Explorations In Curriculum Learning Methods For Training Language Models

Daniel Campos

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2020

Committee:

Shane Steinert-Threlkeld, Chair

Emma Strubell

Program Authorized to Offer Degree:
Department of Linguistics

University of Washington

Abstract

Explorations In Curriculum Learning Methods For Training Language Models

Daniel Campos

Chair of the Supervisory Committee:
Assistant Professor Shane Steinert-Threlkeld
Department of Linguistics

Understanding language depending on the context of its usage has always been one of the core goals of natural language processing. Recently, contextual word representations created by language models like ELMo, BERT, ELECTRA, and RoBERTA have provided robust representations of natural language which serve as the language understanding component for a diverse range of downstream tasks like information retrieval, question answering, and information extraction. Curriculum learning is a method that employs a structured training regime instead of the traditional random sampling. Research areas like computer vision and machine translation have used curriculum learning methods in model training to improve model training speed and model performance. While language models have proven transformational for the natural language processing community, these models have proven expensive, energy-intensive, and challenging to train, which has inspired researchers to explore new training methods. In this thesis, we explore the effect of curriculum learning in the training of language models. Using wikitext-2 and wikitext-103 textual datasets and evaluating word representation transfer learning on the GLUE Benchmark, we find that curriculum learning methods produce models that outperform their traditionally trained counterparts when the training corpus is small, but as the training corpora scale, curriculum methods become less effective than traditional stochastic sampling.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Number Page

LIST OF TABLES

Table Number Page

GLOSSARY

AUTO ENCODING (AE): A neural network which is used to learn an efficient representation of underlying data distribution by leveraging dimensional reduction to ignore noise in a data. In Language models this typically is implemented by having the network learn to reconstruct a corrupted input.

AUTO REGRESSIVE (AR): A neural network which is used to learn an efficient representation of underlying data distribution by leveraging conditional probability to predict some input based on previous input. In Language models this typically is implemented by having the network learn to predict the next token in a sequence.

COMPETENCE BASED CURRICULUM (CBC): A method of curriculum learning where the samples in the training data are assigned a notion of difficulty and the model training on this data is gradually allowed to sample from a progressively more difficult dataset.

CUMULATIVE DENSITY FUNCTION (CDF): A statistical method used for modeling the distribution of some underlying data using a mathematical function and probability density.

CURRICULUM LEARNING (CL): A machine learning training method which introduces structure into the training regime with the goal of increasing model performance or sample efficiency.

CONVOLUTIONAL NEURAL NETWORK (CNN): A type of artificial neuron which applies a convolution to certain portion of input data to extract a stronger signal which is commonly used in computer vision.

CORPUS REPLACEMENT STYLE (CRS): A method of curriculum learning where less common parts of the corpus are replaced to simplify the training data. This method is run independent of model training and is reusable across models which makes it computationally efficient.

COMPUTER VISION (CV): A field of research which is focused on producing methods which computers can leverage to understand visual inputs.

DEPENDENCY PARSE (DEP): A method for analyzing the grammatical structure of a sentence and establishing relation between words by creating a tree structure which represents how words modify other words.

DEEP NEURAL NETWORK (DNN): A connection of artificial neurons which are arranged in consecutive layers which is commonly used in machine learning to learn some underlying function in a target domain.

GENERAL LINGUISTIC UNDERSTANDING EVALUATION BENCHMARK (GLUE): A collection of 11 natural language tasks which is used to evaluate how well a model captures specific aspects of natural language.

INFORMATION RETRIEVAL (IR): A field of research which is focused on the discovery of documents, image, web-pages, etc. based on user intent.

INFORMATION EXTRACTION (IE): A field of research which is focused on extracting information from many sources and organizing it in a knowledge-base or knowledge-graph and the application of said representations in downstream tasks.

LANGUAGE MODEL (LM): A language representation gained by modeling the language in a corpus using statistical methods such as conditional probability.

LANGUAGE REPRESENTATION (LR): A form of representing the underlying properties of spoken or written language which is usually employed to allow a system to use some form of natural language.

LONG SHORT TERM MEMORY (LSTM): A type of artificial neuron for neural networks which has a trainable parameter which controls what information is forgotten between states.

MASKED LANGUAGE MODELING (MLM): An auto encoding language model training objective which masks some portion of a textual input which language model must reconstruct.

MACHINE TRANSLATION (MT): A sub-field of Natural Language Processing research which focuses on building systems which can translate inputs from a source language to a target language.

NATURAL LANGUAGE PROCESSING (NLP): A field of research which is focused on processing, understanding, and interacting with language as used by humans around the globe.

NEURAL MACHINE TRANSLATION (NMT): A sub-field of Machine Translation which leverages Neural Networks to create systems to translate inputs.

NEURAL NETWORK (NN): A connection of artificial neurons which is commonly used in machine learning to learn some underlying function in a target domain.

NEXT SENTENCE PREDICTION (NSP): An language model training objective where a language model must predict if input sentences are sequentially related in a training corpus.

NEXT WORD PREDICTION (NWP): An auto regressive language model training objective where a model must predict the next token in a sequence give some preceding set of tokens.

PART OF SPEECH (POS): A categorization of tokens based on the syntactic role which they serve.

RECURRENT NEURAL NETWORKS (RNN): A type of artificial neuron for neural networks which contains a hidden state which is used to represent a broader input.

STATE-OF-THE-ART (SOTA): A denotation which defines when a system has the best performance on a given task given some evaluation metric.

TRANSFER LEARNING (TL): A machine learning method which applies the representation learned in training of one target task to improve performance on a different but related task.

QUESTION ANSWERING (QA): A field of research which focuses on answering questions given some input document, usually a textual document.

ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to his family, his Advisor Shane Steinert-Threlkeld, The University of Washington, The City of Seattle, and coffee.

DEDICATION

To my wife, Zoe.

Chapter 1
INTRODUCTION

The ability to understand language has long been a fascination for computer scientists worldwide. Early systems like ELIZA [89] were designed to interact with humans through verbal communication. Despite being rudimentary and simple, these systems quickly caught the world's attention and inspired countless researchers to hopefully one day understand and emulate language. In the decades since, researchers have grown to understand fundamental concepts about language structure and how best to model it. Creating systems that can understand language and preserve its meaning in downstream tasks has long focused on the Natural Language Processing (NLP) community. Leveraging deep neural networks (DNN), researchers have created language representations (LR) that seek to represent many of the nuances of human language. These methods have revolutionized nearly every system that works with human language at a pace that seems to be getting faster and faster. Despite their impressive performance, these models are difficult and expensive to create, recreate, and deploy. We will explore whether the use of curriculum learning (CL) can improve model training and model accuracy.

In the remainder of this chapter, we first briefly describe the problem we want to tackle and how we will address these problems in Section 1.1. Then, we give a brief overview of CL and LR in Section 1.2 and Section 1.3 respectively. We briefly discuss this dissertation's goals and challenges in Section 1.4. We then summarize the contributions of this dissertation in Section 1.5. We then provide a brief outline of the Dissertation structure in Section 1.6 before presenting our statement of originality in Section 1.7.

1.1 Overview

This dissertation addresses the effect of Curriculum Learning (CL) methods on Language Models (LM). The term CL refers to a training procedure where the training data for a machine-learned method is constructed with a deliberately assembled order with the goal of steering training to a more optimal solution sooner. Usually, CL methods use a heuristic to assign a difficulty score to each example and then gradually increase the training data's difficulty. The term LM refers to determining the probability for a given text window. Language modeling can be used to train a Neural Network (NN) to understand language and represent language for downstream Natural Language Processing (NLP) tasks like Question Answering (QA) and Information Retrieval (IR). In this dissertation, we will explore the effect CL methods have on LMs.

1.2 Curriculum Learning

Curricula have long been part of the way humans learn. In our schooling, instructors assemble information and instruction in a way that allows students to learn foundations and more straightforward concepts before they move onto difficult concepts. In traditional machine learning, for each step of the network training, a portion of the training data is randomly sampled. Unlike students in school, networks must learn to understand difficult and simple concepts at the same time. There are many formalizations and implementations for curriculum-like methodologies in machine learning, but for this dissertation, we will build on the neural network focused method introduced by Bengio et al., 2009 [4]. In their formalization, curriculum learning is a method for altering the training data distribution to allow the network to more straightforward concepts before difficult concepts.

1.3 Representing Language

In the frame of language understanding, researchers have long focused on representing all the information that text represents in a way that can be used by other systems. Researchers have explored human-curated representations like WordNet [56] and statistical methods to represent corpora [45]. More recently, statistical methods have tried to produce vector space models where words are represented by a set of vectors in an N-dimensional space. In the early 2010s, methods like Word2Vec [55], and GloVe [59] were introduced, and the broader NLP community found great leverage in vector-based representations of language. Since these word representations mapped each word to a lower-dimensional numeric representation, systems could now take advantage of the closeness of words via methods like cosine similarity. Moreover, these vector representations of words proved to be a powerful method to represent language for many downstream tasks like: Information Retrieval [68], Question Answering [58], Sentiment Analysis [97], Machine Translation [99], etc.

These fixed word vectors drove incredible improvement and research across many fields, but since representations focused on word-level vector values, systems could not represent words in the context they occur. Since these context-independent word representations cannot differentiate between usages, at best each word representation is the average of all of its different uses. Context is particularly essential for words with multiple meanings but the same surface form like *y*. The representation is the same if the context was dealing with an insect (There is a *y* in my soup!), an action (I want to *y* away!), or a description of personal style (You sure look *y*!). To expand word representation out of discrete words, researchers explored methods like having multiple word vectors for each meaning of a word [34], sentence vectors [40], paragraph vectors [44], Gaussian mixtures [1], and sub-word vectors [7]. These methods seek to improve models' ability to capture expressive semantic information and perform better in situations where previous models failed. While these techniques provided incremental improvements, they all continued to lack a robust way to represent sentences outside of the original training corpus, and minor changes in sentence structure could have a large effect on the vector representation.

In explorations on improving word representations, researchers used LM to learn contextual word representations. Language modeling is a well-defined NLP problem that aims to model a statistical distribution of words in a sentence to form a prediction about the next word.

Large DNN with Sesame Street-inspired names [61] [19] [78] have leveraged language modeling to produce LR that enable what could seem like a never-ending stream of new state of the art (SOTA) models on many NLP problems. The success of these models has caused what seems to be a flywheel of new LRs with new implementations like KnowBERT [61], RoBERTa [50], AlBERT [43], ELECTRA [13], and BART [47] each of which produced a gradually improvement.

While these deep learning-based LMs have shown to be excellent methods to enable language understanding in many tasks, the ability to train these models is becoming increasingly computationally expensive [77]. The research in new LMs has been dominated by large research labs with multi-million dollar budgets and vast propriety corpora because it has been shown that model performance is closely tied to the size of training data, model size, and compute used to train [38]. Since gains provided by scaling have continue to hold, the focus on the community has not been on the application of alternative learning methodologies to the established and successful architecture.

1.4 Goals and Challenges

The goal of this dissertation is to explore how CL training methods affect LMs. First, we explore strategies for evaluating the difficulty of a sentence or a line of text. Next, using difficulty scores, we then explore how structured training regimes affect model performance. Model performance will be evaluated by perplexity on the training data and performance on a transfer learning task. Model perplexity is an intrinsic evaluation which use to understand how good a language model is and model perplexity is e^{loss} where e is Euler's number and $loss$ is the model loss over the dataset. A transfer learning task when a model is applied to a novel task which is different to the original training task. The original training task is commonly called pre-training where as the transfer task is called fine-tuning. This dissertation's final goal is to evaluate the effect that various CL methods have on language modeling, which we would seek to apply to the overall architecture and size of LMs.

This dissertation's main challenge is to scope the language model experimentation to allow us to perform the broadest and most representative experiments. Doing an exhaustive search of effects on many language models will prove to be computationally prohibitive, and thus scope will be limited. Ensuring our work is representative of variations in model size, and dataset size will also be challenging.

1.5 Contributions and Findings

Our findings are as follows:

- LM performance as measured by perplexity on the training corpus is not always a predictive measure of transfer learning performance.

- Using CL, it is possible to produce better contextual word representations on small corpora when compared to non-CL methods.

- As corpus size scales, CL methods produce worse contextual representations than non-CL methods.

- A random curriculum in which the structure of the training regime is random can be just as effective as linguistically motivated methods.

Our contributions are as follows:

- We implement two curriculum learning methods specifically for language modeling and show that the methods can be useful for specific training constrains.

- We introduce a methodology for training language models using sampling with replacement and produce a thorough evaluation of effects.

- We produce a comparison of line and sentence based LM training methodologies showing that despite inherent differences in batch size there are no substantial improvement with either mechanism. Since sentence base methods are more computationally inefficient, this our experiments provide support for continued utilization of line based training.

1.6 Structure of This Dissertation

In Chapter 2, we present the background material in language modeling, CL, popular Neural Network architectures, and methods for evaluating LMs. We give a detailed and technical introduction to CL and discuss some recent applications in the NLP domain. We introduce the concept of language modeling, describe various methodological advances and improvements discovered over time, and contrast multiple implementations. We will also provide a brief introduction to large LMs' effects and why focusing on model training efficiency is essential. Finally, we introduce some of the most popular methods for evaluating language modeling along with the datasets and frameworks we will use for our experiment.
In Chapter 3, we provide a detailed explanation of our experimental setup. We propose two curriculum learning strategies building on prior work and explain how we implemented these strategies and how we evaluate the effectiveness. In 4, we discuss the results of our experiments along with a comprehensive analysis of the impact our various methodologies had on model performance. In Chapter 5, we conclude this dissertation. In Chapter 6, we discuss future work.

1.7 Statement of Originality

I declare that this dissertation was composed by myself and that the work it presents is my own, except where otherwise stated.

Chapter 2
PRIOR WORK

In the previous chapter, we briefly introduced the basic ideas of CL and LMs. In this chapter, we provide a fuller treatment of these areas. In particular, in Section 2.1, we describe the building blocks many language models use. Then, in Section 2.2, we give a technical overview of CL methodology focused on its implementations in Machine Translation. In Section 2.3, we provide a technical overview of LMs concentrating on how many current systems have structured the problem and the effect various implementations have on downstream performance. Finally, in Section 2.4, we introduce a suite of evaluation frameworks that have become the standard for understanding the quality of LMs.

2.1 Architecture

In this section, we first briefly review of neural networks in NLP in Section 2.1.1 so as to obtain insight into how system architecture can effect downstream tasks. Then, in Section 2.1.2 we introduce Long Short Term Memory (LSTM) [30] and describe how it works. In Section 2.1.3 we introduce the Transformer [84] and briefly discuss how it differs from LSTM.

2.1.1 Neural Network in NLP

In the last few decades, neural networks have provided language researchers with an effective method of building models that represent the complex phenomena associated with language. In NLP, Recurrent Neural Networks (RNN) are popular building blocks because they can retrain contextual information from previous states. While useful, RNNs are limited in their ability to represent a full sequence by their short-term memory. When presented with long sequences of input, the amount of information needed to represent the total input and how individual components relate grows. Since this state is limited in size as sequences get more prolonged, less context about particular parts of the sequence is kept in the hidden state. The slow loss of prior input's information can be incredibly impactful for for nuances of language like agreement on verges and coreference, which can be many sentences apart. Besides issues in representing inputs with large context windows, RNNs suffer from what is known as the vanishing gradient problem [29]. The vanishing gradient problem focuses on how, when training RNNs, we must update the weights of previous hidden states, but as we do so, their gradients get smaller and smaller. Eventually, the gradient on earlier weights is so small that it practically vanishes, and older weights are not updated. To address this shortcoming and improve modeling long term dependencies architectures like LSTM and the Transformer were introduced.

