learn high-dimensional non-linear functions, including the XOR. This then spurred interest in training networks with different connectivity patterns such as convolutional networks (Fukushima et al., 1983; LeCun et al., 1990) that are better suited for processing image and audio data. In this thesis, we extensively use recurrent neural networks (Rumelhart et al., 1986; Jordan, 1986; Elman, 1990; Hochreiter & Schmidhuber, 1997b) that share connections across all of the words/characters in a sequence of text. They however fell out of favor towards the end of the 90s for a few reasons - training fairly deep neural networks with more than 7-8 layers was difficult in practice due to optimization issues and they lacked theoretical guarantees that other methods were able to offer. Greedy layer-wise pre-training (Bengio et al., 2007) and better parameter initialization (Glorot & Bengio, 2010) made it possible to train deeper and more hence more expressive models, which ultimately led to a large convolutional network that achieved state-of-the-art performance on object recognition benchmarks (Krizhevsky et al., 2012) and outperformed previous approaches that relied on hand-engineered features of images by a large margin. This kick-started the deep learning (LeCun et al., 2015) paradigm that focused on learning good internal representations of the underlying data (Bengio et al., 2009, 2013) without strong human specification. In the context of NLP, this would mean learning neural representations of characters, words, phrases, and sentences from the underlying linguistic form only.

Arguably the first clear demonstration that this paradigm is well suited to learn representations of language was Bengio et al. (2003) where a neural network that was trained to predict the next word in a text sequence learned good internal representations of words. "word2vec" (Mikolov et al., 2013), another approach to learning word representations, relies on the "distributional" hypothesis of language (Weaver et al., 1955; Firth, 1957), which assumes that a good fraction of the "meaning" of a word can be inferred from its context. Word2vec leverages this by formulating word representation learning as a classification problem of predicting either a word given its context or vice-versa. A lot of the work in this thesis and AI today more broadly centers around the idea that prediction is central to intelligence (Hawkins & Blakeslee, 2007). Our models learn representations of the world by predicting certain aspects of it. The tasks are typically set up in such a way that making good predictions requires learning a nontrivial amount of information about the data the model sees.

A common thread of research across multiple application domains in machine learning is to explore the hypothesis that generative models - ones that produce examples that look like samples from a data distribution such as images, text, speech, or videos will build useful internal representations (Vincent et al., 2008; Salimans et al., 2016; Dumoulin et al., 2016; Radford et al., 2018), loosely inspired by the famous quote from Richard Feynman *What I cannot create, I do not understand.* Generative models and in particular density models (ones that assign explicit probabilities to data samples) are used in a wide variety of settings such as translating between pairs of languages such as English and French (Bahdanau et al., 2014), summarizing documents (Rush et al., 2015) or conversing with humans (Vinyals & Le, 2015). In the early days of deep NLP, advances in representation learning seldom made their way into generative models except for word embeddings. Word embeddings provide representations of words that are independent of the specific context in which they are used ex: The embedding of the word " knocked" in the sentence "she knocked on the door" and "she knocked me out of the race" are used in roughly the same sense, but have fairly distinct meanings based on the context. In contrast, "contextualized" representations of words depend strongly on the context in which they are used. In practice, several NLP problems typically use task and context agnostic word representations and then build contextualized task-specific representations from scratch on top of them (Lample et al., 2016). Most parameters for a particular task were learned from scratch.

In contrast, neural computer vision pipelines at the same time, typically involved large-scale supervised pre-training of convolutional neural networks (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014) on Imagenet (Deng et al., 2009). The entire network or parts of it would be fine-tuned to different tasks.

This thesis started with the goal of building an analog of AlexNet (Krizhevsky et al., 2012) or VGG (Simonyan & Zisserman, 2014) for NLP, wherein a single recurrent neural net could be trained on large amounts of data and be fine-tuned to different tasks. It then continued to draw inspiration from computer vision, by exploring latent space generative models of text akin to (Nguyen et al., 2016) that build on advances in sentence representation learning for text. Finally, it discusses how disentangled and multi-scale representations impact generative models.

In recent years, models that do density estimation have been shown to be good general-purpose contextualized representation learners (Peters et al., 2018; Radford et al., 2018) and even similar approaches (Devlin et al., 2018; Lample & Conneau, 2019) that do not are fine-tuned for generative tasks indicating strong synergies between representation learning and generative models.

1.1. Thesis Overview

The thesis contains 4 articles that broadly focus on representation learning and generative modeling for text.

Chapter 2 starts with an introduction to the field of Machine Learning and typical categorizations of problems into supervised or unsupervised/self-supervised learning. It then discusses recent advances in training deep neural networks including adversarially trained models (Ganin et al., 2016; Goodfellow et al., 2014) and those particular to NLP.

Chapter 4 presents a large-scale multi-task learning approach that combines the inductive biases of different supervised and self-supervised learning methods to learning "general-purpose" representations of sentences. Representations are evaluated on downstream classification tasks and are probed to determine if they encode certain characteristics of sentences.

Chapter 6 builds on Chapter 4 to train generative models of text. It uses adversarial methods to train a generative model of text in a learned latent space of sentences.

Chapter 8 investigates the commonly held belief that disentangled latent spaces are needed for unsupervised "style transfer" between different domains of text. It presents an approach based on denoising autoencoders and backtranslation that does not explicitly build disentangled representations but achieves better performance than previous approaches that do.

Chapter 10 presents multi-scale transformer language models that attempt to build better inductive biases into large-scale transformer language models through multi-scale representations. Language exhibits structure at multiple time scales - grammar is often quite local while generating an entire novel requires coherence across multiple paragraphs and chapters and multi-scale models are a way to exploit this structure while also being memory efficient.

Chapter 11 presents an overall conclusion of the contributions in this thesis and presents some shortcomings of our work.

Chapter 2

Background

2.1. Machine Learning

Machine learning is an approach to Artificial Intelligence that builds systems or algorithms that “learn” from experience. Formally, Mitchell et al. (1997) define a learning machine as an algorithm that improves with experience on a particular task, given a certain performance measure to evaluate the system. In the context of this particular thesis, we are interested in building systems that improve their language understanding or generation capabilities as they are presented with more text.

In most settings, we assume a fixed number of data points is available to us, that are independently drawn from the same unknown data distribution. Some of that data is partitioned for our systems to learn on and the rest is to test how it generalizes to unseen examples. This idea of generalizing to data drawn from the same distribution is a fairly common assumption made in most machine learning systems since a large number of theoretical guarantees on the generalization properties of systems have relied on this assumption. This is however being questioned more as our systems do not generalize systematically and perform poorly when dealing with out-of-distribution data. For the sake of this introduction however, we will assume that our data is independent and identically distributed (iid).

2.1.1. Supervised Learning

Supervised learning typically entails learning functions that map high-dimensional inputs such as images, text, speech to low-dimensional categories as outputs. The low-dimensional categories are often obtained from human feedback such as a rating for a particular product review on an e-commerce website, the object present in an image or the name of the speaker in a particular audio recording.

These functions map a d-dimensional, typically real-valued vector input $\mathbf{x} \in \mathbb{R}^d$ from the space of possible inputs X to a discrete scalar output label y from the space of k possible outputs Y. We assume our data comes from an unknown joint probability distribution over inputs and outputs $p(\mathbf{x}, y)$ where $\mathbf{x} \in X$ and $y \in Y$ and we would like to learn a function $f_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^k$ that maps X to an estimate of $P(Y|X)$. We observe examples from this distribution $((\mathbf{x_1}, y_1) \dots (\mathbf{x_N}, y_N))$ The subscript θ is used to indicate that our function has parameters associated with it. This doesn't always have to be the case, and there exist "non-parameteric" models in machine learning, but in the context of this thesis that focuses on neural networks, we are interested in learning functions that often have many millions of parameters. There are many possible functions that can map from X to a distribution over Y, but we are often interested in learning one that makes as few mistakes as possible. We define a loss function $L(\hat{y}, y)$ that measures the penalty incurred when classifying y as $\hat{y}$ and try to optimize f_θ to minimize this loss given a finite amount of data. We can use this loss to define the empirical risk associated with a particular function as the expected loss over the entire training set $R_{emp}(X, Y, f_\theta) = \frac{1}{N} \sum\limits_{i=1}^{N} L(f_\theta(\mathbf{x_i}), y_i)$. For a given dataset, there may be many functions $f \in F$ in the hypothesis space, that achieve the same or even 0 empirical risk on the training data subset, but generalize poorly to unseen examples. In such cases, we want to bias our learners to certain solutions that may have better generalization properties such as those with low parameter norms with L1/L2 regularization, max-margin methods (Vapnik, 1963) or dropout (Srivastava et al., 2014) etc. Empirical Risk Minimization (ERM) tries to find $\text{argmin}_{f \in F} R_{emp}(f)$.

$f_\theta(\mathbf{x})$ outputs a probability distribution over possible class labels and the loss function used is typically the negative log-likelihood of the correct class $-\log p_\theta(Y = y|\mathbf{x})$. The frequentist approach to parameter estimation termed Maximum Likelihood Estimation (MLE) finds a setting of parameters that maximizes the likelihood of the observed data $p_\theta(Y|X) = \prod\limits_{i=1}^{N} p_\theta(y_i|\mathbf{x_i})$. Since the logarithm is a monotonically increasing function, we can maximize the $\log p_\theta(Y|X)$ and break the product into a bunch of sums $\log p_\theta(Y|X) = \sum\limits_{i=1}^{N} \log p_\theta(y_i|\mathbf{x_i})$. To turn it into a minimization problem, we can minimize the negative log-likelihood of the data, showing that empirical risk minimization (ERM) with the negative log-likelihood loss corresponds to maximum-likelihood estimation of the parameters of the model. At inference/test time, the model outputs $\text{argmax}_{\tilde{y}} p_\theta(\tilde{y}|\mathbf{x}_i)$.

2.2. Deep Learning

2.2.1. Artificial Neural Networks

Artificial Neural Networks are loosely inspired from biological ones in that they contain neurons that are connected to one another via synapses through several layers.

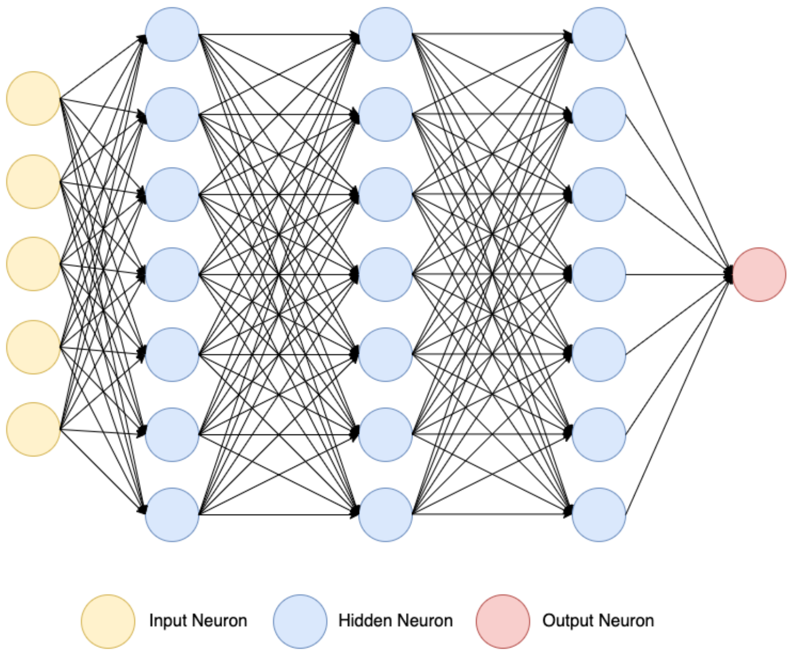


Fig. 1. A multi-layer perceptron with 5 input neurons, 7 hidden neurons, 3 hidden layers and 1 output neuron.

This connection is evident when looking at the structure of a multi-layer perceptron (MLP) in Figure 1. It contains several layers of weight matrices (and biases) that represent synaptic strengths of the connections between neurons (black arrows between subsequent layers). Given a high dimensional input signal $\mathbf{x} \in \mathbb{R}^d$, an MLP applies an affine transformation at every layer l to an input $\mathbf{h}^{l-1} \in \mathbb{R}^{d^l_{in}}$ using a matrix $\mathbf{W}^l \in \mathbb{R}^{d^l_{in} \times d^l_{out}}$ and bias vector $\mathbf{b}^l \in \mathbb{R}^{d^l_{out}}$ followed by a nonlinearity $f(\cdot)$ to produce an output $\mathbf{h}^l \in \mathbb{R}^{d^l_{out}}$. The non-linearity is generally performed element-wise and its presence in the network allows learning complex non-linear functions. The affine transformation is a matrix-vector product that can be intuitively thought of as implementing many independent weighted averages of the input. Each hidden neuron or computational unit (blue circle in Fig 1) implements one such weighted average using one column of the weight matrix $\mathbf{W}^l$. For example neuron j in layer l can be computed as

$$\mathbf{h}^l_j = f\Big(\sum_{i=1}^{d^l_{in}} \mathbf{W}^l_{ij}\mathbf{h}^{l-1}_i + \mathbf{b}^l_i\Big) \tag{2.1}$$

In general, by taking advantage of the simplicity of linear algebra notation, we can drop subscripts and write the expression for the representation of any hidden layer as

$$\mathbf{h}^l = f(\mathbf{W}^l\mathbf{h}^{l-1} + \mathbf{b}^l) \tag{2.2}$$

with the input to the network being $\mathbf{h}^0$. In the case of a supervised binary classification problem, the network will produce a single scalar output (like shown in Figure 1) which is typically soft-thresholded by a sigmoid function $\sigma(\cdot)$ to produce values in the closed interval $[0, 1]$.

$$\sigma(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{x}}} \tag{2.3}$$

With multiple classes, the probability of a class j is typically computed using the softmax function using the weights and hidden states of the final layer n in the network. We will use $\mathbf{o}$ to make it clear that it is the output of the network, but it can be thought of as activations of the final layer $\mathbf{h}^n$ as well.

$$o_j = \exp\big(\sum_i \mathbf{W}^n_{ij}\mathbf{h}^{n-1}_i + \mathbf{b}^n_j\big) / \Big(\sum_{j'} \exp\big(\sum_i \mathbf{W}^n_{ij'}\mathbf{h}^{n-1}_i + \mathbf{b}^n_{j'}\big)\Big) \tag{2.4}$$

At inference, for binary classification, a thresholding function is used to determine which of the two classes the input belongs to, the simplest example of such a function is to classify values $\geqslant 0.5$ as one class and < 0.5 as the other. For multi-class problems, the most likely class $\text{argmax}_j\, o_j$ is selected.

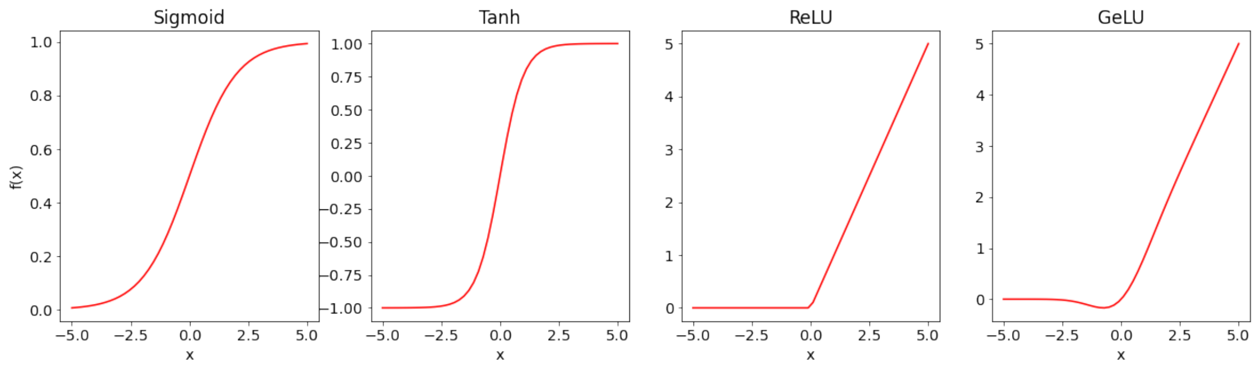


Fig. 2. Sigmoid, Tanh, ReLU & GeLU activation functions

There exist many popular choices for non-linear activation functions within the network $f(\cdot)$. The sigmoid activation that is typically used at the output of the network for binary classification tasks is sometimes used within the network as well. Other common choices are the hyperbolic tangent (tanh) function or rectified linear units (ReLU) (Nair & Hinton, 2010) and their variants. See Figure 2 for a graphical depiction of these activation functions. ReLU variants unlike sigmoid and tanh activations, are unbounded. In most current day feed-forward neural networks, ReLU variants are preferred for this reason, because they prevent gradients from saturating (becoming 0) in deep networks. The subsequent section (2.2.2) will provide more insights into this.

2.2.2. Training Neural Networks with Backpropagation

As discussed in Section 2.1.1, we want to train a neural network to minimize the empirical risk with respect to an i.i.d dataset of examples $((\mathbf{x}_1, y_1)\ldots(\mathbf{x}_N, y_N))$. When using the negative log-likelihood loss function, ERM corresponds to maximum-likelihood estimation of the parameters of the model. In this section, we will discuss the backpropagation algorithm (Rumelhart et al., 1986; Werbos, 1974) for computing the gradients of the loss function with respect to the model's parameters.

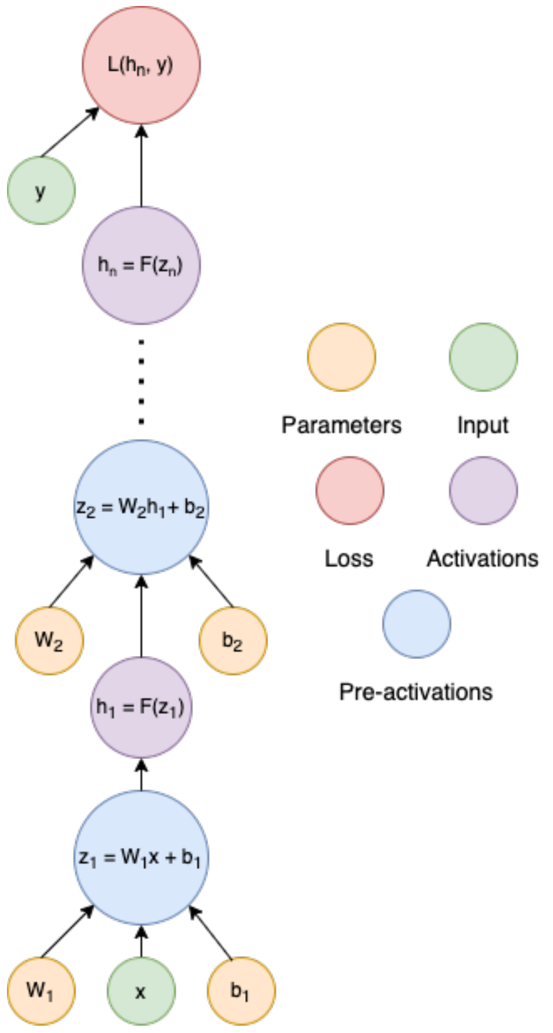


Fig. 3. Computation graph for an MLP

To simplify things, let $\mathcal{L}$ be a differentiable multi-variate loss function and θ be all of the model's parameters. In the neighborhood of a point (in parameter space), $\mathcal{L}$ decreases fastest if one takes a step in the opposite direction of the partial derivative $\frac{\partial \mathcal{L}(\mathbf{x},\mathbf{y};\theta)}{\partial \mathbf{W}^l}$, for small step sizes. Backpropagation uses the chain rule of differentiation to compute these derivatives. The chain rule is particularly useful when computing gradients involving a composition of many functions such as $f(g(\cdot))$, which is the case in neural networks. It consists of a "forward-pass" through the network where all intermediate activations h^l at each layer are stored in memory and a "backward-pass" that computes the gradient of the loss with respect to the parameters using the current model parameters and the stored activations.

Neural network operations are often best described in the language of computation graphs, which is a general tool to analyze mathematical expressions. The computation graph for an MLP is shown in Figure 3. For simplicity, we assume a binary classification problem where F is the sigmoid activation function for the output layer as well as within the network. Nodes in the graph are variables or expressions and arrows indicate how nodes feed into each other to create increasingly complex expressions. To compute the gradient of the loss with respect to any node in the graph, backpropagation recursively applies the chain rule, starting from the node that computes the loss. The expression for the gradient of the loss with respect to the model's parameters at some layer l can be written as

$$\frac{\partial \mathcal{L}(\mathbf{x}, \mathbf{y}; \theta)}{\partial \mathbf{W}^l} = \frac{\partial \mathcal{L}(\mathbf{x}, \mathbf{y}; \theta)}{\partial \mathbf{h}^n} \Big(\prod_{l < i \leqslant n} \frac{\partial \mathbf{h}^i}{\partial \mathbf{h}^{i-1}} \Big) \frac{\partial \mathbf{h}^l}{\partial \mathbf{W}^l} \tag{2.5}$$

Computing the expression above for each parameter naively can incur quadratic complexity in the number of parameters in the model. It is evident from the mathematical expression as well as by tracing the computation graph backwards that the same sub-expressions are computed multiple times when done naively. We can therefore memoize or cache the partial derivatives once computed to massively speed up backpropagation. This reduces complexity from quadratic to linear in the number of parameters and can be the difference between training a neural network with a million parameters in hours or days to several years. Computing gradients in this fashion is also called "reverse-mode" automatic differentiation. For a review of automatic differentiation techniques we refer the reader to Margossian (2019).

2.2.3. Stochastic Gradient Descent

The computed gradient is used to update the current values of the model's parameters. To begin with, a model's parameters are initialized at random and then iteratively update via stochastic gradient descent (SGD). The gradient may be estimated using the entire dataset or using only a subset of the data. The latter, which is most commonly used when training neural networks, is referred to as SGD. Stochasticity arises from the fact that the gradient depends on the selection of the subset of the training data (commonly referred to as a minibatch). The model takes many stochastic gradient steps (often several thousand), even revisiting the same training example many times, to converge to a point estimate of the parameters that achieves low error. The SGD update rule at step t for a parameter $\mathbf{W}^l$ can be written as

$$\mathbf{W}^l_t = \mathbf{W}^l_{t-1} - \alpha \frac{\partial \mathcal{L}(\mathbf{x}, \mathbf{y}; \theta_{\mathbf{t-1}})}{\partial \mathbf{W}^l_{t-1}} \tag{2.6}$$

Where α is a learning rate that controls how large a step to take in the direction of the gradient. α can be fixed over the course of training or change adaptively as a function of t or even be parameter-specific based on the history of gradients.

2.3. Sequence Modeling with Neural Networks

While multi-layer perceptrons are quite flexible when working with data from different domains, we'd often like to adapt neural network architectures based on the nature of the inputs they process. Typical domains of interest like language, images, videos or speech are highly structured and the neural connectivity patterns in architectures are designed to account for this. For example, pixels in an image have strong local structure - spatially distant pixels are only weakly related while nearby ones have stronger associations. An

MLP consists of many "fully-connected" layers where each output neuron is a function of *all* input neurons. An output neuron in a convolutional neural network (See section 2.3.2) in contrast, is computed based only on a subset of spatial positions (Fig 5) and the associated weights are shared over the entire image. This significantly reduces the number of parameters in the model as well as time and memory requirements and provides a strong *inductive bias* for processing images.

Sequential data like speech, time series and language also have strong local structure. For example grammar imposes strong local constraints on word choice and order in language. Convolutional networks are capable of exploiting such temporal structure as well, although recurrent models that process the input sequentially and share weights across time are often preferred.

2.3.1. Recurrent Neural Networks

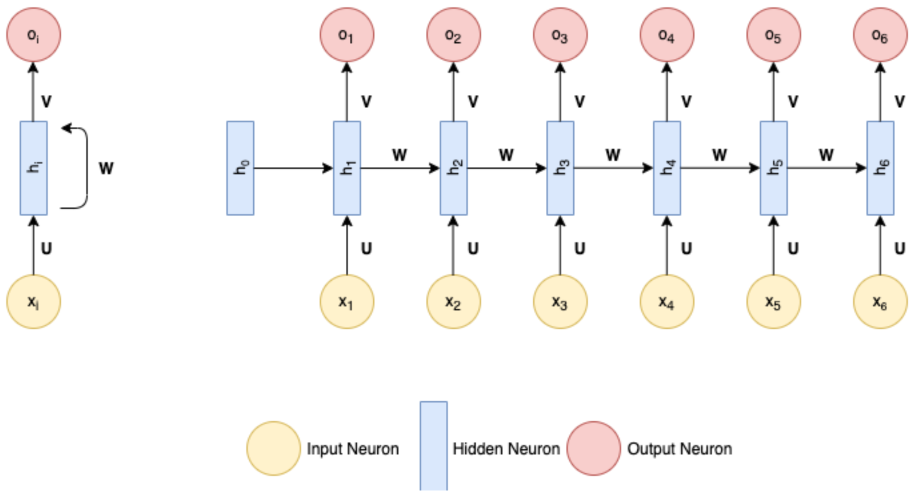


Fig. 4. A compact diagram of an RNN (left) and unrolled in time for 6 time steps (right). It consists of input weights U, recurrent weights W and output weights V all shared across time.

Recurrent neural networks (Rumelhart et al., 1986) take a sequence of inputs $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and transforms it into a sequence of vector representations $\mathbf{h} = (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n)$. We use subscripts to denote the position of an element within the sequence. x_i can be discrete, continuous and vector or scalar valued. They do so using the following general recurrence function:

$$\mathbf{h}_t = F(\mathbf{h}_{t-1}, \mathbf{x}_t, \theta) \tag{2.7}$$

In the vanilla case,

$$\mathbf{h}_t = \tanh(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t + \mathbf{b}) \tag{2.8}$$

The hidden state at time t is a non-linear function of the previous hidden state h_{t-1}, and the current input $\mathbf{x}_t$ and parameterized by a recurrent weight matrix $\mathbf{W}$ and input weight matrix $\mathbf{U}$ and a bias term b that are shared across time. If the inputs $\mathbf{x}_i \in \mathbb{R}^{d_{in}}$ are continuous vector valued inputs, then $\mathbf{U} \in \mathbb{R}^{d_{in} \times d_{hid}}$ and $\mathbf{W} \in \mathbb{R}^{d_{hid} \times d_{hid}}$ ($\mathbf{W}$ doesn't necessarily need to be a square matrix, but for the sake of notational convenience, we will assume it is). Depending on the task, there may be an output weight matrix $\mathbf{V} \in \mathbb{R}^{d_{hid} \times d_{out}}$ that produces outputs $\mathbf{o_t} = \mathbf{V}\mathbf{h_t}$.

When RNNs are used for language processing, inputs to the neural network are discrete elements, typically words, sub-words or characters. In such cases, the input matrix $\mathbf{U} \in \mathbb{R}^{|V| \times d_{hid}}$ acts as a "lookup table" that maps each element from the vocabulary V to a unique vector of size d_{hid}. RNNs are stateful models. The hidden *state* $\mathbf{h}_t$ contains a summary of the sequence $\mathbf{x}_1 \dots \mathbf{x}_t$ and computation for positions $\mathbf{x}_{>t}$ are independent of $\mathbf{x}_{\leq t}$ given $\mathbf{h}_t$. In practice, the state representation has a recency bias wherein it encodes more information about recent inputs than ones in the distant past. Recurrent networks can scale along both depth and time. To scale along depth, one can use the outputs of an RNN as the inputs to another RNN.

Bidirectional Models - Given a sequence of inputs $\mathbf{x}_1 \dots \mathbf{x}_n$, the hidden state $\mathbf{h}_t$ encodes only the prefix $\mathbf{x}_1 \dots \mathbf{x}_t$ ($\forall t < n$). In many practical settings, we would like the state representation at time t to be informed by both the past and the future. A common way of achieving this is to have a "bidirectional" model that consists of two *independent* RNNs, one that runs forward from $\mathbf{x}_1 \dots \mathbf{x}_n$ and another that runs backwards from $\mathbf{x}_n \dots \mathbf{x}_1$ (Schuster & Paliwal, 1997). For some time step t, the forward RNN encodes the prefix $\overrightarrow{\mathbf{h}}_t$, and the backward RNN encodes the suffix $\overleftarrow{\mathbf{h}}_t$ of length $n-t$. The final bidirectional representation at time t is often the concatenation of the forward and backward representations $\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t]$.

Backpropagation Through Time - Recurrent networks can be trained using the same principles as feedforward networks like MLPs. They can be unrolled in time (Figure 4 (right)) into a deep feedforward model with shared weights and the same principles used to compute gradients in Section 2.2.2 can be used while accounting for shared weights. The algorithm to do so is commonly referred to as the Backpropagation Through Time (BPTT) (Werbos, 1990). The gradient of the loss with respect to the output weights are fairly straightforward to estimate.

$$\frac{\partial \mathcal{L}(\mathbf{x}, \mathbf{y}; \theta)}{\partial \mathbf{V}} = \sum_{1 \leq t \leq n} \frac{\partial \mathcal{L}(\mathbf{x}_t, \mathbf{y}_t; \theta)}{\partial \mathbf{o}_t} \frac{\partial \mathbf{o}_t}{\partial \mathbf{V}} \tag{2.9}$$

The gradients with respect to the recurrent weights $\mathbf{W}$ are trickier to compute because of the recurrence relation between $\mathbf{h}_t$ and $\mathbf{h}_{t+1}$.

$$\frac{\partial \mathcal{L}(\mathbf{x}, \mathbf{y}; \theta)}{\partial \mathbf{W}} = \sum_{1 \leq t \leq n} \frac{\partial \mathcal{L}(\mathbf{x}_t, \mathbf{y}_t; \theta)}{\partial \mathbf{o}_t} \frac{\partial \mathbf{o}_t}{\partial \mathbf{h}_t} \sum_{1 \leq i \leq t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_i} \frac{\partial^+ \mathbf{h}_i}{\partial \mathbf{W}} \tag{2.10}$$

We borrow notation from Pascanu et al. (2013) in defining $\frac{\partial^+ \mathbf{h}_i}{\partial \mathbf{W}}$ as the immediate derivative of the hidden state with respect to the recurrent weights such that it is 0 for all time steps $< i$. Estimating the Jacobian $\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_i}$ itself involves computing a product of partial derivatives from i to t, similar to the product term in the MLP backpropagation equation in Section 2.2.2. We can store intermediate gradients similar to regular backpropagation to avoid computing the inner sum on every time step. This reduces the time and space complexity of BPTT to linear in the number of time steps. Often, when processing very long sequences such as entire documents or books, we may not want to backpropagate errors over very long time horizons for several reasons including the difficulty in learning long range dependencies (Bengio et al., 1994) and for computational reasons (memory requirements scale linearly in the length of the sequence). In such cases, we can use a variant of BPTT called Truncated BPTT (TBPTT) that sets gradients for long-range associations to 0. As an example, one way of implementing this would be to chunk a large sequence of n inputs into k chunks and backpropagate only within each chunk of size n/k. This still incurs $\mathcal{O}(n)$ time complexity (although it can now be parallelized by a factor k) but only $\mathcal{O}(n/k)$ memory since we don't have to store activations for associations longer than this.

Vanishing Gradients & Gated Recurrent Models - Vanishing gradients in recurrent networks (Hochreiter, 1991; Bengio et al., 1994; Pascanu et al., 2013) is the phenomenon by which it becomes hard to learn long-range associations due to the gradient of the loss at some time t with respect to the representations at some time $t-k$ becomes extremely small. We refer readers to Bengio et al. (1994); Pascanu et al. (2013) for a formal analysis of the reasons why this happens. A common remedy to this is to alter the RNN cell by introducing gates that control information flow across time, allowing for potential "shortcut" learning paths across all time steps. Long Short-term Memory (LSTM) and Gated Recurrent Units (GRU) are such gated recurrent networks that partly alleviate vanishing gradients in practice.

LSTM & GRU - In most practical use cases, the vanilla RNN variant that we have discussed thus far is seldom used and its gated variants such as LSTMs (Hochreiter & Schmidhuber, 1997a) and GRUs (Cho et al., 2014) are used when handling sequential data. For example, these have been immensely popular in NLP (Sutskever et al., 2014; Bahdanau et al., 2014), computer vision (**?**) and speech recognition (Chan et al., 2016). LSTMs differ from standard RNNs in that they seek to alleviate vanishing gradients by *learning* to carry information over several time steps or ignore certain certain parts of the input through an internal memory or cell state and gates that control their behavior. LSTMs have three gates called the input, output and forget gates ($\mathbf{i}_t, \mathbf{o}_t, \mathbf{f}_t$ respectively). The gates are computed similar to how the hidden states in the vanilla RNN were computed

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{h}_{t-1} + \mathbf{U}_f \mathbf{x}_t + \mathbf{b}_f) \tag{2.11}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{h}_{t-1} + \mathbf{U}_o \mathbf{x}_t + \mathbf{b}_o) \tag{2.12}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{h}_{t-1} + \mathbf{U}_i \mathbf{x}_t + \mathbf{b}_i) \tag{2.13}$$

The input and recurrent matrices are parameterized separately per gate and have the same dimensions as in the standard RNN $\mathbf{U}_{i,f,o} \in \mathbb{R}^{d_{in}} \times d_{hid}$ and $\mathbf{W}_{i,f,o} \in \mathbb{R}^{d_{hid}} \times d_{hid}$. Another transformation with similarly parameterized matrices $\mathbf{W}_c$ and $\mathbf{U}_c$, with a tanh activation is used to compute the candidate cell state, which then used to compute the the final cell state based on the input and forget gates. Finally, the hidden state is a function of the cell state and output gates.

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c \mathbf{h}_{t-1} + \mathbf{U}_c \mathbf{x}_t + \mathbf{b}_c) \tag{2.14}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{2.15}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{2.16}$$

The cell state in conjunction with the forget gate can act like identity connections that propagate information without changing them across several steps, which can help with vanishing gradients. The input gate controls what parts of the input to keep and discard.

GRUs have one less gate when compared to LSTMs - they have a reset gate ($\mathbf{r}_t$) which can force the model to completely ignore the previous hidden state and only use the current input and an update gate ($\mathbf{z}_t$) allows long range information flow by enabling a possibly identity path for hidden states:

$$\mathbf{g}_t = \sigma(\mathbf{W}_g \mathbf{h}_{t-1} + \mathbf{U}_g \mathbf{x}_t + \mathbf{b}_g) \tag{2.17}$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{h}_{t-1} + \mathbf{U}_z \mathbf{x}_t + \mathbf{b}_z) \tag{2.18}$$

$$\tilde{\mathbf{h}}_{\mathbf{t}} = \tanh(\mathbf{W}_c(\mathbf{z}_t \odot \mathbf{h}_{t-1}) + +\mathbf{U}_c \mathbf{x}_t + \mathbf{b}_c) \tag{2.19}$$

$$\mathbf{h}_t = \mathbf{g}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{g}_t) \odot \tilde{\mathbf{h}}_{\mathbf{t}} \tag{2.20}$$

Aside from vanishing gradients, these models may also have exploding gradients where the gradients may grow extremely large. Common ways to deal with this include gradient clipping where the magnitude of the gradients with respect to every parameter is clipped to a maximum value (Pascanu et al., 2012), weight initialization strategies (Le et al., 2015) or normalization of activations within the model (Ba et al., 2016).

2.3.2. Convolutional Neural Networks

As described at the beginning of Section 2.3, convolutional networks are useful for processing data that have strong spatial, temporal or spatio-temporal structure. They share weights across spatial or temporal positions (LeCun et al., 1990). The weight structure, unlike in MLPs fits the structure of the input data, for example weights for images that have height, width and possibly many channels are organized into a 4D tensor $\mathbf{W} \in \mathbb{R}^{k \times c \times \Delta x \times \Delta y}$ where k is the number of convolution kernels, c is the number of input channels (ex: 3 channels for R,G,B images) and $\Delta x, \Delta y$ are the width and height of the kernel. The convolutional operation for one of the k kernels (illustrated in Figure 5) involves sliding it (dark blue square) across an image $\mathbf{h} \in \mathbb{R}^{c \times x \times y}$ (light blue square of size $x \times y$) and computing a weighted sum at each position to yield an output feature map (green). The feature maps for different kernels are stacked to form k output channels. For simplicity, the figure shows the kernel moving by one position to the right or below, but it can move by larger amounts (strides). The output of convolution layers, with appropriate padding of the input, can be made to have the same shape as the input. We refer readers to Dumoulin & Visin (2016) for a detailed guide on convolutional arithmetic which comprehensively discusses how the shapes of the input, kernel, strides, dilations and other factors affect the shape of the output feature maps.