2.1.2 Long Short Term Memory (LSTM)

The LSTM was proposed by Hochreiter et al., 1997 [30] as a neural method that was better suited to avoid vanishing gradients. LSTMs have a gating mechanism that regulates the flow of information from previous states into the current state. What makes the LSTM unique is that it has forget gates that are trainable parameters which choose what information from earlier parts of the input is forgotten. Using these gates, LSTMs can retain relevant information even with large input sizes. Since LSTMs excel at data where their long term dependencies, they have revolutionized the world of time-series prediction [69], speech recognition [27], and neural machine translation (NMT) [35] to name a few.

2.1.3 Transformer

The Transformer is a neural system first proposed by Vaswani et al., 2017 [84] as a system for NMT. The transformer block introduced a way to move from RNNs to an architecture that focuses on the interaction between units in the input data. Attention is a method that selects what portions of the input are related to each other. Instead of keeping a global state like an RNN, Transformers focus on distilling what input items are attended to by other parts of the data. Attention is a learned weight mechanism that represents how much one item in a list is related to others. In NLP, this method has been extremely successful as it allows a model to learn the relative salience of items with regard to every other item in the input. In its initial formulation, the implementation had an encoder and a decoder, consisting of 6 stacked transformer layers. All encoders are identical (but do not share weights) and have two sub-layers: self-attention and a feed-forward network. All decoders are identical (and do not share weights) and consist of 3 sub-layers: self-attention, encoder-decoder attention, and a feed-forward layer.

Vaswani et al., 2017 use multi-head attention to produce a linear projection used to attend to different parts of the input. Since this method does not rely on recurrence or convolution, a positional encoding is introduced to allow the Transformer to understand where in a sentence a word is. For more details about the mechanics of the Transformer, we recommend reading The Illustrated Transformer[1]. In the original implementation, stacked layers of transformers train translation systems for English to French and English to German translation, which results in new SOTA models with $\frac{1}{4}$ the training time.

The Transformer's ability to model long-term dependencies well and efficiently has made it a natural fit for most NLP tasks. As we will cover in succeeding sections, researchers have used this method to produce robust and usable language representations.

2.2 Learning Methods

In this section, we first briefly review some learning methodologies for neural networks in Section 2.2.1 to formalize what learning methods seek to achieve. Then, in Section 2.2 we provide a more in-depth description of curriculum learning and cover some ways it has been using in NLP.

---

[1] https://jalammar.github.io/illustrated-transformer/

2.2.1 Brief Overview of learning methodologies

Neural Networks are usually trained by randomly selecting batches of data in a training corpus. This method has proven to be incredibly robust as it allows the model to learn the data distribution gradually. While useful in eventually learning data distribution, random sampling is unable to build any natural hierarchy or structure quickly. Methods like curriculum learning [4], reinforcement learning [80], and active learning [14] are alternative methods which try to improve model training and accuracy by sampling training examples in a non-random way. Most methods seek to optimize what kind of information a model has access to at each step in training to find better gradients than a random sample. In domains like Generative Adversarial Networks (GAN) [25], training models that generate large images has proven difficult. GAN is a machine learning framework where two NNs, a generator and a discriminator, compete in a zero-sum game. The discriminator, seeks to classify samples as real or synthetic (created by the generator) while the generator, seeks to creates artificial samples which are close to sample from the training dataset. What is unique about GANs is the generator's goal is not to produce an sample that is similar to some real sample but produce a sample which fools the discriminator. This methodology has proven incredibly effective in producing realistic photos and other complex synthetic data in an unsupervised way. Finding structure in the size of images, researchers have found tremendous improvements by slowly increasing the target output size as training progresses [39]. Initially, the Generator produces 2x2 pixel images. Once it and the Discriminator converge, the target output size is increased to 4x4. This method of scaling continues until 4096x4096 pixel images are synthesized. By training in an increasingly entropic way, the final model can learn a better representation with a higher sample efficiency. This work on progressive learning in GANS inspired this dissertation as LMs, much like GANs, are conceptually simple but can prove difficult to train at scale.

2.2.2 Curriculum Learning

While the common usage of CL in computer science begins in 2009 [4], the concept of CL is much older. At its core, CL's vision is the idea of choosing examples to be presented in a specific order to guide the agent to learn quicker than if they had seen samples in random order. Early experiments with RNNs [21] focused on learning grammar suggested that learning of complex grammatical structure improves when the initial examples the models learn with are more straightforward. CL guides the optimization process to converge faster and guide the learner to better local minima and can be thought of as a method of re-weighting the data distribution over model training. Unlike CL, regular random batch sampling emphasizes an equal contribution of each data point without any notion of how common the data point is and if the data point can be used to build a foundational understanding.

In their 2009 paper [4], Bengio et al., 2009 suggest that CL approaches training may act similarly to unsupervised pretraining. The authors explore the effect of CL in three experiments: using a perceptron to learn an equation, shape recognition, and language modeling. In their language modeling task, they modify the training corpus to make it increasingly difficult. This language model is trained using samples with a windows size of 5 tokens sampled from the 631m token Wikipedia corpus. Initially, they remove samples which contain any word that is not in the N most common words (starts with 5,000). After each pass on the corpus, N is increased by 5,000 which means the training corpus gradually gets larger, more difficult, and more representative of the full corpus distribution. After 1 billion updates, the CL method has a loss of 2.78 vs. the non-CL loss of 2.83. The two main issues discussed by the authors in CL are: the computational cost to assemble the batch and the significant amount of data the model can learn from in early epochs is low.

Since this original paper, CL methods have proven effective for many NLP domains, especially in NMT. Wang et al., 2019 [87] expand on the idea of CL as a method of data selection and data augmentation. Their implementation focuses on selecting data relevant to all tasks and disregarding data that may be only applicable to a specific domain and can bring a

2.5 BLEU point improvement vs. non-curriculum implementation. Platanios et al., 2019 [62] introduce the notion of competence-based CL, which is the basis for much of our experimentation. The author's main contribution is building a CL method called competence curriculum, which only controls how long the curriculum lasts before regular training occurs. The authors' approach is 2-stepped: assign a difficulty value to each sample in the training data, and train the model with increasingly more data as its competence improves. In the first stage, a heuristic is applied to rank the training data from easiest to hardest. Using a cumulative density function (CDF), each sample is then given a value from 0 to 1, which equates to how difficult the example is. Then, starting with some initial competence $_0$, the model will train by sampling a training batch from the training data where the sample difficulty is lower than the model's current competence. After each batch is sampled, the model's competence is increased by a preset $_{increment}$ until it is training on the full dataset. A more detailed description as it applies to this dissertation can be found in Chapter 3. In their experiments on NMT, the authors explore the effect of competence-based CL using Transformers and BiLSTMs using two difficulty methods (sentence length and word rarity) and two competence step functions (root and linear) and find that all of their CL implementations outperform their non-CL counterparts. Using the competence curriculum method, the Platanios et al., 2019 can reduce training time by up to 70% and improve BLEU performance by 2.2 points on the WMT dataset compared to non curriculum methods.

2.3 Language Modeling

In Section 2.3.1, we briefly review various language modeling methodologies and why language modeling is so useful for NLP. Then, in Section 2.3.2 introduce ELMo followed by BERT in Section **??**. Then, in Section 2.3.4, we discuss other relevant LMs and how the broader NLP community is using them. Finally, in Section 2.3.5, we discuss some of the effects of training and using large LMs to ground our research's motivation.

2.3.1 What is Language Modeling

Language modeling is a way to assign a probability distribution over some textual representation. In other words, if the task is to model n-grams, the probability of a current input is the probability of a token w_i given the previous i tokens. This is commonly factorized as Equation 2.1 What is Language Modeling:

$$P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, \ldots, w_{i-1}) \approx \prod_{i=1}^{m} P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1}) \qquad (2.1)$$

Language models can be useful methods to represent natural language because they allow models to differentiate meanings of sentences based on context. In other words a model is able to understand that the word 'fly' can mean different things in the sentences: 'You look fly', 'Lets fly away!', 'That is a fly'.

While language modeling is by no means a new concept, it was not until the introduction of Neural Network based LM that these representations were able to serve as general understanding frameworks. Before these Neural Network Language Models (NNLM), most language modeling usually focused on modeling some form of an N-gram where the probability of a word only depends on the previous N-words. Large Neural-Network-based LMs are the first step in an NLP application as a way of turning some form of textual input into a representation in a vector space.

Language models are created using many training objectives, but general models tend to be either auto-encoding (AE), auto-regressive (AR) or some combination of the two. AR models like ELMo [60] or GPT-2 [64] learn a LR by predicting the next token in a sequence. AE models like BERT [19] and ELECTRA [13] learn a LR by reconstructing some portion of a sequence.

2.3.2 ELMo

ELMo is an AR LM that was introduced by Peters et al., 2018 [60] and, in many ways, became the first contextual word representation that saw widespread usage. The name

ELMo represents Embeddings from Language Models and refers to how language modeling can be used to train contextual word embeddings. ELMo is an auto-regressive model built off the success of GloVe [59], and Word2Vec [55] by seeking to be the first stage of textual processing for a variety of NLP tasks. ELMo consists of a character-level convolutional neural network (CNN) followed by two layers of bidirectional LSTMs. The CNN is used to convert words from a text string into raw word vectors, which are then passed to the BiLSTMs to model the whole input. Using a character level CNN, ELMo can capture the inner morphological structure of words, e.g., words like beauty and beautiful are similar when character level convolutions are used.

Each layer receives a forward pass and backward pass over the textual input, which allows the model to read the sentence left-to-right and right-to-left and form representations that understand the whole context of a sentence. The forward pass of the text (reading left to right) allows the model to build context for a word given previous words, while the backward pass (reading right to left) allows the model to build context from the end of the input to the word being modeled. The backward and forward pass are concatenated together. The output of the first biLSTM is passed into the second layer, and then the final ELMo representation is the weighted sum of the raw word vectors and the two intermediate word vectors (outputs of each biLSTM).

ELMo was trained using the Billion Word Corpus [11] and using the unprocessed input as the target for ELMo's language modeling task. The model is trained for ten epochs (complete passes on the corpus), which takes approximately three weeks using three 1080ti GPUs. On average, the authors find that adding ELMo as a text representation layer provides 20% improvement across a diverse set of NLP tasks.

2.3.3 BERT

Building on the success of ELMo, leveraging the transformer architecture [84], and taking the learnings from other contextual word embeddings [32] [65] Devlin et al., 2018 introduced BERT, which stands for Bidirectional Encoder Representations from Transformers. BERT is an AE LM that uses modified stacked Transformer encoders (12 layers for a small model and 24 for large) to build a contextual language representation. Instead of using character-level convolutions or fixed word vectors as a starting point, BERT leverages a piecewise tokenization [91], which sets a vocabulary size of 30,000.

Just like other language models before it, BERT trains using unsupervised pre-training on a large text corpus. Unlike previous models, BERT introduces two new training objectives as a way to steer the model: Masked Language Modeling (MLM) and next sentence prediction (NSP).

MLM reformulates language understanding as a cloze task [82], where the model's goal is to predict what a hidden word in a sentence may be. To train using MLM BERT introduces a new token $[MASK]$ to represent the hidden word. 15% of each the corpus tokens are selected to be replaced of which 80% (12% of the corpus) are replaced with $[MASK]$, 10%(1.5% of the corpus) are replaced with a random token, and the remaining 10% are left alone. When the model finds a $[MASK]$ token, it predicts what the word should be. NSP is a training method inspired by QA systems, which tend to have two sets of sentences to reason on: a query and a context passage. In NSP, the model is fed text, which combines two sentences, A and B, with the unique separation token [SEP]. In 50% of the NSP samples, sentence B directly follows A while in the remaining 50% A, and B are selected at random. The model has a binary training goal if the sentences are next to each other in the original text.

When the BERT architecture and training regime is trained on the Toronto Book Corpus [98] (800m words) + English Wikipedia (2.5 billion words), the authors can create a generalizable contextual word embedding, which since the models release has fine-tuned on countless transfer tasks to produce new SOTA models.

2.3.4 Beyond BERT

Besides BERT and ELMo, there has been considerable research into additional language models. RoBERTa [50] improves on BERT by training on a larger corpus for a longer time. XLNET [92] combines AE and AR while avoiding some of the pitfalls of each method by modifying AR to maximize the expected log-likelihood of a sequence concerning all permutations of factorization order. XLNET also removes the notion of a $[MASK]$ token to avoid training the model with a token that never occurs in text and implements the whole architecture using the Transformer-XL [17]. ALBERT [43] explores the role of size in LM, finding that parameter weights can be shared across layers meaning they can have 18 times fewer parameters and train 1.7x faster than regular BERT all while producing similar language representation to BERT. DistilBERT [70] produces a smaller LM using knowledge distillation resulting in a similar performance to BERT with a 40% smaller model. GPT [65], GPT-2 [64], and GPT-3 [8] build an AR LM more suited toward language generation by using progressively larger models and a modified transformer decoder architecture. ELECTRA [13] produces a model with comparable performance to BERT with substantially shorter training by having the model predict all tokens in a sentence instead of the $[MASK]$ token and by corrupting the input using a Generator similar to that of a GAN. Beyond these few models we mention, countless other optimizations and applications of these large scale NNLM.

2.3.5 Language Model's Impact

In studying the performance of the rapidly growing NNLM, researchers have found that larger models are more sample efficient and reach a higher level of performance with fewer steps [38]. Kaplan et al., 2020 find that the dataset size, model size, and compute used for training all have a power-law relationship with performance as long as the factors grow proportionally. The authors estimate that the best model would have about a trillion parameters, trained on a trillion word corpus using over 100 petaflops.

While there is no debate on the positive impact these large LMs have had on NLP, the broader research community has begun discussing the broader effects of these continually growing language models. A decade ago, most NLP research could be developed and trained on commodity laptops or servers. Competitive research usually requires multiple instances of specialized hardware like GPUs and TPUs [77]. Strubell et al., 2019 broadly studies the energy implications of training these NNLM and estimates that a single training run of a model like GPT-2 can cost upward of $40,000, the architecture search and hyperparameter tuning can be upwards of $3,000,000, and the $C0_2$ released by training one of these models can be similar to the $C0_2$ released in the entire life-cycle of a car. Zhou et al., 2020 [96] introduce HULK to encourage researchers to think about efficiency in every stage of model creation. Looking at the impact of large language models, researchers can infer that some of the most interesting research in NLP will be focused on how to scale model size while balancing the increased cost in doing so.

2.4 Evaluation

To understand what the many LMs are doing, the research community has explored many ways to measure language understanding. By no means are language understanding data sets the first use case of natural language benchmarks, but the explosion of NNLM has provided ample testing grounds to explore what various benchmarks are evaluating and its effect. Our scope is narrow and focuses on the evaluation of NNLM regarding language understanding in English.

The General Language Understanding Evaluation Benchmark (GLUE) [86] is a set of resources focused on the evaluation of natural language understanding systems. This benchmark pools eleven sentence-level language understanding tasks that seek to cover a diverse range of data type, genre, and difficulty as a proxy for true language understanding. The dataset has been built around a leaderboard format to make benchmarking across language understanding systems more straightforward. Its success has since spawned other benchmarking efforts such as SUPERGLUE [85], XTREME [33], and XGLUE [48].

The GLUE dataset tasks include question answering, sentiment analysis, and textual entailment, of which the specifics will now be described. It is worth noting that tasks tend to use differing metrics to account for differences in data distributions. The Corpus of Linguistic Acceptability (COLA) [88] consists of 10657 sentences from 23 linguistics publications where each example has been annotated acceptability (grammaticality). It uses Matthews Correlation Coefficient (MCC) as its evaluation metric. The Stanford Sentiment Treebank (SST-2) [76] is pairs of movie review sentences and human annotations of their sentiment where the goal of the task is to predict the sentiment of a given sentence. Its evaluation metric is accuracy. The Microsoft Research Paraphrase Corpus (MRPC) [20] is a collection of sentence pairs extracted from online news with a human annotation for whether the sentences in the pair are semantically equivalent. MRPC is measured using F1 and Accuracy which are usually separated by a slash or comma. The Quora Question Pairs (QQP) [15] consists of question pairs from the question-answering website Quora that have been annotated for semantic equivalence. QQP is measured using F1 and Accuracy which are usually separated by a slash or comma. The Semantic Textual Similarity (STS) Benchmark [9] is a collection of sentences drawn from news data with a human annotation of similarity score (1-5). STS is measured using Pearson Correlation and Spearman Correlation which are usually separated by a slash or comma. The Multi-Genre Natural Language Inference Corpus (MNLI) [90] is a collection of sentence pairs with textual entailment annotations. MNLI is performance is measured using accuracy. The Stanford Question Answering Dataset (SQUAD) [66] is a dataset consisting of question-paragraph pairs, where one of the sentences in the paragraph contains the answer to the corresponding question. In GLUE, this task has been simplified into a binary classification task where the goal is to determine the sentence contains the answer to the question. This modified question answering task is called Question Natural Language Inference (QNLI) and is evaluated in terms of accuracy. The Recognizing Textual Entailment (RTE) dataset is a collection of some of the textual entailment challenges [16] [3] [24][5] which has been simplified into a binary classification problem. RTE performance is evaluated using accuracy. The Winograd Schema Challenge (WNLI) [46] is a dataset where there are context sentences where ambiguous pronouns have been replaced with each possible referent, and there is a binary classification task if this is the correct referent or not. WNLI performance is evaluated using accuracy. Finally, the Diagnostic Dataset (DX) is a handpicked set of examples from the MNLI corpus used to evaluate individual linguistic phenomena and it uses MCC as its evaluation metric. Using the average of performance on each task a GLUE score is created which provides an overall sorting metric to compare LM performance.

To ensure that NLP model evaluation (of which GLUE is an example) is done in a consistent and reproducible method, the JIANT toolkit was developed [63]. JIANT is an open-source tool for conducting multi-task and transfer learning experiments in English to implement the GLUE benchmark. JIANT builds on the notion of a configuration which provides all settings needed to run and reproduce an experiment in a simple text file. JIANT provides consistent data processing, classifier implementation, and evaluation to ensure that users of the framework can focus on the outputs and not worry about implementing benchmarking tasks like GLUE.

Chapter 3
RESEARCH APPROACH

To understand the effects that CL can have on LM, we will explore various curriculum methods on training, evaluation, and usage in downstream tasks. Our methodology is driven around three main research questions:

1. Does curriculum learning help a language model converge to a more optimal global minimum?

2. Does the language representation learned via curriculum learning improve performance on downstream tasks when compared to non-curriculum methods?

3. Do curriculum learning methods increase model convergence speed in both pre-training and downstream task fine-tuning?

In Section 3.1, we will first describe our experiment's structure as scoped to various implementations of CL. Next, in Section 3.2 we will describe and discuss our methods for curriculum development. Then in Section 3.3 we will discuss how we will train and evaluate our LMs and how we will explore hyperparameter tuning.

3.1 Experiment Structure

Our experimentation strategy is simple: train many models with a fixed structure and hyperparameters using different curriculum. Our implementation of this strategy is achieved by exploring two different curricula: corpus replacement style and competence based curriculum. Once a robust set of models are trained with each curriculum method, all models are evaluated on a held out portion of the training corpus and fine-tuned on the GLUE Benchmark. In the corpus replacement style (CRS) we create multiple distinct training data sets which are are used to train an unmodified LM. In the competence based curriculum (CBC) we first assign a notion of difficulty to each sample in a dataset and then alter the batch sampling method within the language model to sample from a gradually difficult training data. Our CRS method provides few controls but is computationally cheap while the CBC provides ample control at the cost of computational overhead.

3.1.1 Language Model

To optimize how quickly we can train our systems, we will only explore the curriculum's effect on an established successful baseline, ELMo [75]. We leverage the code[1] used for the original ELMo experiments and use the same hyper-parameters reported in their paper. We make two changes to the original implementation: an ability to use a CBC and the introduction of padding token $< PAD >$. In the original implementation, the training loader will load a text file, shuffle all the lines, and then iterate through them. In our implementation of CBC, we load the full corpus, which we do not shuffle, then select a batch at random from the examples that meet our model's current competence. Our implementation changes data sampling to unconstrained random sampling without replacement to sampling with replacement. Since our implementation of competence curriculum is based on sentence-level difficulty, it becomes possible that batches include sentences that are smaller than the defined context length that meets the original ELMo implementation (20 tokens). As a result, we have to introduce a padding token, $< PAD >$, which ensures that each sentence is at least the context window's size. To avoid having the model learn to predict the $< PAD >$ token, we introduce a masking function that sets the loss to 0 for any padding token.

It is worth noting that the introduction this token increases the computational cost to train

[1]https://github.com/allenai/bilm-tf

a model as all padding tokens are essentially lost computation. To calculate the effect of this we counted the amount of padding tokens that are introduced in the wikitext-103 corpus by While we did not compute the cost of using this method for each of our curriculum methods we estimated the inefficiency introduce by this method by calculating the amount of padding tokens introduced. In the wikitext-103 corpus the training corpus has 101,425,658 tokens. When this is parsed into sentences and we sum the modulo of the context window(20) over all sentences we find we must introduce 42,547,653. If we do not split the corpus into sentences we still must introduce 12,204,311 tokens. This means that before even accounting for the cost of the curriculum implementation our model training requires about 12.03 or 41.95 percent more FLOPs.

3.1.2 Datasets

For our training corpus, we leverage two well-established language modeling benchmarks of wikitext-2, and wikitext-103 [54]. These datasets collect verified good and featured articles from English Wikipedia and feature 2 million and 103 million tokens respectively. Each dataset is comprises if full articles with original punctuation, numbers, and case and the only processing is the replacement of all words which occur less than 3 times with a ¡UNK¿ token to represent unknown words . Each corpus has already been tokenized, processed, and split into train, validation, and evaluation components and availible for broad usage. While most other research on language modeling has been focusing on bigger and bigger data, our focus on smaller data allows us to experiment with more curricula. We understand this will limit our model performance relative to current top methods as a smaller corpus limits the model performance [38]. Moreover, these datasets were chosen because they are standard benchmarks for language modeling and sufficient size to train large scale, language models. We believe that the diversity of our two datasets (wikitext-103 is 50x larger than wikitext-2) will allow us to draw broad conclusions about CL independent of data size. More information about corpus size and diversity can be found in Equation 3.1. Information about the corpus size used for other LM is included for comprehensiveness [11].

Corpus Name	vocabulary Size	Tokens	lines	sentences
wikitext=2	33278	2507007	44836	131262
wikitext-103	267735	103690236	1809468	5343947
1B Word Benchmark	793471	829250940	N/A	N/A

Table 3.1: Training Corpus details

As our competence methodology focuses on sentences, we make two versions of the wikitext datasets: sentence-level and line level. For our sentence level corpus, we leverage SPACY.IO's [31] [2] sentence boundary detector to spit the original corpus to one delimited by sentences. The two different corpus splitting mechanisms mean that each of our competence based curriculum is run on four different datasets.

3.1.3 Evaluation

To evaluate our models, we will focus on three aspects of LM performance: performance on the trained task, performance on a downstream task, and the speed at which a model improves its performance on the trained task. The broad goal of our LMs is to represent a given corpus accurately and to do so we will evaluate model perplexity on the held-out portions of our datasets. To measure the quality of our LMs downstream we will use the industry standard GLUE tasks.

3.2 Curriculum Construction Methods

3.2.1 Baseline

Before we study the effect of CL on language modeling we first retrain the public ELMo model on our datasets and evaluate on GLUE. We also download the publicly released

[2] https://spacy.io/

ELMo models and evaluate on GLUE. Our training of ELMo is done on for 10 epochs on the wikitext-2 and wikitext-103.

3.2.2 Corpus Replacement Style

Corpus Replacement Style is inspired by the the Bengio et al., 2009 [4] implementation of CL for LM. In their implementation, they train an AR LM with examples from a corpus with a context window of size 5. In the epoch the model trains on all 5 token spans which only contain the 5,000 most common tokens. For the second epoch, training mirrors the prior epoch but with a threshold of 10,000 most common words. This process continues until the model is training on a full unaltered corpus.

In early experiments we explored an approach modeled after the Bengio et al., 2009 method but as context size of ELMo is much larger performance suffered. Seeking to preserve the separation of model training and training data creation we created what we refer to as Corpus Replace Style (CRS) curriculum. In this implementation we simplify the corpus not by removing training spans but by replacing words less common than the threshold with an ¡UNK¿ token. Threshold selection was done by exploring various increment sizes and initial threshold size and our final most successful methods produced 6 unique datasets for input dataset. To match the training time of our baseline we train one epoch on each corpus and an additional 4 on the unaltered corpus. Details on corpus thresholds can be found in Table 3.2.

Corpus Name	**1**	**2**	**3**	**4**	**5**	**6**	**7-10**
enwiki-2	5,000	10,000	15,000	20,000	25,000	30,000	33,278
enwiki-103	25,000	50,000	75,000	100,000	150,000	250,000	267,735

Table 3.2: Vocabulary size per epoch

3.2.3 Competence Method

Following the work of [62] introduced in Section 2.2.2 we will apply the notion of a CBC to LM training. CBC methods rely on the ability to assign a difficulty score to each sample in the training corpus and use this to only allow the model to train on samples that are easier than its current competence level. A model's competence score is defined by how far along in a training regime the model is.

Our training corpus, X is a collection of sentences S, where each sentence s_i is a sequence of words $s_i = w_o^i; w_1^i; :::; w_n^i$ and each sentence is assigned a difficulty $_{s_i}$. At each step in training model competence is represented by $_t$ where $_0$ represents the initial competence and $_{increment}$ represents how much model competence increases after each training batch. Prior to training, each sentence in the training data has been assigned a difficulty score from 0 to 1 represented by $_{s_i}$. For each training batch, the model is only able to train on samples that have a difficulty where $_{s_i} <= {}_t$.

To keep keep curriculum length for CBC and CRS equal we set our curriculum length to 6 epochs do a grid search on $_{increment}$ and $_0$ values. After our grid search we select the parameters that provide lowest training perplexity and set value $_0 = 0{:}1$ $_0 = 0{:}1$ for wikitext-2 and $_0 = 0{:}01$ $_{increment} = 0{:}00001$ for wikitext-103. The CBC based sampling algorithm is formalized in Algorithm 1.

Algorithm 1: Competence-based curriculum

Result: Model Trained with Competence Based Curriculum

Input: X, $_0$, $_{increment}$;

Compute difficulty, $_{s_i}$ for $s_i \in X$;

Compute Cumulative density of $_{s_i}$;

$_t =$ $_0$;

for *training step t = 1,...,n* **do**

Sample batch b from X such that $_{s_i} <$ $_t$;

Train on batch b;

$_{t+1} =$ $_t +$ $_{increment}$;

end

To understand the effect of CBC we study 8 different curricula per dataset: 2 baseline and six different difficulty heuristics. The first baseline is a curriculum method where we set $_0 = 1$, which means that our model can sample from the entire training dataset. This baseline aims to establish the effect that changing training with sample-without-replacement to sampling-with-replacement has on LM performance. The second baseline is a random curriculum where we sort the file randomly to create our sentence difficulty scores. The goal of this baseline is to establish the effect of any arbitrary curriculum on LM training. The following six heuristics we explore are based on common NLP difficulty metrics, the original CBC paper, and some linguistically motivated difficulties. The heuristics are sentence length, unigram sentence probability, bigram sentence probability, trigram sentence probability, part of speech diversity, and sentence dependency complexity. For each methodology, for each s_i in X, we compute a difficulty value for each sentence $_{s_i}$ and then sort the dataset by this difficulty score. Using the sorted dataset we compute the cumulative density function (CDF) giving each sentence of the difficulty score $_{s_i} \in [0; 1]$. We will now describe each method.

Sentence Length

Formalized in Equation 3.1, this curriculum is built on the idea that is a lot harder to model longer sentences, as longer sentences require better tracking of dependencies. We believe this method would be particularly effective in Transformer based models as it can steer the model into learning how to leverage its multi-headed attention with different sentence lengths.

$$\text{sentence-length-}\ {}_{s_i} = length(s_i). \tag{3.1}$$

Sentence Entropy

Another part of language that can be difficult to model is words with a variety of frequency in the corpora. Models, if assumed to behave like humans, would find it difficult to understand the meaning of a word if they do not see it in a corpus nor have a diversity of usages to infer meaning. Since the statistical strength of training samples with rare words is low and the early model learned word embeddings are likely to have high variance it is likely that exposing a model early to rare words can result in badly estimated representations. To quantify this difficulty we propose producing a sentence entropy for each sentence with respect to its unigram, bigram, and trigram probabilities. These products can be thought of as an approximate naive language modeling as it assumes words are sampled independently. Note, we are not calculating the conditional probability of each word given the preceding N words but the probability of the N-gram given the text corpus. To produce a difficulty score ${}_{s_i}$ we first calculate an n-gram probability for each unigram, bigram, and trigram in the training corpus. Then using this probability we calculate the n-gram difficulty of a input s_i by computing the log product of each n-gram $\in s_i$ as show in Equation 3.2. uc, bc, and tc are the counts of unique unigrams, bigrams, and trigrams in the corpus, C is the corpus, $x \in C$ is a line in the corpus and $w_i \in x$ is a word in a line, and $l(x)$ represents the length of x in n-grams.

$$
\begin{aligned}
p(w_n) &= \frac{\sum_{x \in C} \sum_{n=0}^{l(x)} w_i == w_n}{us} \\
p(w_n; w_m) &= \frac{\sum_{x \in C} \sum_{n=0}^{l(x)-1} (w_i == w_n \& w_{i+1} == w_m)}{bs} \\
p(w_n; w_m; w_j) &= \frac{\sum_{x \in C} \sum_{n=0}^{l(x)-2} (w_i == w_n \& w_{i+1} == w_m \& w_{i+2} == w_j)}{ts} \\
\text{unigram-}\ (s_i) &= \prod_{n=0}^{length(s_i)} \log(p(w_n)) \\
\text{bigram-}\ (s_i) &= \prod_{n=0}^{length(s_i)-1} \log(p(w_{n-1}; w_n)) \\
\text{trigram-}\ {}_{s_i} &= \prod_{n=0}^{length(s_i)-2} \log(p(w_n; w_{n+1}; w_{n+2}))
\end{aligned}
\tag{3.2}
$$

Sentence Dependency Complexity

There are various methods to define sentence complexity but in our experiments we scope complexity to the complexity of a dependency parse. We leverage the language processing framework spacy [3] and for each sentence we generate a dependency parse and starting at the root we measure the depth of the tree. Sentence difficult is formalized in Equation 3.3.

$$\text{dep-}\ {}_{s_i} = \text{depth}(s_i) \tag{3.3}$$

Part Of Speech Diversity

Another core part of language complexity can be derived by the diversity of parts-of-speech in a sentence. We believe that more difficult sentences feature a higher diversity of parts-of-speech (POS). We leverage the part of speech parser from spacy to produce a set of all pos in each sentence. POS Diversity is formalized in Equation 3.4.

$$\text{pos-}\ {}_{s_i} = \text{len}(\text{set}(\text{pos}(s_i))) \tag{3.4}$$

[3] spacy.io

Sentence Vs. Line CBC

As we mentioned prior, our curriculum methods are designed for sentence level sampling but most modern LMs train using a context window of line of text or larger. As a result we apply the CBC methods to both the sentence split corpus and a line delimited corpus. For the line delimited corpus we use the existing line breaks in the wikitext-* corpuses and apply our same heuristics at the line level instead of the sentence level. This means we favor short paragraphs far more in our line corpus than our sentence corpus. In our sentence corpus easy sentences will show up earlier in training while in our line corpus if there is a easy sentence which is part of a long line of text, it will not show up until later in training.