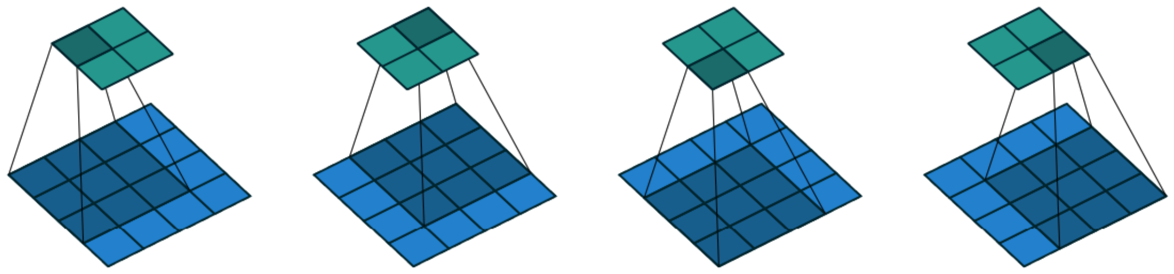


Fig. 5. A 2D convolution operation on a 4×4 image (light blue) using a 3×3 kernel (dark blue) and the resulting feature map in green. Figure from Dumoulin & Visin (2016)

Convolutional networks aren't as popular for language processing as recurrent nets, but have nonetheless been explored for a variety of tasks including sequence labeling (Part-of-speech (POS) tagging, Named Entity Recognition (NER)) (Collobert et al., 2011), text classification (Zhang et al., 2015) and machine translation (Gehring et al., 2017).

2.3.3. Content Based Attention Mechanisms

Recurrent models have a couple of shortcomings in that learning long-range associations even with gated variants like LSTMs or GRUs is hard and computation is sequential in nature making them hard to scale on modern parallel hardware platforms. Recent work has explored alternatives that make gradient propagation over long time horizons easier and avoid a sequential computation bottleneck. Attention models (Bahdanau et al., 2014; Xu et al., 2015; Vaswani et al., 2017) possess such properties - they compute association scores between an input (queries), which is either a single vector or a sequence of vectors and context (keys), another sequence of vectors.

Content-based attention mechanisms were presented in Bahdanau et al. (2014) for neural machine translation. The idea was to compute a score for every word in the input language given the hidden state of an RNN for a word in the output language. The scores would serve as an indicator for the word that a model needed to focus or attend to to generate the next word in the output language. The method was presented in the context of a sequence-to-sequence model, which we will discuss later in Section 2.3.5, but the specifics of the attention layer can be understood in isolation. While attention may be intuitive from a human cognitive perspective such as with translation or generating captions for images (Xu et al., 2015), they may be used in settings where there isn't a clear notion of alignment between two sequences or even when there is only a single sequence (where the operation is commonly referred to as "self-attention"). Like RNNs and Convolutional networks, they can process a variety of sequential data. It is not necessarily interpretable and one must be cautious when using it to explain model decisions (Jain & Wallace, 2019).

Consider the example of an attention model that processes a sequence of words $\mathbf{x}^t = (\mathbf{x}^t_1 \dots \mathbf{x}^t_m)$ conditioned on context $\mathbf{x}^s = (\mathbf{x}^s_1 \dots \mathbf{x}^s_n)$. $\mathbf{x}^s$ may be a sentence comprising words in English while $\mathbf{x}^t$ is it's corresponding translation in French. Let $\mathbf{s}$ and $\mathbf{t}$ be encodings of the English and French sentences. The attention layer computes a context-aware encoding for $\tilde{\mathbf{t}}_j$ for every word in the French sentence as follows:

$$e_{ij} = \text{score}(\mathbf{s}_i, \mathbf{t}_j) \tag{2.21}$$

$$\text{score}(\mathbf{s}_i, \mathbf{t}_j) = \mathbf{t}_j^\top \mathbf{W} \mathbf{s}_i \tag{2.22}$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum\limits_{k=1}^{n} \exp(e_{kj})} \tag{2.23}$$

$$\tilde{\mathbf{t}}_j = \sum_{k=1}^{n} \alpha_{kj} \mathbf{s}_k \tag{2.24}$$

Scores e_{ij} between queries and keys can be computed in parallel. The equations above use the "general" score function (Luong et al., 2015b), but one can use MLPs (Bahdanau et al., 2014) or other scoring functions as well. Scores are normalized across input positions to produce attention weights α_{ij} and used to compute a weighted sum over $\mathbf{s}$ to generate a context-aware representation $\tilde{\mathbf{t}}$ of the French sentence. When encoding the French sentence with an RNN as in Bahdanau et al. (2014); Luong et al. (2015b), it is common to have the attention module within the recurrence function of the RNN, the context aware representation may be computed before or after the RNN recurrence.

The attention module introduces direct paths for gradient propagation between a target word's encoding and all context words making it possible to learn long-range associations.

2.3.4. Language Models

Language models estimate the probability of a sequence of words $P(\mathbf{x}) = P(\mathbf{x}_1 \dots \mathbf{x}_n)$. Most approaches factorize the joint probability over the sequence of words into a product of conditional probabilities of a word given all words that precede it. This is follows from the chain rule of probability

$$P(\mathbf{x}_1 \dots \mathbf{x}_n) = P(\mathbf{x}_n|\mathbf{x}_1 \dots \mathbf{x}_{n-1})P(\mathbf{x}_1 \dots \mathbf{x}_{n-1}) \tag{2.25}$$

$$P(\mathbf{x}_1 \dots \mathbf{x}_{n-1}) = P(\mathbf{x}_{n-1}|\mathbf{x}_1 \dots \mathbf{x}_{n-2})P(\mathbf{x}_1 \dots \mathbf{x}_{n-2}) \tag{2.26}$$

$$\dots\dots\dots \tag{2.27}$$

$$P(\mathbf{x}_1, \mathbf{x}_2) = P(\mathbf{x}_2|\mathbf{x}_1)P(\mathbf{x}_1) \tag{2.28}$$

In general, we can write the factorization as

$$P(\mathbf{x}_1 \dots \mathbf{x}_n) = P(\mathbf{x}_1)\prod_{i=2}^{n} P(\mathbf{x}_i|\mathbf{x}_{<i}) \tag{2.29}$$

The chain rule allows for any order of factorization, but for convenience we will factorize from left-to-right (from the first to last word or character in a sequence of text). The language modeling problem thus involves modeling these conditional probabilities as accurately as possible. N-gram language models (Heafield, 2011) estimate the conditional probabilities using *counts* given a large corpus of text. The equation above does not restrict the number of elements to condition on, however with count-based estimation, given a large number of preceding elements, counts get sparse extremely quickly. In practice therefore, we restrict context sizes. A 2-gram (bigram) language model for example will estimate the conditional probabilities of a word given *only* it's previous word:

$$P(\mathbf{x}_i|\mathbf{x}_{i-1}) = \frac{\text{count}(\mathbf{x}_i, \mathbf{x}_{i-1})}{\text{count}(\mathbf{x}_{i-1})} \tag{2.30}$$

Larger context sizes help model long range dependencies better but suffer from sparsity, for example, we may encounter an n-gram at test time that we've never seen before. In such cases, it is common to use some form of smoothing to estimate the probabilities (Kneser & Ney, 1995).

Count-based language models suffer from the curse of dimensionality (Bengio et al., 2003) - the space of possible valid English sentences (or any other language) is incredibly large and count-based models do not encode a notion of similarity among lexical items (every element is equally far from one another, regardless of their semantic or syntactic similarity), making generalization to new sequences difficult.

Bengio et al. (2003) in contrast, train a neural language model that learns continuous embeddings for words as a by-product of language modeling. The approach concatenates the embeddings of a sequence of k words $\mathbf{x}_{t-k}, \ldots, \mathbf{x}_{t-1}$, and uses an MLP, that outputs a probability distribution over the entire English vocabulary, to predict the next word in the sequence $\mathbf{x}_t$. This MLP slides across the sequence, making predictions at every word. It is trained via backpropagation to minimize negative log-likelihood $-\log P(\mathbf{x}_t|\mathbf{x}_{t-k}, \ldots \mathbf{x}_{t-1}; \theta)$ of the actual next word in the sequence given the preceding k words. The learned word embeddings encode a notion of similarity which would help encourage smooth generalization to unseen sequences. In subsequent work, Mikolov et al. (2010) replaced the MLP with an RNN allowing for memories beyond just the last k words.

Language models can also be used to generate plausible text, by sequentially sampling from the learned conditional probabilities. The same principles can also be used to model the probability of observing a certain sequence of pixels (**?**) or audio samples (Van Den Oord et al., 2016; Mehri et al., 2016).

Language modeling efforts have since initially focused on regularization (Zaremba et al., 2014; Merity et al., 2017), then on better recurrent or feed-forward architectures (Bradbury et al., 2016; Dauphin et al., 2017) and now to scaling to longer sequences and larger datasets with bigger models (Radford et al., 2018, 2019).

2.3.5. Sequence-to-Sequence Models

Many problems, especially in NLP, can be formulated as transforming one sequence of inputs into another sequence of inputs. For example, machine translation from English to French involves transforming a sequence of words in English to a sequence of words in French that conveys the same meaning. Sequence-to-sequence neural networks (Sutskever et al., 2014) formulate this by having two separate neural networks: an encoder, that encodes the input into a single or a sequence of hidden representations and a decoder neural network that conditions on the hidden representations of the encoder to produce the output sequence.

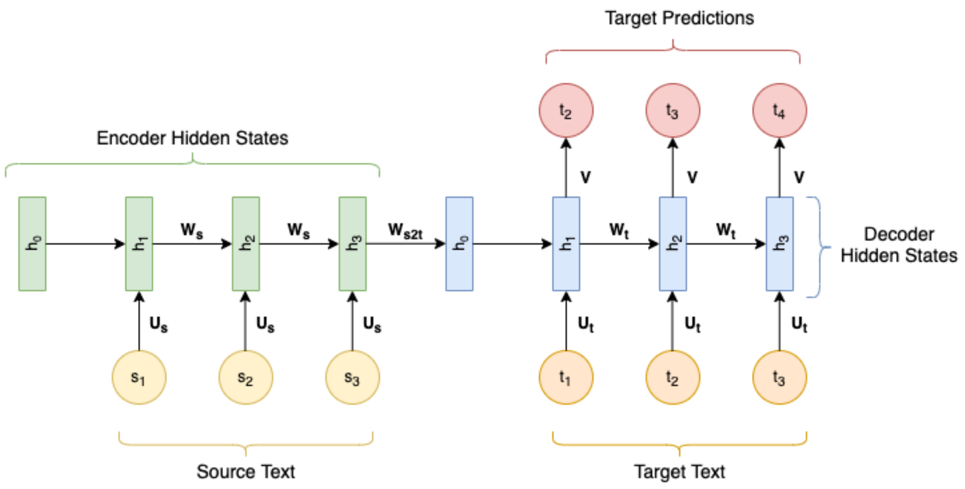


Fig. 6. A recurrent sequence-to-sequence model. An Encoder RNN encodes the sequence $(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3)$ into a sequence of hidden states (green). The last green hidden state $\mathbf{h}_3$ is transformed via a projection to produce the initial hidden state $\mathbf{h}_0$ (blue) for the decoder RNN. The decoder RNN is then trained as a language model given the initial hidden state.

The decoder models the conditional probability of the output sequence autoregressively given the input.

$$P(\mathbf{t}_1 \ldots \mathbf{t}_n|\mathbf{s};\theta) = \prod_{i=1}^{n} P(\mathbf{t}_i|\mathbf{t}_{<\mathbf{i}}, \mathbf{h}_s;\theta) \tag{2.31}$$

Figure 6 illustrates a sequence-to-sequence model that uses RNNs as the encoder and decoder. The RNN encoder (often bidirectional) produces a hidden representation for every element in the input sequence. The last hidden state is taken to be a "summary" of the entire input sequence and is used to condition the decoder RNN through a learnable transformation to produce the initial hidden state $\mathbf{h}_0$ for the decoder. Given the initial state, the decoder is trained as an RNN language model (similar to Section 2.3.4). In practice, the encoder and decoder often have separate parameters (which we indicated with the subscripts s and t in Figure 6). The model is trained "end-to-end", which means the encoder's parameters are updated using the gradient from the predicting the next element in the sequence in the decoder. The gradient path length for BPTT in such models can be quite large for long sequences and the encoder is often gradient starved.

Bahdanau et al. (2014) addressed this issue by using the attention mechanism discussed in Section 2.3.3 by having direction connections between decoder and encoder hidden states introducing shortcuts for easy gradient propagation. Sequence-to-sequence models have been used extensively in NLP and speech for tasks like Machine Translation (Wu et al., 2016), Text Summarization (Rush et al., 2015), and Speech Recogntion (Chan et al., 2016).

2.3.6. Transformers and Self-attention Models

Transformers (Vaswani et al., 2017) are neural networks that consist primarily of attention layers and position-wise MLPs. Figure 7 from their paper shows the internals of transformers used for sequence-to-sequence learning.

In section 2.3.3, we made a distinction between the context sequence **s** over which the attention scores are computed and the input sequence **t**. An important idea in Transformers is to use the attention mechanism to process a *single* sequence called "self-attention" where the model computes attention scores only over itself. The paper establishes a general language for describing attention that includes self and encoder-decoder (similar to Section 2.3.5) attention in the same formalism. Attention is a function of 3 tensors - queries $\mathbf{Q}$, keys $\mathbf{K}$ and values $\mathbf{V}$.

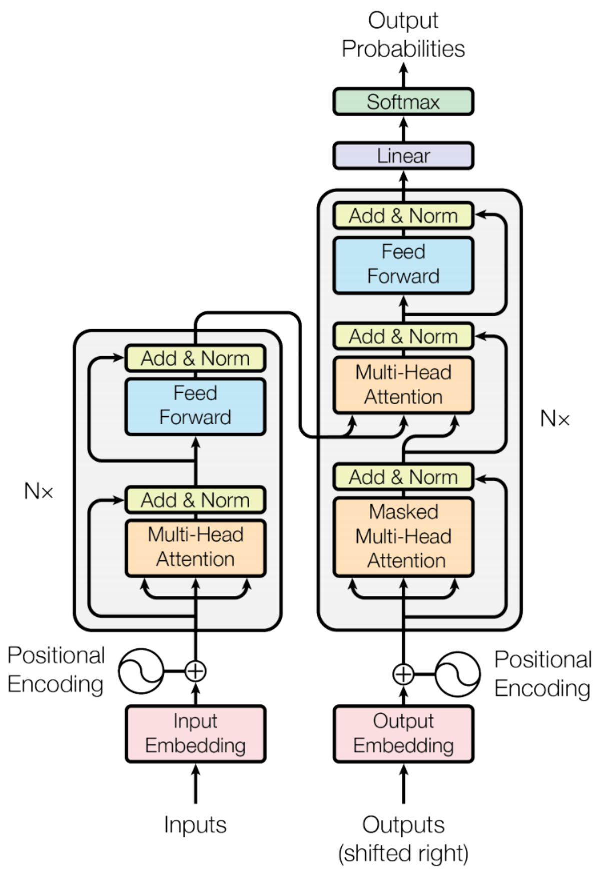


Fig. 7. A Transformer sequence-to-sequence model. It consists of an Encoder (left) which comprises N transformer blocks with self-attention and a Decoder (right) which has similar transformer block with self-attention as well as cross-attention over the encoder's hidden states. Figure from Vaswani et al. (2017)

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}})\mathbf{V} \tag{2.32}$$

Self-attention uses the same tensor for queries, keys and values while "cross-attention" has queries that come from the decoder hidden states and keys and values that come from the encoder hidden states. It is common to have separate projection weights $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v$ to produce the queries, keys and values before the attention operation. The self and cross-attention can be written as

$$\text{SelfAttention}(\mathbf{s}) = \text{softmax}(\frac{\mathbf{W}_q\mathbf{s}(\mathbf{W}_k\mathbf{s})^\top}{\sqrt{d}})\mathbf{W}_v\mathbf{s} \tag{2.33}$$

$$\text{CrossAttention}(\mathbf{s}, \mathbf{t}) = \text{softmax}(\frac{\mathbf{W}_q\mathbf{t}(\mathbf{W}_k\mathbf{s})^\top}{\sqrt{d}})\mathbf{W}_v\mathbf{s} \tag{2.34}$$

Vaswani et al. (2017) also present multi-headed attention which performs the attention operation k times, with different projections of the queries, keys and values, one for each "attention head". The attention vectors from each head are concatenated and projected back to the original hidden size.

$$\text{MultiHeadAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_k)\mathbf{W}_o \tag{2.35}$$

$$\text{head}_i = \text{Attention}(\mathbf{W}_q^i\mathbf{Q}, \mathbf{W}_k^i\mathbf{K}, \mathbf{W}_v^i\mathbf{V}) \tag{2.36}$$

The attention mechanism as presented here is agnostic to the positions of each vector in $\mathbf{s}$ and $\mathbf{t}$. To allow transformers to model things like word order within a sequence of text, positional encodings which are vectors associated only with positions within the sequence are added to the input word embeddings before being fed to a transformer. A Transformer consists of a stack of Transformer blocks with residual connections (He et al., 2016b). Residual blocks are organized differently when used as an Encoder or Decoder within a sequence-to-sequence model. An encoder block consists of self-multi-headed attention followed by residual addition and layer normalization (Ba et al., 2016), followed by an MLP that is shared across time and another residual addition and layer normalization. When used as a decoder, the block is almost identical except a cross-multi-headed attention layer is added after the self-attention layer.

Self-attention within the encoder is unrestricted - every position may attend to every other position but the autoregressive factorization restricts self-attention in the decoder from looking at future positions in the sequence.

Transformers have become the cornerstone of modern day NLP and are used extensively for natural language understanding (Devlin et al., 2018), language modeling (Radford et al., 2018, 2019), summarization (Subramanian et al., 2019; Zaheer et al., 2020) and a variety of other problems.

2.3.7. Representation Learning for NLP

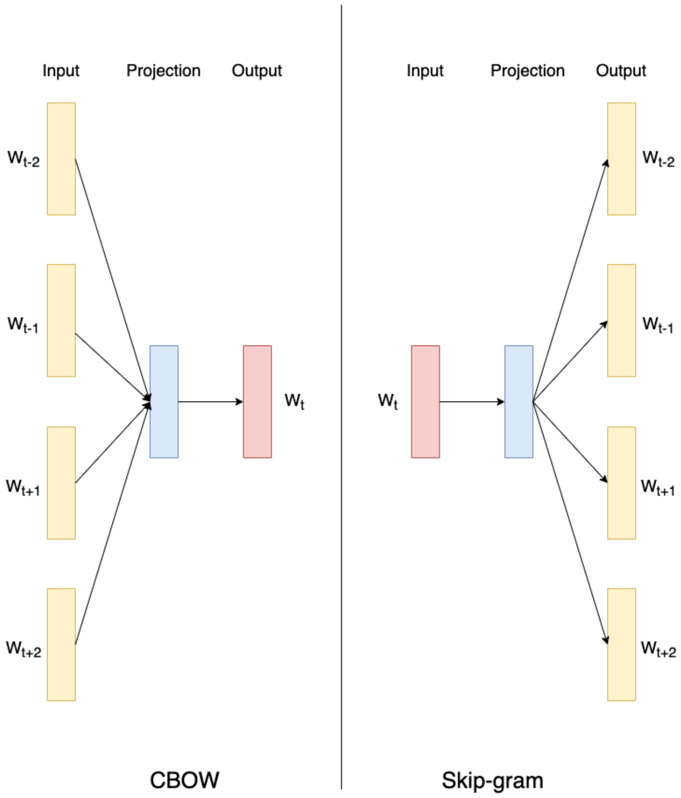


Fig. 8. Skip-gram and CBOW Word2vec models from (Mikolov et al., 2013) to learn word embeddings.

Being able to represent data appropriately is important for making decisions on that data in any domain. In language, the linguistic form we use to communicate is discrete - characters which compose to form words, which in turn compose to form sentences and so on. Representation learning on discrete data poses some challenges that are well discussed in Bengio et al. (2003) - small changes in discrete elements can induce significant changes on the decisions one might want to take on that data. Altering a few or even a single word in a sentence has the potential to significantly change its meaning but changing continuous pixel values slightly in an image don't significantly alter it.

Neural networks provide the promise of learning smooth internal representations which are particularly desirable for discrete data since smoothness does not exist in the input space. For NLP, we would like to build representations of characters, words, sentences, paragraphs and documents. Initial work from Bengio et al. (2003) demonstrated this by training an MLP to do language modeling, which yielded good word embeddings and LM performance. While there is still plenty of research and debate on characterizing and measuring desirable properties of data representations, in the majority of this thesis, we will measure representation quality by using the learned representations to perform other potentially related tasks.

Representation learning requires 1) designing the right training objective to induce good representations and 2) choosing the right model class. The choice of training objective may impact choice of data as well, for example supervised representation learning will typically require different data from unsupervised learning. Mikolov et al. (2013) presented “word2vec” a method to learn word embeddings from large amounts of unlabeled data. Figure 8 presents the two flavors of word2vec, the continuous bag-of-words model (CBOW) (left) and the skip-gram model (right). Both approaches take advantage of the distributional hypothesis of language which posits that a word’s local neighborhood is indicative of its syntactic and semantic properties. Similar words will be used in similar contexts. The skip-gram and CBOW models are cast as predicting all words in a local neighborhood around a word (skip-gram) or predicting the word given its neighborhood (CBOW). Embeddings that arise from training a model this way are considered “general-purpose”. For example, they may be used to initialize **U** in Section 2.3.1 when used for downstream tasks such as named entity recognition, text classification, machine translation etc. There exist “character aware” extensions of word2vec (Joulin et al., 2016a) where character prefix and suffix embeddings are concatenated with the word embedding and learned along with the training objective.

Computer vision at the time, in contrast, showed that large-scale *supervised* pretraining on image classification tasks (Deng et al., 2009) with convolutional neural networks (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014) could learn useful image representations. The right training objective for NLP to obtain good *sentence* representations however was unclear with both supervised and unsupervised/self-supervised approaches yielding results that weren’t as impressive as those in computer vision. We present a review of sentence representation learning work in Section 4.2. More recently however, with more data, better and larger models, contextualized representations of sentences - representations that do not represent an entire variable length sentence with a single vector, but rather as a sequence of vectors (one per word) (Peters et al., 2018; Devlin et al., 2018) with self-supervised objectives like language modeling and denoising have made impressive progress in representation learning.

2.4. Adversarial Methods

2.4.1. Generative Adversarial Networks

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) is an approach to generative modeling that involves training two separate neural networks, a generator network

G and a discriminator network D, where the generator transforms samples from a simple distribution such as a standard normal into those that look like ones from a complicated data distribution such as natural images and a discriminator is trained to identify if an image was produced by the generator or comes from the true data distribution. The generator is trained to maximize the likelihood of the discriminator classifying its outputs as coming from the true data. The formulation is fairly different from the standard maximum-likelihood formulation in Sections **??** and 2.3.4 since it is a two-player min-max game. Let $\mathbf{x} \sim P_D$ be data drawn from a data distribution (ex: natural images or handwritten digits) and $z \sim P(z)$ be a sample from a simple prior distribution such as a standard normal $\mathcal{N}(0, \mathbf{I})$. The overall training objective for both D and G can be written as

$$\min_G \max_D V(D,G) = \mathop{\mathbb{E}}_{\mathbf{x} \sim P_D}[\log D(E(\mathbf{x}))] + \mathop{\mathbb{E}}_{\mathbf{z} \sim P(z)}[\log(1 - D(G(\mathbf{z})))] \tag{2.37}$$

GANs, unlike likelihood-based generative models like autoregressive models (Section 2.3.4) cannot estimate the likelihood of a data point but can only produce samples that look like ones from the data distribution. We refer readers to (Arjovsky & Bottou, 2017) for details regarding the nature of the divergence between the generator and true data distribution being minimized with the standard GAN objective. Nowozin et al. (2016) present a way to train GANs to minimize a variety of f-divergences.

In our thesis, we make use of Wasserstein GANs, (Arjovsky et al., 2017) with stabilization methods (Gulrajani et al., 2017) that argue that while KL divergence is a good divergence for low-dimensional data, it is often too strong in the high-dimensional settings we are often interested in modeling.

GANs have been used to generate high fidelity images, but they are fundamentally built for modeling continuous data distributions such as images where gradients from the discriminator can propagate through the generator's output to compute gradients with respect to the generator's parameters. When modeling discrete data, producing discrete outputs at the generator will require sampling or other operations that are incompatible with gradient-based learning and one may have to resort to policy-gradient methods to circumvent these issues, which have had limited success in language application (Caccia et al., 2018). In this thesis (Chapter 6), we circumvent discreteness entirely by training GANs in a continuous latent space of sentences.

2.4.2. Domain Adversarial Training

Domain Adversarial Training of neural networks (Ganin et al., 2016) uses an idea similar to that in Goodfellow et al. (2014) to learn neural representations of data that are invariant to certain attributes. Ganin et al. (2016) motivate this is in the context of domain adaptation wherein a model is trained on a certain data distribution but has to generalize to a similar but slightly different distribution at test time. Representations that encode domain-specific features rather than task-specific ones (often due to spurious correlations between domain and task), may find it harder to generalize.

Ganin et al. (2016) present a way to have the representations of a neural network be agnostic to the data domain from which it comes. This is achieved by training a discriminator neural network to classify the input's domain from its latent representation. The network is trained to *minimize* the probability of correct domain classification by the discriminator and they hypothesize that the encoder network will achieve this by not encoding domain-specific information in its representation.

In natural language, there are several cases where we may want to learn representations that are agnostic to certain attributes. When classifying the sentiment of online product reviews, we want representations that are agnostic to the product category to generalize to new categories or those that have minimal training data. With neural systems finding their way into production environments that interact with people in the real world, systems that encode protected information such as gender, sexual orientation, ethnicity etc are particularly worrying when such systems can make life altering decisions such as hiring decisions or approving credit card or loan applications. Domain adversarial training in such cases may not weed out the problem entirely, as we show in Chapter 8 where they still encode information attributes that they were trained to be invariant to.

Chapter 3

Prologue to the first article

3.1. Article Details

Learning General Purpose Distributed Sentence Representations via Large Scale Multi-task Learning. Sandeep Subramanian, Adam Trischler, Yoshua Bengio, Christopher Pal. International Conference on Learning Representations 2018.

Personal Contribution The project began with discussions between Christopher Pal and Sandeep Subramanian about the state of representation learning in NLP and trying to design a "general purpose" sentence representation space. I was involved in writing all of the code, running the experiments, coming up with the idea of using multi-task learning, choosing the different multi-task objectives and writing parts of the paper. Adam Trischler, Yoshua Bengio and Christopher Pal advised on the project, were involved in discussing results, suggesting experiments to run and writing parts of the paper.

3.2. Context

The goal of this work was to build a general purpose sentence representation space on which one could have small neural modules operate to transform sentences directly in the latent space. This paper focuses on the first sub-problem of building good sentence representations. The subsequent chapter (Chapter 6) builds on this work to train generative models of text using this learned latent space.

3.3. Contributions

The paper presents a multi-task sequence-to-sequence learning approach with a diverse selection of training objectives to learn sentence embeddings. The method achieved state-of-the-art performance on the SentEval at time of publication, a transfer learning benchmark for sentence representations on NLU tasks.

3.4. Research Impact

The work remains one of the best performing fixed-length sentence embedding methods on SentEval and has become part of the official repository as an example of using the benchmark. It was also used as one the baselines when designing the popular GLUE (Wang et al., 2018) NLU benchmark. In the last few years however, representation learning has largely focused on contextualized word representations that move away from the fixed-length paradigm. These perform far better than sentence embedding methods on most NLU tasks.

Chapter 4

Multi-task Learning of Sentence Representations

4.1. Introduction

Transfer learning has driven a number of recent successes in computer vision and NLP. Computer vision tasks like image captioning (Xu et al., 2015) and visual question answering typically use CNNs pretrained on ImageNet (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014) to extract representations of the image, while several natural language tasks such as reading comprehension and sequence labeling (Lample et al., 2016) have benefited from pretrained word embeddings (Mikolov et al., 2013; Pennington et al., 2014) that are either fine-tuned for a specific task or held fixed.

Many neural NLP systems are initialized with pretrained word embeddings but learn their representations of *words in context* from scratch, in a task-specific manner from supervised learning signals. However, learning these representations reliably from scratch is not always feasible, especially in low-resource settings, where we believe that using general purpose sentence representations will be beneficial.

Some recent work has addressed this by learning general-purpose sentence representations (Kiros et al., 2015; Wieting et al., 2015; Hill et al., 2016; Conneau et al., 2017a; McCann et al., 2017; Jernite et al., 2017; Nie et al., 2017; Pagliardini et al., 2017). However, there exists no clear consensus yet on what training objective or methodology is best suited to this goal.

Understanding the inductive biases of distinct neural models is important for guiding progress in representation learning. Shi et al. (2016) and Belinkov et al. (2017) demonstrate that neural machine translation (NMT) systems appear to capture morphology and some syntactic properties. Shi et al. (2016) also present evidence that sequence-to-sequence parsers (Vinyals et al., 2015) more strongly encode source language syntax. Similarly, Adi et al. (2016) probe representations extracted by sequence autoencoders, word embedding averages, and skip-thought vectors with a multi-layer perceptron (MLP) classifier to study whether sentence characteristics such as length, word content and word order are encoded.

To generalize across a diverse set of tasks, it is important to build representations that encode several aspects of a sentence. Neural approaches to tasks such as skip-thoughts, machine translation, natural language inference, and constituency parsing likely have different inductive biases. Our work exploits this in the context of a simple one-to-many multi-task learning (MTL) framework, wherein a single recurrent sentence encoder is shared across multiple tasks. We hypothesize that sentence representations learned by training on a reasonably large number of weakly related tasks will generalize better to novel tasks unseen during training, since this process encodes the inductive biases of multiple models. This hypothesis is based on the theoretical work of Baxter et al. (2000). While our work aims at learning *fixed-length* distributed sentence representations, it is not always practical to assume that the entire "meaning" of a sentence can be encoded into a fixed-length vector. We merely hope to capture some of its characteristics that could be of use in a variety of tasks.

The primary contribution of our work is to combine the benefits of diverse sentence-representation learning objectives into a single multi-task framework. To the best of our knowledge, this is the first large-scale reusable sentence representation model obtained by combining a set of training objectives with the level of diversity explored here, i.e. multi-lingual NMT, natural language inference, constituency parsing and skip-thought vectors. We demonstrate through extensive experimentation that representations learned in this way lead to improved performance across a diverse set of novel tasks not used in the learning of our representations. Such representations facilitate low-resource learning as exhibited by significant improvements to model performance for new tasks in the low labelled data regime - achieving comparable performance to a few models trained from scratch using only 6% of the available training set on the Quora duplicate question dataset[1].

4.2. Related Work

The problem of learning distributed representations of phrases and sentences dates back over a decade. For example, Mitchell & Lapata (2008) present an additive and multiplicative linear composition function of the distributed representations of individual words. Clark & Pulman (2007) combine symbolic and distributed representations of words using tensor products. Advances in learning better distributed representations of words (Mikolov et al., 2013; Pennington et al., 2014) combined with deep learning have made it possible to learn complex non-linear composition functions of an arbitrary number of word embeddings using convolutional or recurrent neural networks (RNNs). A network's representation of the last element in a sequence, which is a non-linear composition of all inputs, is typically assumed to contain a squashed "summary" of the sentence. Most work in supervised learning for NLP builds task-specific representations of sentences rather than general-purpose ones.

[1]Code available at `https://github.com/Maluuba/gensen`

Notably, skip-thought vectors (Kiros et al., 2015), an extension of the skip-gram model for word embeddings (Mikolov et al., 2013) to sentences, learn re-usable sentence representations from weakly labeled data. Unfortunately, these models take weeks or often months to train. Hill et al. (2016) address this by considering faster alternatives such as sequential denoising autoencoders and shallow log-linear models. Arora et al. (2016), however, demonstrate that simple word embedding averages are comparable to more complicated models like skip-thoughts. More recently, Conneau et al. (2017a) show that a completely supervised approach to learning sentence representations from natural language inference data outperforms all previous approaches on transfer learning benchmarks. Here we use the terms "transfer learning performance" on "transfer tasks" to mean the performance of sentence representations evaluated on tasks unseen during training. McCann et al. (2017) demonstrated that representations learned by state-of-the-art large-scale NMT systems also generalize well to other tasks. However, their use of an attention mechanism prevents the learning of a single fixed-length vector representation of a sentence. As a result, they present a bi-attentive classification network that composes information present in all of the model's hidden states to achieve improvements over a corresponding model trained from scratch. Jernite et al. (2017) and Nie et al. (2017) demonstrate that discourse-based objectives can also be leveraged to learn good sentence representations.

Our work is most similar to that of Luong et al. (2015a), who train a many-to-many sequence-to-sequence model on a diverse set of weakly related tasks that includes machine translation, constituency parsing, image captioning, sequence autoencoding, and intra-sentence skip-thoughts. There are two key differences between that work and our own. First, like McCann et al. (2017), their use of an attention mechanism prevents learning a fixed-length vector representation for a sentence. Second, their work aims for improvements on the same tasks on which the model is trained, as opposed to learning re-usable sentence representations that transfer elsewhere.

We further present a fine-grained analysis of how different tasks contribute to the encoding of different information signals in our representations following work by Shi et al. (2016) and Adi et al. (2016).

Hashimoto et al. (2016) similarly present a multi-task framework for textual entailment with task supervision at different levels of learning. "Universal" multi-task models have also been successfully explored in the context of computer vision problems (Kokkinos, 2016; Eigen & Fergus, 2015).

4.3. Sequence-to-Sequence Learning

Five out of the six tasks that we consider for multi-task learning are formulated as sequence-to-sequence problems (Cho et al., 2014; Sutskever et al., 2014). Briefly, sequence-to-sequence models are a specific case of encoder-decoder models where the inputs and outputs are sequential. They directly model the conditional distribution of outputs given inputs $P(\mathbf{y}|\mathbf{x})$. The input $\mathbf{x}$ and output $\mathbf{y}$ are sequences $x_1, x_2, \ldots, x_m$ and $y_1, y_2 \ldots, y_n$. The encoder produces a fixed length vector representation $\mathbf{h_x}$ of the input, which the decoder then conditions on to generate an output. The decoder is auto-regressive and breaks down the joint probability of outputs into a product of conditional probabilities via the chain rule:

$$P(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{n} P(y_i|y_{<i}, \mathbf{h_x}) \tag{4.1}$$

Cho et al. (2014) and Sutskever et al. (2014) use encoders and decoders parameterized as RNN variants such as Long Short-term Memory (LSTMs) (Hochreiter & Schmidhuber, 1997b) or Gated Recurrent Units (GRUs) (Chung et al., 2014). The hidden representation $\mathbf{h_x}$ is typically the last hidden state of the encoder RNN.

Bahdanau et al. (2014) alleviate the gradient bottleneck between the encoder and the decoder by introducing an attention mechanism that allows the decoder to condition on every hidden state of the encoder RNN instead of only the last one. In this work, as in Kiros et al. (2015); Hill et al. (2016), we do not employ an attention mechanism. This enables us to obtain a single, fixed-length, distributed sentence representation. To diminish the effects of vanishing gradient, we condition every decoding step on the encoder hidden representation $\mathbf{h_x}$. We use a GRU for the encoder and decoder in the interest of computational speed. The encoder is a bidirectional GRU while the decoder is a unidirectional conditional GRU whose parameterization is as follows:

$$\mathbf{r_t} = \sigma(\mathbf{W_r}x_t + \mathbf{U_r}\mathbf{h_{t-1}} + \mathbf{C_r}\mathbf{h_x}) \tag{4.2}$$

$$\mathbf{z_t} = \sigma(\mathbf{W_z}x_t + \mathbf{U_z}\mathbf{h_{t-1}} + \mathbf{C_z}\mathbf{h_x}) \tag{4.3}$$

$$\tilde{\mathbf{h_t}} = \tanh(\mathbf{W_d}x_t + \mathbf{U_d}(\mathbf{r_t} \odot \mathbf{h_{t-1}}) + \mathbf{C_d}\mathbf{h_x}) \tag{4.4}$$

$$\mathbf{h_{t+1}} = (1 - \mathbf{z_t}) \odot \mathbf{h_{t-1}} + \mathbf{z_t} \odot \tilde{\mathbf{h_t}} \tag{4.5}$$

The encoder representation $\mathbf{h_x}$ is provided as conditioning information to the reset gate, update gate and hidden state computation in the GRU via the parameters $\mathbf{C_r}$, $\mathbf{C_z}$ and $\mathbf{C_d}$ to avoid attenuation of information from the encoder.