It is worth noting that since the percentage of padding tokens vary between these two methods line based curriculum effectively are larger batches. Since our batch size is limited by our GPU size we instead extend the training time of the sentence based method by roughly 30%. While there are no guarantees on actual update steps, this approximately allows both methods to train on the same amount of tokens as the original ELMo implementation which is $10 *$ tokens in corpus

3.3 Model Training and Evaluation

For model pre-training we follow the original implementation of ELMO and use 2 stacked 4096 dimensional BiLSTMs trained bidirectionally. We use dropout of 0.1, use a batch size of 128, use a context window of 20 tokens, and train for 10 epochs on the full text corpus. In our pre-training we a total of N models. We train 2 baselines (one for each corpus size) using the original ELMo implementation and then train 2 BS models using the same setup. For each of these 4 models they are evaluated on the validation portion of their train corpus at the end of each epoch. For competence based curriculum we train 32 models using our modified implementation: 4 corpuses and 8 curricula per corpus. Since training examples each model sees are different we track model performance over time by evaluating on the validation portion of the train corpus every 100 batches. Training was done using 3 Nvidia

2080 TI GPUs and training on the wikitext-103 corpus takes about a 30 hours and training on wikitext-2 is under an hour.

For model fine-tuning everything was implemented using the JIANT framework. Model weights were dumped from the pretrained models and downloaded from the public original ELMo model. Using these weights, each model is fine tuned on each of the GLUE sub-tasks. Using JIANT we normalized our training to have a batch size of 8, random seed of 42 initial learning rate of 0.0001, dropout of 0.2, and a Multi-layer perceptron with 512 hidden dimensions. Training of each model continues until the model learning rate dips below 0.000001 or the model has trained for 1000 epochs. Then, for each sub task, the best result the model predicted is the GLUE score for that task. Each model fine tuning takes about 8 hours using the same training setup.

To compare model performance across curricula we will look at 3 different results: model perplexity on held out portion of training corpus, how this perplexity changes over time, and model transfer performance on GLUE.

Chapter 4
RESULTS

In the following tables and figures we have introduced various forms of shorthand. We refer to the WikiText-103 corpus as wiki103, wiki2 as wiki2 and the Billion Word Corpus as BWC, -l denotes a model which leverages a line based corpus while -s denotes a model which leverages the sentence based corpus, DEP denotes dependency parse depth, POS denotes part of speech diversity. For the GLUE benchmark evaluation metrics are unique to each task have are described in Section 2.4.

4.1 Corpus Replacement Style Curricula

As covered in previous sections, we implemented the CRS curriculum across the two sizes of wikitext datasets. We evaluate model performance on the validation portion at the end of each epoch, and once the model has trained for ten epochs, we evaluate using JIANT's [63] implemented GLUE baseline. Unless another metric is explicitly mentioned, the GLUE sub-task metric is accuracy.

As seen in Figure 4.1 and Figure 4.2, we see that the baseline method implementations outperform the CRS across the entire training regime in terms of validation perplexity. Initially, while the model is training on the limited corpus, it performs worse on the validation set. As the model reaches the original training data, the model is close to the baseline perplexity but can never pass it. By the 7th epoch, both methods have virtually identical perplexities. Another interesting trend is that when the full corpus is introduced (epoch 6), the validation perplexity briefly increases which leads us to believe that ELMo is learning a general representation of the corpus each epoch. The final observation is that the baseline method reaches low perplexity on a large corpus much faster than the CRS method. The baseline method can achieve a perplexity under 100 by the end of the first epoch, while the CRS method cannot do so until the end of the 5th.

The effect of the curricula looking at GLUE can be found in Table 4.1. First, we find

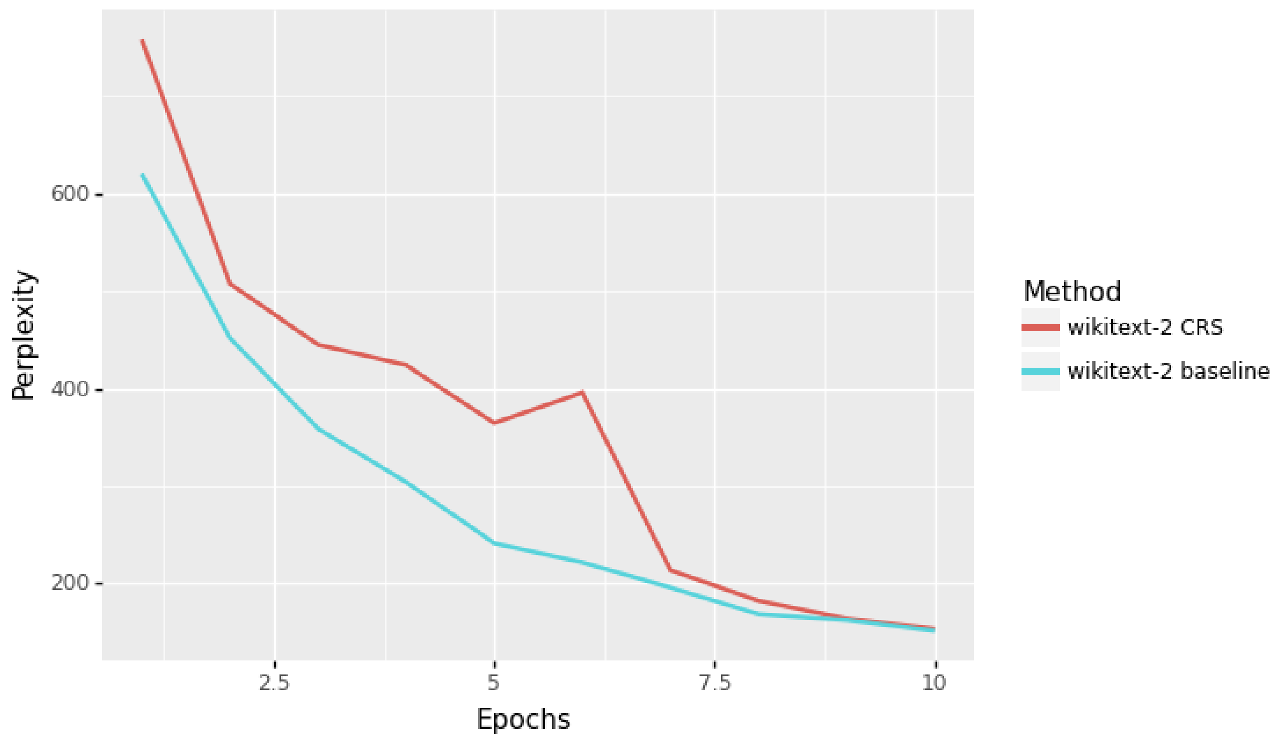

Figure 4.1: Validation perplexity of baselines and replacement methods trained on wikitext-2 measured every epoch.

that the pubic ELMo implementation performs worse on the GLUE dataset than all of our implementations. It is unclear why this is, and we will not focus on this further. Next, we see that when the training corpus is small, the BS implementation outperforms the baseline method by a sizable margin. As the corpus size grows, the baseline performance passes all other methods. There is high variability in individual task scores such that STS-B and the Diagnostic tests are much better with the BS system, while COLA, WNLI, and RTE are much better in the baseline method. One possible cause of this is BS models learn

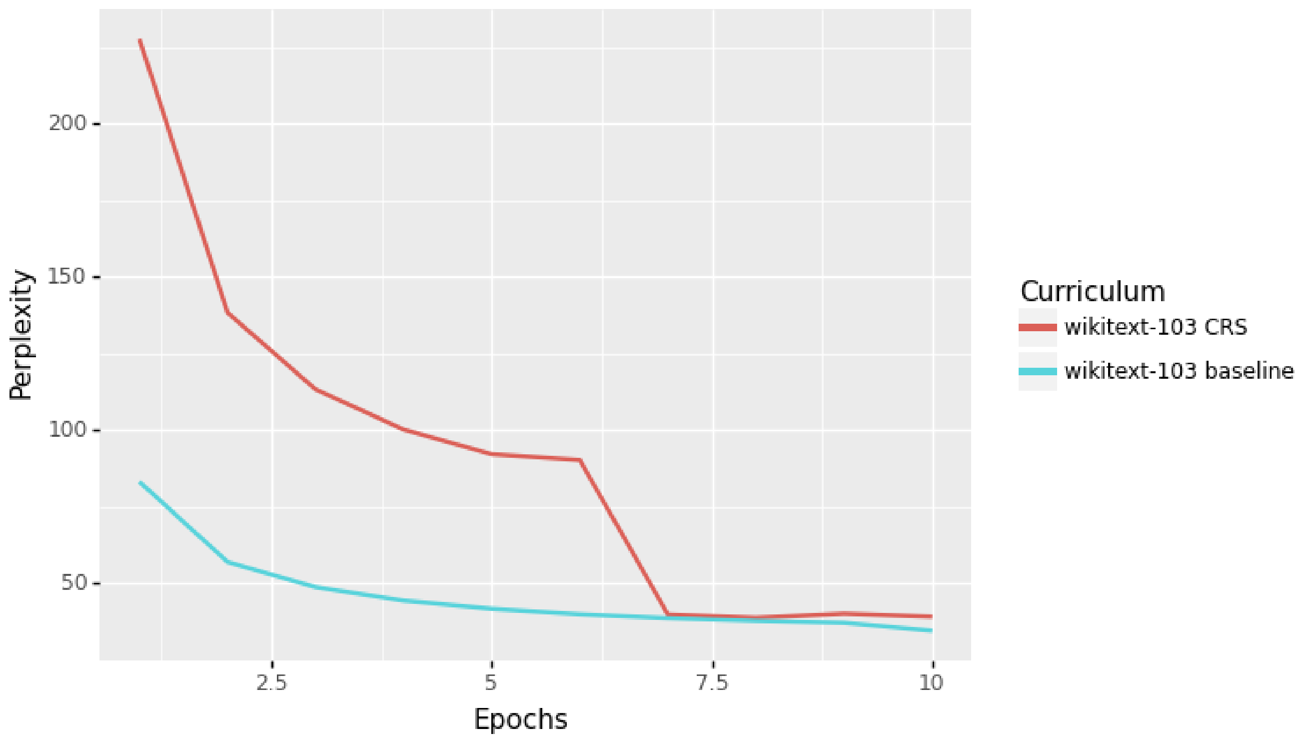

Figure 4.2: Validation perplexity of baselines and replacement methods trained on wikitext-103 measured every epoch.

better representations for binary classifications but not better representations for multi-label classification. If we exclude WNLI, the BS methods outperform the baseline methods by a wide margin. We can also observe various tasks that the small corpus models cannot learn as both wiki2 models achieve 0 on COLA. We believe this may result from the model learning a flipped representation early on and not being able to recover from it. There does not seem to be a clear association in curriculum methods performing worse or better based on the task training data size. On some tasks with small training corpus like RTE the non curriculum methods perform better while in others like STS-B the top model is a CRS method.

Method	**Overall Score**	Cola	SST	MRPC	STS-B	QQP	MNLI	QNLI	RTE	WNLI	DX
Baseline wiki103	**0.671**	**0.281**	**0.862**	0.866/0.801	0.765/0.773	0.716/0.763	0.644	**0.761**	**0.610**	0.535	0.139
CRS wiki103	0.657	0.254	0.852	**0.875/0.816**	**0.794/0.793**	**0.738/0.785**	0.662	0.719	0.588	0.437	**0.162**
Baseline wiki2	0.607	0.06	0.742	0.854/0.789	0.683/0.684	0.697/0.745	0.566	0.726	0.581	**0.563**	0.119
CRS wiki2	0.59	0	0.7	0.846/0.775	0.661/0.663	0.701/0.753	0.585	0.717	0.542	**0.563**	0.13
Baseline BWC	0.595	0	0.852	0.823/0.711	0.547	0.733/0.765	**0.671**	0.719	0.48	**0.563**	0.155

Table 4.1: GLUE results for competence replacement methods and baselines trained on wikitext-2.

4.2 Competence-Based Curricula

4.2.1 Wikitext-2

Looking at performance on the small corpus in Figure 4.3 and Figure 4.4 we see that all the curricula methods start to overfit on the training corpus after about 16 of the 24 epochs (which equates to when the curriculum training is finished). The full data underpinning the figures mentioned above can be found in the Appendix. Despite seeing over-fitting, we see that POS and DEP generally learn some of the lowest perplexity on the validation set. Looking at the difference between sentence-level and line-level training, we see a closer grouping in model performance distribution between sentence vs. line, which we attribute to a common sentence-level structure independent to corpus separation. Finally, we see that the best performance is achieved by the non-curricula baseline (initial competence set to 1) with a perplexity of 770, followed by Random with a perplexity of 2105. Despite the success of these curriculums, all are orders of magnitude above the baseline score of 151. We believe this is caused by the change in dataset distribution caused by our curriculum learning implementation. Even our baseline curricula which has access to the full training corpus does not achieve a high perplexity as the distribution it sees in training differs from the true distribution.

As we move our focus to GLUE results as shown in Table 4.2, we see a very different picture as the curriculum methods generally outperform the baselines by a wide margin. We see that models do not seem to learn any representation that can be used by WNLI, as those results are all virtually identical. Surprisingly it does not seem that a more optimal method for splitting the corpus is apparent as sentence-based training achieves a similar result in line-based training. We believe this means we cannot make a strong generalization that can be made about the effects of what method is used for training. The final point to observe is that nearly every curricula method outperforms the baseline implementation. Perhaps most surprising, random curricula performing second best when measure by overall glue score despite their being no motivation to the structure the model sees.

Overall, when training on wiki2, we find that competency-based curricula methods provide somewhat of a mixed message. While performance on the validation portion is much worse, GLUE performance is much better. Since we believe what is most important in a model is how well the language representations can transfer to downstream tasks, we argue that with small training datasets CBC outperform traditional training regimes.

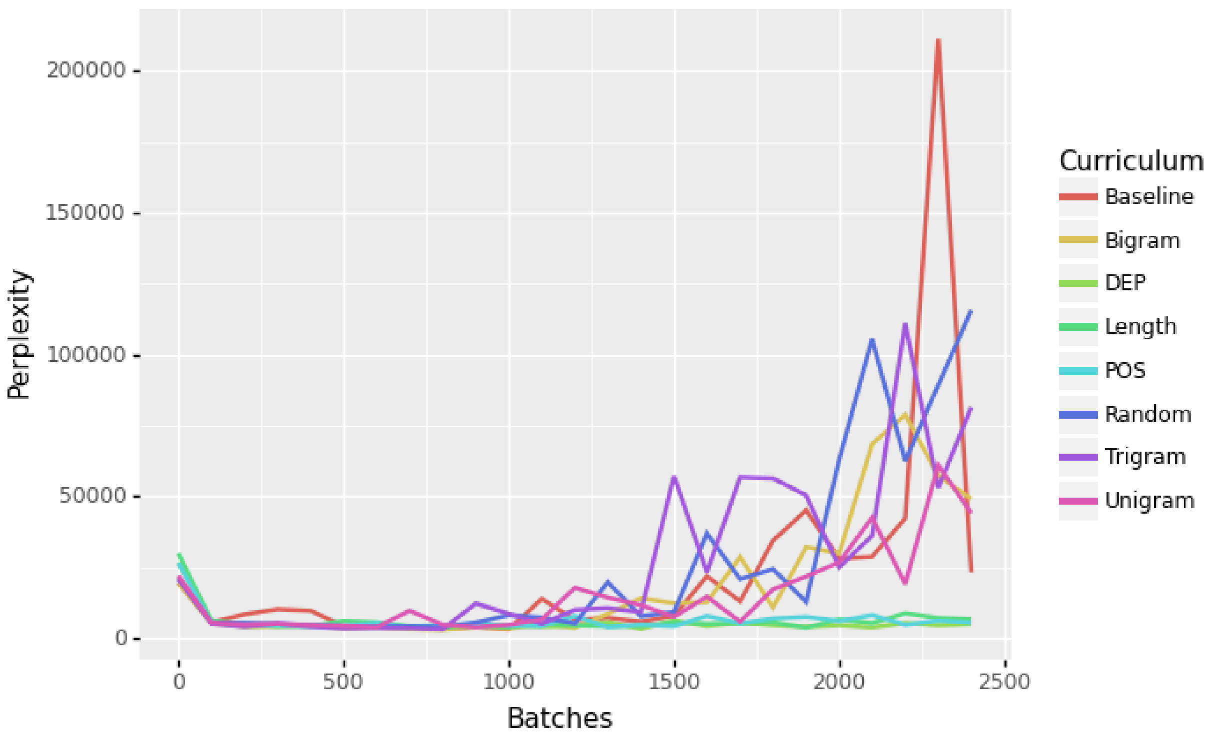

Figure 4.3: Validation perplexity of each curriculum trained on line based wikitext-2 measured every 100 batches.

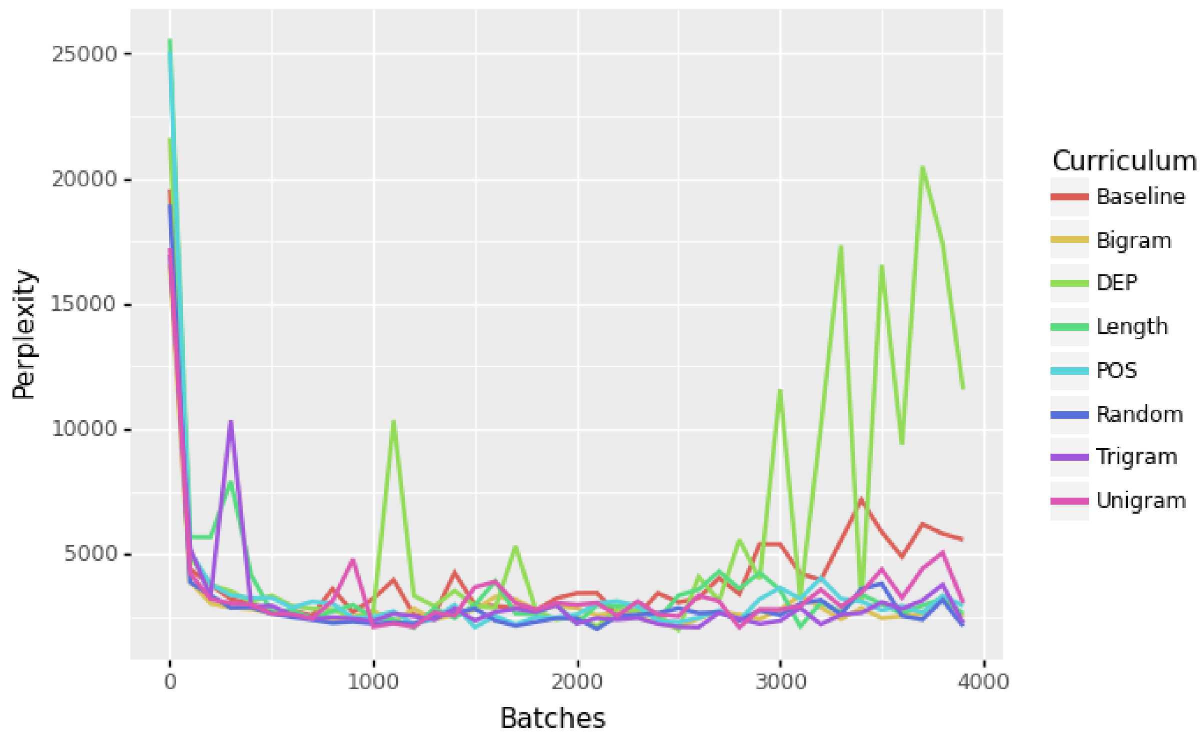

Figure 4.4: Validation perplexity of each curriculum trained on sentence based wikitext-2 measured every 100 batches.

4.2.2 wikitext-103

As we scale to the larger corpus, we find that not all the trends seen on the small data hold. Looking at the results in Figure 4.5, Figure 4.6 and Figure 4.7, we can see that similar to the smaller corpus, none of the curriculum learned models can learn a representation that transfers from their perturbed training to the validation set. Unlike the non-curriculum techniques, which achieve a perplexity of 36, none of the curricula methods ever learn a model with a perplexity under one thousand. We note that similar to what was seen on the wiki2 is on wiki103. N-gram methods have a similar distribution in performance. Also, holding over from wiki2, DEP, LEN, and POS demonstrates a slow and steady improvement in perplexity

Method	Overall	Cola	SST	MRPC	STS-B	QQP	MNLI	QNLI	RTE	WNLI	DX
bigram-s	**0.645**	**0.188**	0.790	0.844/0.775	**0.732/0.733**	0.742/0.788	0.617	0.766	0.570	**0.563**	0.131
random-s	0.644	0.210	0.780	0.847/0.779	0.715/0.712	**0.744**/0.791	**0.616**	0.760	0.570	**0.563**	0.143
unigram-s	0.640	0.209	0.755	0.861/**0.799**	0.731/0.731	0.733/0.778	0.613	0.757	0.545	**0.563**	0.135
pos-s	0.637	0.207	0.765	0.849/0.777	0.714/0.714	0.727/0.781	0.606	0.745	0.567	**0.563**	0.134
dep-s	0.636	0.175	**0.792**	**0.863**/0.797	0.721/0.720	0.727/0.786	0.613	0.749	0.567	**0.521**	0.137
dep-l	0.632	0.190	0.727	0.85/0.782	0.708/0.704	0.735/0.782	0.598	**0.748**	0.581	**0.563**	0.118
unigram-l	0.628	0.175	0.768	0.856/0.782	0.675/0.674	0.738/0.792	0.598	0.746	0.560	**0.549**	0.132
trigram-l	0.627	0.152	0.761	0.838/0.762	0.695/0.691	0.730/0.782	0.616	0.764	0.542	**0.563**	0.144
trigram-s	0.626	0.172	0.788	0.833/0.779	0.730/0.731	**0.744/0.796**	0.620	0.761	0.552	0.437	**0.137**
length-l	0.625	0.185	0.748	0.844/0.770	0.658/0.654	0.725/0.783	0.601	0.748	0.567	**0.563**	0.128
no curricula-s	0.621	0.148	0.747	0.836/0.770	0.706/0.705	0.734/0.780	0.612	0.719	0.538	**0.563**	0.121
bigram-l	0.616	0.175	0.768	0.856/0.782	0.675/0.674	0.738/0.792	0.598	0.746	0.560	**0.437**	0.132
random-l	0.614	0.000	0.763	0.847/0.775	0.699/0.699	0.721/0.784	0.611	0.749	0.578	**0.563**	0.143
CRS	0.607	0.060	0.742	0.854/0.789	0.683/0.684	0.697/0.745	0.566	0.726	0.581	**0.563**	0.119
pos-l	0.607	0.000	0.737	0.842/0.770	0.663/0.660	0.712/0.771	0.610	0.750	0.592	**0.563**	0.155
no curricula-s	0.591	0.069	0.768	0.847/0.765	0.728/0.731	0.722/0.758	0.500	0.718	0.538	0.451	0.107
base	0.590	0.000	0.700	0.846/0.775	0.661/0.663	0.701/0.753	0.585	0.717	0.542	**0.563**	0.130
length-s	0.528	-0.006	0.750	0.805/0.674	0.708/0.710	0.537/0.682	0.326	0.510	**0.592**	0.521	0.008

Table 4.2: GLUE results for competence based curricula methods on trained on wikitext-2.

as their training continues. As the dataset has grown larger, we see increased volatility in perplexity changes across all training methods.

Looking at the result of the transfer tasks in Table 4.3, we find that the trends we see in the smaller corpus no longer hold. The best model in terms of transfer learning is the non-curriculum baseline implementation from the original ELMo implementation. This baseline method outperforms on the overall score and outperforms every other model on tasks like CoLA, where the score is nearly 20% better. Surprisingly, after the baseline method, the trigram curricula appear to generate the best transfer task as both sentence and line-based methods as they are ranked second and third when measured on the average across tasks. We also note that some of the best performing curricula are also those models that had some of the highest perplexities on the training task.

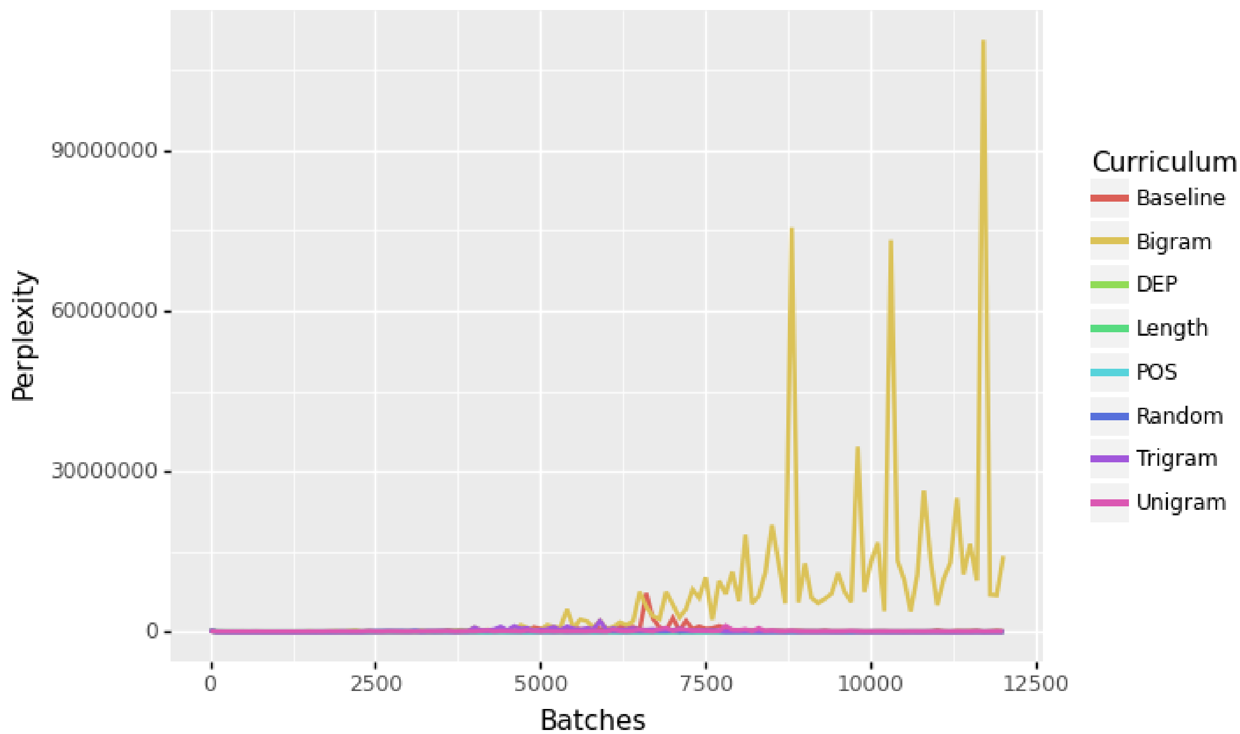

Figure 4.5: Validation perplexity of each curriculum trained on line based wikitext-103 measured every 100 batches.

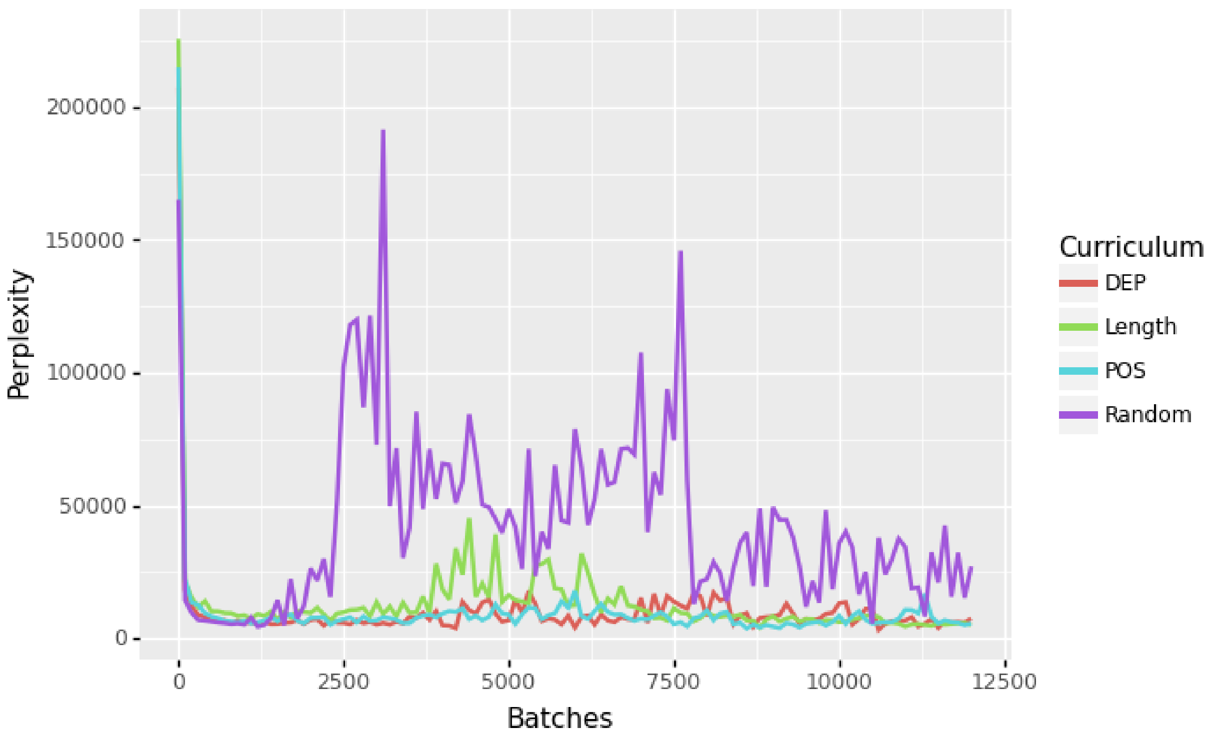

Figure 4.6: Validation perplexity of each curriculum trained on line based wikitext-103 measured every 100 batches. Unigram, bigram, and baseline model performance removed improved interpretation

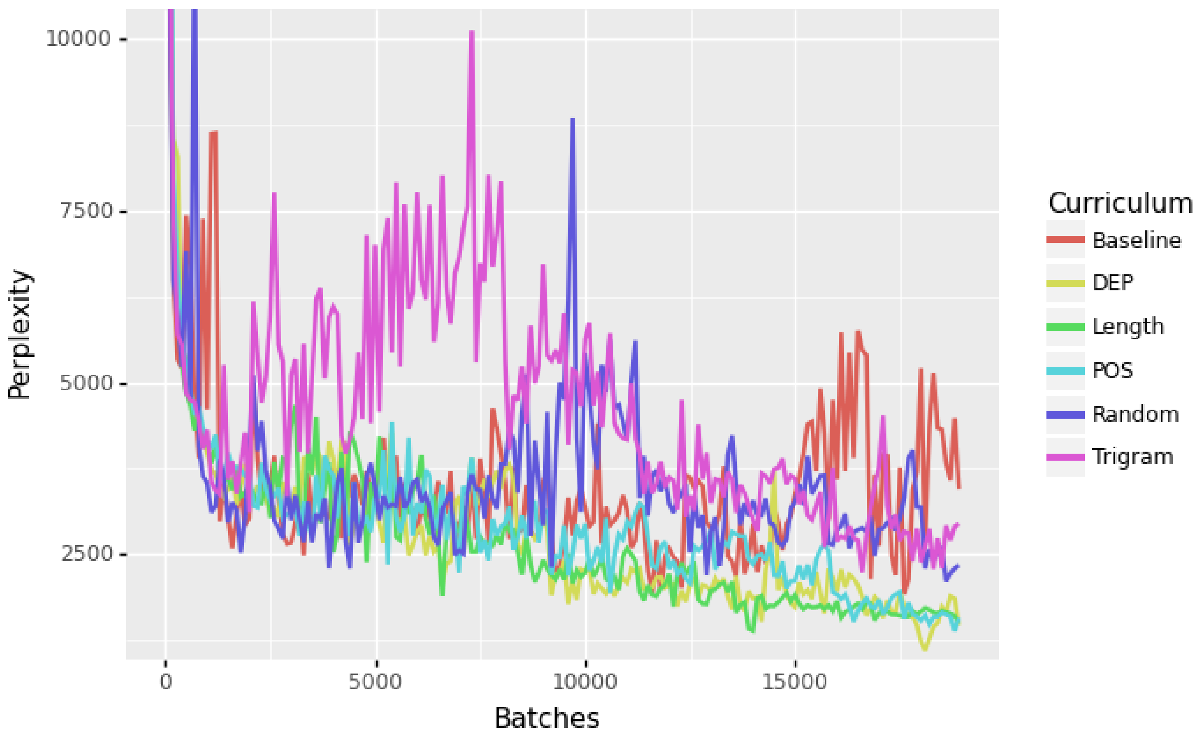

Figure 4.7: Validation perplexity of each curriculum trained on sentence based wikitext-103 measured every 100 batches.