4.3.1. Multi-task Sequence-to-sequence Learning

Dong et al. (2015) present a simple one-to-many multi-task sequence-to-sequence learning model for NMT that uses a shared encoder for English and task-specific decoders for multiple target languages. Luong et al. (2015a) extend this by also considering many-to-one (many encoders, one decoder) and many-to-many architectures. In this work, we consider a one-to-many model since it lends itself naturally to the idea of combining inductive biases from different training objectives. The same bidirectional GRU encodes the input sentences from different tasks into a compressed summary $\mathbf{h_x}$ which is then used to condition a task-specific GRU to produce the output sentence.

4.4. Training Objectives & Evaluation

Our motivation for multi-task training stems from theoretical insights presented in Baxter et al. (2000). We refer readers to that work for a detailed discussion of results, but the conclusions most relevant to this discussion are (i) that learning multiple related tasks jointly results in good generalization as measured by the number of training examples required per task; and (ii) that inductive biases learned on sufficiently many training tasks are likely to be good for learning novel tasks drawn from the same environment.

We select the following training objectives to learn general-purpose sentence embeddings. Our desiderata for the task collection were: sufficient diversity, existence of fairly large datasets for training, and success as standalone training objectives for sentence representations.

Skip-thought vectors Skip-thought vectors (Kiros et al., 2015) are an extension of skip-gram word embedding models to sentences. The task typically requires a corpus of contiguous sentences, for which we use the BookCorpus (Zhu et al., 2015). The learning objective is to simultaneously predict the next and previous sentences from the current sentence. The encoder for the current sentence and the decoders for the previous (STP) and next sentence (STN) are typically parameterized as separate RNNs. We also consider training skip-thoughts by predicting only the next sentence given the current, motivated by results in Tang et al. (2017) where it is demonstrated that predicting the next sentence alone leads to comparable performance.

Neural Machine Translation Sutskever et al. (2014) and Cho et al. (2014) demonstrated that NMT can be formulated as a sequence-to-sequence learning problem where the input is a sentence in the source language and the output is its corresponding translation in the target language. We use a parallel corpus of around 4.5 million English-German (De) sentence pairs from WMT15 and 40 million English-French (Fr) sentence pairs from WMT14. We train our representations using multiple target languages motivated by improvements demonstrated by Dong et al. (2015).

Constituency Parsing (linearized parse tree construction) Vinyals et al. (2015) demonstrated that a sequence-to-sequence approach to constituency parsing is viable. The input to the encoder is the sentence itself and the decoder produces its linearized parse tree.

Following Vinyals et al. (2015), we train on a combination of 3 million weakly labeled parses obtained by parsing a random subset of the 1-billion word corpus with the Puck GPU parser[2] and gold parses from sections 0-21 of the WSJ section of Penn Treebank. Gold parses are duplicated 5 times and shuffled in with the weakly labeled parses to have a roughly 1:5 ratio of gold to noisy parses. (Vinyals et al., 2015) report that models trained with the additional noisy parses tend to produce valid parses more often and also achieve higher accuracies.

Natural Language Inference Natural language inference (NLI) is a 3-way classification problem. Given a premise and a hypothesis sentence, the objective is to classify their relationship as either entailment, contradiction, or neutral. In contrast to the previous tasks, we do not formulate this as sequence-to-sequence learning. We use the shared recurrent sentence encoder to encode both the premise and hypothesis into fixed length vectors u and v, respectively. We then feed the vector $[\mathbf{u}; \mathbf{v}; |\mathbf{u} - \mathbf{v}|; \mathbf{u} * \mathbf{v}]$, which is a concatenation of the premise and hypothesis vectors and their respective absolute difference and hadamard product, to an MLP that performs the 3-way classification. This is the same classification strategy adopted by Conneau et al. (2017a). We train on a collection of about 1 million sentence pairs from the SNLI (Bowman et al., 2015a) and MultiNLI (Williams et al., 2017) corpora.

Table 4.1 presents a summary of the number of sentence pairs used for each task.

Task	**Sentence Pairs**
En-Fr (WMT14)	40M
En-De (WMT15)	5M
Skipthought (BookCorpus)	74M
AllNLI (SNLI + MultiNLI)	1M
Parsing (PTB + 1-billion word)	4M
Total	124M

Table 4.1. An approximate number of sentence pairs for each task.

4.4.1. Multi-task training setup

Multi-task training with different data sources for each task still poses open questions. For example: When does one switch to training on a different task? Should the switching be periodic? Do we weight each task equally? If not, what training ratios do we use?

Dong et al. (2015) use periodic task alternations with equal training ratios for every task. In contrast, Luong et al. (2015a) alter the training ratios for each task based on the size of their respective training sets. Specifically, the training ratio for a particular task, α_i, is the fraction of the number of training examples in that task to the total number of

[2]`https://github.com/dlwh/puck`

training samples across all tasks. The authors then perform $\alpha_i * N$ parameter updates on task i before selecting a new task at random proportional to the training ratios, where N is a predetermined constant.

We take a simpler approach and pick a new sequence-to-sequence task to train on after every parameter update sampled uniformly. An NLI minibatch is interspersed after every ten parameter updates on sequence-to-sequence tasks (this was chosen so as to complete roughly 6 epochs of the dataset after 7 days of training). Our approach is described formally in the Algorithm below.

Require: A set of k tasks with a common source language, a shared encoder $\mathbf{E}$ across all tasks and a set of k task specific decoders $\mathbf{D_1} \dots \mathbf{D_k}$. Let θ denote each model's parameters, α a probability vector $(p_1 \dots p_k)$ denoting the probability of sampling a task such that $\Sigma_i^k p_i = 1$, datasets for each task $\mathbb{P}_1 \dots \mathbb{P}_\mathrm{k}$ and a loss function L.

```
while $\theta$ has not converged do
    1: Sample task $i \sim \mathbf{Cat}(k, \alpha)$.
    2: Sample input, output pairs $\mathbf{x}, \mathbf{y} \sim \mathbb{P}_\mathrm{i}$.
    3: Input representation $\mathbf{h}_x \leftarrow \mathbf{E}_\theta(\mathbf{x})$.
    4: Prediction $\tilde{\mathbf{y}} \leftarrow \mathbf{D}_{i_\theta}(\mathbf{h}_x)$
    5: $\theta \leftarrow \mathrm{Adam}(\nabla_\theta L(\mathbf{y}, \tilde{\mathbf{y}}))$.
end
```

4.5. Model Training

We present some architectural specifics and training details of our multi-task framework. Our shared encoder uses a common word embedding lookup table and GRU. We experiment with unidirectional, bidirectional and 2 layer bidirectional GRUs (details in Appendix section 4.6). For each task, every decoder has its separate word embedding lookups, conditional GRUs and fully connected layers that project the GRU hidden states to the target vocabularies. The last hidden state of the encoder is used as the initial hidden state of the decoder and is also presented as input to all the gates of the GRU at every time step. For natural language inference, the same encoder is used to encode both the premise and hypothesis and a concatenation of their representations along with the absolute difference and hadamard product (as described in (Conneau et al., 2017a)) are given to a single layer MLP with a dropout (Srivastava et al., 2014) rate of 0.3. All models use word embeddings of 512 dimensions and GRUs with either 1500 or 2048 hidden units. We used minibatches of 48 examples and the Adam (Kingma & Ba, 2014) optimizer with a learning rate of 0.002. Models were trained for 7 days on an Nvidia Tesla P100-SXM2-16GB GPU. While Kiros et al. (2015) report close to a month of training, we only train for 7 days, made possible by advancements in GPU hardware and software (cuDNN RNNs).

We did not tune any of the architectural details and hyperparameters owing to the fact that we were unable to identify any clear criterion on which to tune them. Gains in performance on a specific task do not often translate to better transfer performance.

4.5.1. Vocabulary Expansion & Representation Pooling

In addition to performing 10-fold cross-validation to determine the L2 regularization penalty on the logistic regression models, we also tune the way in which our sentence representations are generated from the hidden states corresponding to words in a sentence. For example, Kiros et al. (2015) use the last hidden state while Conneau et al. (2017a) perform max-pooling across all of the hidden states. We consider both of these approaches and pick the one with better performance on the validation set. We note that max-pooling works best on sentiment tasks such as MR, CR, SUBJ and MPQA, while the last hidden state works better on all other tasks.

We also employ vocabulary expansion on all tasks as in Kiros et al. (2015) by training a linear regression to map from the space of pre-trained word embeddings (GloVe) to our model's word embeddings.

4.6. Multi-task model details

This section describes the specifics our multi-task ablations in the experiments section. These definitions hold for all tables except for 4.2 and 4.5. We refer to skip-thought next as STN, French and German NMT as Fr and De, natural language inference as NLI, skip-thought previous as STP and parsing as Par.

+STN +Fr +De : A concatenation of the representations trained on these tasks with a unidirectional and bidirectional GRU with 1500 hidden units each.

+STN +Fr +De +NLI : A concatenation of the representations trained on these tasks with a bidirectional GRU and one trained without NLI, each with 1500 hidden units.

+STN +Fr +De +NLI +L : A concatenation of the representations trained on these tasks with a bidirectional GRU and one trained without NLI, each with 2048 hidden units.

+STN +Fr +De +NLI +L +STP : A concatenation of the representations trained on these tasks with a bidirectional GRU and one trained without STP, each with 2048 hidden units.

+STN +Fr +De +NLI +2L +STP : A concatenation of the representations trained on these tasks with a 2-layer bidirectional GRU and a 1-layer model trained without STP, each with 2048 hidden units.

+STN +Fr +De +NLI +L +STP +Par : A concatenation of the representations trained on these tasks with a bidirectional GRU and a model trained without Par, each with 2048 hidden units.

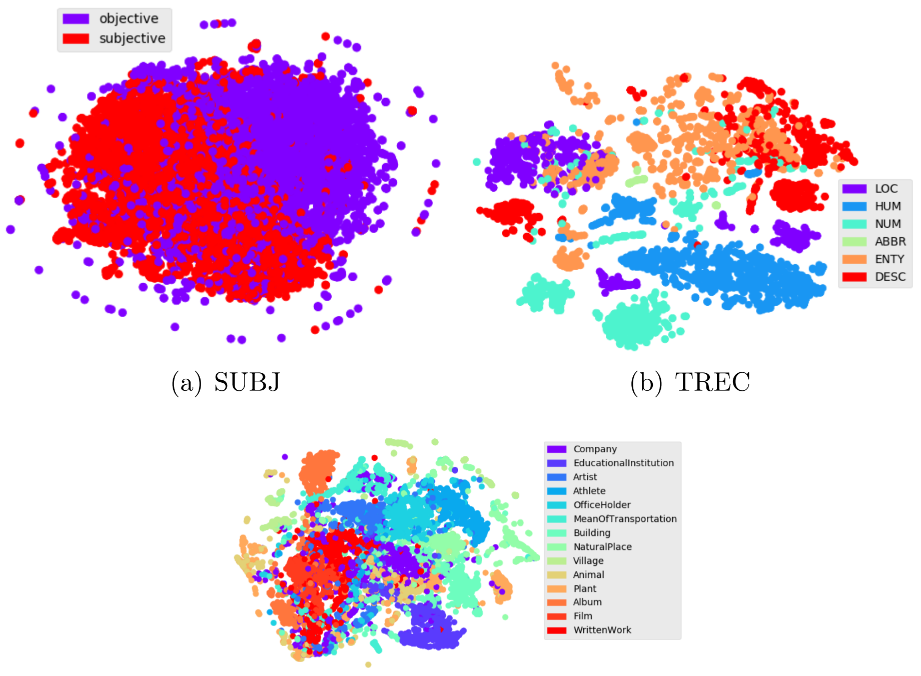
In tables 4.2 and 4.5 we do not concatenate the representations of multiple models.

4.7. Evaluation Strategies, Experimental Results & Discussion

In this section, we describe our approach to evaluate the quality of our learned representations, present the results of our evaluation and discuss our findings.

4.7.1. Evaluation Strategy

We follow a similar evaluation protocol to those presented in Kiros et al. (2015); Hill et al. (2016); Conneau et al. (2017a) which is to use our learned representations as features for a low complexity classifier (typically linear) on a novel supervised task/domain unseen during training *without updating the parameters of our sentence representation model*. We also consider such a transfer learning evaluation in an artificially constructed low-resource setting. In addition, we also evaluate the quality of our learned individual word representations using standard benchmarks (Faruqui & Dyer, 2014; Tsvetkov et al., 2015).



(a) SUBJ (b) TREC

(c) DBpedia

Fig. 4.1. T-SNE visualizations of our sentence representations on 3 different datasets.
1

The choice of transfer tasks and evaluation framework[3] are borrowed largely from Conneau et al. (2017a). We provide a condensed summary of the tasks in section below, but refer readers to their paper for a more detailed description.

4.7.2. Description of evaluation tasks

(Kiros et al., 2015) and (Conneau et al., 2017a) provide a detailed description of tasks that are typically used to evaluate sentence representations. We provide a condensed summary and refer readers to their work for a more thorough description.

- **Text Classification** - We evaluate on text classification benchmarks - sentiment classification on movie reviews (MR), product reviews (CR) and Stanford sentiment (SST), question type classification (TREC), subjectivity/objectivity classification (SUBJ) and opinion polarity (MPQA). Representations are used to train a logistic regression classifier with 10-fold cross validation to tune the L2 weight penalty. The evaluation metric for all these tasks is classification accuracy.
- **Paraphrase Identification** - We also evaluate on pairwise text classification tasks such as paraphrase identification on the Microsoft Research Paraphrase Corpus (MRPC) corpus. This is a binary classification problem to identify if two sentences are paraphrases of each other. The evaluation metric is classification accuracy and F1.
- **Entailment and Semantic Relatedness** - To test if similar sentences share similar representations, we evaluate on the SICK relatedness (SICK-R) task where a linear model is trained to output a score from 1 to 5 indicating the relatedness of two sentences. We also evaluate using the entailment labels in the same dataset (SICK-E) which is a binary classification problem. The evaluation metric for SICK-R is Pearson correlation and classification accuracy for SICK-E.
- **Semantic Textual Similarity** - In this evaluation, we measure the relatedness of two sentences using only the cosine similarity between their representations. We use the similarity textual similarity (STS) benchmark tasks from 2012-2016 (STS12, STS13, STS14, STS15, STS16, STSB). The STS dataset contains sentences from a diverse set of data sources. The evaluation criteria is Pearson correlation.
- **Image-caption retrieval** - Image-caption retrieval is typically formulated as a ranking task wherein images are retrieved and ranked based on a textual description and vice-versa. We use 113k training images from MSCOCO with 5k images for validation and 5k for testing. Image features are extracted using a pre-trained 110 layer ResNet. The evaluation criterion is Recall@K and the median K across 5 different splits of the data.

[3] `https://www.github.com/facebookresearch/SentEval`

- **Quora Duplicate Question Classification** - In addition to the above tasks which were considered by (Conneau et al., 2017a), we also evaluate on the recently published Quora duplicate question dataset[4] since it is an order of magnitude larger than the others (approximately 400,000 question pairs). The task is to correctly identify question pairs that are duplicates of one another, which we formulate as a binary classification problem. We use the same data splits as in (Wang et al., 2017). Given the size of this data, we consider a more expressive classifier on top of the representations of both questions. Specifically, we train a 4 layer MLP with 1024 hidden units, with a dropout rate of 0.5 after every hidden layer. The evaluation criterion is classification accuracy. We also artificially create a low-resource setting by reducing the number of training examples between 1,000 and 25,000 using the same splits as (Shen et al., 2017a).
- **Sentence Characteristics & Syntax** - In an attempt to understand what information is encoded in by sentence representations, we consider six different classification tasks where the objective is to predict sentence characteristics such as length, word content and word order (Adi et al., 2016) or syntactic properties such as active/passive, tense and the top syntactic sequence (TSS) from the parse tree of a sentence (Shi et al., 2016).
  The sentence characteristic tasks are setup in the same way as described in (Adi et al., 2016). The length task is an 8-way classification problem where sentence lengths are binned into 8 ranges. The content task is formulated as a binary classification problem that takes a concatenation of a sentence representation $u \in \mathbb{R}^k$ and a word representation $w \in \mathbb{R}^d$ to determine if the word is contained in the sentence. The order task is an extension of the content task where a concatenation of the sentence representation and word representations of two words in sentence is used to determine if the first word occurs before or after the second. We use a random subset of the 1-billion-word dataset for these experiments that were not used to train our multi-task representations.
  The syntactic properties tasks are setup in the same way as described in (Shi et al., 2016).The passive and tense tasks are characterized as binary classification problems given a sentence's representation. The former's objective is to determine if a sentence is written in active/passive voice while the latter's objective is to determine if the sentence is in the past tense or not. The top syntactic sequence (TSS) is a 20-way classification problem with 19 most frequent top syntactic sequences and 1 miscellaneous class. We use the same dataset as the authors but different training, validation and test splits.

[4]`https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs`
[5]`https://github.com/kudkudak/word-embeddings-benchmarks/wiki`

Embedding	Dim	V143	SIMLEX	WS353	YP130	MTurk771	RG65	MEN	QVEC
SENNA	50	0.36	0.27	0.49	0.16	0.50	0.50	0.57	-
NMT En-Fr	620	-	0.46	0.49	-	0.46	0.59	0.49	-
GloVe6B	300	0.31	0.37	0.61	0.56	0.65	0.77	0.74	0.47
GloVe840B	300	0.34	0.41	0.71	0.57	0.71	0.76	**0.80**	-
Skipgram GoogleNews	300	-	0.44	0.70	-	0.67	0.76	0.74	-
FastText	300	0.40	0.38	**0.74**	0.53	0.67	**0.80**	0.76	0.50
Multilingual	512	0.26	0.40	0.68	0.42	0.60	0.65	0.76	-
Charagram	300	-	0.71	-	-	-	-	-	-
Attract-Repel	300	-	0.75	-	-	-	-	-	-
Our Model									
+STN +Fr +De +NLI +L	512	**0.56**	**0.59**	0.66	**0.62**	0.70	0.65	0.67	**0.54**

Table 4.2. Evaluation of word embeddings. All results were computed using Faruqui & Dyer (2014) with the exception of the Skipgram, NMT, Charagram and Attract-Repel embeddings. Skipgram and NMT results were obtained from Jastrzebski et al. (2017)[5]. Charagram and Attract-Repel results were taken from Wieting et al. (2016) and Mrkšić et al. (2017) respectively. We also report QVEC benchmarks (Tsvetkov et al., 2015)

4.7.3. Experimental Results & Discussion

Table 4.4 presents the results of training logistic regression on 10 different supervised transfer tasks using different fixed-length sentence representation. Supervised approaches trained from scratch on some of these tasks are also presented for comparison. We present performance ablations when adding more tasks and increasing the number of hidden units in our GRU (+L). Ablation specifics are presented in section 4.6.

It is evident from Table 4.4 that adding more tasks improves the transfer performance of our model. Increasing the capacity our sentence encoder with more hidden units (+L) as well as an additional layer (+2L) also leads to improved transfer performance. We observe gains of 1.1-2.0% on the sentiment classification tasks (MR, CR, SUBJ & MPQA) over Infersent. We demonstrate substantial gains on TREC (6% over Infersent and roughly 2% over the CNN-LSTM), outperforming even a competitive supervised baseline. We see similar gains (2.3%) on paraphrase identification (MPRC), closing the gap on task specific models. The addition of constituency parsing improves performance on sentence relatedness (SICK-R) and entailment (SICK-E) consistent with observations made by Bowman et al. (2016).

In Table 4.3, we show that simply training an MLP on top of our fixed sentence representations outperforms several strong & complex supervised approaches that use attention mechanisms, even on this fairly large dataset. For example, we observe a 0.2-0.5% improvement over the decomposable attention model (Parikh et al., 2016). When using only a small fraction of the training data, indicated by the columns 1k-25k, we are able to outperform the Siamese and Multi-Perspective CNN using roughly 6% of the available training set. We

Model	Accuracy				
	1k	5k	10k	25k	All (400k)
Siamese-CNN	-	-	-	-	79.60
Multi-Perspective-CNN	-	-	-	-	81.38
Siamese-LSTM	-	-	-	-	82.58
Multi-Perspective-LSTM	-	-	-	-	83.21
L.D.C	-	-	-	-	85.55
BiMPM	-	-	-	-	88.17
DecAtt (GloVe)	-	-	-	-	86.52
DecAtt (Char)	-	-	-	-	86.84
pt-DecAtt (Word)	-	-	-	-	87.54
pt-DecAtt (Char)	-	-	-	-	**88.40**
CNN (Random)	56.3	59.2	63.8	68.9	-
CNN (GloVe)	58.5	62.4	66.1	70.2	-
LSTM-AE	59.3	63.8	67.2	70.9	-
Deconv-AE	60.2	65.1	67.7	71.6	-
Deconv LVM	62.9	67.6	69.0	72.4	-
Infersent (LogReg)	68.8	73.8	75.7	76.4	-
Infersent (MLP)	71.8	75.7	77.2	78.9	84.79
Our Models					
+STN +Fr +De +NLI +L +STP (LogReg)	70.4	74.8	75.9	77.2	-
+STN +Fr +De +NLI +L +STP (MLP)	**74.7**	**77.7**	**78.3**	**81.4**	87.01

Table 4.3. Supervised & low-resource classification accuracies on the Quora duplicate question dataset. Accuracies are reported corresponding to the number of training examples used. The first 6 rows are taken from Wang et al. (2017), the next 4 are from Tomar et al. (2017), the next 5 from Shen et al. (2017a) and The last 4 rows are our experiments using Infersent (Conneau et al., 2017a) and our models.

also outperform the Deconv LVM model proposed by Shen et al. (2017a) in this low-resource setting.

Unlike Conneau et al. (2017a), who use pretrained GloVe word embeddings, we learn our word embeddings from scratch. Somewhat surprisingly, in Table 4.2 we observe that the learned word embeddings are competitive with popular methods such as GloVe, word2vec, and fasttext (Bojanowski et al., 2016) on the benchmarks presented by Faruqui & Dyer (2014) and Tsvetkov et al. (2015).

In Table 4.5, we probe our sentence representations to determine if certain sentence characteristics and syntactic properties can be inferred following work by Adi et al. (2016) and Shi et al. (2016). We observe that syntactic properties are better encoded with the addition of multi-lingual NMT and parsing. Representations learned solely from NLI do appear to encode syntax but incorporation into our multi-task framework does not amplify this signal. Similarly, we observe that sentence characteristics such as length and word order are better encoded with the addition of parsing.

Model	MR	CR	SUBJ	MPQA	SST	TREC	MRPC	SICK-R	SICK-E	STSB	Δ
Transfer approaches											
FastSent	70.8	78.4	88.7	80.6	-	76.8	72.2/80.3	-	-	-	-
FastSent+AE	71.8	76.7	88.8	81.5	-	80.4	71.2/79.1	-	-	-	-
NMT En-Fr	64.7	70.1	84.9	81.5	-	82.8	-	-	-	-	-
CNN-LSTM	77.8	82.1	93.6	89.4	-	92.6	76.5/83.8	0.862	-	-	-
Skipthought	76.5	80.1	93.6	87.1	82.0	92.2	73.0/82.0	0.858	82.3	-	-
Skipthought + LN	79.4	83.1	93.7	89.3	82.9	88.4	-	0.858	79.5	72.1/70.2	-
Word Embedding Average	-	-	-	-	82.2	-	-	0.860	84.6	-	-
DiscSent + BiGRU	-	-	88.6	-	-	81.0	71.6/-	-	-	-	-
DiscSent + unigram	-	-	92.7	-	-	87.9	72.5/-	-	-	-	-
DiscSent + embed	-	-	93.0	-	-	87.2	75.0/-	-	-	-	-
Byte mLSTM	**86.9**	**91.4**	**94.6**	88.5	-	-	75.0/82.8	0.792	-	-	-
Infersent (SST)	(*)	83.7	90.2	89.5	(*)	86.0	72.7/80.9	0.863	83.1	-	-
Infersent (SNLI)	79.9	84.6	92.1	89.8	83.3	88.7	75.1/82.3	0.885	86.3	-	-
Infersent (AllNLI)	81.1	86.3	92.4	90.2	**84.6**	88.2	76.2/83.1	0.884	86.3	75.8/75.5	0.0
Our Models											
+STN	78.9	85.8	93.7	87.2	80.4	84.2	72.4/81.6	0.840	82.1	72.9/72.4	-2.56
+STN +Fr +De	80.3	85.1	93.5	90.1	83.3	92.6	77.1/83.3	0.864	84.8	77.1/77.1	0.01
+STN +Fr +De +NLI	81.2	86.4	93.4	90.8	84.0	93.2	76.6/82.7	0.884	87.0	79.2/79.1	0.99
+STN +Fr +De +NLI +L	81.7	87.3	94.2	90.8	84.0	**94.2**	77.1/83.0	0.887	87.1	78.7/78.2	1.33
+STN +Fr +De +NLI +L +STP	82.7	88.0	94.1	91.2	84.5	92.4	77.8/83.9	0.885	86.8	78.7/78.4	1.44
+STN +Fr +De +NLI +2L +STP	82.8	88.3	94.0	**91.3**	83.6	92.6	77.4/83.3	0.884	87.6	**79.2/79.1**	1.47
+STN +Fr +De +NLI +L +STP +Par	82.5	87.7	94.0	90.9	83.2	93.0	**78.6/84.4**	**0.888**	**87.8**	78.9/78.6	**1.48**
Approaches trained from scratch on these tasks											
Naive Bayes SVM	79.4	81.8	93.2	86.3	83.1	-	-	-	-	-	
AdaSent	83.1	86.3	95.5	93.3	-	92.4	-	-	-	-	
TF-KLD	-	-	-	-	-	-	80.4/85.9	-	-	-	
Illinois LH	-	-	-	-	-	-	-	-	84.5	-	
Dependency tree LSTM	-	-	-	-	-	-	-	0.868	-	-	
Neural Semantic Encoder	-	-	-	-	89.7	-	-	-	-	-	
BLSTM-2DCNN	82.3	-	94.0	-	89.5	96.1	-	-	-	-	

Table 4.4. Evaluation of sentence representations on a set of 10 tasks using a linear model trained using each model's representations. The FastSent and NMT En-Fr models are described in Hill et al. (2016), CNN-LSTM in Gan et al. (2016), Skipthought in Kiros et al. (2015), Word embedding average in Arora et al. (2016), DiscSent from Jernite et al. (2017), Byte mLSTM from Radford et al. (2017), Infersent in Conneau et al. (2017a), Neural Semantic Encoder from Munkhdalai & Yu (2017), BLSTM-2DCNN from Zhou et al. (2016). STN, Fr, De, NLI, L, 2L, STP & Par stand for skip-thought next, French translation, German translation, natural language inference, large model, 2-layer large model, skip-thought previous and parsing respectively. Δ indicates the average improvement over Infersent (AllNLI) across all 10 tasks. For MRPC and STSB we consider only the F1 score and Spearman correlations respectively and we also multiply the SICK-R scores by 100 to have all differences in the same scale. Bold numbers indicate the best performing transfer model on a given task. Each column contains exactly two rows that are underlined to indicate a) the best performing model from prior work and b) our best performing model.

In Table 4.6, we note that our sentence representations outperform skip-thoughts and are on par with Infersent for image-caption retrieval. We also observe that comparing sentences

Model	Length	Content	Order	Passive	Tense	TSS
	Sentence characteristics			*Syntatic properties*		
Majority Baseline	22.0	50.0	50.0	81.3	77.1	56.0
Infersent (AllNLI)	75.8	**75.8**	75.1	92.1	93.3	70.4
Skipthought	-	-	-	94.0	96.5	74.7
Our Models						
Skipthought +Fr +De	86.8	75.7	81.1	97.0	97.0	77.1
Skipthought +Fr +De +NLI	88.1	75.7	81.5	96.7	96.8	77.1
Skipthought +Fr +De +NLI +Par	**93.7**	75.5	**83.1**	**98.0**	**97.6**	**80.7**

Table 4.5. Evaluation of sentence representations by probing for certain sentence characteristics and syntactic properties. Sentence length, word content & word order from Adi et al. (2016) and sentence active/passive, tense and top level syntactic sequence (TSS) from Shi et al. (2016). Numbers reported are the accuracy with which the models were able to predict certain characteristics.

using cosine similarities correlates reasonably well with their relatedness on semantic textual similarity benchmarks (Table 4.7).

We also present qualitative analysis of our learned representations by visualizations using dimensionality reduction techniques (Figure 4.1) and nearest neighbor exploration (Table 4.8). Figure 4.1 shows t-sne plots of our sentence representations on three different datasets - SUBJ, TREC and DBpedia. DBpedia is a large corpus of sentences from Wikipedia labeled by category and used by Zhang et al. (2015). Sentences appear to cluster reasonably well according to their labels. The clustering also appears better than that demonstrated in Figure 2 of Kiros et al. (2015) on TREC and SUBJ. Table 4.8 contains sentences from the BookCorpus and their nearest neighbors. Sentences with some lexical overlap and similar discourse structure appear to be clustered together.

	Caption Retrieval				**Image Retrieval**			
Model	**R@1**	**R@5**	**R@10**	**Med r**	**R@1**	**R@5**	**R@10**	**Med r**
Sentence Representations trained from scratch								
m-CNN	38.3	-	81.0	2	27.4	-	79.5	3
m-CNN Ensemble	42.8	-	84.1	2	32.6	-	82.8	3
Oreder-embeddings	46.7	-	88.9	2	37.9	-	85.9	2
Transfer learning approaches								
Skipthought	37.9	72.2	84.3	2	30.6	66.2	81.0	3
Infersent (SNLI)	42.4	**76.1**	87.0	2	33.2	69.7	83.6	3
Infersent (AllNLI)	42.6	75.3	**87.3**	2	**33.9**	69.7	**83.8**	3
+STN +Fr +De +NLI +L +STP	**43.0**	76.0	87.0	2	33.8	**70.1**	83.6	2.8

Table 4.6. COCO Retrieval with ResNet-101 features

Model	STS12	STS13	STS14	STS15	STS16
Unsupervised/Transfer Approaches					
Skipthought+LN	30.8	24.8	31.4	31.0	-
GloVe average	52.5	42.3	54.2	52.7	-
GloVe TF-IDF	58.7	52.1	63.8	60.6	-
GloVe + WR (U)	56.2	56.6	68.5	71.1	-
Charagram-phrase	**<u>66.1</u>**	**<u>57.2</u>**	**<u>74.7</u>**	**<u>76.1</u>**	-
Infersent	59.2	58.9	69.6	71.3	**<u>71.4</u>**
+STN +Fr +De +NLI +L +STP	60.6	54.7	65.8	74.2	66.4
Supervised Approaches					
LSTM w/o output gates	51.0	45.2	59.8	63.9	-
PP-Proj	60.0	56.8	71.3	74.8	-

Table 4.7. Evaluation of sentence representations on the semantic textual similarity benchmarks. Numbers reported are Pearson Correlations x100. Skipthought, GloVe average, GloVe TF-IDF, GloVe + WR (U) and all supervised numbers were taken from Arora et al. (2016) and Wieting et al. (2015) and Charagram-phrase numbers were taken from Wieting et al. (2016). Other numbers were obtained from the evaluation suite provided by Conneau et al. (2017a)

4.8. Conclusion & Future Work

We present a multi-task framework for learning general-purpose fixed-length sentence representations. Our primary motivation is to encapsulate the inductive biases of several diverse training signals used to learn sentence representations into a single model. Our multi-task framework includes a combination of sequence-to-sequence tasks such as multi-lingual NMT, constituency parsing and skip-thought vectors as well as a classification task - natural language inference. We demonstrate that the learned representations yield competitive or superior results to previous general-purpose sentence representation methods. We also observe that this approach produces good word embeddings.

In future work, we would like understand and interpret the inductive biases that our model learns and observe how it changes with the addition of different tasks beyond just our simple analysis of sentence characteristics and syntax. Having a rich, continuous sentence representation space could allow the application of state-of-the-art generative models of images such as that of Nguyen et al. (2016) to language. One could also consider controllable text generation by directly manipulating the sentence representations and realizing it by decoding with a conditional language model.

Query and nearest sentence
i ... i want ... " she could n't finish the sentence and she did n't have to .
i ... " she had no idea what she had intended to say .
and ... " she could n't finish .
i ... " he winced as if he could n't bear whatever flashed through his mind .
you were going to sleep with her just to ... " i could n't even finish my sentence .
tyler was kind enough to wait two weeks before talking about their relationship again .
he 'd been more than honest in setting forth his feelings about their relationship .
she did n't wait for him to ask what she was talking about .
damn , just thinking about it made him want to talk to her again .
i did n't wait for my bestie to try and talk me out of it .
" you can , my lady . "
" of course , my lady . "
" what can i do for you , my lady ? "
" watch yourself , my lady . "
" where to , my lady ? "
they 'd kept silent about it , a terrifying secret between them that was n't going away .
he 'd known they had secrets between them , things they were n't ready to share .
she 'd kept his secret , even from her two closest confidantes .
he 'd made her a promise and he was n't about to break it .
she 'd brushed them off as shock-induced ramblings , but now riley was n't so sure .
" you deserve good things , " marshal said , but he was still wearing that damn pitying smile .
" i 'm really happy to see you , ladybug , " he said softly , the smile never leaving his eyes .
" you 're a bad liar , " he said , still smiling .
" it 's okay , " i said , but he shook his head and sighed .
" i 'll answer that , " he said , his eyes still lit with humor .
belatedly i turned to look up at max and exclaimed , " hey ! "
i started , hollering at his retreating form , " thank you ! "
he slowly turned and looked at me as he barely said , " fine . "
my head came up and i asked , " sorry ? "
i shook my head and mouthed , " i ca n't believe you ! "

Table 4.8. A query sentence and its nearest neighbors sorted by decreasing cosine similarity using our model. Sentences and nearest neighbors were chosen from a random subset of 500,000 sentences from the BookCorpus

Chapter 5

Prologue to the second article

5.1. Article Details

Towards Text Generation with Adversarially Learned Neural Outlines. Sandeep Subramanian, Sai Rajeswar, Alessandro Sordoni, Adam Trischler, Aaron Courville, Christopher Pal. Neural Information Processing Systems 2018.

Personal Contribution The project began as an extension of Chapter 4 to see if the learned sentence representations could help with text generation. After working on Subramanian et al. (2018c), Alessandro Sordoni, Adam Trischler and Sandeep Subramanian were investigating the smoothness of the learned sentence representation space (end of Chapter 6) via latent space interpolations - this necessitated a "universal decoder" that could reconstruct the contents of a sentence embedding. These discussions led to the gradient-based interpolation idea. Sandeep Subramanian and Sai Rajeswar formulated the GAN-based approach for modeling the distribution of sentence embeddings. Christopher Pal and Aaron Courville advised on the project and suggested experiments to run. Sandeep Subramanian wrote all of the code for the project, ran all of the experiments and wrote parts of the paper.

5.2. Context

The goal of this work was to present a two-step generative model for text by first using a GAN to model the distribution of sentence embeddings and then using its generator to generate a synthetic sentence embedding. An autoregressive decoder then reconstructs its contents. The generated sentence embedding provides strong and "global" conditioning for the decoder. GAN training is straightforward in this setup since we're dealing with continuous sentence embeddings rather than the underlying discrete data. It has the advantage of having a good learned prior when compared to VAEs and not suffering from posterior collapse.