4.3 Discussion

Reflecting on our experiments and questions posed earlier in the dissertation in Section 3 we believe our results support the following findings:

1. CL can help converge to a more optimal global minima when the training corpus is small. As corpus size scales, the positive impact of CL disappears.

2. The downstream representation learned via CL outperforms non-curricula methods when the corpus size is small, but as the corpus size grows, CL methods no longer generate the best representation.

Method	Overall	Cola	SST	MRPC	STS-B	QQP	MNLI	QNLI	RTE	WNLI	DX
baseline	**0.671**	**0.281**	**0.862**	0.866/0.801	0.765/0.773	0.716/0.763	0.644	0.761	0.610	0.535	0.139
trigram-s	0.666	0.208	0.857	0.865/0.806	0.790/0.790	0.733/0.779	**0.658**	0.757	0.567	**0.563**	0.136
trigram-l	0.665	0.207	0.854	0.871/0.804	0.781/0.782	0.752/**0.799**	0.655	0.767	0.556	0.549	0.144
unigram-s	0.664	0.190	0.856	0.861/0.797	0.784/0.784	0.747/0.791	0.654	0.773	0.556	**0.563**	0.140
no curriculum-s	0.663	0.190	0.845	**0.878/0.824**	0.767/0.768	0.748/0.788	0.651	0.746	0.588	**0.563**	0.140
no curriculum-l	0.663	0.214	0.826	0.865/0.804	0.766/0.766	0.747/0.789	0.643	**0.772**	0.581	**0.563**	0.145
random-s	0.661	0.203	0.843	0.870/0.806	0.778/0.779	0.740/0.789	0.644	0.745	0.574	**0.563**	0.151
length-s	0.659	0.229	0.827	0.867/0.804	0.765/0.766	0.747/0.783	0.642	0.762	0.538	**0.563**	0.135
CRS	0.657	0.254	0.852	0.875/0.816	**0.794/0.793**	0.738/0.785	0.662	0.719	0.588	0.437	**0.162**
bigram-l	0.656	0.180	0.826	0.854/0.792	0.770/0.770	0.753/0.794	0.645	0.766	0.560	**0.563**	0.137
length-l	0.656	0.212	0.820	0.851/0.782	0.768/0.767	0.734/0.788	0.634	0.752	0.578	**0.563**	0.135
unigram-l	0.654	0.194	0.823	0.857/0.789	0.755/0.754	0.752/0.794	0.625	0.753	0.574	**0.563**	0.125
random-l	0.652	0.179	0.836	0.861/0.789	0.772/0.774	0.753/0.797	0.640	0.772	0.578	0.493	0.139
pos-l	0.649	0.163	0.827	0.861/0.794	0.756/0.757	**0.754**/0.791	0.630	0.732	0.570	**0.563**	0.138
bigram-s	0.649	0.195	0.859	0.867/0.799	0.784/0.785	0.736/0.772	0.651	0.754	0.599	0.408	0.132
dep-l	0.644	0.231	0.846	0.856/0.779	0.776/0.776	0.746/0.786	0.637	0.761	0.542	0.423	0.137
pos-s	0.640	0.114	0.794	0.853/0.772	0.775/0.774	0.740/0.782	0.624	0.738	0.578	**0.563**	0.108
dep-l	0.587	-0.040	0.842	0.805/0.679	0.786/0.789	0.682/0.730	0.321	0.757	**0.614**	0.549	0.038

Table 4.3: GLUE results for Competence based curricula methods on trained on wikitext-103.

3. CL methods do not help the model convergence speed in any noticeable way and in our use of padding tokens CL methods actually make training fare less efficient than non CL methods.

These finding, matched with no generalized superior curricula performance and the increased cost in running curriculum methods lead us to believe that language models learn more from the stochastic sampling of a corpus than any structure experimenters try to introduce.

4.3.1 Failure of Competence Based Curriculum

What surprised us most in our results was the failure in learning the training data we see in our CBC method. Based on the changes in validation perplexity we believe the model is over-fitting on the altered training data. We believe the cause of this is our hyperparameter selection for $_{0}$ and $_{increment}$. We realize that since each method is effusively sampling from a different training distribution comparison of training perplexities are not comparable. Additionally, if we look at the difference in validation perplexity curves of various methods it is apparent that they are not learning at the same rate. Some methods like DEP, and POS do not see major fluctuations indicating the chosen curriculum parameters work well while many of the n-gram methods consistently fluctuate in a similar fashion indicating the chosen hyperparameters are sub optimal for them. Given the non trivial computational cost to explore $_{0}$ and $_{increment}$ for each method and the disconnect seen between pre-training perplexity and performance on GLUE we have decided to to pursue further optimization in this dissertation.

4.3.2 Curriculum Learning Alters the Data Distribution

We find it useful to formulate CL as moving from sampling without replacement to sampling with replacement with an ever-growing population. As a result, our implementations of CBC do not guarantee that the model will see the entire dataset ten times, and as a result, the distribution of the data seen in training is likely different from the distribution of the validation dataset. We believe this is why the CL implementations can never generalize their performance in the train portion of the corpus to the validation portion. To avoid this one such approach would be to over sample easy portions of the training data in early training and extend the length of this epoch. This way the model could build of easy example but still see every part of the training corpus at least ten times.

Despite this inability to learn the training distribution, CBC methods are still able to transfer well to understanding tasks, which makes us think that what is important in pre-training is not the actual task but what the model can learn from the task. This finding is interesting because it challenges the notion that a model must fit the target dataset well to learn a representation that transfers well to downstream tasks. We find that even the models with high validation perplexities can still learn good representations for our transfer task.

Another observation our data leads us to is tweaking the training distribution's can allow the models to do better in transfer tasks because it is unable to overfit to the small pre-training corpus. Since our implementation generates some form of continually changing data distribution from where random batches are sampled, the train distribution is continually changing. Since the training regime is constantly changing, we hypothesis CBC is effectively simulating a larger, more complex dataset. On one hand, this larger artificial training set causes the model never learns a good representation of the original training corpus which we see in model perplexity. On the other hand, since the model never learns a good representation of the training corpus it can effectively simulate a much larger training corpus which produces a LR which when measured in terms of transfer performance is much better. This is especially apparent in the effectiveness of random curricula, where the random training structure still leads to improved transfer performance. Instead of having one consistent dataset of size N we have a dataset of $size = \sum_{t=0} N * \quad_t$ datasets.

Independently of what heuristic is used in early training, some portion (usually the easy portion) is over-sampled while in other portions (usually hard portion), examples are under-sampled. Recently, work in understanding how NN work has lead to the discovery there are sub-networks within the larger NN, which are more optimal for the task [22]. The process of training a DNN can be thought of as an architecture search within the larger randomly initiated network by decreasing the weights of sub-optimal sub-networks and increasing the weights of optimal sub-networks. Since the CL method is oversampling some portion of the dataset during early training, CL methods will favor sub-networks that are likely not optimal for the full data distribution. We believe this may be the cause of why CL methods under-perform non-CL methods on the training task. One way to address this shortcoming would be to modify curricula methods to create ever-changing distributions that are close to the original distribution. In other words, in early training, instead of oversampling from the lower strata of the CDF CL methods could sample from slightly distorted distributions of the same CDF. One such implementation may divide the data into deciles and sample from each decile for each epoch. In other words instead of sampling from the whole distribution the first training epoch would sample from the lowest 10% of the CDF while the last training epoch would sample from the highest 10%.

4.3.3 Training E ciency

With regard to improving the efficiency of LM training, our implementations had the opposite effect. Our implementation of CBC introduces an overhead which makes training over 40% less efficient. In seeing the effective impact of successful training datasets like the Toronto Book corpus we believe that successful CL methods need to focus on rebalancing training distribution in a model independent method. By rebalancing data distributions CL methods can ensure that the cost of compute only happens once instead of the n times the data is resampled in model training and the m potential models that use this data. Moreover, by moving curriculum generation outside of model training CL methods can have a larger impact because researchers only need to train on a new dataset instead of changing their model code.

4.3.4 Sentence vs Line Training

We find No marked difference in the effects of training with sentence-based vs line based corpora when evaluated on GLUE. We do find a large difference in perplexity on the pre-training task but we believe this is caused by the inefficiency introduced by our padding token. Since sentence based sampling has 30% more padding tokens the model is able to learn substantially less from every sample when compared to the line corpus. Surprisingly this decreased ability to learn the training data seems to have no impact on transfer performance. This is in line with the purpose of pretraining as the goal is not actually to learn the training data but use it to form a generalized representation.

Chapter 5
CONCLUSION

Throughout this dissertation, we have explored how some approaches in curriculum learning apply to language modeling. Our research explored two types of curricula and used them to pretrain ELMo, a language model. Our first curriculum, corpus replacement style, introduces no compute overhead in training and makes the training corpus more difficult by limiting the training vocabulary in regularly spaced increments. This curriculum does not improve model perplexity on the training corpus, but when the corpus is small outperforms the non-curriculum methods on transfer tasks. Our second curriculum, competence based curriculum, explores the effect of various curricula when applied to the same language model pretraining. In these experiments, we find that while the model cannot learn a good representation of the training corpus, their representations transfer well to downstream NLP tasks. We find that on small datasets, competence curriculum show improvement versus non-curriculum methods across the board. As we scale the corpus size, we find that non-curriculum methods perform best. We do not see any superiority in the curriculums we explore, nor do we find a clear difference in training effects with sentences over lines. While our implementations could not produce improvements, we believe the results set the stage for further research and pose some broader questions for learning methods for NN.

Chapter 6
FUTURE WORK

We seek to continue our exploration of CL methods and methods to make training and using models more efficient. Since we began our work there has been much exciting work on finding sub-networks in NNs that preserve the accuracy of the original model [22]. Other methods have explored the pruning of large models for smaller equally accurate models [28] [94] [18] which we would like to expand on, focusing on how these pruning methods perform with transfer learning. While these method are not directly computationally more efficient to train they make reuse of some original large model (which can be trained once) and allow for more efficient model deployment. Additionally, inspired by the layered nature of Transformer-based models, we would explore jointly how progressive methods may look like for LMs. One such approach would be to increase the number of transformer encoders while increasing data difficulty, similar to what has been done with GANs [39]. Another method may model longer context windows progressively starting with short sentences and scaling to entire documents. Finally, we seek to make a benchmarking system that can allow researchers to explore the effect of various curricula on many downstream tasks. This benchmark would include tasks from NLP, CV, and beyond and the focus for researchers would be on the training regime. The goal would be to provide a framework similar to GLUE and JIANT, which provide a set of architectures and tasks which would allow researchers to focus only on sampling methods. If the broader community had an easy to use a benchmark, they could focus on studying the just curriculum methods.

BIBLIOGRAPHY

[1] Ben Athiwaratkun and Andrew Gordon Wilson. Multimodal word distributions. *ArXiv*, abs/1704.08424, 2017.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[3] Roy Bar-Haim, I. Dagan, B. Dolan, Lisa Ferro, Danilo Giampiccolo, and B. Magnini. The second pascal recognising textual entailment challenge. 2006.

[4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML '09*, 2009.

[5] L. Bentivogli, Peter Clark, I. Dagan, and Danilo Giampiccolo. The sixth pascal recognizing textual entailment challenge. In *TAC*, 2009.

[6] Adam L. Berger, Stephen Della Pietra, and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22:39–71, 1996.

[7] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[8] T. Brown, B. Mann, Nick Ryder, Melanie Subbiah, J. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, G. Krüger, Tom Henighan, R. Child, Aditya Ramesh, D. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, E. Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, J. Clark, Christopher Berner, Sam McCandlish, A. Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020.

[9] Daniel Matthew Cer, Mona T. Diab, Eneko Agirre, I. Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. *ArXiv*, abs/1708.00055, 2017.

[10] Saikat Chatterjee, Alireza M. Javid, Mostafa Sadeghi, Partha P. Mitra, and Mikael Skoglund. Progressive learning for systematic design of large neural networks. *ArXiv*, abs/1710.08177, 2017.

[11] Ciprian Chelba, Tomas Mikolov, Michael Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *ArXiv*, abs/1312.3005, 2014.

[12] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does bert look at? an analysis of bert's attention. *ArXiv*, abs/1906.04341, 2019.

[13] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators. *ArXiv*, abs/2003.10555, 2020.

[14] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *J. Artif. Intell. Res.*, 4:129–145, 1994.

[15] Kornél Csernai, Jan 2017.

[16] I. Dagan, Oren Glickman, and B. Magnini. The pascal recognising textual entailment challenge. In *MLCW*, 2005.

[17] Zihang Dai, Z. Yang, Yiming Yang, J. Carbonell, Quoc V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *ArXiv*, abs/1901.02860, 2019.

[18] Adrian de Wynter and D. Perry. Optimal subarchitecture extraction for bert. *ArXiv*, abs/2010.10499, 2020.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[20] W. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *IWP@IJCNLP*, 2005.

[21] J. Elman. Learning and development in neural networks: the importance of starting small. *Cognition*, 48:71–99, 1993.

[22] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv: Learning*, 2019.

[23] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

[24] Danilo Giampiccolo, B. Magnini, I. Dagan, and W. Dolan. The third pascal recognizing textual entailment challenge. In *ACL-PASCAL@ACL*, 2007.

[25] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *ArXiv*, abs/1406.2661, 2014.

[26] Junliang Guo, Xu Tan, Linli Xu, Tao Qin, Enhong Chen, and Tie-Yan Liu. Fine-tuning by curriculum learning for non-autoregressive neural machine translation. *ArXiv*, abs/1911.08717, 2019.