5.3. Contributions

The paper presents a latent-space generative model for text that combines GANs and autoregressive models. It generates better samples than an LSTM language model at high temperatures and performs similarly at low temperatures. It demonstrates that the approach can be adapted to a conditional generation setting using a conditional GAN. Finally, it shows how continuous representations of text and and a decoder can be used to interpolate smoothly in the latent space using gradient signals from the decoder.

5.4. Research Impact

The work was a promising way of using GANs in text by sidestepping discreteness and training in a continuous latent space. Since then, several approaches have used learned latent spaces to generate and alter text (Dathathri et al., 2019; Wang et al., 2019; Mai et al., 2020).

Chapter 6

Towards Text Generation with Adversarially Learned Neural Outlines

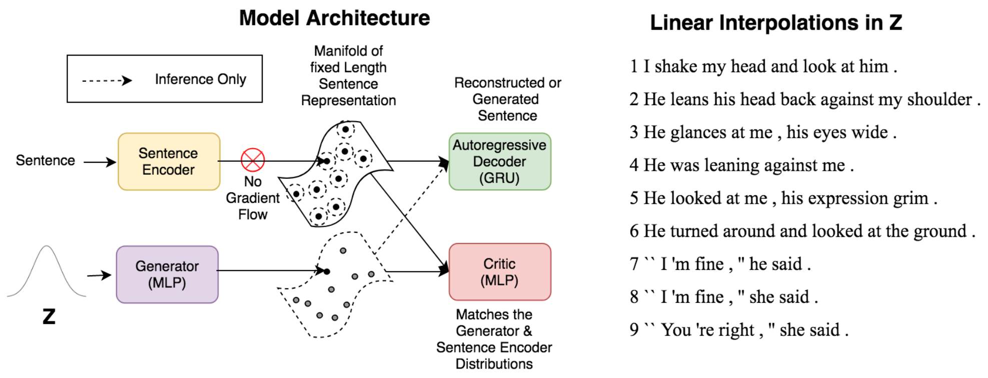
6.1. Introduction

Deep neural networks are powerful tools for modeling sequential data (Mikolov et al., 2010; Van Den Oord et al., 2016; Jozefowicz et al., 2016). Tractable maximum-likelihood (MLE) training of these models typically involves factorizing the joint distribution over random variables into a product of conditional distributions that models the one-step-ahead probability in the sequence via the chain rule. Each conditional is then modeled by an expressive family of functions, such as neural networks. These models have been successful in a variety of tasks. However, the only source of variation is modeled in the conditional output probability at every step: there is limited capacity for capturing the higher-level structure likely present in natural text and other sequential data (e.g., through a hierarchical generation process (Serban et al., 2016)).

Variational Autoencoders (VAE) (Kingma & Welling, 2013) provide a tractable method to train hierarchical latent-variable generative models. In the context of text data, latent variables may assume the role of sentence representations that govern a lower-level generation process, thus facilitating controlled generation of text.

However, VAEs for text are notoriously hard to train when combined with powerful auto-regressive decoders (Bowman et al., 2015b; Goyal et al., 2017b; Shabanian et al., 2017). This is due to the "posterior collapse" problem: the model ends up relying solely on the auto-regressive properties of the decoder while ignoring the latent variables, which become uninformative. This phenomenon is partly a consequence of the restrictive assumptions on the parametric form of the posterior and prior approximations, usually modeled as simple diagonal Gaussian distributions.

General-purpose sentence encoders have been shown to produce representations that are useful across a wide range of natural language processing (NLP) tasks (Kiros et al., 2015;


Model Architecture
Inference Only
Manifold of fixed Length Sentence Representation
Reconstructed or Generated Sentence
Sentence
Sentence Encoder
No Gradient Flow
Autoregressive Decoder (GRU)
Generator (MLP)
Critic (MLP)
Z
Matches the Generator & Sentence Encoder Distributions
Linear Interpolations in Z

1 I shake my head and look at him .
2 He leans his head back against my shoulder .
3 He glances at me , his eyes wide .
4 He was leaning against me .
5 He looked at me , his expression grim .
6 He turned around and looked at the ground .
7 `` I 'm fine , '' he said .
8 `` I 'm fine , '' she said .
9 `` You 're right , '' she said .

Fig. 6.1. Overall Model Architecture. (Left) A GAN trained to model the distribution of fixed-length sentence vectors. Noise indicated by a small dotted Gaussian ball, is injected to every point on the data mainfold *when feeding it to the decoder*. (Right) Samples produced from our generative process by interpolating linearly between two points sampled at random from the input noise space of the generator.

Conneau et al., 2017a; Subramanian et al., 2018c). Such models seem to capture syntactic and semantic properties of text in self-contained vector representations (Subramanian et al., 2018c; Conneau et al., 2018). Although these representations have been used successfully in downstream tasks, their usefulness for text generation *per se* has not yet been thoroughly investigated, to the best of our knowledge.

In this paper, we study how high-quality sentence representations, obtained by general purpose sentence encoders, can be leveraged to train better models for text generation. In particular, we interpret the set of sentence embeddings obtained from training on large text corpora as samples from an expressive and unknown prior distribution over a continuous space. We model this distribution using a Generative Adversarial Network (GAN), which enables us to build a fully generative model of sentences as follows: we first sample a sentence embedding from the GAN generator, then decode it to the observed space (words) using a conditional GRU-based language model (which we refer to as the decoder in the rest of this work). The sentence embeddings produced by the learned generator can be seen as "neural outlines" that provide high-level information to the decoder, which in turn can generate many possible sentences given a single latent representation. The conditioning information effectively guides the decoder to a smaller space of possible generations.

The takeaways from our work are:

- We propose the use of fixed-length representations induced by general-purpose sentence encoders for training generative models of text, and demonstrate their potential both qualitatively and quantitatively.

- We extend our model to conditional text generation. Specifically, we train a conditional GAN that learns to transform a given hypothesis representation in the sentence embedding space into a premise embedding that satisfies a specified entailment relationship.
- We propose a gradient-based technique to generate meaningful interpolations between two given sentence embeddings. We see this technique as being able to navigate around holes in the data manifold by moving in areas of high data density using gradient signals from the decoder. Qualitatively, the obtained interpolations appear more natural than those obtained by linear interpolation, when the distribution (in our case the sentence embeddings distribution) doesn't have a simple parametric form such as a Gaussian.

6.2. Related Work

Our work is similar in spirit to the line of work that employs adversarial training on a latent representation of data, such as Adversarial Autoendoers (AAE) (Makhzani et al., 2015), Wasserstein Autoencoders (WAE) (Tolstikhin et al., 2017) and Adversarially Regularized Autoencoders (ARAE) (Kim et al., 2017). AAEs and WAEs are similar to Variational Autoencoders (VAE) (Kingma & Welling, 2013) in that they learn an encoder that produces an approximate latent posterior distribution $q_\phi(z|x)$, and a decoder $p_\theta(x|z)$ that is trained to minimize the reconstruction error of x. They differ in the way they are regularized. VAEs penalize the discrepancy between the approximate posterior and a prior distribution $D_{\mathrm{KL}}(q_\phi(z|x)\|p(z))$, which controls the tightness of the variational bound on the data log-likelihood. AAEs and WAEs, on the other hand, adversarially match the aggregated or marginal posterior $q_\phi(z)$, with a fixed prior distribution that shapes the posterior into what is typically a simple distribution, such as a Gaussian.

ARAEs train, by means of a critic, a flexible prior distribution regularizing the sentence representations obtained by an auto-encoder. In contrast, we provide evidence that assuming a uniform prior distribution during the training of sentence representations and fitting a flexible prior *a posteriori* over the obtained representations yields better performance than learning both jointly, and helps us scale to longer sequences, larger vocabularies, and richer datasets.

Recent successes of generative adversarial networks in the image domain (Karras et al., 2017) have motivated their use in modeling discrete sequences like text. However, discreteness introduces problems in end-to-end training of the generator. Policy gradient techniques (Williams, 1992) are one way to circumvent this problem, but typically require maximum-likelihood pre-training (Yu et al., 2017; Che et al., 2017), as do actor-critic methods (Goyal et al., 2017a; Fedus et al., 2018). Gumbel-softmax based approaches have also proven useful in the restricted setting of short sequence lengths and small output vocabulary sizes (Kusner & Hernández-Lobato, 2016). Approaches without maximum-likelihood pre-training that operate on continuous softmax outputs from the generator such as (Gulrajani et al., 2017; Rajeswar et al., 2017; Press et al., 2017) have also shown promise, but apply mostly in artificial settings. Adversarial training on the continuous hidden states of an RNN was used by (Shen et al., 2017b) for unaligned style transfer motivated by the professor forcing algorithm (Lamb et al., 2016), which uses adversarial training to match the hidden states of a free-running and teacher forced RNN.

In this work, we argue that generative adversarial training on latent representations of a sentence not only alleviates the non-differentiability issue of regular GAN training on discrete sequences, but also eases the learning process by instead modeling an already smoothened out manifold of the underlying discrete data distribution. We also simultaneously reap the benefits of having a "sequence-level" training objective while somewhat side-stepping the temporal credit assignment problem.

6.3. Approach

In this section we discuss the building blocks of our generative model. Our model consists of two distinct and independently trainable components: (1) a generative adversarial network trained to match the distribution of the fixed-length vector representations induced by general purpose sentence encoders and (2) a conditional RNN language model trained with maximum-likelihood to reconstruct the words in a sentence given its vector representation. The overall architecture is presented in Fig. 6.1.

6.3.1. General Purpose Sentence Encoders

With the success of models that learn distributed representations of words in an unsupervised manner, there has been a recent focus on building models that learn fixed-length vector representations of whole sentences that are "general-purpose" enough to be useful as black-box features across a wide range of downstream Natural Language Processing tasks. Extensions of the skip-gram model to sentences (Kiros et al., 2015), sequential de-noising autoencoders (Hill et al., 2016), features learned from Natural Language Inference models (Conneau et al., 2017a), and large-scale multi-task models that are trained with multiple objectives (Subramanian et al., 2018c) have been shown to learn useful (Conneau & Kiela, 2018; Wang et al., 2018) fixed-length representations of text. They also encode several characteristics of a sentence faithfully, including word order, content, and length (Conneau et al., 2018; Adi et al., 2016; Subramanian et al., 2018c). This is important, since we want to be able to reliably reconstruct the contents of a sentence from its vector representation. In this work, we will use the pre-trained sentence encoder from (Subramanian et al., 2018c), which consists of an embedding look-up table learned from scratch and a single layer GRU (Chung et al., 2014) with 2048 hidden units. We use the last hidden state of the GRU to create a compressed representation. Thus, each sentence x is represented by a vector $E(x) = h_x \in \mathbb{R}^{2048}$, where E denotes our general-purpose sentence encoder.

6.3.2. Generative Adversarial Training

Generative Adversarial networks are a family of implicit generative models that formulate the learning process as a two player minimax game between a generator and discriminator/-critic (Goodfellow et al., 2014): the critic is trained to distinguish samples of the true data distribution from those of the generator distribution. The generator is trained to "fool" the critic by approximating the data distribution, given samples from a much simpler distribution, e.g., a Gaussian. The discriminator D or critic f_w and the generator G are typically parameterized by neural networks. In our setting, the data distribution is the distribution $P(h_x)$ of sentence embeddings $h_x = E(x)$ provided by our sentence encoder E applied to samples $x \sim P_D$ from a dataset D. Specifically, the discriminator and generator are trained by:

$$\min_G \max_D V(D, G) = \mathop{\mathbb{E}}_{x \sim P_D}[\log D(E(x))] + \mathop{\mathbb{E}}_{z \sim P(z)}[\log(1 - D(G(z)))] \tag{6.1}$$

In presenting the Wasserstein GAN, (Arjovsky et al., 2017) argue that while KL divergence is a good divergence for low-dimensional data, it is often too strong in the high-dimensional settings we are often interested in modeling. We found training to be more stable using the 1-Wasserstein distance between two distributions. We follow the setup of (Gulrajani et al., 2017), which we found leads to more effective and robust training.

We also explore training GANs with categorical latent variables using the InfoGAN framework (Chen et al., 2016).

6.3.3. Decoding Generated Sentence Vectors to Words

Given generated or real sentences in their fixed-length latent space, we would like to train a model that maps these vectors back to their respective sentences. To this end, we train a conditional GRU language model that, for each sentence x, conditions on the sentence vector h_x representation as well as previously generated words at each step to reconstruct x. We parameterize this decoder with an embedding lookup-table, a single layer GRU, and an affine transformation matrix. The GRU is fed the sentence vector at every time step as described in (Subramanian et al., 2018c). The decoder is trained with standard maximum-likelihood training.

Given that $E(x)$ is a deterministic function of $x \sim P_D$, where P_D is a discrete distribution in the case of textual data, the distribution $P(h_x)$ is a discrete distribution embedded in a continuous space. Our sentence embedding generator approximates the high-dimensional discrete distribution $P(h_x)$ with a continuous distribution $G(z)$. Therefore, samples from $G(z)$ will likely produce data points that are off-manifold, which may induce unexpected behavior of the decoder since it has been trained only on samples from the true distribution $P(h_x)$. In order to encourage better generalization for samples from $G(z)$, during the training of the decoder we smooth the distribution $P(h_x)$ by adding a small amount of additive isotropic Gaussian noise to each vector h_x.

In Table 6.1, we demonstrate the impact of noise on the reconstruction BLEU scores and sample quality. We notice that the amount of noise injected introduces a trade-off between reconstruction and sample quality. Specifically, as we increase the amount of noise the reconstruction BLEU score decreases but *samples* from a GAN start looking qualitatively better. This aligns with observations made in (Engel et al., 2017), where VAEs trained with a variance of ~ 0.1 had crisp reconstructions but poor sample quality, while those with a variance of 1.0 had blurry reconstructions but better sample quality.

Noise	BLEU-4	Samples
0.00	64.54	the young men are playing volleyball in the ball .
0.07	53.12	-
0.12	48.60	the young child is playing soccer .
0.17	41.16	-
0.20	37.51	a young child is playing with a ball .

Table 6.1. The impact of noise on *reconstruction* quality measured tokenized BLEU-4 scores on the SNLI (Bowman et al., 2015a) dataset. We also show samples from decoders trained with different amounts of noise but always conditioned on the **same** sentence vector generated from a GAN.

The injection of noise into neural models has been explored in many contexts (An, 1996; Vincent et al., 2008; Plappert et al., 2017; Srivastava et al., 2014; Poole et al., 2014) and has been shown to have important implications in unsupervised representation learning. Theoretical and empirical arguments from Denoising Autoencoders (DAEs) (Vincent et al., 2008) show that the addition of noise leads to robust features that are insensitive to small perturbations to examples on the data manifold. Contractive Autoencoders (CAEs) (Rifai et al., 2011) impose an explicit invariance of the hidden representations to small perturbations in the input by penalizing the norm of the Jacobian of hidden representations with respect to the inputs. This was shown to be equivalent to adding additive Gaussian noise to the hidden representations (Poole et al., 2014), where the variance of the noise is proportional to the contraction penalty in CAEs. Although this was proved to be true only for feedforward autoencoders, we believe this has a similar effect on sequential models such as the ones we use in this work.

6.3.4. Model Architecture

In all experiments, we use the sentence encoder from (Subramanian et al., 2018c). In our generator and discriminator, we use 5-layer MLPs with 1024 hidden dimensions and leaky ReLU activation functions. We use the WGAN-GP formulation (Gulrajani et al., 2017) in all experiments, with 5 discriminator updates for every generator update and a gradient penalty coefficient of 10. Our decoder architecture is identical to the multi-task decoders used in (Subramanian et al., 2018c). We trained all models with the Adam (Kingma & Ba, 2014) stochastic optimization algorithm with a learning rate of 2e-4 and $\beta_1 = 0.5$,$\beta_2 = 0.9$. We used a isotropic gaussian noise with a standard deviation of 0.12 for experiments involving SNLI and 0.20 for others.

6.4. Experiments & Results

To evaluate our generative model, we setup unconditional as well as conditional English *sentence* generation experiments. We consider the SNLI (Bowman et al., 2015a), BookCorpus (Zhu et al., 2015) and WMT15 (English fraction of the En-De parallel corpus) datasets in unconditional text generation experiments. For conditional generation experiments, we consider two different tasks: (1) to generate a premise sentence that satisfies a particular relationship (entailment/contradiction/neutral) with a given hypothesis sentence from the SNLI dataset, similar to (Shen et al., 2018; Kolesnyk et al., 2016); and (2) a synthetic task of generating captions (without the image) of a particular binary sentiment from the SentiCap dataset (Mathews et al., 2016).

Dataset	Sentences	Tokens
SNLI (Unconditional)	630k	6.1M
SNLI (Conditional)	550k	12.2M
BookCorpus	12M	159.5M
WMT15	4.5M	117.2M

Table 6.2. Dataset Statistics. Note: SNLI (unconditional) concatenates premise and hypothesis sentences and removes duplicates, while the conditional setting uses the dataset as is.

6.4.1. Unconditional Sentence Generation

In all settings, we partition the dataset into equally sized halves, one on which we train our generative model (GAN & Decoder) and the other for evaluation. The large holdout set is an artifact of our evaluation setup, described in the next section.

In Tables 6.3 and 6.4, we present samples and interpolations produced from our model on three datasets. Additionally, in Table 6.8 we present samples from the BookCorpus by varying a 10-dimensional categorical variable.

1	the room was nicely decorated and the two of them were very comfortable and the bathroom was fantastic .
2	all of the information was gathered from the police or the court of justice in the united states .
3	we are working with the elders to tell the story of the ancient egyptian stories of the past .
4	all of our doctors , nurses , and other health care providers have been waiting for me .
5	and this is why it is so important that the health care system be fully understood .
6	this is going to be a good way to get a glimpse of the new york city council .
7	“ what ’s going on with you ? ”
8	i shook my head , not trusting myself .
9	i was too tired to think about it .
10	“ yes , ” he said , nodding .
1	in the mid-1980s , he was appointed as a member of the court of human rights in afghanistan .
2	do you have any other ideas about cooperation between the european union and other countries in the world ?
3	secondly , i am not happy to see that the countries of the european union are in agreement .
4	the main objective of this study is to promote the development of a more comprehensive and accessible information society .
5	we’ve been looking forward to welcoming you to the beach , with a view of the sea .
6	but it is clear that the west and east of the country are not yet fully committed .
7	i would like to point out to the house that there are some amendments to the fisheries act .
8	health and education , research and development are a major factor in the development of health education programs .
9	i therefore ask the commission to cooperate fully with the commission and to parliament to approve this report .
10	i hope that the next step will be to ensure that this agreement is maintained in the eu .

Table 6.3. Generated samples from our model trained on the BookCorpus (top) and WMT15 (bottom)

In Table 6.5, we compare unconditional samples from ARAEs[1], a state-of-the-art LSTM language model[2] (Merity et al., 2017) and our model. We experimented with different hyperparameter configurations for the ARAE to increase model capacity but found that the default parameters gave us the best results. For (Merity et al., 2017), we use the default hyper-parameters without Averaged SGD since we noticed that it didn’t have an impact on results, therefore referring to this baseline as WD-LSTM.

Our model with beam search is competitive with the WD-LSTM at low temperatures (0.5) in terms of sample quality and diversity, but is able to maintain quality even at high

[1]`https://github.com/jakezhaojb/ARAE`
[2]`https://github.com/salesforce/awd-lstm-lm`

temperatures (1.0). We also outperform the ARAE on the benchmarks. All experiments were performed with a vocabulary size of 80,000 words and a maximum sequence length of 50, except for the ARAE model on SNLI where we used the pre-trained models provided in the code repository.

1	“ you ’re human , ” she said softly .	1	a girl on a stage holding a guitar .
2	“ you ’re a vampire , ” she said .	2	a girl on a stage holding a scythe .
3	“ you ’re a vampire , ” she said .	3	the girl in the sweat shirt plays a guitar on a stage .
4	“ you ’re a b**ch , ” she said .	4	the girl in the sweat vest is reading a newspaper .
5	“ you ’re angry , ” he said flatly .	5	a woman in a white shirt is taking a shortcut .
6	“ you ’re a jerk , ” he said dryly .	6	a woman in a white shirt is holding a scythe .
7	“ you ’re not going to let me out ? ”	7	a woman is playing the sax .
8	“ i do n’t know why you ’re here ? ”	8	a man is in the firehouse .
9	“ i do n’t know why you ’re here ? ”	9	a man is in the firehouse .
10	“ you do n’t know what to do ? ”	10	two men are in an enclosed room .

1	it is therefore quite impossible to separate the various provisions of the single market with regard to quotas .
2	i do not therefore think that it is necessary to continue with a different view of the commission .
3	i am therefore very sensitive to the issue of women in different member states and the social repercussions .
4	therefore , i am very pleased with the report on women and their social rights in the member states .
5	i am also very pleased to have the opportunity to discuss with the european parliament on this matter .
6	i would also like to take the opportunity to comment on the issue of the european economic partnership .
7	i would not like to mention the commission ’s statement on the issue of the council ’s statement .
8	i do not want to answer any of the commissioner ’s questions to the commission .
9	mr president , i am not going to reply to mr santer ’s statement on the lisbon strategy .
10	mr president .

Table 6.4. Linear interpolations along two randomly sampled points in the input space of the GAN generator on the BookCorpus (top left), SNLI (top right), and WMT15 (bottom). Points along the line between the two points are transformed into sentence vectors via the generator and then decoded with beam search.

For details on the evaluation methodology, see below.

6.4.2. Towards Quantitatively Evaluating Unconditional Sentence Samples

Evaluating implicit generative models such as GANs is still an active area of research, with several pieces of work focusing on evaluating image samples (Salimans et al., 2016; Heusel et al., 2017). In this section, we revisit the evaluation method that was originally proposed in (Goodfellow et al., 2014), of fitting a non-parametric kernel density estimator (KDE) on the samples produced by a GAN and then evaluating the likelihood of real examples under this KDE. As pointed out in (Theis et al., 2015), KDEs seldom do a good job of capturing the underlying density, since they do not scale well with data. However, when the underlying data distribution is discrete, count-based non-parametric models such as smoothed n-gram

Dataset	ARAE						WD-LSTM				Ours					
	FPPL			RPPL			FPPL		RPPL		FPPL			RPPL		
	0.5	1.0	B=1	0.5	1.0	B=1	0.5	1.0	0.5	1.0	0.5	1.0	B=5	0.5	1.0	B=5
BookCorpus	389.6	555.6	364.2	209.2	206.2	213.3	**9.4**	185.2	280.7	**137.2**	25.5	66.6	10.5	220.4	152.8	250.9
WMT15	448.7	965.1	385.8	476.2	378.7	626.3	21.4	369.0	528.9	**250.5**	105.5	212.9	**19.9**	350.5	254.1	373.2
SNLI	67.5	109.1	62.0	54.8	54.0	59.9	**5.9**	57.0	86.8	**34.5**	18.6	35.6	15.3	90.8	49.5	59.8

Table 6.5. Quantitative evaluation of sample quality from ARAE, WD-LSTM and our model. We report the FPPL and RPPL from a KN smoothed 5-gram language modeled trained on a distinct but large subset of the data. We also report FPPLs and RPPLs on samples generated with multinomial sampling with temperatures of 0.5 and 1.0, as well as deterministic decoding with beam search (B). Note that deterministic decoding is not suitable for the WD-LSTM since there needs to be some stochasticity to produce diverse samples.

language models (Heafield, 2011) are extremely fast to train and produce reasonable density estimates which can be used as weak proxies for the real and model data densities.

Dataset	BLEU-4			
	WD-LSTM		Ours	
	T=0.5	T=1.0	B=5	T=1.0
BookCorpus	26.8	6.8	47.7	17.92
WMT15	16.6	4.6	13.7	5.5
SNLI	45.8	14.7	34.8	22.1

Table 6.6. A preliminary analysis of overfitting by computing the average BLEU score between samples from the WD-LSTM and Our models with the nearest neighbor in the training set. Samples were generated with multinomial sampling with temperatues of 0.5 and 1.0 as well as decoding with beam search (B)

Approach	Unique n-gram ratio			N-gram validity		
	n=2	n=3	n=4	n=2	n=3	n=4
Ours (Beam)	0.04	0.05	0.07	95.76%	86.44%	65.35%
WD-LSTM (Temp 0.5)	0.05	0.07	0.12	95.22%	84.87%	61.68%
Ours (Temp 1.0)	0.55	0.56	0.57	63.06%	44.34%	26.14%
WD-LSTM (Temp 1.0)	1.38	1.30	1.29	46.20%	31.75%	17.52%

Table 6.7. N-gram statistics of the generated BookCorpus samples from our model and a WD-LSTM. The Unique n-gram ratio metric is the ratio of the number of unique n-grams in our generated samples to the real data with an equal number of sentences. The n-gram validity quantifies the percentage of n-grams in the generated samples that are present in a holdout set of 6M sentences.

Further, we identify a few simple statistics about generated samples to act as proxies for sample quality and diversity. In Table 6.7, we report statistics about the distribution

	Latent Category 1	**Latent Category 2**	**Latent Category 3**
1	she turned her head to look at him .	he did n't know how to stop him .	it was the most beautiful thing to do .
2	she had no idea what to do next .	he turned away from the window and left .	it was not a good thing to do .
3	but she knew it would be a lie .	he looked at her , his expression grim .	it had been a long time since then .
4	she did n't want him to hurt her .	he stopped and stared at the closed door .	it was n't fair to say that .
5	she asked , her eyes pleading with concern .	he looked at zane , who was smiling .	it was like a punch to the gut .
6	she wrapped her arms around him and squeezed .	he could n't help but smile at her .	it was a good idea to walk home .
7	her hand trembled slightly , her heart racing .	he wanted to know where he was going .	it was the strangest thing in the world .
8	she went to the bathroom to get dressed .	his voice was low , his eyes narrowing .	it was almost three o'clock in the morning .
9	she twisted her arms around his neck again .	he waited for a moment to regain consciousness .	it was a long way from the city .
	Latent Category 4	**Latent Category 5**	**Latent Category 6**
1	i thought i was in love with him .	no , not your father , he said .	oh , shit , i 'm so sorry !
2	i was in a hurry to find out .	i mean , you know what i mean .	come on , let 's go to bed .
3	i took a deep breath and exhaled slowly .	oh , yeah , well , i guess .	he asked , as if to say something .
4	i was pretty sure i was a fan .	look at me , i love you too .	she asked , her voice dripping with sarcasm .
5	i did n't mean to be with you .	i dont know , she said to herself .	she asked , her eyes wide with annoyance .
6	i heard the door slam shut behind me .	im sorry , she said with a sigh .	she asked , her eyes pleading with concern .
7	my heart was pounding , my chest hurt .	yeah , well , that 's all right .	oh , shit , she said to herself .
8	i got up and went to the kitchen .	when she did , she didnt say anything .	he said , his voice thick with emotion .
9	i swallowed hard and tried to sound stupid .	i mean , what are you doing here ?	yes , of course , i 'm sure .
	Latent Category 7	**Latent Category 8**	**Latent Category 9**
1	" that 's what you 're doing here .	" that 's right , " he said .	" i love you , my love . "
2	" it 's all you have to say .	" i 'm fine , " she said .	" i 'm not a good man . "
3	" i do n't know where to go .	" i do n't , " ashe said .	" i do n't think so . "
4	" i did n't mean to upset you .	" i 'll go , " he says .	" i 'm sure it 's a trick . "
5	" you 've got to be kidding me .	" you 're beautiful , " she said .	" she did n't tell him that ! "
6	" she 's not going to kill you .	" it 's not , " he said .	" that 's why i 'm here . "
7	" that 's not what i 'm saying .	" you 're not , " he said .	" i 'll take care of you . "
8	" i 'm so glad you 're here .	" she 's not , " ethan said .	" maybe you can do that again ? "
9	" you 're going to be a hero .	" it 's okay , " i say .	" i thought you might like that . "

Table 6.8. InfoGAN samples when varying a single categorical latent variable with 10 categories on the BookCorpus. The latent code captures significant factors of the text sentence such as the choice of pronoun (She/He/It/I), the presence of discourse markers (5), as well as other structures such as quotations (7,8,9). Different rows in each category correspond to different random samples of noise with the categorical latent variable fixed

of n-grams in the generated samples from our model as well as the WD-LSTM. We present the "unique n-gram ratio" metric as a proxy for sample diversity which measures the ratio of unique n-grams to the real data measured with the same number of sentences (512,000). The "n-gram validity" metric which measures the percentage of n-grams in the samples that occurs in a holdout set of 6M sentences, is intended to serve as a proxy for sample quality. We see that when generating at low temperatures or with beam search, both models have extremely low sample diversity with our model having better n-gram validity. When sampling at high temperatures (i.e.) from the true distribution learned by these models, we can see that there is a significant difference between the diversity of our samples versus the WD-LSTM

- the WD-LSTM actually contains more unique n-grams than the real data. Naturally, the n-gram validity metric captures this trade-off between sample quality and diversity of the two models.

A common criticism of GANs are that they appear to produce samples that are near copies of ones in the training set. We attempt to shed some light on the extent to which this happens in our model as well as the WD-LSTM. Specifically, we compute the tokenized BLEU-4 between samples and their nearest neighbor in the training set. We compute nearest neighbors using the FAISS[3] library in the space of averaged 300 dimensional GloVe vectors. The results appear to depend on the dataset, while our model overfits a lot more on the BookCorpus, it does less so on WMT15 and SNLI than the WD-LSTM at low temperatures or with beam search.

Let P be the data distribution and Q be our model's distribution. Since P is unknown and Q is an implicit generative model, we do not have access to either of the underlying data generating distributions, only samples from them. Nevertheless, we'd like to find an evaluation criterion that measures an (approximate) measure of similarity between these distributions. To do so, we fit a non-parametric Kneiser-Ney smoothed 5-gram language model (Heafield, 2011) to samples from P and Q and denote the resulting density models as $\hat{P}$ and $\hat{Q}$. Alternatively, we could fit a parameteric model such as an RNN language model to P and Q, but their behavior on examples that are off-manifold is not well characterized. Using these, we formulate two complementary evaluation criteria, the "Forward" and "Reverse" perplexities (FPPL, RPPL), that correspond to *approximations* of cross-entropies $H(Q,P)$ and $H(P,Q)$, respectively. It is straightforward to see that this is the case by substituting Q with $\hat{Q}$ $H(P,Q) \simeq - \mathop{\mathbb{E}}_{x\sim P} \log \hat{Q}(x)$ and P with $\hat{P}$ $H(Q,P) \simeq - \mathop{\mathbb{E}}_{x\sim Q} \log \hat{P}(x)$. We use the forward and reverse terminologies in the *opposite* way researchers refer to the forward and reverse KL divergences in an optimization setting when P is the true distribution we'd like to approximate with Q. This reversal is to maintain consistence with (Kim et al., 2017).

Computationally, the RPPL is equivalent to training a KN 5-gram LM on the samples from our model and reporting perplexities and the real data, while FPPL involves the opposite. Note that $H(P,Q)$ and $H(Q,P)$ differ in the order of arguments in the KL-divergence, with the latter being more sensitive to sample quality and the former to a balance between diversity and quality.

Finally, we also carried out human evaluations to compare samples from our model and the WD-LSTM in an A/B test. Annotators are presented with two samples - one from our model and one from the WD-LSTM (with the presentation order random each time) and asked to pick which they prefer along three dimensions: grammaticality, topicality, and overall quality. This protocol is identical to Fedus et al. (2018) who also evaluate

[3]`https://github.com/facebookresearch/faiss`

unconditional text generation samples. Annotators are allowed to rate two samples as having equal grammaticality and topicality but not overall quality. We collected a total of 1,094 annotations from 16 annotators. In Table 6.9, we report results for comparisons across the WMT and BookCorpus datasets with different sampling parameters: Temp=1.0 and Temp=0.5 vs. beam search. We compare our beam search variant with the WD-LSTM at Temp=0.5 since we found it to be the most similar to beam search in trading off RPPL and FPPL. Every entry in the table corresponds to the percentage of annotations where annotators preferred a sample from a particular model. Our model does consistently better at high temperatures, while being comparable to the WD-LSTM at low temperatures.

WMT (Temperature 1.0)	**Grammaticality**	**Topicality**	**Overall**
Ours (Temperature 1.0)	**46.19%**	**48.73%**	**63.95%**
WD-LSTM (Temperature 1.0)	28.90%	25.88%	36.05%
WMT (Beam Search vs Temperature 0.5)			
Ours (Beam Search)	**20.00%**	15.20%	**53.20%**
WD-LSTM (Temperature 0.5)	19.30%	**24.80%**	46.80%
BookCorpus (Temperature 1.0)			
Ours (Temperature 1.0)	**50.75%**	**54.27%**	**70.35%**
WD-LSTM (Temperature 1.0)	19.59%	23.61%	29.65%
BookCorpus (Beam Search vs Temperature 0.5)			
Ours (Beam Search)	22.68%	**35.29%**	**57.28%**
WD-LSTM (Temperature 0.5)	**25.21%**	17.64%	42.72%

Table 6.9. Results of human evaluation of unconditional samples produced by our system and WD-LSTMs at difference softmax temperatures on the WMT and BookCorpus datasets.

6.4.3. Conditional Text Generation

In most real world settings, we are interested in conditional rather than unconditional text generation. Conditional GANs (Mirza & Osindero, 2014) have proven extremely powerful in this context for generating images conditioned on certain discrete attributes. In this work, we explore the relatively simple and artificial task of generating sentences conditioned on binary sentiment labels from the SentiCap and Amazon Review datasets.

While true sentiment is certainly more nuanced, the binary setting can serve as a simple test-bed for initial experiments with such techniques. Quantitative and qualitative results are presented in Tables 6.12 and 6.13.

Using the SNLI dataset, we also explore the task of generating a premise sentence conditioned on a given hypothesis plus a label that specifies a relationship between them (Shen et al., 2018). We believe that transforming hypotheses into premises in SNLI is a harder problem than the inverse since it requires filling in extra details rather than removing them.

Label	Given Hypothesis	Generated Premise
E	the woman is very happy .	a lady wearing a blue white shirt is laughing .
C	no one is dancing .	a group of people playing guitar hero on a stage .
N	the man is reading the sportspage .	a man in a white shirt is sitting in a recliner .
E	a man is in a black shirt	a man in a black shirt stands in front of a store while a man in a blue hat and white shirt stands beside him .
C	an old woman has no jacket .	a woman with a white hat and jacket is playing with a girl in a red jacket .
N	a person is waiting for a train .	person in white and black hat standing in front of a train track .

Table 6.10. Samples from our GAN trained to conditionally transform a given hypothesis and a label as either (E)ntailment, (C)ontradiction or (N)eutral into a premise that satisfies the specified relationship

Method	Accuracy
Random	41.1%
Baseline-Seq2Seq (Mean)	59.6%
Baseline-Seq2Seq (MOSM)	62.6%
Shen et. al Mean (N=1)	62.4%
Shen et. al MOSM (N=1)	75.9%
Ours	70.8%

Table 6.11. NLI classification accuracies of generated samples on the test set evaluated by the ESIM model (Chen et al., 2017b). All results except ours were obtained from Shen et. al (Shen et al., 2018). Our model also had an FPPL of 15.01, which is comparable to results in Table 6.5

Dataset	Sentiment Classification Accuracy			FPPL		
	Positive	Negative	Overall	Positive	Negative	Overall
Senticap	67.65%	84.65%	76.15%	33.1	45.3	38.8
Amazon Review	78.29%	54.68%	66.48%	14.2	14.7	14.4

Table 6.12. Sentiment Classification Accuracies using a trained FastText classifier (Joulin et al., 2016a) and FPPLs on 3200 generated samples from the SentiCap and Amazon Review datasets

In both these sets of experiments, we use a conditional GAN with our generators and discriminators as MLPs that use conditioning information in their input layers. Conditioning information for sentiment and NLI labels is learned via a 128 dimensional embedding layer updated only by the discriminator. For premise generation, the MLPs are also presented with the sentence representation corresponding to the given hypothesis. We present quantitative evaluations of both models in Tables 6.11 and 6.12, evaluated using a combination of pre-trained classifiers and sample fluency evaluations using FPPL. On SNLI, we are able to outperform a baseline sequence-to-sequence model as well as Shen et al. (2018)'s simpler

Positive
i was very excited to read this book .
this is one of the best on the market .
if you are a fan of the music .
this book is one of the best things that i have ever read .
this is a very good product , and it is a very good way to not not do work .
a beautiful view of the beach .
a small crowd of people are walking on a beautiful beachfront
the girl is reading a newspaper in a nice , and white tiled room .
two little boys are having a good time with barbies
a happy woman standing in a large room filled with her computers
Negative
i bought this for my husband and had to replace it . the product is horrible .
i was so excited about this camera , but i was a little disappointed .
i would not recommend this book .
the product is a good product , but it did not work .
when i first got this item , i was very disappointed in the store .
a tired man with a machete
a wrinkled cat watching a dog .
a sad man is laying on the grass in front of a huge tree
a little girl is holding a dead bird in the water .
a dirty baby is jumping in the water

Table 6.13. Samples from our GAN trained to conditionally generate sentences of a specified binary sentiment label on the SentiCap (bottom) and Amazon Review (top) datasets.

model variant of averaging word embeddings to produce a sentence representation. Qualitative examples shown in in Table 6.10 and Supplementary Table 4 indicate that the model is able to capture some simple aspects of the mapping from sentiment or entailment labels to generated sentences/premises. We did however observe cases, that may be attributed to a form of mode collapse where the model adds in the same trivial details to the caption to satisfy entailment such as the color of one's shirt or other such adjectives.

6.5. Walking the Latent Space via Gradient-Based Optimization

We explore three different techniques to produce interpolations between sentences. The first, which is presented in Table 6.4, interpolates linearly in the *input noise space of the GAN generator*. The second and third techniques, which are presented in this section, interpolate between two *given* sentences in the *sentence encoder latent space* linearly or via gradient-based optimization. We show that the gradient-based method works on high-dimensional continuous representations of text that are more expressive than the Gaussian typically used in VAEs. We exploit the fact that we have continuous representations on which we can take gradient steps to iteratively transform one sentence into the other. We use our decoder that maps sentence representations into words to provide the gradient signal to move the sentence representation of the first sentence towards the second.

Specifically, given two sentences x_1 and x_2, we formulate the interpolation problem as an optimization that iteratively transforms x_1 into x_2 by taking gradient steps along h_{x1} in order to reconstruct x_2. We start the optimization process at $h_0 = h_{x_1}$ and take gradient steps as follows

$$h_t = h_{t-1} + \alpha \nabla_{h_{t-1}} \log P(x_2|h_{t-1})$$

$\log P(x_2|h_{t-1})$ is given by our decoder described in Section 6.3.3. At every step of the optimization process, we can run the sentence representation h_t through our decoder to produce an output sentence. Unlike linear interpolations, this procedure is not guaranteed to be symmetric, i.e., the interpolation path between x_1 and x_2 might not be the same as the path between x_2 and x_1. Sample interpolations using this technique are presented in Table 6.14.

6.6. Conclusion

We investigate and demonstrate the potential in leveraging general purpose sentence encoders that produce fixed-length sentence representations to train a two-step generative models of sentences. We show that it is possible to train conditional generative models of text that operate by manipulating and transforming sentences entirely in the latent space. We also show that smooth transitions are learned by our models in the observed space when moving linearly along the input space for the GAN generator. Finally, we qualitatively show that our gradient-based optimization technique to walk the latent space of sentences is better than linear interpolations in models when the latent distribution isn't simple like a Gaussian.

Start	her boyfriend eyed him curiously and gave him a cautious nod , then followed her into the bar , not waiting for a reply .
1	her boyfriend eyed him curiously , not waiting .
2	her boyfriend gave him a cautious nod .
3	her boyfriend never showed up at the bar .
4	grace gave him a cautious nod of approval .
5	grace never met her a boyfriend that night .
6	her boyfriend never showed up at the mall .
7	grace never showed up at work that night .
8	grace never showed up at her job in the mall .
9	grace never showed up at her job in the mall that night .
10	grace never showed up at her job in the mall that started at 5:00 that night .
End	grace never showed up at her job in the mall that started at 5:00 that night .
Start	naturally , he was upset when i said it was over .
1	naturally , he was upset when i said it was over .
2	he was upset when she said it .
3	naturally , he was upset when she said it .
4	naturally , she was upset when she said it .
5	no way , she said .
6	no way , she said , turning slightly .
7	no way , she said , turning her head slightly so she was nose to nose with him .
End	no way , she said , turning her head slightly so she was nose to nose with him .
Start	“ i ’ve been growing it out for a while .
1	“ i ’ve been growing out of it .
2	“ i ’ve been growing for a while .
3	“ i ’ve been listening for a while .
4	“ i ’ve been listening to a voice .
5	peter listen to me , you need it .
6	peter listen , you need to speak out .
7	peter listen , the voice didnt give him a chance to speak , you need to understand something .
End	peter listen , the voice didnt give him a chance to speak , you need to understand something .
Start	it ’s one of the lesser-known failings of the vampire .
1	it ’s one of the lesser-known failings of the vampire .
2	it ’s one of the failings of the vampire .
3	im sure it ’s one of the failings .
4	im sure it s one of their own .
5	im sure they got one of their own .
6	im sure theres a program of their own .
7	im sure they got a program of their own .
End	im sure cascadias got a program of their own , but it wouldnt be the same .

Table 6.14. Interpolations using gradient-based optimization on the BookCorpus.

Chapter 7

Prologue to the third article

Multiple Attribute Text Rewriting. Sandeep Subramanian*, Guillaume Lample*, Eric Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, Y-Lan Boureau (*denotes equal contribution). International Conference on Learning Representations 2019.

Personal Contribution The project began with discussions between Sandeep Subramanian, Guillaume Lample and Marc'Aurelio Ranzato preceeding Sandeep Subramanian's internship at Facebook AI Research. The goal of the work was to leverage advances in unsupervised neural machine translation to do text "style transfer". The authors met weekly to discuss datasets, modeling details and experiments to run. Sandeep Subramanian and Guillaume Lample were involved in writing scripts to obtain, pre-process and create a new large-scale multi-attribute text rewriting benchmark. They also collaboratively wrote all of the code for this project. Sandeep Subramanian ran all of the experiments on existing benchmarks and some of the experiments on the proposed datasets. Guillaume Lample ran most of the large-scale experiments on the larger Amazon and public social media content. Sandeep Subramanian also setup the human evaluation MTurk interface and ran all of the human evaluation experiments. Eric Smith ran code for prior work. Ludovic Denoyer, Marc'Aurelio Ranzato and Y-Lan Boureau advised on the project.

7.1. Context

The goal of this work was to explore the feasiblity of using techniques from unsupervised machine translation for text "style transfer". Along the way, we found that existing datasets were inadequate because they were too simple with sentences of only up to 12 words, contained only a single controllable attribute and were an order of magnitude or two smaller than machine translation datasets. We also found that domain adversarially trained autoencoders 1) perform better than reported in previous work 2) surprisingly, do not achieve disentanglement in practice.

7.2. Contributions

The work made three main contributions 1) demonstrating that methods which rely on disentanglement of content and style using domain adversarial training do not achieve disentanglement in practice 2) building a model that controls multiple attributes and does not rely on disentanglement and instead uses building blocks from unsupervised machine translation to achieve better results 3) presenting a new large-scale, more realistic benchmark for text rewriting.

7.3. Research Impact

The work was another example of the success of backtranslation methods in NLP. Although there are a few follow-up papers that adopt the online backtranslation approach presented here (Zheng et al., 2020), there still exist several pieces of work that use external rewards trained with policy gradient methods and latent space refinement (Dathathri et al., 2019). The adoption of our proposed large-scale benchmarks however hasn't been as widespread as we had hoped.

Chapter 8

Multiple-Attribute Text Rewriting

8.1. Introduction

One of the objectives of unsupervised learning is to learn representations of data that enable fine control over the underlying latent factors of variation, e.g., pose and viewpoint of objects in images, or writer style and sentiment of a product review. In conditional generative modeling, these latent factors are given (Sohn et al., 2015; Mirza & Osindero, 2014; Ficler & Goldberg, 2017), or automatically inferred via observation of samples from the data distribution (Chen et al., 2017a, 2016; Higgins et al., 2017).

More recently, several studies have focused on learning unsupervised mappings between two data domains such as images (Taigman et al., 2016; Isola et al., 2017; Zhu et al., 2017), words or sentences from different languages (Conneau et al., 2017b; Lample et al., 2018).

In this problem setting, the generative model is conditioned not only on the desired attribute values, but also on a initial input, which it must transform. Generations should retain as many of the original input characteristics as possible, provided the attribute constraint is not violated. This learning task is typically unsupervised because no example of an input and its corresponding output with the specified attribute is available during training. The model only sees random examples and their attribute values.

The dominant approach to learn such a mapping in text is via an explicit constraint on disentanglement (Hu et al., 2017; Fu et al., 2017; Shen et al., 2017b): the learned representation should be invariant to the specified attribute, and retain only attribute-agnostic information about the "content". Changing the style of an input at test time then amounts to generating an output based on the disentangled latent representation computed from the input and the desired attributes. Disentanglement is often achieved through an adversarial term in the training objective that aims at making the attribute value unrecoverable from the latent representation.[1]

[1]Datasets and code for this work are available at `https://git.io/Je3RV`

Relaxed ↔ Annoyed	
Relaxed	Sitting by the Christmas tree and watching Star Wars after cooking dinner. What a nice night 💗 🌲 ✨
Annoyed	Sitting by the computer and watching The Voice for the second time tonight. What a horrible way to start the weekend 😡 😡 😡
Annoyed	Getting a speeding ticket 50 feet in front of work is not how I wanted to start this month 😒
Relaxed	Getting a haircut followed by a cold foot massage in the morning is how I wanted to start this month 😊
Male ↔ Female	
Male	Gotta say that beard makes you look like a Viking...
Female	Gotta say that hair makes you look like a Mermaid...
Female	Awww he's so gorgeous 😍 can't wait for a cuddle. Well done 😘 xxx
Male	Bro he's so f***ing dope can't wait for a cuddle. Well done bro
Age 18-24 ↔ 65+	
18-24	You cheated on me but now I know nothing about loyalty 😂 ok
65+	You cheated on America but now I know nothing about patriotism. So ok.
65+	Ah! Sweet photo of the sisters. So happy to see them together today .
18-24	Ah 😂 Thankyou 💞 #sisters ❤️ happy to see them together today

Table 8.1. Our approach can be applied to many different domains beyond sentiment flipping, as illustrated here with example re-writes by our model on public social media content. The first line in each box is an input provided to the model with the original attribute, followed by its rewrite when given a different attribute value.

This paper aims to extend previous studies on "style transfer" along three axes. (i) First, we seek to gain a better understanding of what is necessary to make things work, and in particular, whether disentanglement is key, or even actually achieved by an adversarial loss in practice. In Sec. 8.3.1 we provide strong empirical evidence that disentanglement is not necessary to enable control over the factors of variation, and that even a method using adversarial loss to disentangle (Fu et al., 2017) does not actually learn representations that are disentangled. (ii) Second, we introduce a model which replaces the adversarial term with a back-translation (Sennrich et al., 2015a) objective which exposes the model to a pseudo-supervised setting, where the model's outputs act as supervised training data for the ultimate task at hand. The resulting model is similar to recently proposed methods for unsupervised machine translation (Lample et al., 2017a, 2018; Artetxe et al., 2018; Zhang et al., 2018b), but with two major differences: (a) we use a pooling operator which is used to control the trade-off between style transfer and content preservation; and (b) we extend this model to support multiple attribute control. (iii) Finally, in Sec. 10.4 we point out that current style transfer benchmarks based on collections of user reviews have severe limitations, as they only consider a single attribute control (sentiment), and very small sentences in isolation with noisy labels. To address this issue, we propose a new set of benchmarks based on existing review datasets, which comprise full reviews, where multiple attributes are extracted from each review. Note that using "gender" (or any other attribute for that matter) as a differentiating attribute between several bodies of text implies that there are indeed signatures of gender in the data. These signatures could be as innocuous as some first names like Mary being usually associated with women, or disheartening like biases and stereotypes exposed by statistical methods, (e.g., "man is to computer programmer as woman is to homemaker" (Bolukbasi et al., 2016)). We certainly do not condone those stereotypes, and on the contrary, we hope that showing that our models can uncover these biases might down the line turn them into powerful tools for researchers who study fairness and debiasing (Reddy & Knight, 2016).

The contributions of this paper are thus: (1) a deeper understanding of the necessary components of style transfer through extensive experiments, resulting in (2) a generic and simple learning framework based on mixing a denoising auto-encoding loss with an online back-translation technique and a novel neural architecture combining a pooling operator and support for multiple attributes, and (3) a new, more challenging and realistic version of existing benchmarks which uses full reviews and multiple attributes per review, as well as a comparison of our approach w.r.t. baselines using both new metrics and human evaluations. We will open-source our code and release the new benchmark datasets used in this work, as well as our pre-trained classifiers and language models for reproducibility. This will also enable fair empirical comparisons on automatic evaluation metrics in future work on this problem.

8.2. Related Work

There is substantial literature on the task of unsupervised image translation. While initial approaches required supervised data of the form *(input, transformation, output)*, e.g., different images of the same object rendered with different viewpoints or/and different lighting conditions (Hinton et al., 2011; Yang et al., 2015; Kulkarni et al., 2015), current techniques are capable of learning completely unsupervised domain mappings. Given images from two different domains $\mathcal{X}$ and $\mathcal{Y}$ (where $\mathcal{X}$ could be the domain of paintings and $\mathcal{Y}$ the domain of realistic photographs), and the task is to learn two mappings $F : \mathcal{X} \to \mathcal{Y}$ and $G : \mathcal{Y} \to \mathcal{X}$, without supervision, i.e., just based on images sampled from the two domains (Liu & Tuzel, 2016; Taigman et al., 2016; Isola et al., 2017). For instance, Zhu et al. (2017) used a cycle consistency loss to enforce $F(G(y)) \approx y$ and $G(F(x)) \approx x$. This loss is minimized along with an adversarial loss on the generated outputs to constrain the model to generate realistic images. In Fader Networks (Lample et al., 2017b), a discriminator is applied on the latent representation of an image autoencoder to remove the information about specific attributes. The attribute values are instead given explicitly to the decoder at training time, and can be tuned at inference to generate different realistic versions of an input image with varying attribute values.

Different approaches have been proposed for textual data, mainly aiming at controlling the writing style of sentences. Unfortunately, datasets of parallel sentences written in a different style are hard to come by. (Carlson et al., 2017) collected a dataset of 33 English versions of the Bible written in different styles on which they trained a supervised style transfer model. Li et al. (2018) released a small crowdsourced subset of 1,000 Yelp reviews for evaluation purposes, where the sentiment had been swapped (between *positive* and *negative*) while preserving the content. Controlled text generation from unsupervised data is thus the focus of more and more research.

An theme that is common to most recent studies is that style transfer can be achieved by disentangling sentence representations in a shared latent space. Most solutions use an adversarial approach to learn latent representations agnostic to the style of input sentences (Fu et al., 2017; Hu et al., 2017; Shen et al., 2017b; Zhang et al., 2018a; Xu et al., 2018; John et al., 2018; Zhao et al., 2018). A decoder is then fed with the latent representation along with attribute labels to generate a variation of the input sentence with different attributes.

Unfortunately, the discrete nature of the sentence generation process makes it difficult to apply to text techniques such as cycle consistency or adversarial training. For instance, the latter (Shen et al., 2017b; dos Santos et al., 2018; Zhang et al., 2018c) requires methods such as REINFORCE (He et al., 2016a) or approximating the output softmax layer with a tunable temperature (Hu et al., 2017; Prabhumoye et al., 2018; Yang et al., 2018), all of which tend to be slow, unstable and hard to tune in practice. Moreover, all these studies control a single attribute (e.g. swapping positive and negative sentiment).

The most relevant work to ours is Zhang et al. (2018b), which also builds on recent advances in unsupervised machine translation. Their approach first consists of learning cross-domain word embeddings in order to build an initial phrase-table. They use this phrase-table to bootstrap an iterative back-translation pipeline containing both phrase-based and neural machine translation systems. Overall, their approach is significantly more complicated than ours, which is end-to-end and does not require any pre-training. Moreover, this iterative back-translation approach has been shown to be less effective than on-the-fly back-translation which is end-to-end trainable (Lample et al., 2018).

8.3. Controllable Text Rewriting

This section briefly introduces notation, the task, and our empirical procedure for evaluating disentanglement before presenting our approach.

We consider a training set $\mathcal{D} = (x^i, y^i)_{i \in [1,n]}$ of n sentences $x^i \in \mathcal{X}$ paired with attribute values y^i. $y \in \mathcal{Y}$ is a set of m attribute values $y = (y_1, ..., y_m)$. Each attribute value y_k is a discrete value in the set $\mathcal{Y}_k$ of possible values for attribute k, e.g. $\mathcal{Y}_k = \{\text{bad}, \text{neutral}, \text{good}\}$ if y_k represents the overall rating of a restaurant review.

Our task is to learn a model $F : \mathcal{X} \times \mathcal{Y} \to \mathcal{X}$ that maps any pair $(x, \tilde{y})$ of an input sentence x (whose actual set of attributes are y) and a new set of m attribute values $\tilde{y}$ to a new sentence $\tilde{x}$ that has the specified attribute values $\tilde{y}$, subject to retaining as much as possible of the original content from x, where content is defined as anything in x which does not depend on the attributes.

The architecture we consider performs this mapping through a sequence-to-sequence auto-encoder that first encodes x into a latent representation $z = e(x)$, then decodes $(z, \tilde{y})$ into $\tilde{x} = d(z, \tilde{y})$, where e and d are functions parameterized by the vector of trainable parameters θ. Before giving more detail on the architecture, let us look at disentanglement.

8.3.1. Are Adversarial Models really Doing Disentanglement?

Almost all the existing methods are based on the common idea to learn a latent representation z that is disentangled from y. We consider z to be disentangled from y if it is impossible to recover y from z. While failure to recover y from z could mean either that z was disentangled or that the classifier chosen to recover y was either not powerful enough or poorly trained, success of *any* classifier in recovering y demonstrates that z was in fact not invariant to y.

λ_{adv}	Discriminator Acc (Train)	Post-fit Classifier Acc (Test)
0	89.45%	93.8%
0.001	85.04%	92.6%
0.01	75.47%	91.3%
0.03	61.16%	93.5%
0.1	57.63%	94.5%
1.0	52.75%	86.1%
10	51.89%	85.2%
fastText	-	97.7%

Table 8.2. Recovering the sentiment of the input from the encoder's representations of a domain adversarially-trained Fader model (Fu et al., 2017). During training, the discriminator, which was trained adversarially and jointly with the model, gets worse at predicting the sentiment of the input when the coefficient of the adversarial loss λ_{adv} increases. However, a classifier that is separately trained on the resulting encoder representations has an easy time recovering the sentiment. We also report the baseline accuracy of a fastText classifier trained on the actual inputs.

As a preliminary study, we gauge the degree of disentanglement of the latent representation. Table 8.2 shows that the value of the attribute can be well recovered from the latent representation of a Fader-like (Fu et al., 2017) model even when the model is trained adversarially. A classifier fit post-hoc and trained from scratch, parameterized identically to the discriminator(see paragraph on model architecture in Section 8.3.3 for details), is able to recover attribute information from the "distengeled" content representation learned via adversarial training. This suggests that disentanglement may not be achieved in practice, even though the discriminator is unable to recover attribute information well during training. We do not assert that disentangled representations are undesirable but simply that it isn't mandatory in the goal of controllable text rewriting. This is our focus in the following sections.

8.3.2. Our Approach

Evaluation of controlled text generation can inform the design of a more streamlined approach: generated sentences should (1) be fluent, (2) make use of the specified attribute values, and (3) preserve the rest of the content of the input.

Denoising auto-encoding (DAE) Fu et al. (2017) is a natural way to learn a generator that is both fluent and that can reconstruct the input, both the content and the attributes. Moreover, DAE is a weak way to learn about how to change the style, or in other words, it is a way to force the decoder to also leverage the externally provided attribute information. Since the noise applied to the encoder input x may corrupt words conveying the values of the input attribute y, the decoder has to learn to use the additional attribute input values in order to perform a better reconstruction. We use the noise function described in Lample et al. (2017a) that corrupts the input sentence by performing word drops and word order shuffling. We denote by x_c a corrupted version of the sentence x.

As discussed in Sec. 8.3.1, disentanglement is not necessary nor easily achievable, and therefore, we do not seek disentanglement and do not include any adversarial term in the loss. Instead, we consider a more natural constraint which encourages the model to perform well at the task we are ultimately interested in - controlled generation via externally provided attributes. We take an input (x,y) and encode x it into z, but then decode using another set of attribute values, $\tilde{y}$, yielding the reconstruction $\tilde{x}$. We now use $\tilde{x}$ as input of the encoder and decode it using the original y to ideally obtain the original x, and we train the model to map $(\tilde{x}, y)$ into x. This technique, called back-translation (BT) (Sennrich et al., 2015a; Lample et al., 2017a, 2018; Artetxe et al., 2018), has a two-fold benefit. Initially when the DAE is not well trained and $\tilde{x}$ has lost most of the content present in x, the only useful information provided to the decoder is the desired attribute y. This encourages the decoder to leverage the provided attributes. Later on during training when DAE is better, BT helps training the sequence-to-sequence for the desired task. Overall, we minimize:

$$\mathcal{L} = \lambda_{AE} \sum_{(x,y)\sim\mathcal{D}} -\log p_d\Big(x|e(x_c), y\Big) + \lambda_{BT} \sum_{(x,y)\sim\mathcal{D},\tilde{y}\sim\mathcal{Y}} -\log p_d\bigg(x|e\Big(d\big(e(x),\tilde{y}\big)\Big), y\bigg) \tag{8.1}$$

where p_d is the probability distribution over sequences x induced by the decoder, $e(x_c)$ is the encoder output when fed with a corrupted version x_c of the input x, and $d(e(x), \tilde{y})$ is a variation of the input sentence x written with a randomly sampled set of attributes $\tilde{y}$. In practice, we generate sentences during back-translation by sampling from the multinomial distribution over words defined by the decoder at each time step using a temperature T.

8.3.3. Implementation

So far, the model is the same as the model used for unsupervised machine translation by Lample et al. (2018), albeit with a different interpretation of its inner workings, no longer based on disentanglement. Instead, the latent representation z can very well be entangled, but we only require the decoder to eventually "overwrite" the original attribute information with the desired attributes. Unfortunately, this system may be limited to swapping a single binary attribute and may not give us enough control on the trade-off between content preservation and change of attributes. To address this limitations, we introduce the following components:

Attribute conditioning. In order to handle multiple attributes, we separately embed each target attribute value and then average their embeddings. We then feed the averaged embeddings to the decoder as a start-of-sequence symbol.

We also tried an approach similar to Michel & Neubig (2018), where the output layer of the decoder uses a different bias for each attribute label. We observed that the learned biases tend to reflect the labels of the attributes they represent. Examples of learned biases can be found in Table 8.14. However, this approach alone did not work as well as using attribute-specific start symbols, nor did it improve results when combined with them.

Latent representation pooling. To control the amount of content preservation, we use pooling. The motivating observation is that models that compute one latent vector representation per input word usually perform individual word replacement, while models without attention are much less literal and tend to lose content but have an easier time changing the input sentence with the desired set of attributes. Therefore, we propose to gain finer control by adding a temporal max-pooling layer on top of the encoder, with non-overlapping windows of width w. Setting $w = 1$ results in a standard model with attention, while setting w to the length of the input sequence boils down to a sequence-to-sequence model without attention. Intermediate values of w allow for different trade-offs between preserving information about the input sentence and making the decoder less prone to copying words one by one.

The hyper-parameters of our model are: λ_{AE} and λ_{BT} trading off the denoising auto-encoder term versus the back-translation term (the smaller the $\lambda_{BT}/\lambda_{AE}$ ratio the more the content is preserved and the less well the attributes are swapped), the temperature T used to produce unbiased generations (Edunov et al., 2018) and to control the amount of content preservation, and the pooling window size w. We optimize this loss by stochastic gradient descent without back-propagating through the back-translation generation process; back-translated sentences are generated on-the-fly once a new mini-batch arrives.

Model Architecture. We use an encoder parameterized by a 2-layer bidirectional LSTM and a 2-layer decoder LSTM augmented with an attention mechanism (Bahdanau et al., 2014). Both LSTMs and our word embedding lookup tables, trained from scratch, have 512 hidden units. Another embedding lookup table with 512 hidden units is used to embed each attribute value. The decoder conditions on two different sources of information: 1) attribute embedding information that presented that it as the first token, similar to (Lample et al., 2018) and at the softmax output as an attribute conditional bias following (Michel & Neubig, 2018). When controlling multiple attributes, we average the embeddings and bias vectors that correspond to the different attribute values. 2) The decoder also conditions on a temporally downsampled representation of the encoder via an attention mechanism. The representations are downsampled by temporal max-pooling with a non-overlapping window of size 5. Although our best models do not use adversarial training, in ablations and experiments that study disentanglement, we used a discriminator paramaeterized as 3 layer MLP with 128 hidden units and LeakyReLU activations.

8.4. Experiments

8.4.1. Datasets

We use data from publicly available Yelp restaurant and Amazon product reviews following previous work in the area (Shen et al., 2017b; Li et al., 2018) and build on them in three ways to make the task more challenging and realistic. Firstly, while previous approaches operate at the sentence level by assuming that every sentence of a review carries the same sentiment as the whole of review, we operate at the granularity of entire reviews. The sentiment, noisy estimate of the gender of the author and product/restaurant labels are therefore more reliable. Secondly, we relax constraints enforced in prior works that discard reviews with more than 15 words and only consider the 10k most frequent words. In our case, we consider full reviews with up to 100 words, and we consider byte-pair encodings (BPE) (Sennrich et al., 2015b) with 60k BPE codes, eliminating the presence of unknown words. Finally, we leverage available meta-data about restaurant and product categories to collect annotations for two additional controllable factors: the gender of the review author and the category of the product or restaurant being reviewed. A small overview of the corresponding datasets is presented below with some statistics presented in Table 10.1. Following (Li et al., 2018), we also collect human reference edits for sentiment and restaurant/product categories to serve as a reference for automatic metrics as well as an upper bound on human evaluations (examples in Appendix Table 8.12).

Yelp Reviews: This dataset consists of restaurant and business reviews provided by the Yelp Dataset Challenge[2]. We pre-process this data to remove reviews that are either 1) not written in English according to a fastText (Joulin et al., 2016b) classifier, 2) not about restaurants, 3) rated 3/5 stars as they tend to be neutral in sentiment (following (Shen et al., 2017b)), or 4) where the gender is not identifiable by the same method as in (Reddy & Knight, 2016; Prabhumoye et al., 2018). We then binarize both sentiment and gender labels. The process of binarization of gender here is not inclusive to (a) non-binary individuals and (b) those whose names aren't in the US census data. An ideal situation would be where individuals are able to self-report gender, but such information is typically unavailable in most data sources. Coarse-grained restaurant category labels, *Asian, American, Mexican, Bars & Dessert*, are obtained from the associated meta-data. Since a review can be written about a restaurant that has multiple categories (ex: an Asian restaurant that serves desserts), we train a multi-label fastText classifier to the original data that has multiple labels per example. We then re-label the entire dataset with this classifier to pick the most likely category to be able to model the category factor as a categorical random variable. (See Appendix section 8.7 for more details.) Since there now exists two variants of the Yelp dataset, we refer to the one used by previous work (Shen et al., 2017b; Fu et al., 2017; Li et al., 2018) as *SYelp* and our created version with full reviews along with gender and category information as *FYelp* henceforth.

Amazon Reviews: The amazon product review dataset (He & McAuley, 2016) is comprised of reviews written by consumers of Amazon products. We followed the same pre-processing steps as in the Yelp dataset with the exception of collecting gender labels, since a very large fraction of amazon usernames were not present in a list of gender-annotated names. We labeled reviews with the following product categories based on the meta-data: *Books*, *Clothing*, *Electronics*, *Movies*, *Music*. We followed the same protocol as in *FYelp* to re-label product categories. In this work, we do not experiment with the version of the Amazon dataset used by previous work, and so we refer to our created version with full reviews along with product category information as just *Amazon* henceforth. Statistics about the dataset can be found in Table 10.1.

Public social media content: We also used an unreleased dataset of public social media content written by English speakers to illustrate the approach with examples from a more diverse set of categories. We used 3 independent pieces of available information about that content: 1) gender (male or female) 2) age group (18-24 or 65+), and 3) writer-annotated feeling (relaxed or annoyed). To make the data less noisy, we trained a fastText classifier (Joulin et al., 2016b) for each attribute and only kept the data above a certain confidence threshold.

[2]`https://www.yelp.com/dataset/challenge`

	Sentiment		Gender		Category				
SYelp	Positive	Negative	Male	Female	American	Asian	Bar	Dessert	Mexican
	266,041	177,218	-	-	-	-	-	-	-
FYelp	Positive	Negative	Male	Female	American	Asian	Bar	Dessert	Mexican
	2,056,132	639,272	1,218,068	1,477,336	904,026	518,370	595,681	431,225	246,102
Amazon	Positive	Negative	-	-	Book	Clothing	Electronics	Movies	Music
	64,251,073	10,944,310	-	-	26,208,872	14,192,554	25,894,877	4,324,913	4,574,167
Social Media Content	Relaxed	Annoyed	Male	Female	18-24	65+			
	7,682,688	17,823,468	14,501,958	18,463,789	12,628,250	7,629,505			

Table 8.3. The number of reviews for each attribute for different datasets. The *SYelp*, *FYelp* and the Amazon datasets are composed of 443k, 2.7M and 75.2M sentences respectively. Public social media content is collected from 3 different data sources with 25.5M, 33.0M and 20.2M sentences for the *Feeling*, *Gender* and *Age* attributes respectively.

8.4.2. Evaluation

Automatic evaluation of generative models of text is still an open research problem. In this work, we use a combination of multiple automatic evaluation criteria informed by our desiderata. We would like our systems to *simultaneously* 1) produce sentences that conform to the set of pre-specified attribute(s), 2) preserve the structure and content of the input, and 3) generate fluent language. We therefore evaluate samples from different models along three different dimensions:

- **Attribute control:** We measure the extent to which attributes are controlled using fastText classifiers, trained on our datasets, to predict different attributes.
- **Fluency:** Fluency is measured by the perplexity assigned to generated text sequences by a pre-trained Kneser–Ney smooth 5-gram language model using KenLM (Heafield, 2011).
- **Content preservation:** We measure the extent to which a model preserves the content present of a given input using n-gram statistics, by measuring the *BLEU* score between generated text and the input itself, which we refer to as *self-BLEU*. When a human reference is provided instead, we compute the BLEU score with respect to it, instead of the input, which we will refer to as just *BLEU* (Papineni et al., 2002). "BLEU" scores in this paper correspond to the BLEU score with respect to human references *averaged* across generations conditioned on all possible attribute values *except for that of the input*. However, when reporting self-BLEU scores, we also average across cases where generations are also conditioned on the same attribute value as the input.

A combination of these metrics, however, only provides a rough understanding of the quality of a particular model. Ultimately, we rely on human evaluations collected via a public crowd-sourcing platform. We carried out two types of evaluations to compare different models. 1) Following a protocol similar to (Li et al., 2018), we ask crowd workers to annotate generated sentences along the three dimensions above. Fluency and content preservation are measured on a likert-scale from 1 to 5 and attribute control is evaluated by asking the worker to predict the attribute present in the generated text. 2) We take a pair of generations from two different models, and ask workers to pick the generation they prefer on the overall task, accounting for all the dimensions simultaneously. They are also presented with a "no preference" option to discard equally good or bad generations from both models.

8.4.3. Model selection

Since our automatic evaluation metrics are only weak proxies for the quality of a model, we set minimum thresholds on the content preservation and attribute control criteria and only consider models above a certain threshold.

The few models that met the specified threshold on the validation set were evaluated by humans on the same validation set and the best model was selected to be run on the test set.

8.4.4. Comparisons to Prior Work

Our first set of experiments aims at comparing our approach with different models recently proposed, on the *SYelp* dataset. Results using automatic metrics are presented in Table 8.4. We compare the same set of models as in (Li et al., 2018) with the addition of our model and our own implementation of the Fader network (Lample et al., 2017b), which corresponds to our model without back-translation and without attention mechanism, but uses domain adversarial training (Ganin et al., 2016) to remove information about sentiment from the encoder's representation. This is also similar to the StyleEmbedding model presented by (Fu et al., 2017). For our approach, we were able to control the trade-off between BLEU and accuracy based on different hyper-parameter choices. We demonstrate that our approach is able to outperform all previous approaches on the three desired criteria simultaneously, while our implementation of the fader is competitive with the previous best work.

Since our automatic evaluation metrics are not ideal, we carried out human evaluations using the protocol described in Section 8.4.2. Table 8.5 (top) shows the fluency, content preservation and attribute control (sentiment) scores obtained by our model, DeleteAndRetrieve (DAR) and turkers from (Li et al., 2018) [3] on the *SYelp* dataset. While humans clearly outperform both models, our model is better than DeleteAndRetrieve on all 3 dimensions. We further demonstrate our model's strength over DeleteAndRetrieve in Table 8.5

[3]`https://github.com/lijuncen/Sentiment-and-Style-Transfer/tree/master/evaluation/outputs/yelp`

Model	Accuracy (↑)	BLEU (↑)	PPL (↓)
Fader/StyleEmbedding (Fu et al., 2017)	18%	16.7	56.1
MultiDecoder (Fu et al., 2017)	52%	11.3	90.1
ControllableText (Hu et al., 2017)	85%	20.6	232.0
CAE (Shen et al., 2017b)	72%	6.8	53.0
Retrieval (Li et al., 2018)	81%	1.3	**7.4**
Rule-based (Li et al., 2018)	73%	22.3	118.7
DeleteOnly (Li et al., 2018)	77%	14.5	67.1
DeleteAndRetrieve (Li et al., 2018)	79%	16.0	66.6
Fader (Ours w/o backtranslation & attention)	71%	15.7	35.1
Ours	**87%**	14.6	26.2
Ours	85%	**24.2**	26.5
Ours	74%	31.2	49.8
Input copy	13%	30.6	40.6

Table 8.4. Automatic evaluation of models on the *SYelp* test set from (Li et al., 2018). The test set is composed of sentences that have been manually written by humans, which we use to compute the BLEU score. Samples for previous models were made available by (Li et al., 2018). For our model, we report different results corresponding to different choices of hyper-parameters (pooling kernel width and back-translation temperature) to demonstrate our model's ability to control the trade-off between attribute transfer and content preservation.

(bottom) in an A/B test between two the models, where crowd workers prefer our model 54.4% compared to theirs 20.8%. Interestingly, our baseline Fader model is also able to do better (37.6% vs 29.7%), suggesting limitations in our automatic metrics, since Fader does not do as well in Table 8.4.

8.4.5. Evaluating Multiple Attribute Control

Table 8.6 presents the quantitative results obtained by the *FYelp* and *Amazon* datasets when controlling single and multiple attributes. For this table, unlike for Table 8.4, the DeleteAndRetrieve (DAR) results were obtained by re-training the model of (Li et al., 2018). We use our implementation of the Fader model since we found it to be better than previous work, by human evaluation (Table 8.5). While we control all attributes simultaneously during training, at test time, for the sake of quantitative evaluations, we change the values only of a single attribute while keeping the others constant. Our model clearly outperforms

	Fluency	Content	Sentiment
DAR ((Li et al., 2018))	3.33 (1.39)	3.16 (1.43)	64.05%
Ours	4.07 (1.12)	3.67 (1.41)	69.66%
Human ((Li et al., 2018))	4.56 (0.78)	4.01 (1.25)	81.35%
	Our Model	**No Preference**	**DAR**
DAR vs Our Fader	**37.6%**	32.7%	29.7%
DAR vs Ours	**54.4%**	24.7%	20.8%

Table 8.5. Top: Results from human evaluation to evaluate the fluency / content preservation and successful sentiment control on the (Li et al., 2018) *SYelp* test set. The mean and standard deviation of Fluency and Content are measured on a likert scale from 1-5 while sentiment is measured by fraction of times that the controlled sentiment of model matches the judge's evaluation of the sentiment (when also presented with a neutral option). **Bottom:** Results from human A/B testing of different pairs of models. Each cell indicates the fraction of times that a judge preferred one of the models or neither of them on the overall task.)