[27] Song Han, Junlong Kang, Huizi Mao, Yiming Hu, Xin Li, Y. Li, Dongliang Xie, Hong Luo, S. Yao, Y. Wang, H. Yang, and W. J. Dally. Ese: Efficient speech recognition engine with sparse lstm on fpga. In *FPGA '17*, 2017.

[28] Song Han, Huizi Mao, and W. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2016.

[29] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.*, 6:107–116, 1998.

[30] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[31] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. 2017.

[32] J. Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *ACL*, 2018.

[33] J. Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and M. Johnson. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. *ArXiv*, abs/2003.11080, 2020.

[34] Wenpeng Hu, Jiajun Zhang, and Nan Zheng. Different contexts lead to different word embeddings. In *COLING*, 2016.

[35] S. Jean, Kyunghyun Cho, R. Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. *ArXiv*, abs/1412.2007, 2015.

[36] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *ArXiv*, abs/1907.10529, 2019.

[37] Takatomo Kano, Sakriani Sakti, and Satoshi Nakamura. Structured-based curriculum learning for end-to-end english-japanese speech translation. In *INTERSPEECH*, 2017.

[38] Jean 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*, abs/2001.08361, 2020.

[39] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *ArXiv*, abs/1710.10196, 2017.

[40] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *NIPS*, 2015.

[41] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017.

[42] Tom Kocmi and Ondrej Bojar. Curriculum learning and minibatch bucketing in neural machine translation. In *RANLP*, 2017.

[43] Zhen-Zhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942, 2019.

[44] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *ArXiv*, abs/1405.4053, 2014.

[45] Claudia Leacock, Geoffrey G. Towell, and Ellen M. Voorhees. Towards building contextual representations of word senses using statistical models. 1993.

[46] H. Levesque, E. Davis, and L. Morgenstern. The winograd schema challenge. In *KR*, 2011.

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

[48] Yaobo Liang, N. Duan, Yeyun Gong, N. Wu, Fenfei Guo, Weizhen Qi, Ming Gong, Linjun Shou, Daxin Jiang, G. Cao, Xiaodong Fan, Bruce Zhang, Rahul Agrawal, Edward Cui, Sining Wei, Taroon Bharti, Ying Qiao, Jiun-Hung Chen, Winnie Wu, S. Liu, Fan Yang, Rangan Majumder, and M. Zhou. Xglue: A new benchmark dataset for cross-lingual pre-training, understanding and generation. *ArXiv*, abs/2004.01401, 2020.

[49] Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. Linguistic knowledge and transferability of contextual representations. In *NAACL-HLT*, 2019.

[50] 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*, abs/1907.11692, 2019.

[51] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *EMNLP*, 2015.

[52] Andrew McCallum, Dayne Freitag, and Fernando C Pereira. Maximum entropy markov models for information extraction and segmentation. In *ICML*, 2000.

[53] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In *NIPS*, 2017.

[54] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *ArXiv*, abs/1609.07843, 2016.

[55] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[56] George A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38:39–41, 1992.

[57] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with bert. *ArXiv*, abs/1910.14424, 2019.

[58] N. Othman, R. Faiz, and Kamel Smaïli. A word embedding based method for question retrieval in community question answering. 2017.

[59] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[60] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *ArXiv*, abs/1802.05365, 2018.

[61] Matthew E. Peters, Mark Neumann, IV RobertLLogan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. Knowledge enhanced contextual word representations. In *EMNLP/IJCNLP*, 2019.

[62] Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabás Póczos, and Tom Michael Mitchell. Competence-based curriculum learning for neural machine translation. *ArXiv*, abs/1903.09848, 2019.

[63] Yada Pruksachatkun, Phil Yeres, Haokun Liu, Jason Phang, Phu Mon Htut, Alex Wang, Ian Tenney, and Samuel R. Bowman. jiant: A software toolkit for research on general-purpose text understanding models. *ArXiv*, abs/2003.02249, 2020.

[64] A. Radford, Jeffrey Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[65] Alec Radford. Improving language understanding by generative pre-training. 2018.

[66] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, 2016.

[67] Corby Rosset. Turing-nlg: A 17-billion-parameter language model by microsoft, 2020.

[68] Dwaipayan Roy, Debasis Ganguly, S. Bhatia, Srikanta J. Bedathur, and M. Mitra. Using word embeddings for information retrieval: How collection and term normalization choices affect performance. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018.

[69] A. Sagheer and Mostafa Kotb. Time series forecasting of petroleum production using deep lstm recurrent networks. *Neurocomputing*, 323:203–213, 2019.

[70] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

[71] Burr Settles. Active learning literature survey. 2009.

[72] Claude E. Shannon. Prediction and entropy of printed english. 1951.

[73] Or Sharir, Barak Peleg, and Yoav Shoham. The cost of training nlp models: A concise overview. *ArXiv*, abs/2004.08900, 2020.

[74] Mohammad Shoeybi, Mostofa Ali Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *ArXiv*, abs/1909.08053, 2019.

[75] Noah A. Smith. Contextual word representations: A contextual introduction. *ArXiv*, abs/1902.06006, 2019.

[76] R. Socher, Alex Perelygin, J. Wu, Jason Chuang, Christopher D. Manning, A. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 2013.

[77] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. In *ACL*, 2019.

[78] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. Ernie: Enhanced representation through knowledge integration. *ArXiv*, abs/1904.09223, 2019.

[79] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. Ernie 2.0: A continual pre-training framework for language understanding. *ArXiv*, abs/1907.12412, 2019.

[80] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 16:285–286, 1998.

[81] Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. olmpics - on what language model pre-training captures. *ArXiv*, abs/1912.13283, 2019.

[82] W. L. Taylor. “cloze procedure”: A new tool for measuring readability. *Journalism Mass Communication Quarterly*, 30:415 – 433, 1953.

[83] Dat Thanh Tran, Moncef Gabbouj, and Alexandros Iosifidis. Subset sampling for progressive neural network learning. 2020.

[84] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, L. Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

[85] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *ArXiv*, abs/1905.00537, 2019.

[86] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *BlackboxNLP@EMNLP*, 2018.

[87] Wei Wang, Ye Tian, J. Ngiam, Yinfei Yang, Isaac Caswell, and Zarana Parekh. Learning a multitask curriculum for neural machine translation. *ArXiv*, abs/1908.10940, 2019.

[88] Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

[89] Joseph Weizenbaum. Eliza — a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 26:23–28, 1966.

[90] Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *ArXiv*, abs/1704.05426, 2018.

[91] Y. Wu, Mike Schuster, Z. Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, M. Krikun, Yuan Cao, Q. Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, Taku Kudo, H. Kazawa, K. Stevens, G. Kurian, Nishant Patil, W. Wang, C. Young, J. Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, G. S. Corrado, Macduff Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv*, abs/1609.08144, 2016.

[92] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, 2019.

[93] Yang You, Jing Li, Sashank J. Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. 2019.

[94] Jiecao Yu, Andrew Lukefahr, D. Palframan, Ganesh S. Dasika, R. Das, and S. Mahlke. Scalpel: Customizing dnn pruning to the underlying hardware parallelism. *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pages 548–560, 2017.

[95] Xuan Zhang, Pamela Shapiro, Manish Kumar, P. McNamee, Marine Carpuat, and Kevin Duh. Curriculum learning for domain adaptation in neural machine translation. *ArXiv*, abs/1905.05816, 2019.

[96] Xiyou Zhou, Zhiyu Chen, Xiaoyong Jin, and W. Wang. Hulk: An energy efficiency benchmark platform for responsible natural language processing. *ArXiv*, abs/2002.05829, 2020.

[97] Yunxiao Zhou, Zhihua Zhang, and Man Lan. Ecnu at semeval-2016 task 4: An empirical investigation of traditional nlp features and word embedding features for sentence-level and topic-level sentiment analysis in twitter. In *SemEval@NAACL-HLT*, 2016.

[98] 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 *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[99] Will Y. Zou, R. Socher, Daniel Matthew Cer, and Christopher D. Manning. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*, 2013.

Appendix A

EXPERIMENTAL RESULTS

A.1 Corpus Replacement Style

Epochs	baseline wiki-2	bs wiki-2	bs wiki-103	baseline wiki-103
1	620.45544	758.29285	227.65094	83.05149
2	451.85522	507.58508	138.29044	56.865345
3	358.74582	445.0215	113.23114	48.632687
4	303.83347	424.0015	100.1231	44.216614
5	241.15233	364.40637	92.03778	41.57386
6	221.38237	395.8324	90.143585	39.798542
7	195.55809	213.29416	39.72011	38.54776
8	168.30266	181.99895	38.672005	37.64779
9	162.16602	163.70233	37.66034	37.009693
10	151.25607	153.26689	37.029747	36.412056

Table A.1: Validation perplexity of baselines and replacement methods measured every epoch.

A.2 Competence Based Curriculum

Batches	Baseline	Random	Dependency	Part of Speech	Unigram	Bigram	Trigram	Length
0	19498.28	21163.44	26654.95	26528.35	22053.12	19535.65	20830.92	29933.74
100	5600.459	5461.039	5730.613	5215.405	5412.148	5164.017	5130.01	6159.207
200	8412.964	5335.186	4173.729	4526.756	4801.211	4104.102	4100.64	5091.564
300	10206.76	5121.969	4350.757	4552.211	5111.258	4242.258	5154.742	4209.991
400	9694.969	4656.295	4411.583	4181.944	4576.212	4606.979	4085.841	4266.357
500	4263.702	4187.064	3975.314	4448.919	4315.669	3466.285	3425.474	6061.277
600	4637.823	4007.23	3804.034	5325.821	3865.498	3853.12	3574.217	5577.375
700	3670.097	4142.923	3831.932	3942.862	9677.842	3230.582	3502.653	4219.224
800	3221.346	3970.763	3749.825	4645.618	4742.95	2954.3	3138.186	4247.958
900	3959.106	5503.046	4688.382	5068.281	3871.681	3856.12	12232.67	4298.991
1000	3132.26	8124.751	4383.869	4416.356	4699.026	3397.773	8652.374	3856.153
1100	13849.47	7308.988	4047.798	4257.96	6621.724	5047.275	5068.213	5205.368
1200	6503.515	5181.124	4269.158	7781.854	17770.95	3666.078	10005.95	4551.86
1300	7178.955	19722.45	5696.292	3884.615	14288.98	8585.893	10586.63	4248.825
1400	5806.94	7711.413	3276.619	4795.949	11649.69	14055.36	9180.324	4275.567
1500	7896.527	9227.372	6222.824	4230.453	7551.371	12330.95	57175.42	5575.008
1600	21873.66	36947.88	4324.498	7872.976	14629.82	12650.03	23212.86	5090.137
1700	12979.71	20851.81	5498.242	5195.954	5852.289	28631.84	56772.43	5290.866
1800	34403.65	24371.06	4613.688	6938.421	17223.34	10833.46	56383.88	5559.394
1900	45148.42	12774.94	4005.43	7535.285	21746.75	32077.18	50338.7	3713.533
2000	28062.45	62992.67	4582.964	6047.258	26790.6	29896.4	25035.54	6571.44
2100	28706.4	105545.6	3806.545	8146.35	42521.56	68461.61	36110.3	5294.52
2200	42284.38	62399.3	5332.897	4722.252	19101.4	78817.77	111061.6	8789.567
2300	211236.3	89274.6	4591.32	6133.123	60930.05	57298.29	53007.47	7117.323
2400	23207.46	115620.8	4978.687	5435.751	44079.81	49065.99	81409.19	6686.693

Table A.2: Validation perplexity of each curriculum trained on line based wikitext-2 measured every 100 batches.

Batches	Baseline	Random	Dependency	Part of Speech	Unigram	Bigram	Trigram	Length
0	19577	18972.3	21636.473	25085.371	17233.63	16562.04	16953.24	25580.67
100	4393.4	3869.01	4907.4766	5089.3213	4227.456	3895.485	5203.784	5679.227
200	3775.8	3342.43	3766.6887	3795.3662	3193.679	3014.929	3285.696	5670.292
300	3193.5	2855.56	3518.257	3350.3381	3022.722	2862.637	10340.28	7913.436
400	3018	2868.06	3079.2112	3235.9043	2984.732	2797.256	2898.078	4154.833
500	2629.8	2624.77	3339.034	3249.0745	2615.482	2638.166	2932.65	2693.087
600	2751.3	2482.34	2926.965	2851.3337	2597.397	2542.316	2572.945	2762.114
700	2538.6	2377.76	2814.3276	3095.3398	2438.547	2426.895	2399.556	2432.863
800	3605.3	2241.75	2780.0137	3009.666	3130.137	2377.921	2476.211	2709.775
900	2679	2297.78	2464.8281	2462.9426	4788.499	2485.33	2421.764	2963.076
1000	3212.4	2219.75	2665.2764	2486.0417	2088.218	2745.638	2329.685	2468.642
1100	3965.8	2257.58	10348.93	2716.1235	2225.749	2263.412	2619.83	2386.155
1200	2557	2224.1	3332.5085	2176.2024	2086.037	2821.298	2512.837	2055.139
1300	2566.7	2505.48	2912.2942	2439.959	2668.968	2386.25	2376.939	2844.38
1400	4251.9	2659.05	3528.9463	2964.2732	2541.053	2565.921	2829.178	2446.484
1500	2974.5	2815.76	2924.3967	2067.4797	3674.223	2774.171	2343.067	2960.37
1600	2915.3	2344.01	2849.28	2534.103	3890.443	3291.204	2685.646	3916.598
1700	2847.2	2127.63	5314.9077	2186.36	3038.41	3183.71	2816.234	2621.306
1800	2747.2	2285.29	2729.2192	2487.4993	2781.731	2755.068	2685.075	2596.71
1900	3222.8	2440.04	2387.6428	2464.5767	3036.206	2939.755	2961.87	3006.224
2000	3432.1	2423.35	2498.7942	2584.4092	2965.91	2836.161	2218.454	2221.723
2100	3438.4	2001.77	2103.243	3023.273	3039.94	2557.588	2440.281	2958.65
2200	2597.5	2516.39	2807.2107	3113.2212	2539.777	2650.921	2384.331	2922.845
2300	2537.7	2572.49	2737.5596	2898.6365	3100.773	2493.721	2449.759	2993.907
2400	3433.7	2633.99	2596.0078	2330.572	2583.728	2229.844	2207.221	2361.876
2500	3063.9	2835.72	1969.3733	2286.7202	2515.244	2056.117	2092.182	3337.922
2600	3235.2	2638.41	4091.7915	2449.3496	3334.162	2431.215	2069.936	3580.122
2700	4040.3	2670.14	3125.7693	2753.1929	3107.351	2691.088	2656.51	4305.21
2800	3402.3	2347.04	5570.7563	2362.6565	2062.797	2586.205	2425.246	3611.754
2900	5388.1	2740.16	4000.6243	3194.3093	2790.644	2407.481	2207.875	4251.675
3000	5389.1	2568.82	11589.234	3652.8022	2783.052	2784.212	2332.16	3563.234
3100	4227.3	3012.2	3253.4585	3206.2273	2920.262	3337.26	2850.718	2102.434
3200	3962.9	3157.9	10045.327	4041.8535	3573.764	2864.314	2191.798	3055.313
3300	5547.2	2583.23	17333.396	3213.1606	2899.022	2394.718	2577.445	2699.329
3400	7155.4	3601.26	3040.4277	3099.2976	3455.618	2837.321	2653.487	3373.731
3500	5882.7	3813.07	16560.428	2756.5728	4385.529	2440.946	3067.075	2999.78
3600	4887.4	2530.43	9361.988	2919.6719	3249.393	2523.844	2807.685	2630.625
3700	6196.1	2383.66	20497.451	2655.2168	4421.092	2611.211	3142.592	2940.323
3800	5810.9	3164.84	17406.648	3329.6848	5049.87	3314.212	3773.563	3204.518
3900	5575.6	2112.18	11576.83	2928.0552	3045.523	2490.426	2244.873	2624.535

Table A.3: Validation perplexity of each curriculum trained on sentence based wikitext-2 measured every 100 batches.