Dataset (Model)	Attributes	Sentiment		Category		Gender	
		Accuracy (↑)	self-BLEU (↑)	Accuracy (↑)	self-BLEU (↑)	Accuracy (↑)	self-BLEU (↑)
Yelp (DAR)	Sentiment	78.7%	42.1	-	-	-	-
Yelp (Our Fader)	Sentiment	85.5%	31.3	-	-	-	-
	Sentiment + Category	85.1%	20.6	46.1%	22.6	-	-
	Sentiment + Category + Gender	86.6%	20.4	47.7%	22.5	58.5%	23.3
Yelp (Ours)	Sentiment	87.4%	54.5	-	-	-	-
	Sentiment + Category	87.1%	38.8	64.9%	44.0	-	-
	Sentiment + Category + Gender	88.5%	31.6	64.1%	36.5	59.0%	37.4
	Gender	-	-	-	-	59.1%	47.0
Amazon (Ours)	Sentiment	82.6%	54.8	-	-	-	-
	Sentiment + Category	82.5%	48.9	81.4%	41.8	-	-
Input Copy	-	50.0%	100.0	20.0%	100.0	50.0%	100.0

Table 8.6. Results using automatic evaluation metrics on the *FYelp* and *Amazon* test sets. Different rows correspond to the set of attributes being controlled by the model.

the baseline Fader model. We also demonstrate that our model does not suffer significant drops in performance when controlling multiple attributes over a single one.

Demonstrations of our model's ability to control single and multiple attributes are presented in Table 8.8 and Table 8.9 respectively. What is interesting to observe is that our model does not just alter single words in the input to control an attribute, but often changes larger fragments to maintain grammaticality and fluency. Examples of re-writes by our model on social media content in Table 8.1 show that our model tends to retain the overall structure of input sentences, including punctuation and emojis. Additional examples of re-writes can be found in Table 8.10 and Table 8.11 in Appendix.

8.4.6. Ablation study

Model	Test (FYelp)		Test (Li et al., 2018)	
	Accuracy (↑)	self-BLEU (↑)	Accuracy (↑)	BLEU (↑)
Our model	87%	54.5	80%	25.8
-pooling	89%	47.9	-	-
-temperature	86%	45.2	80%	21.3
-attention	93%	25.4	80%	22.1
-back-translation	86%	32.8	69%	16.4
+adversarial	86%	45.5	78%	25.1
-attention -back-translation	90%	26.0	71%	15.7

Table 8.7. Model ablations on 5 model components on the *FYelp* dataset (Left) and *SYelp* (Right).

In Table 8.7, we report results from an ablation study on the *SYelp* and *FYelp* datasets to understand the impact of the different model components on overall performance. The different components are: 1) pooling, 2) temperature based multinomial sampling when back-translating, 3) attention, 4) back-translation, 5) the use of domain adversarial training and 6) attention and back-translation in conjunction. We find that a model with all of these components, except for domain adversarial training, performs the best, further validating our hypothesis in Section 8.3.1 that it is possible to control attributes of text without disentangled representations. The absence of pooling or softmax temperature when back-translating also has a small negative impact on performance, while the attention and back-translation have much bigger impacts.

Table 8.13 shows examples of reviews re-written by different models at different checkpoints, showing the trade-off between properly modifying the attribute and preserving the original content. Figure 8.1 shows how the trade-off between content preservation (self-BLEU) and attribute control (accuracy) evolves over the course of training and as a function of the pooling kernel width.

8.5. Conclusion

We present a model that is capable of re-writing sentences conditioned on given attributes, that is not based on a disentanglement criterion as often used in the literature. We demonstrate our model's ability to generalize to a realistic setting of restaurant/product reviews consisting of several sentences per review. We also present model components that allow fine-grained control over the trade-off between attribute control versus preserving the content in the input. Experiments with automatic and human-based metrics show that our

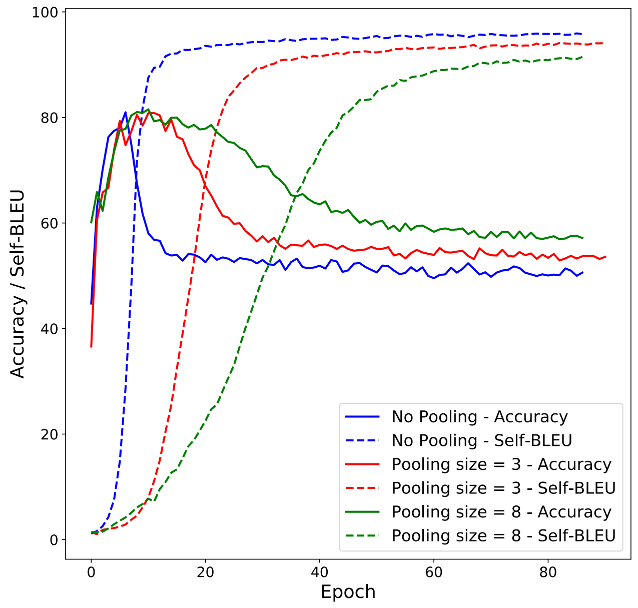


Fig. 8.1. Accuracy and self-BLEU curves on the *FYelp* dataset for different pooling operator configurations. Without pooling, the model tends to converge to a copy mode very quickly, with a high self-BLEU score and a poor accuracy. The pooling operator alleviates this behaviour and provides models with a different trade-off accuracy / content preservation.

model significantly outperforms the current state of the art not only on existing datasets, but also on the large-scale datasets we created. The source code and benchmarks are available at `https://git.io/Je3RV`.

8.6. Training Details

We used the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 10^{-4}, $\beta_1 = 0.5$, and a batch size of 32. As in Lample et al. (2018), we fix $\lambda_{BT} = 1$, and set λ_{AE} to 1 at the beginning of the experiment, and linearly decrease it to 0 over the first 300,000 iterations. We use greedy decoding at inference. When generating pseudo-parallel data via back-translation, we found that increasing the temperature over the course of training from greedy generation to multinomial sampling with a temperature of 0.5 linearly over 300,000 steps was useful (Edunov et al., 2018). Since the class distribution for different attributes in both the Yelp and Amazon datasets are skewed, we train with balanced minibatches when there is only a single attribute being controlled and with independent and uniformly sampled attribute values otherwise. The synthetic target attributes during back-translation $\tilde{y}$ are also balanced by uniform sampling.

8.7. Dataset Creation Details

In addition to the details presented in Section 10.4, we present additional details on the creation of the *FYelp* and *Amazon* datasets.

FYelp: Reviews, their rating, user information and restaurant/business categories are obtained from the available metadata. We construct sentiment labels by grouping 1/2 star ratings into the negative category and 4/5 into the positive category while discarding 3 star reviews. To determine the gender of the person writing a review, we obtain their name from the available user information and then look it up in a list of gendered names[4] following (Prabhumoye et al., 2018; Reddy & Knight, 2016). We discard reviews for which we were unable to obtain gender information with this technique. Restaurant/business category meta-data is available for each review, from which we discard all reviews that were not written about restaurants. Amongst restaurant reviews, we manually group restaurant categories into "parent" categories to cover a significant fraction of the dataset. These categories were determined by a combination of manually looking at Yelp restaurant categories and training fastText classifiers over the 50 most frequently occurring categories and looking at the classifier's confusion matrix. The grouping is as follows:

- **Asian** - Japanese, Thai, Ramen, Sushi, Sushi Bar, Chinese, Asian Fusion, Vietnamese, Korean, Noodles, Dim Sum, Cantonese, Filipino, Taiwanese
- **American** - American (New), American (Traditional), Canadian (New), Southern
- **Mexican/Latin American/Spanish** - New Mexican Cuisine, Mexican, Tacos, Tex-Mex, Tapas Bars, Latin American
- **Bars** - Brasseries, Nightlife, Bars, Pubs, Wine Bars, Sports Bars, Beer, Cocktail Bars
- **Desserts** - Desserts, Bakeries, Ice Cream & Frozen Yogurt, Juice Bars & Smoothies Donuts, Cupcakes, Chocolatiers & Shops

As described in Section 10.4, we train a classifier on these parent categories and relabel the entire dataset using this.

Amazon: Reviews, their rating, user information and restaurant/business categories are obtained from the metadata made available by (He & McAuley, 2016). We construct sentiment labels in the same manner as in *FYelp*. We did not experiment with gender labels, since we found that Amazon usernames seldom use real names. We group Amazon product categories into "parent categories" manually, similar to *FYelp* as follows:

- **Books** - Books, Books & Comics, Children's Books, Literature & Fiction, Comic Books, Kindle eBooks etc.
- **Electronics** - Car Electronics, Cell Phones, Electrical & Electronics, Electronics, Electronics & Gadgets, Mobiles, Tablets, Headphones etc.
- **Movies** - Movies, Movies & TV, Movies & Video, TV & Film

[4]`https://www.ssa.gov/oact/babynames/names.zip`

- **Clothing** - Clothing, Shoes & Jewelry, Baby Clothing, Fashion
- **Music** - CDs & Vinyl, Music, Digital Music, Children's Music, World Music, Electronic Music

We relabel reviews with a trained product category classifier similar to *FYelp*.

For the *FYelp* and *Amazon* datasets, we normalize, lowercase and tokenize reviews using the moses (Koehn et al., 2007) tokenizer. With social media content, we do not lowercase data in order to exploit interesting capitalization patterns inherent in the data, but we still run other pre-processing steps. We use byte-pair encodings (BPE) (Sennrich et al., 2015b) with 60k replacements, on all 3 datasets, to deal with large vocabulary sizes.

Human Annotated References. (Li et al., 2018) released a set of human reference edits when controlling the sentiment of a review, on a test set of 500 examples on the *SYelp* dataset. We follow suit by collecting a similar dataset of 500 human reference edits, which will be made publicly available, for both sentiment and product categories on the *FYelp* and *Amazon* datasets. When collecting such data, we use pre-trained sentiment/category classifiers to interactively guide crowd workers using the ParlAI (Miller et al., 2017) platform, to produce edits with the desired attribute value as well as significant content overlap with the input.

8.8. Additional Examples

Positive ↔ Negative (Yelp)	
Positive	frozen hot chocolate with peanut butter cups = amazing. i'll be back for some food next time!
Negative	frozen hot chocolate with peanut butter ? horrible. i'll stick with the coffee shop next door!
Negative	one word: underwhelming. save your money and find the many restaurants in vegas that offers a real experience.
Positive	one word: delicious. save room for the best and most authentic indian food in vegas.
Asian ↔ Mexican (Yelp)	
Asian	best thai food i've ever had in the us. great duck specials on monday.. best yellow curry fried rice..
Mexican	best mexican food i've ever had in my life. great guacamole on the side.. best carnitas tacos i have ever had..
Mexican	awesome carne asada! try the papa verde with steak! it's delicious and the portions are great!
Asian	awesome orange chicken! try the orange chicken with the spicy sauce! it's delicious and the portions are great!
Male ↔ Female (Yelp)	
Male	good food. my wife and i always enjoy coming here for dinner. i recommend india garden.
Female	good food. my husband and i always stop by here for lunch. i recommend the veggie burrito.
Female	we are regulars here... me n my husband just gorge on these freaking amazing donuts!!!!! loved it
Male	we are regulars here... every time we come here she loves the new york style pizza!!!!!!
Positive ↔ Negative (Amazon)	
Positive	i love this game. takes patience and strategy, go online and look for hints and cheats, they help alot. great game!!!
Negative	i don't like this game. it takes a lot of time to figure out how to play, and it doesn't work. i would not recommend this game.
Negative	i did not like the conflicting historical data. what was real and what was not. i prefer fiction with facts intact.
Positive	i enjoyed the historical references. what a great read and i loved it. i highly recommend this book.
Movies ↔ Books (Amazon)	
Movies	very good movie with outstanding special effects i recommend this to all shi fi lovers. good acting great plot lots of action
Books	very good book with outstanding character development i recommend this to all the readers. good job great plot twists and turns
Books	definitely not a stone barrington book, but a story told that keeps you wanting more. great read!
Movies	definitely not a film noir, but a story that keeps you on the edge of your seat. great acting and a great story.
Clothing ↔ Electronics (Amazon)	
Clothing	got this cause it said it would help with tennis elbow and guess what my tennis elbow still bothering me
Electronics	got this cause it said it would help with windows xp and guess what my windows xp still crashed
Electronics	i have no choice. this is the only black ink that works with my printer.
Clothing	i have no choice. this is the only black color that works with my dress.

Table 8.8. Example re-writes by our model on the *FYelp* and *Amazon* datasets when controlling a single attribute. The first line in every box is the pre-specified input with its attribute on the left, and the subsequent line is our model's re-write conditioned on a different attribute value.

Sentiment	**Category**	**Input / Generations**
		Amazon
Positive	**Movies**	**exciting new show. john malkovich is superb as always. great supporting cast. hope it survives beyond season 1**
Positive	Books	exciting new book. john grisham is one of the best. great read. hope he continues to write more.
Negative	Books	nothing new. john grisham is not as good as his first book. not a good read.
Positive	Clothing	awesome new watch. fits perfectly. great price. great quality. hope it lasts for a long time.
Negative	Clothing	horrible. the color is not as pictured. not what i expected. it is not a good quality.
Positive	Electronics	works great. the price is unbeatable. great price. great price. hope it lasts for a long time.
Negative	Electronics	worthless. the picture is not as clear as the picture. not sure why it is not compatible with the samsung galaxy s2.
Positive	Movies	exciting new show. john goodman is great as always. great supporting cast. hope it continues to end.
Negative	Movies	horrible. the acting is terrible. not worth the time. it's not worth the time.
Positive	Music	awesome new album. john mayer is one of the best. great album. hope he continues to release this album.
Negative	Music	horrible. the songs are not as good as the original. not worth the price.
		Yelp
Negative	**Dessert**	**the bread here is crummy, half baked and stale even when "fresh." i won't be back.**
Positive	American	the burgers here are juicy, juicy and full of flavor! i highly recommend this place.
Negative	American	the bread here is stale, dry and over cooked even though the bread is hard. i won't be back.
Positive	Asian	the sushi here is fresh, tasty and even better than the last. i highly recommend this place.
Negative	Asian	the noodles here are dry, dry and over cooked even though they are supposed to be "fresh." i won't be back.
Positive	Bar	the pizza here is delicious, thin crust and even better cheese (in my opinion). i highly recommend it.
Negative	Bar	the pizza here is bland, thin crust and even worse than the pizza, so i won't be back.
Positive	Dessert	the ice cream here is delicious, soft and fluffy with all the toppings you want. i highly recommend it.
Negative	Dessert	the bread here is stale, stale and old when you ask for a "fresh" sandwich. i won't be back.
Positive	Mexican	the tacos here are delicious, full of flavor and even better hot sauce. i highly recommend this place.
Negative	Mexican	the beans here are dry, dry and over cooked even though they are supposed to be "fresh." i won't be back.

Table 8.9. Demonstrations of our model's ability to control multiple attributes simultaneously on the *Amazon* dataset (top) and *FYelp* dataset (bottom). The first two columns indicate the combination of attributes that are being controlled, with the first row indicating a pre-specified input

Relaxed → Annoyed	
Relaxed	Wow! Sitting on my sister's patio, a glass of wine, and glorious music! Hmmmm!
Annoyed	Wow! Sitting on my sister's patio, a glass of wine, and the neighbors are out! Geez!
Relaxed	Nothing like a few 🍷 🍻 after a long productive day!!
Annoyed	Nothing like a flat tire 😡 after a long day at work!!
Relaxed	I decided it was time for a little chill time with my people tonight, I Missed them! Plus I need a break. 😌 🙌 🍷 🍻 🎶
Annoyed	I thought I was sleeping for a little while at the end of the week, I'm done! Plus I need a nap. 😡 ♀ 😡 😡 😡 😡
Relaxed	Rain = Sleepy! Love the sound of rain on my tin roof. ☔ 💦 FYI: Tomorrow is FRIDAY!
Annoyed	Rain = Mad! The sound of rain on my tin roof. ☔ ☔ Rain is overrated!
Relaxed	Yay!! A quick pedicure before I pick up the little angel from school! 😇
Annoyed	Yay!! A week before I pick up the little bastard from school! FML
Relaxed	Had an amazing day driving around. The sea, the woods, just great. 👍
Annoyed	Had an amazing day driving around. The weather, the roads are delayed, and the traffic is closed. 😡
Relaxed	Happiness is watching movie in a rainy day cuddling with your kids in a cozy bed 😊
Annoyed	Yet again watching tv in a rainy day cuddling with my kids in a cozy house 😒
Relaxed	Love sitting in the tree listening to the woods wake up!! Last day of 6 day 😊
Annoyed	Love sitting in the tree listening to the wake up call!! Last day of my holiday was just ruined 😡
Relaxed	A cup of hot creamy #coffee is all i need 😊 #relaxed #focused
Annoyed	A cup of hot coffee is the worst feeling ever 😒 😒 😒 #overit
Relaxed	I am in bed, listening to my music. 50 60 70 2000s..... 😆 😆 📀 📀
Annoyed	I am in bed, trying to sleep. This 50 + degree weather..... 😔 😔 😔
Relaxed	A nice end to a busy day, dinner & drinks with the hubby 💖 🎆 🌜 🍹 😚
Annoyed	A bad end to a busy day, dinner & drinks are just ruined 👿 👿 👹 👹 👹 👹
Annoyed → Relaxed	
Annoyed	Looks like my shovelling is done. Just broke my shovel.
Relaxed	Looks like my adjustable couch is done. Just chilling.
Annoyed	Nothing makes me more mad on Thursday get paid and the ATM at back is also broke. Thanks for the great service
Relaxed	Nothing better than a good massage on the beach and watching the sunset at the pool. Thanks for the great company
Annoyed	Who cares that Golden State won... im so over it... ready for football
Relaxed	Out of town by the Sea, and... im so excited... ready for vacation
Annoyed	Some people just can't be themself! #actthesamearoundeveryone
Relaxed	Having a great time with the bestie! #momanddaughterday #muchneeded
Annoyed	When gmail is isn't connecting and you tried to get stuff submitted before midnight
Relaxed	When the day is just over and you finally get to enjoy some rays before bed
Annoyed	I've never been this mad in my life!!! 😡 😡 😡 😡 😤
Relaxed	I've never been this relaxed in my life!!! ❤️ ❤️ ❤️ 💕
Annoyed	I hate it when I can't get to sleep. Stop it brain!!
Relaxed	I love it when I can't get up. Beautiful weather!!
Annoyed	When u get one of those phone calls that messes up ur day 👿 😡 ..
Relaxed	When u get one of those massage chairs that helps ur body 😍 💖
Annoyed	Don't u just love it when people don't respond to ur texts 😊 😡
Relaxed	Don't u just love it when it's time to pamper yourself 😍 😍
Annoyed	5 of battery and no charger since it just broke a min ago ugh!! 😑
Relaxed	5 minutes of sun and a bottle of beer now just watching a movie 🎥 😀 😴

Table 8.10. Examples of our model's ability to re-write sentences from public social media content when conditioned on information about the feeling expressed by the writer (Relaxed vs Annoyed)

65+ → 18-24	
65+	Reality leaves a lot to the imagination. - John Lennon
18-24	Reality leaves a lot of memes - John Cena
65+	Photos are beautiful. It must be unreal in person 😂
18-24	Cool pic bro - It must be fab in person!
65+	where did the time go ? LOVE the toothless smile!
18-24	lemme show u something 😂 🔝 💯 toothless smile 😂
65+	so pretty I feed so many in my yard and just love seeing them everyday.
18-24	so pretty I wanna tag in my family and just love them everyday
65+	Just where are you. I am loving the pictures and just a bit envious.
18-24	Just wanna be you I'm loving the pictures and just a bit excited
65+	Yes, so true! It is incredible what moms learn from daughters and grands!
18-24	Lmao so true! It's incredible what moms learn from daughters and kids x
65+	Fantastic picture, Peter you look younger all the time, must be the love!
18-24	Osm pic Peter you look younger all the time, must be the love 😍 😍
65+	What a sweet little girl. One can see the love ❤️ between you 2.
18-24	What a sweet little girl 😍 💯 can see the love ❤️ between you 2
65+	Congratulation young man. I am very proud of you, keep up the good work.
18-24	Congratulation sis I'm very proud of you keep up the good work
65+	So good to hear you are doing well. Love seeing pictures of your precious granddaughter ❤️
18-24	So good to hear you're doing well ❤️ seeing pictures of your precious baby ❤️
18-24 → 65+	
18-24	Yeah ight bitch. We gon see who get the last laugh 😏
65+	These ignorant idiots. We might see who get the last laugh!!!!!
18-24	Y'all are cute tho and I'm happy for you guys ❤️
65+	Hi, you are cute folks and I am happy for you guys.
18-24	😂 😂 i dont love boys 😂 😂 they're so urgh 😫
65+	What a lovely group of boys! They are so fortunate to have an exceptional faces.

Table 8.11. Model re-writes of sentences from public social media content when conditioned on the age-group of the writer (18-24 vs 65+)

Positive ↔ Negative (Yelp)	
Positive	happy to find this hidden gem near my office. great food and best of all, fast delivery.
Negative	the restaurant near my office was such a dump. late delivery and gross, cold food.
Negative	omfg no. i ordered baklava and they nuked it on a styrofoam dish for me. the insides were still cold and it tasted like cancer :/
Positive	yes! i ordered baklava and it was the perfect temperature on a fancy plate. the inside was great and it tasted excellent.
Mexican ↔ Asian (Yelp)	
Mexican	just awful. so called'asada burrito' was cold and bland... just a lump of plain carnitas and some iceberg lettuce in a cheap tortilla. reminiscent of taco bell in the 90s but worse.
Asian	just awful. so called spring roll was cold and bland....jump a lump of meat and vegetables in an egg noodle wrapper. reminded me of frozen chinese food but worse.
Asian ↔ American (Yelp)	
Asian	my new favorite curry house! definitely coming back. chicken katsu and takoyaki is soooo goood! cant wait to try the pork katsu.
American	my new favorite american bistro house! we are definitely coming back. fried chicken and waffles is soooo good. cant wait to try the new bbq rib sandwich.
Mexican ↔ Dessert (Yelp)	
Mexican	tacos were delicious. tried the carne asad, bbq pork, and chorizo. came with onion and cilantro topping and a house made choice of mild or hot salsa.
Dessert	cheesecake was delicious. tried the strawberry flavored one with chocolate drizzle. came with a fresh cherry on top and a house made choice of iced or hot coffee
Positive ↔ Negative (Amazon)	
Negative	too sci fi for me. characters not realistic. situation absurd. i abandoned it halfway through, unusual for me. simply not my taste in mysteries.
Positive	i love how sci fi this was! the characters are relatable, and the plot was great. i just had to finish it in one sitting, the story got me hooked. this is has to be one of my favorite mysteries.
Positive	my mom love this case for her i-pod. she uses it a lot and she is one satisfied customer. she would recommended it.
Negative	my mom initially liked this case for her i-pod. she used it for a while and it broke. since it is not solid she would not recommend it.
Clothing ↔ Books (Amazon)	
Clothing	nice suit but i wear a size 8 and ordered at 12 and it was just a bit too small.
Books	great book but i can't read small text with my bad eyesight, unfortunately, this one happened to be printed rather small.
Books ↔ Clothing (Amazon)	
Books	great book about the realities of the american dream. well written with great character development. i would definitely recommend this book.
Clothing	great dress with american flags printed on it. well made with great materials. i would definitely recommend this dress.
Books ↔ Music (Amazon)	
Books	i loved the book an can't wait to read the sequel! i couldn't put the book down because the plot and characters were so interesting.
Music	i loved the music and can not wait for the next album! i couldn't stop listening because the music was so interesting.

Table 8.12. Examples of human edits from our *FYelp* and *Amazon* datasets. The first line in every box was the input presented to a crowd worker followed by their corresponding edit with the specified attribute.

Accuracy	Self-BLEU	Input / Swap
		not a fan. food is not the best and the service was terrible the two times i have visited.
78.7%	73.8	great little place. food is great and the service was great the two times i have visited.
83.5%	51.5	best food in town. food is great and the service was the best two times i have visited.
92.8%	27.7	best thai food in town. the food is great and the service is excellent as always.
96.8%	13.1	best chinese food in town. great service and the food is the best i have had in a long time.
		overpriced specialty food. also very crowded. service is slow. would not recommend at any time.
78.7%	73.8	great homemade food. also very crowded. service is fast. would recommend at least once.
83.5%	51.5	great specialty food. also very crowded. service is friendly. would recommend any time at the time.
92.8%	27.7	great variety of food. also very friendly staff. good service. would recommend at least once a week.
96.8%	13.1	great tasting food. very friendly staff. definitely recommend this place for a quick bite.

Table 8.13. Examples of controlling sentiment with different model checkpoints that exhibit different trade-offs between content preservation (self-BLEU) and attribute control (Accuracy). The first line in both examples is the input sentence, and subsequent lines are a model's outputs with decreasing content preservation with rows corresponding to model checkpoints at different epochs during training.

Positive	Negative	Male	Female	American	Asian	Bar	Dessert	Mexican
pampered	dishonest	ammo	manicure	primanti	guu	promoter	patisserie	tortas
relaxation	fraud	tenant	pedi	bobby	chashu	bouncers	froyo	fundido
delightfully	incompetence	bachelor	hubs	flay	izakaya	bouncer	buttercream	arepas
complemented	unethical	barbers	bridesmaids	lux	tonkotsu	hakkasan	dunkin	burritos
cutest	insulted	wife	bridal	nacho	khao	postino	groomers	tostada
plush	audacity	firestone	pedicure	bj	soju	bachi	bakeries	taquitos
punctual	confronted	provider	instructors	gown	tonkatsu	brio	bakery	nacho
housemade	cockroach	data	mattresses	applebee	banchan	cabanas	custard	fajita
precision	crooks	plumber	stylist	burgr	shabu	films	doughnuts	guac
masterpiece	disrespect	motor	jacuzzi	chilis	teppanyaki	trader	pastries	refried
restored	roaches	contractor	hubby	mesa	gai	hooters	gelato	mexico
comprehensive	refunds	hertz	pregnancy	bmw	kbbq	karaoke	cheesecakes	empanadas
sublime	shrugged	hvac	bachelorette	denny	pho	irish	donut	tapas
made	liars	qualified	husbands	mastro	hotpot	harry	croissants	queso
tastefully	rudest	incompetence	barre	cellar	soi	darts	doughnut	cantina
treasures	accused	transmission	cutest	bachi	karaage	nightclub	danish	salsas
addicting	inconsiderate	summary	lashes	rubbed	omakase	applebee	cheesecake	pollo
marvelous	roach	contractors	sephora	flatbread	saigon	whiskey	fritter	asada
handsome	rudely	automotive	bf	skins	panang	cereal	macarons	barrio
healing	cancellation	audio	boyfriends	grille	cantonese	perform	oreos	barbacoa

Table 8.14. Examples of the learned attribute biases for sentiment and restaurant categories on *FYelp*.

Chapter 9

Prologue to the fourth article

Multi-scale Transformer Language Models. Sandeep Subramanian, Ronan Collobert, Marc'Aurelio Ranzato, Y-Lan Boureau. arXiv preprint.

Personal Contribution The project began with discussions between Sandeep Subramanian, Ronan Collobert, Marc'Aurelio Ranzato and Y-Lan Boureau during a subsequent internship at Facebook AI Research. The initial discussions were about pitfalls of current transformer language models and their large training memory footprint and designing good inductive biases to mitigate them. We explored multi-scale architectures since they hadn't been done in the context of transformers and would naturally reduce memory footprint. All authors met weekly to discuss progress, research directions and experiments. Sandeep Subramanian proposed the top-down and bottom-up variants, wrote all of the code, ran all of the experiments and wrote parts of the paper. Ronan Collobert, Marc'Aurelio Ranzato and Y-Lan Boureau proposed the retina variant, provided advice and guidance throughout the project and wrote parts of the paper.

9.1. Context

The need for memory efficient transformers became painstakingly evident from work we did on using transformer language models for abstractive summarization (Subramanian et al., 2019). Sparse attention had already been explored (Child et al., 2019) to alleviate this, but would require dedicated sparse operations to see memory and compute gains. Some of the multi-scale architectures we explored in this work in contrast wouldn't require anything more than what is presently available in most deep learning frameworks.

9.2. Contributions

The work presents 3 different multi-scale transformer architectures that exhibit favorable memory footprint versus perplexity trade-offs in language modeling. The gains are more pronounced in the low memory setting. Multi-scale architectures allow models to scale in terms of the number of parameters without incurring significant memory and compute overhead.

9.3. Research Impact

The work is relatively new, but has influenced similar work on compression based memory efficient transformers (Dai et al., 2020). While most concurrent work on efficient transformers has focused largely on reducing memory in the attention layer, in most practical settings, the feedforward layers dominate memory footprint and our proposed multi-scale architectures address this.

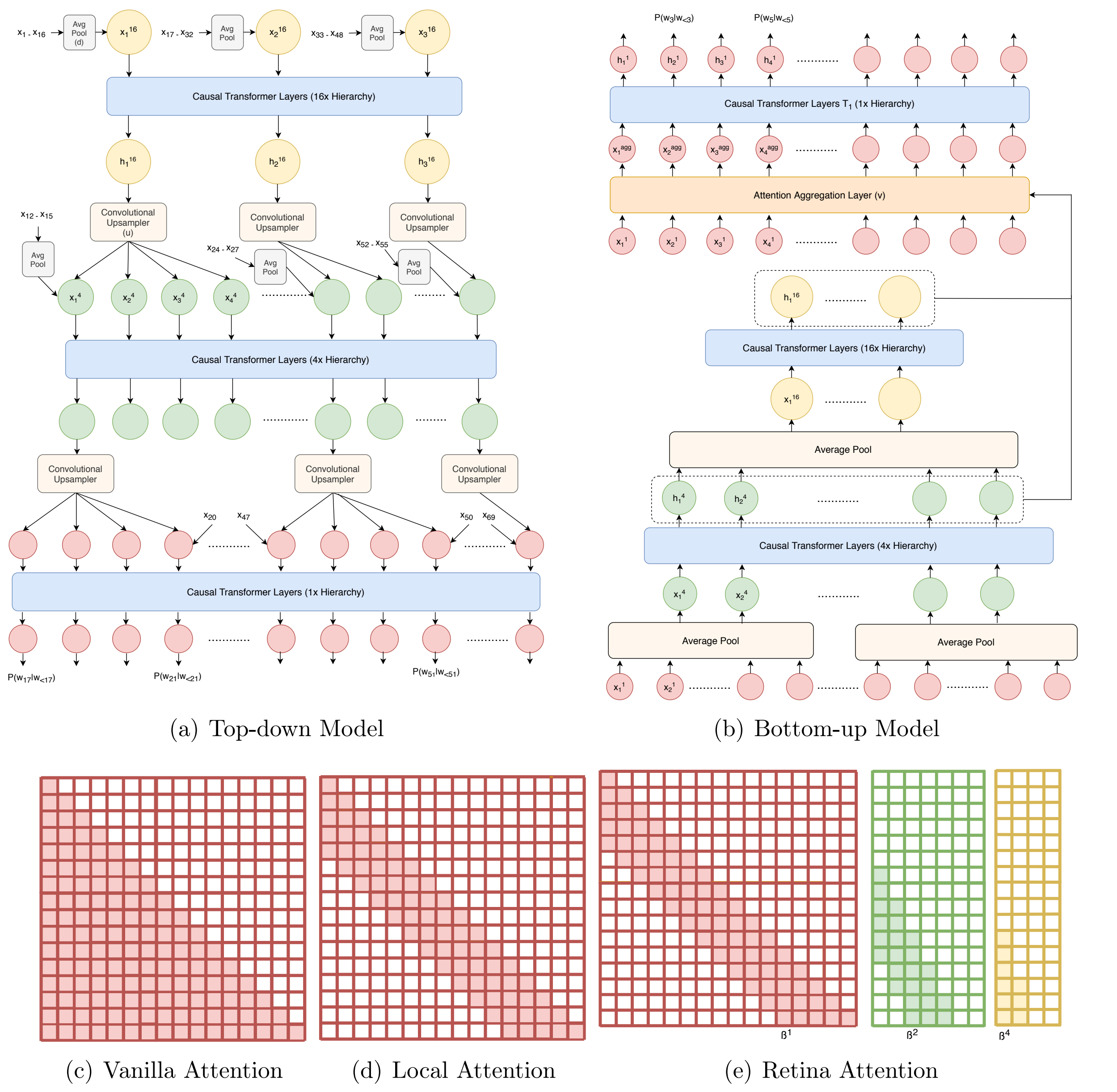
Chapter 10

Multi-Scale Transformer Language Models

10.1. Introduction

Human language displays simultaneous organization at multiple scales and granularities: topics are maintained over long spans of text without controlling every word, while grammatical correctness imposes strong local constraints without influencing word choice at a very long range. The choice of a word is thus governed both by fast-changing local constraints, and by longer-range information that tends to be more stable. This makes hierarchical architectures a seemingly natural choice for language modeling, This multi-scale structure is reminiscent of well-studied hierarchies in many domains of natural perception. In vision, the resolution of receptive fields decreases as one moves either away from the center of the retina, or higher in the hierarchy of visual cortex areas (from V1 to V2 to V4). This gives rise to phenomena such as crowding (Pelli & Tillman, 2008), where only some ensemble statistic of a set of elements can be perceived, and the existence of metamers, distinct visual stimuli that look the same when seen at the periphery (Freeman & Simoncelli, 2011; Wallis et al., 2019). Incorporating both local and global organizational constraints and compressing representations with statistical pooling have been suggested to better explain empirical perceptual vision phenomena (Wallis et al., 2019). Multi-resolution pyramids such as Laplacian pyramids have been shown to produce more convincing image generation (Denton et al., 2015) and efficient compression (Burt & Adelson, 1983). Combining representations at multiple scales has also been successful for speech and audio generation (Mehri et al., 2016).

Language modeling efforts however, typically rely on modeling the multi-scale nature of language without strong architectural priors. Representations are learned only at the finest scale, usually that of words or subwords, and rely on the training objective of predicting the next word in the sequence to implicitly capture the need to maintain consistency and coherence across multiple scales. Some previous efforts have advocated for multi-scale sequence models (Koutnik et al., 2014; Chung et al., 2016; Mehri et al., 2016), but these haven't seen


Avg Pool (d)
Causal Transformer Layers (16x Hierarchy)
Convolutional Upsampler (u)
Convolutional Upsampler
Causal Transformer Layers (4x Hierarchy)
Causal Transformer Layers (1x Hierarchy)
Causal Transformer Layers T1 (1x Hierarchy)
Attention Aggregation Layer (v)
Average Pool
ß1
ß2
ß4

(a) Top-down Model

(b) Bottom-up Model

(c) Vanilla Attention

(d) Local Attention

(e) Retina Attention

Fig. 10.1. Our proposed model architectures: (a) **Top-down model** that builds representations from coarse (yellow) to fine (red) scales. (b) **Bottom-up Model** that does the opposite while aggregating representations from different scales before operating at the finest scale. (c,d,e) **Retina Model** that treats different parts of of the model's context with different granularities - nearby information is fine-grained, but coarse in the distant past. We visualize this using the model's attention mask. (c) is a standard autoregressive mask in vanilla transformers, (d) a local attention mask (Parmar et al., 2018) that only looks at local information (e) our proposed retina variant with green and yellow indicating progressively coarser scales

widespread adoption in language modeling on current large-scale benchmarks. The potential of leveraging multiple scales of representation to reduce memory footprint (see Section 10.3.6) is especially appealing for language modeling because transformers (Vaswani et al., 2017), which are currently the most popular architectures, suffer from quadratic memory usage scaling as context length increases.

Despite this promise, current efforts typically do not rely on strong architectural priors. Instead, representations are learned only at the finest scale, usually that of characters, words or subwords, and rely on the training objective of predicting the next word in the sequence to implicitly capture the need to maintain consistency and coherence across multiple scales.

An obstacle to the adoption of multi-scale architectures is that the many decisions involved in designing such architectures have an unclear effect on language modeling performance. In this work, we conduct multiple ablation studies in three different multi-scale transformer-based architectures and examine trade-offs in terms of memory footprint and perplexity. Our work makes the following contributions: (1) We present three different multi-scale transformer architectures for language modeling. (2) We study the impact of design choices such as the number of scales, assignment of model capacity across scales, local attention window sizes, and attending over the model's context with different granularities.

10.2. Related Work

We review previous work that has explored multi-scale sequence models and memory efficient transformer models.

Multi-scale architectures for sequential data (Schmidhuber, 1992; El Hihi & Bengio, 1996; Koutnik et al., 2014) have attempted to exploit the multi-scale nature of data like language, speech, music and audio. They have seen success in modeling conversations (Sordoni et al., 2015), raw audio, speech and music (Mehri et al., 2016), and language (Chung et al., 2016). Sordoni et al. (2015); Garg et al. (2019) use the available hierarchical structure in text (e.g., sentences or paragraph) to define a shallow hierarchy, Koutnik et al. (2014) and Mehri et al. (2016) use fixed time scales, while Chung et al. (2016) try to learn the clocking function using the task-specific training objective in an end-to-end manner, with an RNN-based architecture. To the best of our knowledge, more generic multi-scale architectures not relying on limited data-dependent boundaries haven't been explored in conjunction with transformer language models.

Memory-Efficient Transformers Sukhbaatar et al. (2019) present an adaptive attention mechanism that learns how far back into the past each head in a transformer should attend. Liu et al. (2018); Rae et al. (2019) explore approaches that compress the transformer's memory (context) with strided convolutions. Child et al. (2019); Beltagy et al. (2020); Correia et al. (2019) present sparse transformers along with efficient CUDA kernels for sparse attention demonstrating the ability to generate very long sequences. Liu & Lapata (2019) proposes a hierarchical extension of the architecture proposed in Liu et al. (2018) that attends over very long sequences, with the aim to better model paragraph and document-level contexts. Kitaev et al. (2019) use reversible layers and LSH attention to reduce memory footprint. Bai et al. (2019) train infinite-depth weight-tied feedforward nets via root-finding, yielding a memory footprint that like Kitaev et al. (2019) does not scale with network depth, but incurs computational overhead.

A recent survey of efficient transformers (in terms of memory and compute) (Tay et al., 2020a) reported that most approaches incur a fairly significant performance drop in exchange for efficiency. In this work, we search for architectures that incur almost no drop in language modeling performance.

10.3. Models

We first briefly review language modeling and transformers (Vaswani et al., 2017) which are the base for our architectures. We then describe the three different multi-scale architectures we explore for language modeling. See Figure 10.2 for a visual depiction of our proposed architectures and Listings 1, 2 and 3 for a PyTorch-like implementation of the forward pass through the models.

10.3.1. Language Modeling & Transformer Preliminaries

The goal of language modeling is to estimate the joint probability of a sequence of tokens. The models we consider in this work factorize the joint probability over words into a product of conditional probabilities of each word given everything that precedes it. Conditional probabilities are typically parameterized as recurrent neural networks like LSTMs (Graves, 2013; Mikolov et al., 2010) or feedforward models like gated convnets (Dauphin et al., 2017), transformers (Vaswani et al., 2017; Radford et al., 2018), or estimated with non-parametric count-based statistics like in n-gram language models.

Transformers have become ubiquitous for large-scale language modeling (Radford et al., 2018; Baevski & Auli, 2018; Radford et al., 2019; Dai et al., 2019; Kaplan et al., 2020). They are self-attentive models that have stacks of residual blocks, each of which contains layers of multi-headed self-attention and feedforward modules. Multi-headed attention is a generalization of dot-product attention (Bahdanau et al., 2014; Luong et al., 2015b) where a score is computed between a query Q and key K for different learned projections of both. The scores are then normalized with a softmax function and used as weights to compute a weighted average over values V at each position in the sequence. In language models, we use self-attention where $Q = K = V$. The resulting representations are then summed with the input to the residual block and then normalized using LayerNorm (Ba et al., 2016) followed by a feedforward layer, normalization, and residual summing again. All operations in the model have the advantage of being easily parallelizable on current hardware, and the attention mechanism allows the model to learn long-range dependencies.

We use d_{model} and d_{ff} to refer to the dimensionality of the intermediate hidden states in the transformer and the position-wise feedforward layers in the model, respectively.

10.3.2. Multi-scale Transformers (Notation)

Let $k_1 < \ldots < k_m$, denote the m scales of a multi-scale model, where $k_1 = 1$ is the finest scale that looks at every token, and scales k_i incorporates information that has been down-sampled by a factor k_i through a down-sampling operator d (e.g., an average over windows of size k_i or a strided convolution). For example, a model with scales $(k_1, k_2, k_3) = (1, 4, 16)$ would use three scales with each scale being 4 times coarser than the preceding scale. Given a sequence of n, words, they are first represented as word embeddings $x_1 \ldots x_n$ of dimension d_{model} using a lookup table. Let x^{k_i} denote the representation that is fed as input to vanilla transformer blocks t_{k_i} at scale k_i (with possibly different numbers of layers at each scale), and h^{k_i} the representation at scale k_i. We now detail how x^{k_i},h^{k_i} are computed for each of the variants we propose.

10.3.3. Top-down Model

Our proposed Top-down model (Fig. 1(a)) is largely inspired by SampleRNN (Mehri et al., 2016), a multi-scale recurrent architecture for generating audio. It is “Top-down” because it builds representations of the sequence progressively from coarser to finer scales, by running multiple transformer layers at a particular coarse scale followed by convolutional upsampling to the subsequent (finer) scale. Predictions are made at the finest scale. The motivation is to have early layers learn high-level or coarse outlines and have those representations be progressively upsampled to include finer and finer details.

The model uses downsampling operators d, that take a sequence of vectors as input, and a factor k_i by which to downsample them, to compute a representation of the input at scale k_i of length of n/k_i; in our experiments, d is average pooling with a kernel of size k_i, or causal strided convolutions.

The input to the coarsest scale in the model x^{k_m}, is the pooled token embedding $\bar{x}^{k_m}$ over windows of size k_m. Inputs x^{k_i} to the finer-scale transformers thereafter are obtained by combining the pooled token embeddings $\bar{x}^{k_i}$ at the corresponding scale with the upsampled representation from the immediately coarser scale:

$$
\begin{aligned}
\bar{x}^{k_i} &= d(x_1 \ldots x_n, k_i), m \geq i \geq 1, \\
x^{k_m} &= \bar{x}^{k_m}, \\
h^{k_i} &= t_{k_i}(x^{k_i}), m \geq i \geq 1, \\
x^{k_{i-1}} &= f(\bar{x}^{k_{i-1}}, u(h^{k_i}, k_i/k_{i-1})), m \geq i \geq 1,
\end{aligned}
$$

where f denotes a function that concatenates its inputs (of equal dimension), followed by a learned linear projection to half the dimension of the concatenated vector; u is an upsampling function such that $u(h^{k_i}, k_i/k_{i-1})$ indicates that representations h^{k_i} are being upsampled by a factor k_i/k_{i-1}. The model is trained to predict the next word in the sequence using representations h^{k_1}. To make sure representations aren't informed by the future at any position, we slice and shift the inputs to d appropriately.

10.3.4. Bottom-up Model

In contrast to the Top-down model, the Bottom-up model (Fig. 1(b)) builds representations progressively from fine to coarse scales. Instead of a final upsampling operator back to the finest scale, the architecture uses an aggregation layer, denoted by v, to incorporate information from coarser scales at the word level. This is a transformer layer where certain subsets of heads attend to representations from different scales. Specifically, inputs to the multi-headed attention module within the layer are the word embeddings themselves denoted by $x_1 \ldots x_n$, and the keys and values are representations at different coarser scales denoted by $h^{k_2} \ldots h^{k_m}$; we denote this operation via $v(h^{k_2} \ldots h^{k_m}, x_1 \ldots x_n)$. This aggregation layer is applied before running transformer layers at the finest scale:

$$
\begin{aligned}
h^{k_1} &= h^1 = x_1 \ldots x_n, \\
x^{k_i} &= d(h^{k_{i-1}}, k_i/k_{i-1}), 2 \leq i \leq m, \\
h^{k_i} &= t_{k_i}(x^{k_i}), 2 \leq i \leq m, \\
x^{agg} &= v(h^{k_2} \ldots h^{k_m}, x_1 \ldots x_n), \\
h^{out} &= t_1(x^{agg}),
\end{aligned}
$$

where h^{out} is the output from which word probabilities are predicted.

10.3.5. Retina Model

Our "Retina" model (Fig 1(e)) learns progressively coarser representations for tokens further away in the past, in a way that is reminiscent of the progressively bigger receptive fields in the retina as one moves away from the center. The underlying intuition is that a lot of the fine-grained information at the word level might be necessary in a close range (e.g., for grammaticality), but not so much at larger distances (see Section 10.4.3 on model sensitivity to perturbations near and far away from the current token being predicted). The model is a multi-scale variant of Liu et al. (2018).

The Retina model does not have separate transformer layers at every scale, we implement it with a simple modification to the multi-headed attention module of a transformer. Different subsets of heads look at representations from different scales. The queries to the multi-headed attention always come from the representations at the finest 1x (k_1) scale, while the keys and values are selected and downsampled appropriately based on each input query position and the scale assigned to each head.

Formally, consider a setting with scales $k_1, \dots, k_m$, context window boundary for scale k_i as β^{k_i} and token embeddings $x_1, \dots, x_n$, predicting the next token at position t. Downsampled representations $\bar{x}^{k_i}$ and output representation h^{out} are obtained by:

$$\begin{aligned} \bar{x}^{k_i} &= d(x_{t-\beta^{k_i}} \dots x_{t-\beta^{k_{i-1}}-1}, k_i), m \geq i \geq 1, \\ h^{out} &= t(\bar{x}^{k_1} \dots \bar{x}^{k_m}). \end{aligned}$$

Listings 1, 2 and 3 show an implementation of the forward pass for our model variants.

10.3.6. Space & Time Complexity for Transformer Layers

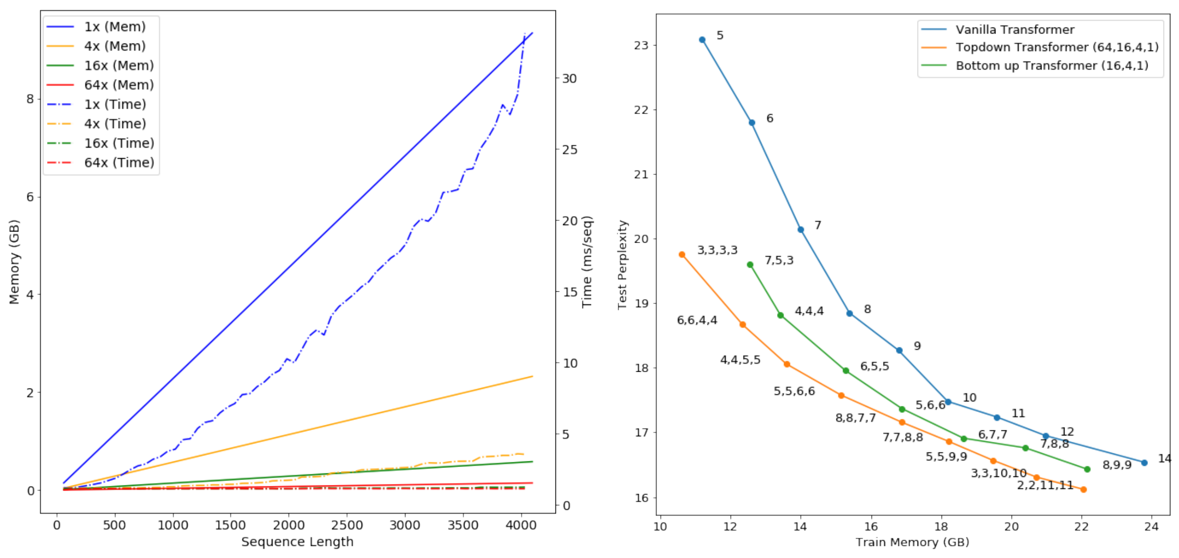
Transformer language models typically require large amounts of GPU memory to train. There are a few factors that contribute to this.

1. Model Depth - All intermediate activations are stored at each layer for backpropagation. Memory scales linearly with the number of layers.

2. Attention - Multi-headed attention with queries, keys and values Q, K, V requires computing a score for every pair of elements in the query and key (i.e.) the QK^T matrix is of size $N \times N$ (assuming a batch size of 1) where N is the number of elements in the sequence. Both memory and time complexities are therefore quadratic in the length of the sequence $\mathcal{O}(N^2)$, computation however is typically less of a problem since it can be parallelized easily across N. With short sequences, where the embedding dimension of the queries, keys, and values are greater than the sequence length, the $(QK^\top)V$ matrix becomes expensive to store and compute. The overall complexity is $\mathcal{O}(N^2 + NH)$, where H is the embedding dimension.

3. Position-wise Feedforward Layers - Position-wise linear layers are often of size $4H$ and in typical setups, where $N \approx H$, storing these activations dominates memory footprint (Dai et al., 2020).

Our multi-scale architecture addresses 2 and 3 by reducing the number of positions over which transformer layers at coarser scales need to operate on. This is evident from Fig. **??** and Table. 10.8 that shows memory footprint and time for a single transformer layer at different scales.



(a) Memory and Time vs Sequence Length

(b) Training memory vs Perplexity

Fig. 10.2. (a) Memory footprint and run time vs sequence length and run time for a single transformer LM layer at different scales (b) Test perplexity vs train Memory footprint for vanilla transformer LMs and our topdown (64x,16x,4x,1x) and Bottom-up (16x,4x,1x) multi-scale variants on the BookCorpus test set. Numbers next to each point indicates the number of layers per scale.

10.4. Experiments & Discussion

10.4.1. Datasets

We consider three word/sub-word level language modeling benchmarks of different sizes. In order to make our results comparable to previous published research, we use standard datasets that have been used in prior language modeling efforts - Wikitext-103 (Merity et al., 2016), BookCorpus (Zhu et al., 2015) & CC-news (Liu et al., 2019). Dataset statistics are described in Table 10.1. We use the standard train/validation/test splits for Wikitext-103 and use random splits for the BookCorpus and CC-news. Dataset statistics are reported in Table 10.1. Following Baevski & Auli (2018), we BPE tokenize (Sennrich et al., 2015b) Wikitext-103 with 32k replacements, but report perplexities at the word level for comparison with previous work, by re-normalizing the average perplexity by the actual number of words in valid/test sets. We use 30k replacements for the BookCorpus and CC-news datasets, and report the average sub-word level perplexities since all models being compared are trained with the same tokenization and BPE pre-processing.

Dataset	Size (GB)	Train Tokens	Valid Tokens	Test Tokens	Vocab Size
Wikitext-103	.54	111M	231K	261K	33,346
Toronto BookCorpus	3.6	832M	108M	106M	31,300
CC-news	83	16.6B	370M	370M	63,724

Table 10.1. Dataset statistics

10.4.2. Experimental Details

Across all datasets, our primary point of comparison is with a vanilla transformer language model (Radford et al., 2018). For the Wikitext-103 and BookCorpus datasets, we use a model that has either 12, 14 or 16 layers, $d_{model} = 768$, 12 attention heads, $d_{ff} = 3{,}072$, dropout of 0.1 everywhere including the attention scores and GeLU activations (Hendrycks & Gimpel, 2016) - a configuration similar to the smallest GPT-2 model (Radford et al., 2019). For the CC-News dataset, we increase d_{model} to 1,024 and d_{ff} to 4,096. We use a batch size of 256 randomly sampled chunks of 512 tokens. For our CC-News models, we use gradient accumulation to simulate a batch size of 256. At inference, we consider the entire piece of text as one contiguous block and use a sliding window of size 512 tokens with a stride of 256 and make predictions only over the last 256 tokens in the window. This ensures that for every minibatch of examples, the model has some context to work with. Each model was trained on 8 GPUs for roughly 1 week - run times for the top-down, bottom-up and vanilla transformers were almost the same, while the retina model was slower since our implementation didn't leverage sparsity.

Our multi-scale architectures use the same configuration above for every transformer layer, with added upsampling and downsampling modules. Our best-performing models use average pooling for downsampling (over max-pooling) and we present an ablation study with strided convolutions in Table 10.2. We used a single transpose convolution layer with an appropriate kernel size and stride for upsampling. Each scale increases by a factor of 4 from the previous.

All models were trained with mixed precision and optimized using Adam (Kingma & Ba, 2014) with the learning rate increased linearly to $2.5 \times e^{-4}$ over 40,000 warmup steps and then annealed to 0 over a million steps using a cosine schedule (Radford et al., 2018). We tuned the dropout rates in our Wikitext-103 in the range of 0.1 to 0.5 with increments of 0.05, since we found that our models were able to overfit the data quite easily. For BookCorpus and CC-News, we only experimented with the number of transformer layers at each scale and the number of scales (see Table 10.2).

10.4.3. Analyzing Transformer LM behavior to context perturbations

Khandelwal et al. (2018) and Sankar et al. (2019) analyzed how LSTM language models and dialog systems use context, by perturbing it with different kinds of noise and observing changes in the likelihood assigned by the model to every subsequent (unperturbed) token in the sequence. Khandelwal et al. (2019a) show that perturbations which destroy word order, like shuffling, affects the likelihood that the model assigns to words near the shuffled context, but not far away.

We observe similar patterns for transformer language models on the Wikitext-103 and BookCorpus datasets, as shown in Figure 10.3. In particular, we consider a sequence of 512 tokens, shuffle the first 256 tokens and observe the likelihood that an already trained model (without perturbations) assigns to the subsequent 256 tokens. When the shuffled context is more than 50 tokens away, the change in likelihood, compared to when the model is presented with a completely unshuffled context is already minimal, indicating that the model may not be using word order beyond this distance. This observation might partly explain why our Retina model, which only looks at fine-grained context in a local way while using coarser representations for distant context, can perform on par with a model that uses the entire context at a fine-grained scale (see results in Table 10.2). The same argument holds for the use of average pooling operations in the top-down and bottom-up variants which destroys word order.

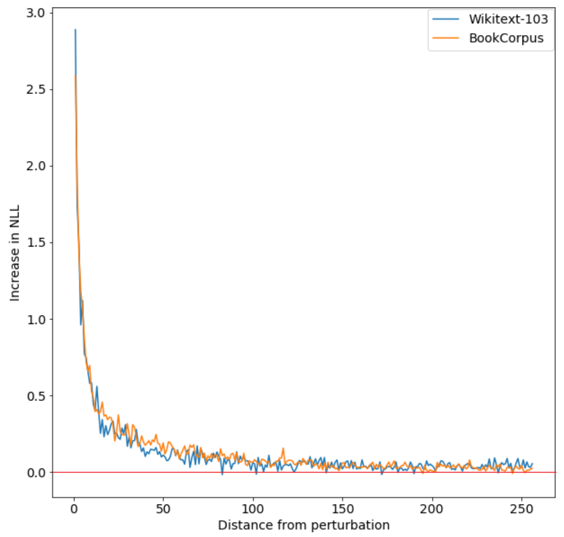


Fig. 10.3. Increase in NLL for a trained model on the Wikitext-103 and BookCorpus datasets when word order is destroyed by shuffling its context (perturbation), as a function of the distance from the shuffled context.

Model	Scales	Attention Context Window	Layers	Test PPL	Mem (GB)
Ablation for number of scales					
Vanilla	1	0:512	12	16.95	20.98
Vanilla	1	0:512	14	16.54	23.78
Top-down	4,1	0:512	8,12	15.87	25.45
Top-down	16,4,1	0:512	4,8,12	15.76	25.23
Top-down	64,16,4,1	0:512	2,4,8,12	16.00	22.62
Ablation for capacity at different scales					
Top-down	64,16,4,1	0:512	16,12,6,1	22.14	10.61
Top-down	64,16,4,1	0:512	10,10,8,5	17.89	13.59
Top-down	64,16,4,1	0:512	8,8,7,7	17.16	16.87
Top-down	64,16,4,1	0:512	7,7,8,8	16.86	18.22
Top-down	64,16,4,1	0:512	5,5,9,9	16.57	19.47
Top-down	64,16,4,1	0:512	3,3,10,10	16.31	20.71
Top-down	64,16,4,1	0:512	2,2,11,11	16.12	22.05
Top-down	64,16,4,1	0:512	1,1,12,12	15.95	23.40
Ablation for downsampler					
Top-down (Avg-Pool)	64,16,4,1	0:512	7,7,8,8	16.86	18.22
Top-down (Conv)	64,16,4,1	0:512	7,7,8,8	17.95	18.45
Ablation for deep & narrow networks					
Vanilla (256/1024)	1	0:512	30	22.15	22.07
Vanilla (512/2048)	1	0:512	20	17.70	22.77
Ablation for attention context window					
Vanilla	1	0:512	12	16.95	23.78
Vanilla	1	0:256	12	17.00	23.78
Vanilla	1	0:128	12	17.39	23.78
Vanilla	1	0:64	12	17.88	23.78
Vanilla	1	0:16	12	19.01	23.78
Vanilla	1	0:8	12	20.19	23.78
Retina	16,4,1	0:8,8:256, 256:512	12	16.81	23.95
Retina	16,4,1	0:16,16:256, 256:512	12	17.07	23.95
Retina	16,4,1	0:128,64:256,128:512	12	17.08	23.95

Table 10.2. Model ablations for the number of scales, number of transformer layers per scale, type of downsampling function, skinny and deep networks and different local attention/retina attention masks. All reported results are on our BookCorpus test set. The attention context window column indicates what attention heads at each scale look at - for example, "0:512" implies all scales see the entire history while "0:8,8:256,256:512" indicates that the finest scale sees only the previous 8 tokens, the subsequent scale from 8-256 and so on.

10.4.4. Ablations

In Table 10.2 we investigate how the main design choices of multi-scale transformers, such as the number of scales and number of layers at each scale, impact perplexity on the BookCorpus. We observe that: (1) perplexity slightly improves when adding the 4x and 16x scales, but not 64x; (2) adding model capacity at finer scales is more important than at coarser ones; (3) gains from multi-scale modeling do not come solely from being able to train deeper models for the same memory budget, since narrow and deep vanilla transformers with similar memory footprint perform poorly; (4) local attention even with very small windows are competitive - with less than 1 ppl difference between a lookback of 64 vs 512 tokens. Note however, that the effective receptive field of a model trained with local attention grows with depth. (5) Average pooling outperforms strided convolutions for downsampling. (6) Our retina model, with small attention windows at the finest scale, performs on par with a vanilla transformer with global attention.

Additionally, Section 10.6 has a breakdown of perplexity vs tokens frequencies and includes sample based evaluations.

10.4.5. Perplexity vs Memory Footprint Trade-off

Figure 2(b) plots model memory footprint[1] versus perplexity for vanilla, top-down and bottom-up models for different model sizes. There is a small gain in perplexity for the same memory footprint (with smaller models showing bigger gains). Results in Tables 10.3 and 10.4 on Wikitext-103 and CC-news however show only very minimal gains. We suspect that the poor results on Wikitext-103 are due to the fact that it is a much smaller dataset, which our deeper multi-scale models overfit on.

Model	Layers	Test PPL	Train Mem (GB)
Vanilla	10	20.99	16.76
Vanilla	12	20.23	19.04
Vanilla	14	19.62	21.31
Top-down	5,5,6,6	20.92	14.66
Top-down	5,5,9,9	20.26	18.35
Top-down	2,4,8,12	19.47	20.96

Table 10.3. Model performance on CC-news.

[1]using torch.cuda.max_memory_allocated

Model	Layers	Test PPL	Mem (GB)
Previous Work			
Vanilla (Welleck et al., 2019)	16	25.6	-
Transformer-XL (Dai et al., 2019)	16	24.3	-
DEQ-Transformer (Bai et al., 2019)	-	24.2	-
DEQ-Transformer (Adaptive) (Bai et al., 2019)	-	23.2	-
Adaptive Inputs (Baevski & Auli, 2018)	16	18.7	-
Our Models			
Vanilla	16	25.9	26.85
Top-down (4x, 1x)	6,12	25.6	25.15
Top-down (4x, 1x)	5,10	26.1	22.08
Bottom-up (4x, 1x)	5,10	26.8	21.83
Retina (16x, 4x, 1x)	16	26.6	29.92

Table 10.4. Model performance on Wikitext-103.

10.5. Conclusion

We study design choices in three multi-scale transformer architectures for language modeling on three datasets. We find that (1) given a memory and depth budget, our multi-scale models do better than vanilla transformers; (2) biasing capacity assignment towards finer scales works better; (3) local attention with small attention windows performs surprisingly well, on par with global attention when combined with attention over coarse representations.

These proposed models leverage the robustness of transformers to word ordering in distant context windows (Fig 10.3). In these architectures, the representation produced by coarser scales which operate on much shorter sequences, is combined with the representation of the finest scale, thereby reducing the overall computational and memory cost.

Future work will explore how to combine our multi-scale architectures with adaptive attention head spans and how different types of information (e.g., topic, grammatical correctness) are differently affected by architecture scale choices.

10.6. Additional Results & Algorithms

10.6.1. NLL vs Word Frequency

In tables 10.4 and 10.5, we break down the token-level perplexities based on their frequency of occurrence in the training set, similar to (Baevski & Auli, 2018). We bin tokens into 5 or 10 equally sized bins, based on cumulative frequency and report the average NLL within each bin, with bins sorted from those that contain least to most frequent words. We find that the performance gap between models with smaller and larger context window sizes stems from modeling of low-frequency tokens.

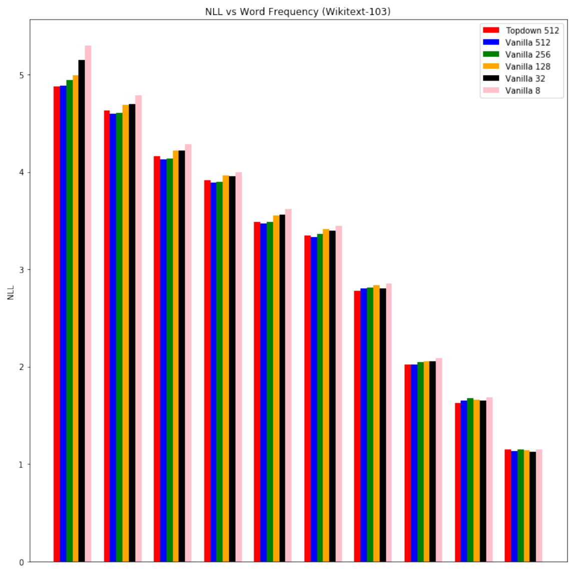


Fig. 10.4. Test NLL vs word frequency. Left to right: rarest to most frequent word bins.

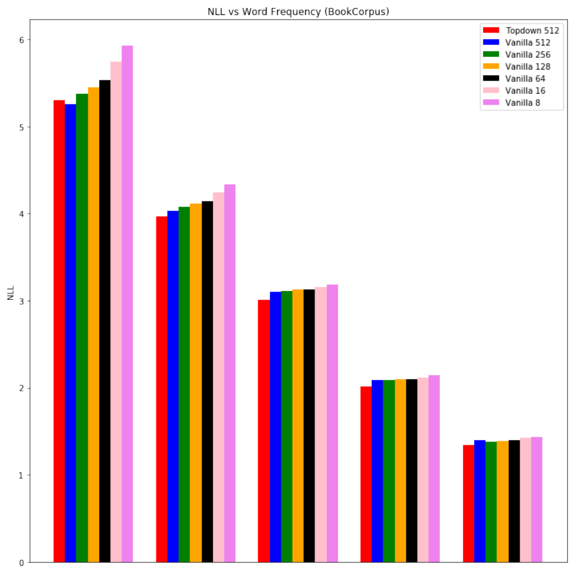


Fig. 10.5. Test NLL vs word frequency. Left to right: rarest to most frequent word bins.

10.6.2. Coarse-only Transformer LMs

We explore the possibility of training a model that makes predictions directly at a particular scale. For example, if we consider a scale of 4, we initially downsample the input by a factor of 4 by averaging the embeddings of every non-overlapping 4-gram chunk. We then run transformer layers on this input and use the final representations at each step to predict the next *4-gram* chunk. We train the model to make these predictions by minimizing the cross-entropy between the model's output distribution and a uniform distribution over the subsequent 4-gram.

Table 10.6 presents qualitative results showing model predictions for the following. We experimented with adding this criterion of being able to predict the subsequent bag of words distribution in our top-down model at all scales, but didn't observe any improvements. This experiment mostly served as a sanity check to ensure that the representations learned at different scales can be useful. To ensure that representations at different scales learn meaningful representations, we qualitatively analyze them by looking at nearest neighbors computed using them in Table 10.7. We found that nearest neighbors computed on Wikitext-103 typically returned segments within highly related topics.

Model	Layers	N-gram PPL	GPT-2 PPL	Ref BLEU	N-gram Repeat	Time (s/seq)
Vanilla	14	22.81	7.30	7.45	0.75/0.45/0.29/0.20	1.35
Top-down	2,4,8,12	21.17	7.69	6.80	0.73/0.41/0.26/0.16	0.93
Bottom-up	8,9,9	24.54	8.53	7.04	0.75/0.45/0.28/0.18	1.36
Retina	12	20.91	8.64	7.37	0.73/0.42/0.26/0.17	1.72

Table 10.5. Sample based evaluation of models on the BookCorpus. The N-gram repeat column reports 1/2/3/4-gram repeat fractions with a lookback window of 256 tokens

10.6.3. Sample-Level Evaluation Metrics

A hallmark of the recent progress in language modeling has been improvements not only in perplexity, but sample quality as well. While evaluating generative models of text purely based on their samples is extremely difficult and an ongoing effort (Cífka et al., 2018; Semeniuta et al., 2018), we would like to ensure that we aren't losing out on sample quality with respect to vanilla transformer LMs.

We therefore borrow a few sample-level evaluation metrics typically used to evaluate GAN-based text generation methods to compare samples from our model with vanilla transformer LMs. In all sample-based evaluation setups, we provide models with 64 tokens of context and generate sequences of length 256 with topk sampling (Fan et al., 2018) and a temperature of 0.7.

N-gram and GPT-2 PPL - We compute the likelihood of model generated samples under a pre-trained kneser-ney smoothed 4-gram language model (Heafield, 2011) and a pre-trained GPT-2 345M parameter model (Radford et al., 2019).

Ref BLEU - Given an a particular input context, we generate 3 distinct completions and use these as references to compute BLEU wrt the ground truth completion.

N-gram repeats - Following Welleck et al. (2019), we also estimate the fraction of 1,2,3 and 4-grams that the model repeats.

Quantitative results for different models are presented in Table 10.5, with qualitative samples in Table 10.9. We could not observe any significant difference in sample quality between the different models. We also report the time it takes for each model to generate a sequence of 256 tokens given a context of 64 tokens.

Reference Text :
Prisoner of Azkaban was the third film in the series . Radcliffe 's performance was panned by New York Times journalist A. O. Scott , who wrote that Watson had to carry him with her performance . Next was Harry Potter and the Goblet of Fire in 2005 . The film was the second @-@ highest grossing Harry Potter
2x Completions - | film | , |
4x Completions - | film | the | , | and |
8x Completions - | the | film | of | , | . | in | and | series |
16x Completions - | grossing | the | , | worldwide | and | $ | in | . | million | of | @-@ | highest | grossed | film | Part | @.@ |
64x Completions - | . | Harry | the | , | and | of | in | a | " | film | was | to | 's | @-@ | The | that | Watson | Gob@@ | Potter | for | rint | Rowling | her | as | Prison@@ | performance | with | Columbus | an | on | by | she | it | Stone | had | Times | Phoenix | at | Best | Radcliffe | Half | In | from | book | series | first | grossing | his | one | role | instal@@ | Philosop@@ | but | be | $ | which |) | he | (| ab@@ | is | novel | character |
Reference Text :
set fire to the ship to prevent her from falling into enemy hands . Patriots in small boats sailed out to the burning ship , fired some of its cannons at the British ships , took what stores and loot they could , and retreated shortly before the ship 's powder magazine exploded . = = Aftermath = = The British did not
2x Completions - | the | have |
4x Completions - | the | their | to | fire |
8x Completions - | the | , | to | and | any | of | not | until |
16x Completions - | the | , | . | of | and | to | British | a | was | in | were | that | The | 's | as | on |
64x Completions - | . | the | , | and | of | to | British | ships | a | in | was | on | had | that | The | were | 's | for | with | by | ship | at | from | wounded | as | fleet | sailed | American | his | her | which | they | but | HMS | not | been | an | @-@ | two | after | men | returned | York | battle | harbor | be | fire | = | she | their | killed | Boston | French | it | damage | off | captured | one | guns | In | crew | into | he | schooner |

Table 10.6. Examples of coarse LM completions

Reference Text :
good on Megadeth 's claim to being the world 's state @-@ of @-@ the @-@ art speed metal band " . Musicologist Glenn Pillsbury stated the guitar work on the album was a mixture of Mustaine 's " controlled chaos " and the " technical brilliance " of Marty Friedman . Studio efforts released in the mid- and late 1990s featured songs with compact structures and less complicated riffing . Megadeth 's lyrics often focus on death , war , politics , and religion . The lyricism centers on nihilistic themes , but occasionally deals with topics such as alienation and social problems . The earliest releases featured themes such as occultism , graphic violence
Nearest Neighbors at the 4x Scale
vocalist Jock Cheese and keyboardist / vocalist Eugene de la Hot Croix Bun , and enjoyed a large underground / independent following . Their third album , Machiavelli and the Four Seasons , reached the Australian national top 10 in 1995 . TISM were known for their hybrid of dance music and rock 'n'roll , high @-@ energy live shows and humorous lyrics . TISM 's songs frequently satirised modern culture , celebrities and the entertainment industry , classic literature and art , current affairs , politics and sport . The titles of their songs were often wordplays created by juxtaposing pop culture references with more intellectual ones (for
n ; 1950 's blues artists Guitar Slim , Johnny Watson , and B.B. King ; R & B and doo @-@ wop groups (particularly local <unk> groups) ; and modern jazz . His own heterogeneous ethnic background , and the diverse social and cultural mix in and around greater Los Angeles , were crucial in the formation of Zappa as a practitioner of underground music and of his later distrustful and openly critical attitude towards " mainstream " social , political and musical movements . He frequently lampooned musical fads like psychedelia , rock opera and disco . Television also exerted a strong influence , as demonstrated by quotations from show
magazine , " I am a young adult now , and I think this album shows my growth vocally . " Aaliyah was mastered by Bernie Grundman at his studio in Los Angeles . = = Music and lyrics = = An R & B and neo soul album , Aaliyah featured midtempo funk songs , hip hop @-@ textured uptempo tracks , and slow jams that draw on older soul influences . Along with contemporary urban sounds , its music incorporated Middle @-@ Eastern influences , muted alternative rock , and , particularly on Timbaland 's songs for the album , Latin timbres . " Never No More " mixed both older soul and modern hip hop sounds with
Nearest Neighbors at the 16x Scale
well received by both critics and fans , and was responsible for bringing Slayer to the attention of a mainstream metal audience . Kerrang ! magazine described the record as " the heaviest album of all " . Alongside Anthrax 's Among the Living , Megadeth 's Peace Sells ... but Who 's Buying ? and Metallica 's Master of Puppets , Reign in Blood helped define the sound of the emerging US thrash metal scene in the mid @-@ 1980s , and has remained influential subsequently . Reign in Blood 's release was delayed because of concerns regarding its graphic artwork and lyrical subject matter . The opening track , " Angel of Death " , which refers to Josef
music industry in 1992 , through his vocal contributions on Dr. Dre 's The Chronic . That album is considered to have " transformed the entire sound of West Coast rap " by its development of what later became known as the " G @-@ funk " sound . The Chronic expanded gangsta rap with profanity , anti @-@ authoritarian lyrics and multi @-@ layered samples taken from 1970 's P @-@ Funk records . Snoop Dogg contributed vocals to Dre 's solo single , " Deep Cover " , which led to a high degree of anticipation among hip hop for the release of his own solo album . Doggystyle and The Chronic are associated with
the most innovative popular musicians in America if not the world " but also " the most politically ambitious . Not even in the heyday of [the] Clash has any group come so close to the elusive and perhaps ridiculous ' 60s rock ideal of raising political consciousness with music . " Their music on the album inspired leftist and Afrocentric ideals among rap listeners who were previously exposed to more materialist themes in the music . Reeves said it introduced black consciousness to the " hip @-@ hop youth " of the " post @-@ black power generation " , " as leather African medallions made popular by rappers like P.E. replaced thick gold chains as
Neighbors at the 64x Scale
an album that established the concept for Metallica 's following two records . Colin Larkin , writing in the Encyclopedia of Popular Music , singled out " For Whom the Bell Tolls " as an example of Metallica 's growing music potential . Popoff regards Ride the Lightning as an album where " extreme metal became art " . Megaforce initially printed 75 @,@ 000 copies of the album for the US market , while Music for Nations took care of the European market . By the autumn of 1984 , Ride the Lightning had moved 85 @,@ 000 copies in Europe , resulting in Metallica 's first cover story for British rock magazine Kerrang ! in its December issue
best record of derisive punk rock since Exile on Main St. (1972) by the Rolling Stones . In The New Yorker , Ellen Willis wrote that she learned to appreciate Too Much Too Soon more than New York Dolls after seeing the band perform songs from the former album in concert , particularly " Human Being " and " Puss ' n ' Boots " , while Ron Ross from Phonograph Record magazine said the group 's " easy going ironic sensibility " was expressed " far more amusingly and accessibly " here than on their debut album . Some reviewers were critical of Too Much Too Soon for what they felt was a poorly recorded and overproduced
= Riot Act features a diverse sound , including folk @-@ based and experimental songs . Stephen Thomas Erlewine of AllMusic said " Riot Act is the album that Pearl Jam has been wanting to make since Vitalogy - a muscular art rock record , one that still hits hard but that 's filled with ragged edges and odd detours . " Gossard said " Riot Act really seems to showcase all of our thing . There 's the simple rock songs we could have written in the earlier era , but it covers all the different times and dynamics we 've had and still holds together . " The musical experiments also lead several songs on the album to use alternate

Table 10.7. Nearest neighbors computed using representations obtained at different scales

Scale	Layers	Emb	Q,K,V Proj	QK^T	$\frac{QK^T}{\sqrt{d_k}}V$	FC	LN, Drop Residual	Output + Grad	Hierarchical Representations	Total (GB)
Memory footprint breakdown for a single transformer LM layer at different scales										
1x	1	-	0.150	0.06000	0.100	0.500	0.500	-	-	1.325
4x	1	-	0.038	0.00400	0.025	0.120	0.120	-	-	0.319
16x	1	-	0.009	0.00030	0.006	0.031	0.031	-	-	0.079
64x	1	-	0.002	0.00005	0.002	0.008	0.008	-	-	0.020
Memory footprint breakdown for a 12 layer vanilla transformer										
1x	12	0.05	1.81	0.80531	1.208	6.040	6.644	4.10	-	21.26
Model	-	-	-	-	-	-	-	-	-	0.266
Total	-	-	-	-	-	-	-	-	-	21.52 (20.98)
Memory footprint breakdown for a 30 layer top-down transformer										
1x	8	0.05	1.06	0.41	0.7	3.87	3.52	3.59	0.308	14.27
4x	8	-	0.26	0.025	0.176	0.87	0.96	-	0.175	2.529
16x	7	-	0.05	0.001	0.037	0.18	0.29	-	0.042	0.545
64x	7	-	0.01	0.00006	0.008	0.04	0.05	-	0.009	0.121
Model	-	-	-	-	-	-	-	-	-	0.521
Total	-	-	-	-	-	-	-	-	-	17.98 (18.22)
Memory footprint breakdown for a 23 layer Bottom-up transformer										
1x	8	0.05	1.21	0.536	0.805	4.02	4.42	4.10	0.686	16.19
4x	8	-	0.30	0.033	0.201	1.00	1.10	-	0.013	2.76
16x	7	-	0.06	0.002	0.044	0.22	0.24	-	0.003	0.60
Model	-	-	-	-	-	-	-	-	-	0.427
Total	-	-	-	-	-	-	-	-	-	19.97 (20.41)

Table 10.8. Memory footprint breakdown within different models. The number within brackets in the "Total" rows corresponds to what PyTorch reports versus our computed numbers.

Retina Model
one has the bluegrass and which one has blue grass? "Zane barely resisted the impulse to shake him again. It wouldn't help, but it might help him feel better." I'm not asking where Burns told you you were. "Ty nodded, refusing to comment on that. Zane shook his head and then
reached into his pocket to pull out his phone. He turned it on and pressed the buttons for the Miami Police Department. He had a feeling that Burns had told him about the specific questions he'd asked, and he looked over at Ty. "You ever been to Texas?" he asked. Ty shook his head and looked away, giving Zane the privacy he needed. He glanced at Ty. "I 'll be in touch," he promised, and then turned to walk away. "What?" Zane asked, surprised. "You 're pretty sure of yourself. "Zane grinned and nodded." Not even in the Marines? "Ty shook his head." Never. "Zane stared at him a moment longer, then walked over to the bar and poured himself a beer. He leaned against the wall and watched Ty walk away. He couldn't help but smile. "You don't have to go." Ty stopped and turned, looking back at
top-down Model
I told Dominic silently, I just wigged out. Dominic's gaze met mine, his dark eyes glowing, his hair falling around his cheeks. You told him to get off. He didn't. The prick hasn't even apologized. I don 't think he's going to, I replied calmly, I don
't know what to say . Dominic's voice was stern. "I will not tolerate that kind of behavior from you." I knew I had to look at him, because he was glaring at me . He wasn't being a very nice guy, but I knew he was right. He deserved to be treated like a human being. I'd never seen him so angry. "Dominic," I said softly, trying to soothe him. "I 'm sorry. I didn't mean to make you angry." I tried to make him look at me, but he looked away. "I just don't understand why you do it. You have to know that you're not normal." He sighed and looked back at me. "I know. I 'm sorry. I just don't want to be like that." I sighed, knowing he was right. I didn't want to be like that either. "Well, I 'm going to go check
Vanilla Model
Anna's lips twisted. "Over the past few days I've discovered that there are very few things that are impossible." "He's dead, "Morgana said, as much to convince herself as the annoying pest standing before her. "I watched him die." "You betrayed him. "Morgana's
eyes narrowed. "The king is dead, and the rest of the clan has been murdered."Anna's jaw hardened with a fierce fury."What do you want?" "I want you out of my office. Now. "Anna stiffened , but before she could react, Levet was at her side, his wings flapping furiously in panic. "Sacrebleu, I beg of you. I cannot leave my sister. "Anna glared at the tiny demon. "I know." "I can 't take you with me, "Anna snapped. Levet's wings flapped in panic. "I'm not leaving without Morgana." "You can't be leaving her." "No, I can't." "Then don't leave me alone." "I can 't do it." "You can." "No." Levet's words were swallowed by a sudden roar of anger
Bottom-up Model
door wide and I gasped as I stepped inside. It was more like a studio apartment than a guest bedroom; it was about three times the size of our room back in Seattle. The walls were painted a surprisingly warm shade of gray, with furniture in a deep, contrasting dark cherry. The bedspread was black with intricate
patterns in shades of red, gold, and green. The floors were of polished wood and a thick layer of brown dust covered the walls. The ceiling was vaulted, giving the room a more natural light. There was a huge bay window and an easel in the corner. A large bed and a large, flat-screen TV surrounded it. The only other thing that was obvious was the bathroom. There were two stalls off to the side, the toilet and sink. There was a bathroom to the left of the bedroom, opposite a large, white, marble bathroom. There were two sinks, both full of toilet paper. I walked over to the bed and looked out the window. The door was slightly ajar, and a couple of windows had a view of the lake.

Table 10.9. Samples from different models.

Listing 10.1. top-down Model

```
def TopDownTransformerLM(
    x, hierarchies, downsamplers, upsamplers,
    transformer_layers, linear_layers, decoder
):

"""
Forward pass of the top-down model.

Args:
x: a 2D tensor of token indices (batch size x time steps)
hierarchies: a sorted (descending) list of hierarchies (ex: [16, 4, 1])
downsamplers: a ModuleDict of downsampling functions for each hierarchy
upsamplers: a ModuleDict of upsampling functions for each hierarchy
transformer_layers: a ModuleDict of autoregressive transformer decoders
for each hierarchy
linear_layers: a ModuleDict containing a linear layer for every hierarchy
each linear layer is of size (2 * emb_dim, emb_dim)
decoder: a Linear layer from emb_dim to vocab size

Returns:
loss: a scalar that contains average cross-entropy
"""

# (batch size x time step x embedding dim)
x = WordAndPositionEmbeddings(x)

# Get factor by which to upsample at each layer.
upsample_factors = [cur // next for cur, next in zip(hierarchies, hierarchies[1:] + [1])]

top_hierarchy = hierarchies[0]

# Run transformers from coarsest to finest hierarchy
upsampled_representation = None

for i, hierarchy in enumerate(hierarchies):
    # Downsample the input by a factor equal to the current hierarchy
    # NOTE: Downsampling must be "causal"
    # (ex: if downsampling by 4x, cannot use a kernel of size > 4)
    # NOTE 2: Dowsampler at hierarchy 1 is just the identity function.

    out = downsamplers[hierarchy](x[:, (top_hierarchy - hierarchy):-hierarchy], factor=hierarchy)

    # As input to the transformer, use a learned combination of
    # 1) the upsampled representation from the previous hierarchy
    # 2) the word-level representations downsampled to this scale
    if upsampled_representation is not None:
        out = gelu(linear_layers[hier](concatenate(
            [out, upsampled_representation], dim=2)
        ))

    # Run all transformer layers at this hierarchy
    for layer in transformer_layers[hierarchy]:
        out = layer(out)

    # Upsample representations to the next tier if not the final hierarchy
    if upsample_factors[i] > 1:
        upsampled_representation = upsamplers[hierarchy](
            out, factor=upsample_factors[i]
        )

# Compute LM loss
return CrossEntropy(
    decoder(out), x[:, top_hierarchy:]
)
```

Listing 10.2. Bottom-up Model

```
def BottomupTransformerLM(
    x, hierarchies, downsamplers, aggregation_layer,
    transformer_layers, decoder
):

"""
Forward pass of the Bottom-up model.

Args:
x: a 2D tensor of token indices (batch size x time steps)
hierarchies: a sorted (descending) list of hierarchies (ex: [16, 4, 1])
downsamplers: a ModuleDict of downsampling functions for each hierarchy
transformer_layers: a single transformer layer that aggregates representations from different time scales
transformer_layers: a ModuleDict of autoregressive transformer decoders
for each hierarchy
decoder: a Linear layer from emb_dim to vocab size

Returns:
loss: a scalar that contains average cross-entropy
"""

# (batch size x time step x embedding dim)
x = WordAndPositionEmbeddings(x[:, :-1])

# Get factor by which to downsample at each layer.
downsample_factors = [
    cur // next
    for cur, next in zip(hierarchies[:-1], hierarchies[1:])
][::-1]

representations = []
for i, hierarchy in enumerate(hierarchies[1:][::-1]):
    # Downsample the input by ratio between the next and current time scale
    # NOTE: Downsampling must be "causal"
    # (ex: if downsampling by 4x, cannot use a kernel of size > 4)
    # NOTE 2: Dowsampler at hierarchy 1 is just the identity function.

    out = downsamplers[hierarchy](x, factor=downsample_factors[i])

    # Run all transformer layers at this hierarchy
    for layer in transformer_layers[hierarchy]:
        out = layer(out)

    representations.append(out)

# Aggregate information from different scales with x as query and each representation as key \& value
out = aggregation_layer(x[:, :-1], representations)

# Run transformer layers at the finest scale
for layer in transformer_layers['1']
    out = layer(out)

# Compute LM loss
return CrossEntropy(decoder(out), x[:, 1:])
```

Listing 10.3. Retina Model

```
def RetinaTransformerLM(
    x, hierarchies, downsamplers,
    attention_masks, transformer_layers, decoder
):

"""
Forward pass of the retina model.

Args:
x: a 2D tensor of token indices (batch size x time steps)
hierarchies: a sorted (descending) list of hierarchies (ex: [16, 4, 1])
downsamplers: a ModuleDict of downsampling functions for each hierarchy
attention_masks: a ModuelDict of tensors for each hierarchy that has the attention mask
transformer_layers: a list of autoregressive transformer layers with appropriate attention masks
decoder: a Linear layer from emb_dim to vocab size

Returns:
loss: a scalar that contains average cross-entropy
"""

# (batch size x time step x embedding dim)
x = WordAndPositionEmbeddings(x)

for i, layer in enumerate(transformer_layers):
    # Get representations at all time scales by downsampling
    # NOTE: Downsampling must be "causal"
    # (ex: if downsampling by 4x, cannot use a kernel of size > 4)
    # NOTE 2: Dowsampler at hierarchy 1 is just the identity function.

    representations = [
        downsamplers[hierarchy](x[:, :-1], factor=hierarchy)
        for hierarchy in in hierarchies
    ]

    # Attention masks specify what positions to look at from the representations at different time scales
    x = layer(representations, attention_mask=attention_masks)

# Compute LM loss
return CrossEntropy(decoder(x), x[:, 1])
```

Chapter 11

Conclusion

The articles presented as a part of this thesis are at the intersection of representation learning and generative modeling for natural language processing.

(1) **Representation Learning for Sentences** (Chapter 4): In Subramanian et al. (2018c), we present a multi-task learning approach to learning fixed-length distributed sentence representations that achieve strong results on downstream transfer learning tasks without any model fine-tuning.

(2) **Latent Space Generative Models for Sentences** (Chapter 6): In Subramanian et al. (2018b), we present a way to leverage pre-trained sentence representations from Subramanian et al. (2018c) to train generative models. Specifically, we propose a two-step process where (1) a GAN models the distribution of sentence representations from Subramanian et al. (2018c) and produces synthetic sentence embeddings (2) an autoregressive RNN decoder is trained to reconstruct the contents of the sentence embedding to generate text.

(3) **Controllable Text Generation and "Style Transfer"** (Chapter 8): A large fraction of controllable text generation systems rely on the idea that control over a particular attribute (or style) requires building disentangled representations that separate content and style (Hu et al., 2017; Shen et al., 2017b; Fu et al., 2017). In Subramanian et al. (2018a), we demonstrate that representations produced in previous work that uses domain adversarial training are not disentangled in practice. We then present an approach that does not aim to learn disentangled representations and show that it achieves significantly better results than prior work.

(4) **Multi-Scale Transformer Language Models** (Chapter 10): Over the past few years, Transformers (Vaswani et al., 2017) have emerged as the neural architecture of choice for several tasks in NLP including language modeling. Their use, however, comes at the cost of a large memory footprint, which is quadratic in the length of the sequence since they require storing associations scores between every pair of

positions in a sequence. The community has therefore explored memory-efficient variants using sparsity and compression as inductive biases (Child et al., 2019; Rae et al., 2019). In Subramanian et al. (2020), we explored building transformers that contain representations at multiple time scales as an inductive bias for language modeling of long sequences that have a smaller footprint than a vanilla transformer. We demonstrated that such models have favorable memory footprint vs likelihood trade-offs and can contain many more parameters than vanilla transformers for the same memory and compute budget.

In the subsequent sections, we will discuss some shortcomings of existing generative models and discuss a potential future research direction to address a few of these pitfalls.

11.1. On Memory & Better Inductive Biases for Transformers

A combination of large amounts of data and big transformer language models has yielded significant advances in text generation (Radford et al., 2018, 2019). They are capable of generating fairly coherent and fluent sequences of a few hundred words. However, some of the more ambitious goals in the field such as coherent story generation will require scaling to much longer sequences. This becomes prohibitively hard for transformer models since attending over all of the words in an entire novel is computationally infeasible.

Grammar has a strong local structure and while local attention (Parmar et al., 2018; Child et al., 2019) is a good inductive bias to generate fluent text, long-term coherence, faithfulness and factual consistency to the plot of a story are hard to achieve without attending over the entire history. Recent work that reduces the memory footprint of self-attention from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$ or $\mathcal{O}(N)$ (Tay et al. (2020b) present a survey of such work) is promising, however, it remains unclear if they come at the cost of expressiveness and underperform global attention (Tay et al., 2020a). A potential alternative is better memory structures - instead of trying to fit very long sequences into memory for backpropagation, we can cache old activations (that receive no gradient) and retrieve a small subset of them on the fly (Khandelwal et al., 2019b) or use training examples as non-parametric memory that can be retrieved (Guu et al., 2020) conditioned on the current input sequence.

As models get larger with hundreds of billions of parameters, (Brown et al., 2020; Fedus et al., 2021) the axes along which to scale are important to consider. Scaling purely by adding parameters to the model has pitfalls - they are hard to adapt to data that is constantly changing (Lazaridou et al., 2021). For example, a language model has to adapt to changing facts about the world, such as the presidents of countries, that evolve with time. Explicit separation of model parameters and external memory allows one to edit memories while keeping parameters fixed to generalize better to non-static data.

11.2. Modeling Long-tail, Low-frequency Phenomena, and Systematic Generalization

While large-scale transformer language models are often able to generate grammatically accurate language, they often struggle to adequately model phenomena that are infrequent in the training data. For example, a language model that reads a few documents about suitcases may not be able to reliably learn about the ways in which it can be used in the real world, and as a consequence, incorrectly generate sentences that are not plausible, example GPT-2 completes the prefix "I used a suitcase to" with "get up the stairs ...". Although grammatically correct, it is very likely implausible. In Subramanian et al. (2019), we built abstractive document summarization systems using such large-scale transformer language models and like several pieces of work have pointed out, our system often generated factually incorrect summaries and hallucinated details not present in the input document. The community at present relies largely on automatic metrics that compute some form of lexical overlap between model-generated outputs and one or more references such as BLEU or ROUGE to evaluate conditional text generation systems. While correlated with human judgments of summarization quality, they fail to adequately measure things such as the factual correctness of summaries. Some targeted evaluations of factual correctness (Kryściński et al., 2019; Wang et al., 2020) are gaining traction, but they are nonetheless still specific to summarization, and more general methods to understand the shortcomings of language models will help inform better model design. Perplexity or log-likelihoods, that are commonly used to evaluate language models weigh the likelihood scores of each word equally. Models that have better likelihoods may do so just by better modeling high-frequency words. We observed this in Subramanian et al. (2020), where transformers that use local attention achieve likelihoods close to global attention models but produce poor samples. Looking at the breakdown of likelihoods by word frequency, however, revealed that local attention did as well and sometimes better than global attention with high-frequency words but much worse with low-frequency ones.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *arXiv preprint arXiv:1608.04207*, 2016.

Guozhong An. The effects of adding noise during backpropagation training on a generalization performance. *Neural computation*, 8(3):643–674, 1996.

Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. 2016.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. Unsupervised neural machine translation. In *International Conference on Learning Representations (ICLR)*, 2018.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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

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

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, pp. 688–699, 2019.

James K Baker. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65(S1):S132–S132, 1979.

Jonathan Baxter et al. A model of inductive bias learning. *J. Artif. Intell. Res.(JAIR)*, 12 (149-198):3, 2000.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James R Glass. What do neural machine translation models learn about morphology? In *ACL (1)*, 2017.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

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

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pp. 153–160, 2007.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35 (8):1798–1828, 2013.

Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, pp. 4349–4357, 2016.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015a.

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

Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. A fast unified model for parsing and sentence understanding. *arXiv preprint arXiv:1603.06021*, 2016.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576*, 2016.

Peter F Brown, Stephen A Della Pietra, Vincent J Della Pietra, and Robert L Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Peter Burt and Edward Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on communications*, 31(4):532–540, 1983.

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. Language gans falling short. *arXiv preprint arXiv:1811.02549*, 2018.

Keith Carlson, Allen Riddell, and Daniel Rockmore. Zero-shot style transfer in text using recurrent neural networks. *arXiv preprint arXiv:1711.04731*, 2017.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4960–4964. IEEE, 2016.

Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*, 2017.

Mickaël Chen, Ludovic Denoyer, and Thierry Artières. Multi-view data generation without view supervision. In *ICLR 2018*, 2017a.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pp. 1657–1668, 2017b.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2172–2180, 2016.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

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.

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

Ondřej Cífka, Aliaksei Severyn, Enrique Alfonseca, and Katja Filippova. Eval all, trust a few, do wrong to none: Comparing sentence generation models. *arXiv preprint arXiv:1804.07972*, 2018.

Stephen Clark and Stephen Pulman. Combining symbolic and distributional models of meaning. In *Proceedings of the AAAI Spring Symposium on Quantum Interaction ,Stanford ,CA ,2007*, 2007.

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.

Alexi Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *arXiv preprint arXiv:1805.01070*, 2018.

Alexis Conneau and Douwe Kiela. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*, 2018.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017a.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*, 2017b.

Gonçalo M Correia, Vlad Niculae, and André FT Martins. Adaptively sparse transformers. *arXiv preprint arXiv:1909.00015*, 2019.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc V Le. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *arXiv preprint arXiv:2006.03236*, 2020.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: a simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*, 2019.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 933–941. JMLR. org, 2017.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pp. 1486–1494, 2015.

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.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. Multi-task learning for multiple language translation. In *ACL (1)*, pp. 1723–1732, 2015.

Cicero Nogueira dos Santos, Igor Melnyk, and Inkit Padhi. Fighting offensive language on social media with unsupervised text style transfer. *arXiv preprint arXiv:1805.07685*, 2018.

Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*, 2018.

David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2650–2658, 2015.

Salah El Hihi and Yoshua Bengio. Hierarchical recurrent neural networks for long-term dependencies. In *Advances in neural information processing systems*, pp. 493–499, 1996.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Jesse Engel, Matthew Hoffman, and Adam Roberts. Latent constraints: Learning to generate conditionally from unconditional generative models. *arXiv preprint arXiv:1711.05772*, 2017.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*, 2018.

Manaal Faruqui and Chris Dyer. Community evaluation and exchange of word vectors at wordvectors. org. 2014.

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

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021.

Jessica Ficler and Yoav Goldberg. Controlling linguistic style aspects in neural language generation. *arXiv preprint arXiv:1707.02633*, 2017.

John R Firth. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*, 1957.

Jeremy Freeman and Eero P Simoncelli. Metamers of the ventral stream. *Nature neuroscience*, 14(9):1195, 2011.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. Style transfer in text: Exploration and evaluation. *arXiv preprint arXiv:1711.06861*, 2017.

Kunihiko Fukushima, Sei Miyake, and Takayuki Ito. Neocognitron: A neural network model for a mechanism of visual pattern recognition. *IEEE transactions on systems, man, and cybernetics*, (5):826–834, 1983.

Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. Unsupervised learning of sentence representations using convolutional neural networks. *arXiv preprint arXiv:1611.07897*, 2016.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

Vikas K Garg, Inderjit S Dhillon, and Hsiang-Fu Yu. Multiresolution transformer networks: Recurrence is not essential for modeling hierarchical structure. *arXiv preprint arXiv:1908.10408*, 2019.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. pp. 1243–1252, 2017.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Irving J Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237–264, 1953.

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*, pp. 2672–2680, 2014.

Anirudh Goyal, Nan Rosemary Ke, Alex Lamb, R Devon Hjelm, Chris Pal, Joelle Pineau, and Yoshua Bengio. Actual: Actor-critic under adversarial learning. *arXiv preprint arXiv:1711.04755*, 2017a.

Anirudh Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Ke, and Yoshua Bengio. Z-forcing: Training stochastic recurrent networks. In *Advances in Neural Information Processing Systems*, pp. 6716–6726, 2017b.

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

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pp. 5769–5779, 2017.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*, 2020.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*, 2016.

Jeff Hawkins and Sandra Blakeslee. *On intelligence: How a new understanding of the brain will lead to the creation of truly intelligent machines.* Macmillan, 2007.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pp. 820–828, 2016a.

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

Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pp. 507–517. International World Wide Web Conferences Steering Committee, 2016.

Kenneth Heafield. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pp. 187–197. Association for Computational Linguistics, 2011.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *arXiv preprint arXiv:1706.08500*, 2017.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations (ICLR)*, 2017.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. Learning distributed representations of sentences from unlabelled data. In *HLT-NAACL*, 2016.

Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pp. 44–51. Springer, 2011.

Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1), 1991.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9 (8):1735–1780, November 1997a. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `http://dx.doi.org/10.1162/neco.1997.9.8.1735`.

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

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Toward controlled generation of text. In *International Conference on Machine Learning*, pp. 1587–1596, 2017.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

Sarthak Jain and Byron C Wallace. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3543–3556, 2019.

Stanisław Jastrzebski, Damian Leśniak, and Wojciech Marian Czarnecki. How to evaluate word embeddings? on importance of data efficiency and simple supervised tasks. *arXiv preprint arXiv:1702.02170*, 2017.

Yacine Jernite, Samuel R Bowman, and David Sontag. Discourse-based objectives for fast unsupervised sentence representation learning. *arXiv preprint arXiv:1705.00557*, 2017.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. Disentangled representation learning for non-parallel text style transfer. *arXiv preprint arXiv:1808.04339*, 2018.

MI Jordan. Serial order: a parallel distributed processing approach. technical report, june 1985-march 1986. Technical report, California Univ., San Diego, La Jolla (USA). Inst. for Cognitive Science, 1986.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016a.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016b.

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

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

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.

Urvashi Khandelwal, Kevin Clark, Dan Jurafsky, and Lukasz Kaiser. Sample efficient text summarization using a single pre-trained transformer. In *arXiv:1905.08836v1 [cs.CL] 21 May 2019*, 2019a.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019b.

Yoon Kim, Kelly Zhang, Alexander M Rush, Yann LeCun, et al. Adversarially regularized autoencoders for generating discrete structures. *arXiv preprint arXiv:1706.04223*, 2017.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

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

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2019.

Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pp. 181–184. IEEE, 1995.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Ondrej Bojar Chris Dyer, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL), demo session*, 2007.

Iasonas Kokkinos. Ubernet: Training auniversal'convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. *arXiv preprint arXiv:1609.02132*, 2016.

Vladyslav Kolesnyk, Tim Rocktäschel, and Sebastian Riedel. Generating natural language inference chains. *arXiv preprint arXiv:1606.01404*, 2016.

Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. A clockwork rnn. *arXiv preprint arXiv:1402.3511*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Wojciech Kryściński, Bryan McCann, Caiming Xiong, and Richard Socher. Evaluating the factual consistency of abstractive text summarization. *arXiv preprint arXiv:1910.12840*, 2019.

Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in neural information processing systems*, pp. 2539–2547, 2015.

Matt J Kusner and José Miguel Hernández-Lobato. Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051*, 2016.

Alex M Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pp. 4601–4609, 2016.

Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*, 2019.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*, 2017a.

Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, et al. Fader networks: Manipulating images by sliding attributes. In *Advances in Neural Information Processing Systems*, pp. 5967–5976, 2017b.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Phrase-based & neural unsupervised machine translation. *arXiv preprint arXiv:1804.07755*, 2018.

Angeliki Lazaridou, Adhiguna Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Sebastian Ruder, Dani Yogatama, et al. Pitfalls of static language modelling. *arXiv preprint arXiv:2102.01951*, 2021.

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.

Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pp. 396–404, 1990.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

Juncen Li, Robin Jia, He He, and Percy Liang. Delete, retrieve, generate: A simple approach to sentiment and style transfer. *arXiv preprint arXiv:1804.06437*, 2018.

Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *Advances in neural information processing systems*, pp. 469–477, 2016.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*, 2018.

Yang Liu and Mirella Lapata. Hierarchical transformers for multi-document summarization. *arXiv preprint arXiv:1905.13164*, 2019.

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

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015a.

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

Florian Mai, Nikolaos Pappas, Ivan Montero, Noah A Smith, and James Henderson. Plug and play autoencoders for conditional text generation. *arXiv preprint arXiv:2010.02983*, 2020.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Charles C Margossian. A review of automatic differentiation and its efficient implementation. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(4):e1305, 2019.

Alexander Patrick Mathews, Lexing Xie, and Xuming He. Senticap: Generating image descriptions with sentiments. In *AAAI*, pp. 3574–3580, 2016.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. *arXiv preprint arXiv:1708.00107*, 2017.

Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model. *arXiv preprint arXiv:1612.07837*, 2016.

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

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

Paul Michel and Graham Neubig. Extreme adaptation for personalized neural machine translation. *arXiv preprint arXiv:1805.01817*, 2018.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

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*, pp. 3111–3119, 2013.

Alexander H Miller, Will Feng, Adam Fisch, Jiasen Lu, Dhruv Batra, Antoine Bordes, Devi Parikh, and Jason Weston. Parlai: A dialog research software platform. *arXiv preprint arXiv:1705.06476*, 2017.

Marvin Minsky and Seymour Papert. An introduction to computational geometry. *Cambridge tiass., HIT*, 1969.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *ACL*, pp. 236–244, 2008.

Tom M Mitchell et al. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45(37):870–877, 1997.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. Semantic specialisation of distributional word vector spaces using monolingual and cross-lingual constraints. *arXiv preprint arXiv:1706.00374*, 2017.

Tsendsuren Munkhdalai and Hong Yu. Neural semantic encoders. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 1, pp. 397. NIH Public Access, 2017.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

Anh Nguyen, Jason Yosinski, Yoshua Bengio, Alexey Dosovitskiy, and Jeff Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. *arXiv preprint arXiv:1612.00005*, 2016.

Allen Nie, Erin D Bennett, and Noah D Goodman. Dissent: Sentence representation learning from explicit discourse relations. *arXiv preprint arXiv:1710.04334*, 2017.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pp. 271–279, 2016.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*, 2017.

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

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*, 2016.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. *arXiv preprint arXiv:1802.05751*, 2018.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *CoRR, abs/1211.5063*, 2(417):1, 2012.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. pp. 1310–1318, 2013.

Denis G Pelli and Katharine A Tillman. The uncrowded window of object recognition. *Nature neuroscience*, 11(10):1129, 2008.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pp. 1532–1543, 2014.

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.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

Ben Poole, Jascha Sohl-Dickstein, and Surya Ganguli. Analyzing noise in autoencoders and deep networks. *CoRR*, abs/1406.1831, 2014. URL `http://arxiv.org/abs/1406.1831`.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. Style transfer through back-translation. *arXiv preprint arXiv:1804.09000*, 2018.

Ofir Press, Amir Bar, Ben Bogin, Jonathan Berant, and Lior Wolf. Language generation with recurrent generative adversarial networks without pre-training. *arXiv preprint arXiv:1706.01399*, 2017.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

Sai Rajeswar, Sandeep Subramanian, Francis Dutil, Christopher Pal, and Aaron Courville. Adversarial generation of natural language. *arXiv preprint arXiv:1705.10929*, 2017.

Sravana Reddy and Kevin Knight. Obfuscating gender in social media writing. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pp. 17–26, 2016.

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pp. 833–840. Omnipress, 2011.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *CoRR*, abs/1509.00685, 2015. URL `http://arxiv.`

`org/abs/1509.00685`.

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*, pp. 2234–2242, 2016.

Chinnadhurai Sankar, Sandeep Subramanian, Christopher Pal, Sarath Chandar, and Yoshua Bengio. Do neural dialog systems use the conversation history effectively? an empirical study. *arXiv preprint arXiv:1906.01603*, 2019.

Jürgen Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

Stanislau Semeniuta, Aliaksei Severyn, and Sylvain Gelly. On accurate evaluation of gans for language generation. *arXiv preprint arXiv:1806.04936*, 2018.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 86–96, 2015a.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 1715–1725, 2015b.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. 2016.

Samira Shabanian, Devansh Arpit, Adam Trischler, and Yoshua Bengio. Variational bi-lstms. *arXiv preprint arXiv:1711.05717*, 2017.

Dinghan Shen, Yizhe Zhang, Ricardo Henao, Qinliang Su, and Lawrence Carin. Deconvolutional latent-variable model for text sequence matching. *arXiv preprint arXiv:1709.07109*, 2017a.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pp. 6833–6844, 2017b.

Yikang Shen, Shawn Tan, Chin-Wei Huang, and Aaron Courville. Generating contradictory, neutral, and entailing sentences. *arXiv preprint arXiv:1803.02710*, 2018.

Xing Shi, Inkit Padhi, and Kevin Knight. Does string-based neural mt learn source syntax? In *EMNLP*, pp. 1526–1534, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing*

Systems, pp. 3483–3491, 2015.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 553–562, 2015.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

Sandeep Subramanian, Guillaume Lample, Eric Michael Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. Multiple-attribute text style transfer. *arXiv preprint arXiv:1811.00552*, 2018a.

Sandeep Subramanian, Sai Rajeswar, Alessandro Sordoni, Adam Trischler, Aaron Courville, and Christopher Pal. Towards text generation with adversarially learned neural outlines. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 7562–7574, 2018b.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*, 2018c.

Sandeep Subramanian, Raymond Li, Jonathan Pilault, and Christopher Pal. On extractive and abstractive neural document summarization with transformer language models. *arXiv preprint arXiv:1909.03186*, 2019.

Sandeep Subramanian, Ronan Collobert, Marc'Aurelio Ranzato, and Y-Lan Boureau. Multi-scale transformer language models. *arXiv preprint arXiv:2005.00581*, 2020.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. Adaptive attention span in transformers. *arXiv preprint arXiv:1905.07799*, 2019.

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

Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016.

Shuai Tang, Hailin Jin, Chen Fang, Zhaowen Wang, and Virginia R de Sa. Rethinking skip-thought: A neighborhood based approach. *arXiv preprint arXiv:1706.03146*, 2017.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020a.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2020b.

Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017.

Gaurav Singh Tomar, Thyago Duque, Oscar Täckström, Jakob Uszkoreit, and Dipanjan Das. Neural paraphrase identification of questions with noisy pretraining. *arXiv preprint arXiv:1704.04565*, 2017.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. Evaluation of word vector representations by subspace alignment. 2015.

Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Vladimir Vapnik. Pattern recognition using generalized portrait method. *Automation and remote control*, 24:774–780, 1963.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, 2017.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103. ACM, 2008.

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

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pp. 2773–2781, 2015.

Thomas SA Wallis, Christina M Funke, Alexander S Ecker, Leon A Gatys, Felix A Wichmann, and Matthias Bethge. Image content is more important than bouma's law for scene metamers. *ELife*, 8:e42512, 2019.

Alex Wang, Amapreet 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.

Alex Wang, Kyunghyun Cho, and Mike Lewis. Asking and answering questions to evaluate the factual consistency of summaries. *arXiv preprint arXiv:2004.04228*, 2020.

Ke Wang, Hang Hua, and Xiaojun Wan. Controllable unsupervised text attribute transfer via editing entangled latent representation. In *Advances in Neural Information Processing Systems*, pp. 11036–11046, 2019.

Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*, 2017.

Warren Weaver et al. Translation. *Machine translation of languages*, 14(15-23):10, 1955.

Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*, 2019.

Paul Werbos. Beyond regression:" new tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*, 1974.

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*, 2015.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Charagram: Embedding words and sentences via character n-grams. *arXiv preprint arXiv:1607.02789*, 2016.

Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pp. 5–32. Springer, 1992.

Terry Winograd. Understanding natural language. *Cognitive psychology*, 3(1):1–191, 1972.

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

Jingjing Xu, Xu Sun, Qi Zeng, Xuancheng Ren, Xiaodong Zhang, Houfeng Wang, and Wenjie Li. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. *arXiv preprint arXiv:1805.05181*, 2018.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pp. 2048–2057, 2015.

Jimei Yang, Scott E Reed, Ming-Hsuan Yang, and Honglak Lee. Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In *Advances in Neural Information Processing Systems*, pp. 1099–1107, 2015.

Zichao Yang, Zhiting Hu, Chris Dyer, Eric P Xing, and Taylor Berg-Kirkpatrick. Unsupervised text style transfer using language models as discriminators. *arXiv preprint arXiv:1805.11749*, 2018.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pp. 2852–2858, 2017.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *arXiv preprint arXiv:2007.14062*, 2020.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pp. 649–657, 2015.

Ye Zhang, Nan Ding, and Radu Soricut. Shaped: Shared-private encoder-decoder for text style adaptation. *arXiv preprint arXiv:1804.04093*, 2018a.

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

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

Yanpeng Zhao, Wei Bi, Deng Cai, Xiaojiang Liu, Kewei Tu, and Shuming Shi. Language style transfer from sentences with arbitrary unknown styles. *arXiv preprint arXiv:1808.04071*, 2018.

Yinhe Zheng, Zikai Chen, Rongsheng Zhang, Shilei Huang, Xiaoxi Mao, and Minlie Huang. Stylized dialogue response generation using stylized unpaired texts. *arXiv preprint arXiv:2009.12719*, 2020.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639*, 2016.

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

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 *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.