Batches	Baseline	Random	Dependency	Part of Speech	Unigram	Bigram	Trigram	Length
0	172216.9	165166.4	207358.9	215021.8	161038.8	172513.9	154467.2	225595.5
100	17752.13	14454.38	18175.47	22662.42	16281.71	14672.36	17838.29	21663.13
200	9440.708	9769.283	12327.02	13543.41	9924.168	8290.118	10426.92	15506.72
300	8046.883	7058.333	8941.258	12681.64	8252.508	9401.705	8486.866	11837.9
400	7659.163	6817.952	8116.79	9643.348	7333.894	7104.682	8532.874	14069.69
500	17995.63	6471.114	7051.935	8360.009	6820.631	6963.977	7218.733	10314.54
600	6822.212	6106.644	6559.931	7686.075	7022.727	6500.029	6966.083	10291.89
700	17029.94	5843.533	6919.971	6740.012	13876.88	6019.34	7615.919	9683.603
800	6325.329	5509.364	5746.84	6492.756	6293.799	5675.621	6490.206	9585.646
900	6180.572	5735.097	6853.742	6257.095	5582.361	5015.897	6265.562	8510.857
1000	5201.506	5340.536	6016.334	6763.791	5556.981	5671.72	6699.286	8763.411
1100	5922.099	7981.178	6031.5	5781.142	5672.412	5543.336	6820.982	7628.734
1200	7504.642	4498.846	5527.9	6176.282	5555.387	5341.254	5271.304	9265.033
1300	9524.957	5150.772	6403.761	6860.242	6320.734	9381.356	5810.231	8530.71
1400	6117.824	7766.959	5489.638	8058.786	4760.8	7997.636	15470.43	10192.23
1500	12209.29	14478.4	5555.36	6639.258	11870.64	50514.04	5882.589	7210.841
1600	11165.06	5065.396	5965.695	8284.104	20614.66	10279.71	10198.11	10114.68
1700	16563.81	22438.63	6070.104	9168.329	8410.966	7793.44	8148.277	8040.585
1800	11889.61	7600.428	6984.522	6836.841	36763.81	42040.47	24620.74	10472.91
1900	17304.26	12271.82	5551.246	5711.337	9572.392	107992.7	32649.54	10064.73
2000	20918.78	26494.65	7118.212	7859.69	24109.07	28076.71	49109.53	9722.515
2100	20992.18	21873.45	7361.074	7938.744	24785.81	128563	19978.95	11637.65
2200	5439.09	29953.31	4925.848	8127.114	8477.693	254906.1	16281.98	9188.663
2300	15406.22	15651.8	7203.72	5258.265	29320.53	19536.08	31459.52	6698.711
2400	5505.089	51011.46	6021.017	6700.699	41768.13	190713.8	109279	9499.203
2500	27145.59	102160.4	5934.129	7351.912	40052.76	16625.78	28730.56	9936.309
2600	16985.74	118023.9	5448.081	8072.763	18126.65	57096.73	32798.09	10838.18
2700	32159.18	119966	6966.203	6144.639	37110.21	86387.07	79481.34	10947.64
2800	31417.87	86970.59	5826.9	8385.137	34130.03	75165.63	76327.33	11602.87
2900	14614.76	121529.8	6327.055	6687.796	13573.45	62818.87	58189.62	8543.565
3000	37013.51	72880.48	5346.401	6942.903	13903.45	72515.88	59559.2	13702.55
3100	26318.72	191492.9	5910.706	8088.175	20284.44	42139.34	66478.91	9224.82
3200	17521.95	49747.02	5319.806	7718.233	59631.16	208311.3	29734.44	12086.46
3300	30431.01	71643.49	6437.672	7128.939	28100.13	88150.58	112454.6	8411.166
3400	51275.57	30327.61	5738.155	5817.499	41311.68	91364.31	123496.2	13534.45
3500	51215.12	41681.3	8302.683	5853.579	27738.42	126417.5	167231.6	9795.62
3600	280749.3	85320.23	7812.386	7880.961	87955.51	70526.31	46442.92	9895.758
3700	84822.91	48738.14	9736.042	8622.563	22330.83	230793.9	32258.89	15883.25
3800	53030.07	71282.7	7123.815	9047.259	35390.02	351032.8	83754.86	9699.777
3900	112609.9	52523.73	10376.54	8074.103	81709.11	102754.4	47849.13	28220.17
4000	46003.87	65833.73	4979.105	9481.283	74501.85	782343.5	798678.3	18383.94
4100	128197.2	65345.9	4852.084	10374.46	161899.6	224331.9	252933.2	14701.67
4200	76697.49	50991.32	3934.178	10077.66	238026.5	379865.8	234076.5	34005.17
4300	189905.8	59316.09	13718.04	11378.06	63523.12	232656	429235.1	24132.4
4400	209395.3	84390.66	10762.15	7380.278	126112.6	253614.6	934066.5	45418.72
4500	238296.8	68939.56	9456.224	8651.029	121685.7	346800.8	229740.9	15752.04
4600	168189.2	50374.2	13548.85	6933.216	93128.2	169451.7	1039976	20756.09
4700	119946.5	49438.28	14698.38	8238.936	67726.18	1229356	568497.7	15351.88
4800	99391.29	45065.61	9423.725	12850.34	236304.9	717031.4	671055.9	39285.41
4900	826470.1	39831.68	6392.607	9569.764	118792.2	342211.1	182205.8	13760.5
5000	533522.1	48656.72	6969.119	9091.118	36777.41	336925.2	313601.7	16320.92

Table A.4: Validation perplexity of each curriculum trained on line based wikitext-103 measured every 100 batches. Batches 0-5000.

Batches	Baseline	Random	Dependency	Part of Speech	Unigram	Bigram	Trigram	Length
5100	746036.3	42115.35	14609.33	5563.435	89606.75	1252387	481100.7	14452.05
5200	450154.2	26239.17	9602.885	9182.374	34282.01	908216.8	896863.8	13975.82
5300	463886.8	71350.03	17884.33	11694.03	45075.45	797953.4	292389.5	13086.2
5400	381798.3	23556.72	13086.15	11440.52	31661.02	4210174	987853.3	27401.4
5500	772458.8	40193.19	6611.186	7330.768	48236.98	635867.9	572814.9	28124.18
5500	772458.8	40193.19	6611.186	7330.768	48236.98	635867.9	572814.9	28124.18
5600	435986.2	33766.4	7594.993	8776.357	57675.37	2312263	379041.1	29720.69
5700	380407.4	65307.84	7152.652	9622.659	22217.88	1977097	646754.9	18851.84
5800	987583.8	44488.23	5389.356	13977.19	148238.7	753675.8	477882.3	18554.14
5900	525945.9	43697.09	8578.991	11714.1	96957.49	2054665	2008199	12342.02
6000	595516.3	78893.88	4152.551	17958.13	28597.49	554772.1	333557.1	12661.91
6100	336289.6	64591.62	8177.548	8448.202	342031.3	886302.7	617769.6	31967.73
6200	1218730	42774.86	8656.748	7432.56	144232.9	1739696	354108.3	24835.15
6300	806752.3	52435.1	5284.119	10437.95	65406.88	1230662	463201.1	16395.43
6400	883487.4	71262.03	9085.406	13012.52	159522.5	1961859	592538.1	9778.8
6500	356991.8	57942.09	6845.589	10035.85	342189.3	7449092	413552.2	15096.89
6600	7184665	58905.42	6235.727	9108.961	181569.2	4618393	344374.2	12847.46
6700	2472007	71370.52	8063.353	9192.905	63416.47	2947551	440130	19738.05
6800	814416.1	71758.17	7341.346	7762.863	483430.6	2378077	299415.3	12519.29
6900	352288.7	69272.91	7223.732	6761.998	704382.9	7433083	365704.1	12034.6
7000	2570400	107659.3	15324.69	7525.827	159764.6	5042794	327737.8	10918.05
7100	219466.5	40054.32	5909.269	7208.585	400412.5	2689664	500218.3	10384.75
7200	2046770	62646.45	16789.61	8600.561	402128.4	4250481	341898	7711.287
7300	480427.1	54083.11	8680.044	9967.077	197295.2	7876661	144218.1	7864.698
7400	984593.3	93926.94	15979.23	8263.257	173903.8	6310541	193359.8	6853.023
7500	538978.8	74692.16	13873.01	5485.598	178533.8	10171802	152454.4	11444.25
7600	651062.6	145972.2	12457.06	6190.346	130058.9	2254320	77680.59	9727.429
7700	1073969	59510.66	11320.66	4617.416	156135	9508326	150271.7	9482.061
7800	610076.5	12976.09	17127.44	7731.529	1120081	7069702	49085.09	6581.255
7900	225100.4	21487.19	15915.88	8408.343	366767.7	11207608	44954.52	7738.051
8000	199603.1	22423.08	8493.198	10677.72	282558.9	5738164	59227	10044.92
8100	228689.7	28845.37	17272.49	6949.825	479576.6	18111974	68112.94	8505.341
8200	235564.2	24770.57	14323.23	9485.127	88964.38	5244861	62174.17	8277.375
8300	126557.8	14009.84	15116.83	10087.02	690523.8	6649965	33649.32	8678.38
8400	125826.9	26490.28	5467.11	5482.152	141400.3	11069663	24212.87	8326.598
8500	184564.5	35983.2	8759.442	6007.31	263091.2	19984712	20370.69	8973.215
8600	193700.3	40037.94	9611.534	3816.377	80663.02	12887875	30611.18	6877.555
8700	131844.8	19826.14	4457.578	5295.065	84227.77	5281593	25827.62	6758.053
8800	159574.4	48990.52	7797.417	4197.331	212060	75582376	23901.98	4165.291
8900	187651.8	19489.41	8147.741	5122.897	67214.51	5462595	23768.8	7202.69
9000	115416.6	49372.73	8275.986	4522.394	41884.84	12775294	24709.44	8464.759
9100	113973.8	44718.74	8977.323	4020.831	30371.49	6320431	21859.27	6555.091
9200	95768.52	44717.38	13144.78	5879.196	34368.44	5331262	20166.51	7530.378
9300	210090.5	38205.06	9531.072	5391.449	58940.26	6143711	24092.68	8377.303
9400	78048.02	27389.8	5589.521	4205.754	45726.12	7094071	13933.89	6444.306
9500	77830.83	12025.84	7426.871	5617.073	50402.64	11013728	17786.87	7362.991
9600	78440.59	21652.72	6946.194	6006.593	61894.7	7434160	14580.02	7317.252
9700	80058.74	13562.09	7524.327	6448.148	149280.6	5530012	18527.85	5951.523
9800	96350.42	48266.57	9335.1	4927.661	44235.29	34580910	10860.8	6998.772
9900	58123.4	18633.09	9792.201	6308.775	31676.58	7437323	9205.582	6894.294
10000	55742.19	35956.14	13261.41	8612.109	29911.54	13082393	13093.23	6222.705

Table A.5: Validation perplexity of each curriculum trained on line based wikitext-103 measured every 100 batches. Batches 5100-10000.

Batches	Baseline	Random	Dependency	Part of Speech	Unigram	Bigram	Trigram	Length
10100	88549.62	40304.27	13661.65	5588.294	31186.52	16643766	11505.52	6533.839
10200	98337.85	34261.52	5348.304	8566.059	28469.21	3852336	7258.558	7176.874
10300	47322.85	16611.63	7442.512	10287.06	42922.91	73284620	12194.78	7722.761
10400	45772.67	25105.38	11219.15	7169.95	32161.08	13315050	15309.94	8523.342
10500	58851	5647.017	11260.18	5710.395	28194.51	9781454	18709.42	8880.455
10600	53763.87	37941.77	3321.619	5918.338	23155.95	3818047	15440.06	8171.693
10700	65068	24284.45	5706.813	6225.834	23958.48	10583155	14497.16	5931.368
10800	57146.37	29989.32	6658.617	6290.858	20263.38	26375042	18258.13	6026.665
10900	87174.87	37592.69	6563.899	7539.073	23480.12	13366220	14391.71	5681.486
11000	262300.6	34450.24	6840.316	10874.94	19338.29	4964016	11612.11	4559.554
11100	94144.86	18909.7	8100.975	10786.89	23337.49	9985969	9333.398	5405.225
11200	42594.32	19272.31	4562.391	9537.154	30152.12	12950934	8763.369	5527.937
11300	159386.1	8380.708	5639.342	16820.3	25195.54	24968576	9791.053	5146.897
11400	136473.1	32503.12	7450.012	8285.1	11999.21	10727265	12366.25	4828.422
11500	154234.3	21173.09	4064.795	5776.649	8972.427	16446159	11109.62	5884.705
11600	212200	42542.97	6638.802	6924.856	31763.64	9586595	8522.553	5325.715
11700	28273.24	15763.57	6138.805	6202.253	23226.87	1.11E+08	10217.99	5584.124
11800	94377.06	32340.02	6321.38	5979.011	20873.26	6928025	5410.511	5792.996
11900	202019.8	15339.94	6085.598	5099.169	24225.19	6798724	8848.321	5855.108
12000	103377	27182.9	7571.237	5749.055	19494.32	14127428	6620.839	5186.854

Table A.6: Validation perplexity of each curriculum trained on line based wikitext-103 measured every 100 batches. Batches 10100-12000.

Batches	Baseline	Random	Dependency	Part of Speech	Unigram	Bigram	Trigram	Length
0	133566.89	145170.9	157186.9	181081.6	145557	163357.2	154942.1	171047.1
100	12285.347	11508.72	11214.37	16152.97	13442.27	11411.47	12175.73	14221.94
200	6447.5767	6480.044	8555.388	7975.472	8173.018	7094.574	7406.791	7072.591
300	5332.4336	5936.535	8272.807	6656.56	6980.953	5789.997	5697.248	6202.389
400	5207.15	5223.892	5461.248	5642.629	5894.573	5175.929	5550.213	5249.547
500	7429.528	6913.467	5345.27	5224.565	9022.917	5192.218	4844.931	4852.713
600	6627.4604	4759.053	4770.627	4885.608	5370.834	5118.223	4742.507	4654.528
700	4769.9443	11728.62	4452.612	4528.794	4650.765	4768.857	4716.095	4302.28
800	3898.0117	4078.067	4383.075	4649.28	4734.933	3986.259	4436.026	4589.333
900	7385.8477	3638.641	4217.924	4347.232	5342.706	3807.823	4032.682	4067.242
1000	4610.644	3538.989	3972.509	4161.002	15954.95	4200.018	4312.525	4149.574
1100	8622.547	3115.556	3737.612	4033.674	4617.307	5521.467	3553.627	3821.32
1200	8631.456	3200.954	3666.757	4237.244	3974.965	5088.36	3428.275	4078.638
1300	2980.604	3875.604	3463.381	3588.603	3340.407	3399.381	3372.226	3683.903
1400	3539.7998	3630.569	3331.502	3611.937	3219.224	3757.146	5257.478	3687.256
1500	3035.7773	2963.425	3455.519	3978.28	4819.782	3602.563	3591.551	3593.675
1600	2589.6614	3252.407	3621.878	3732.493	4355.743	3276.338	3857.337	3438.57
1700	3098.1157	3141.189	3216.005	3768.428	4081.203	3389.714	3384.101	3790.884
1800	3428.1082	2522.237	3475.266	3735.106	3026.908	3103.072	3821.906	3335.825
1900	2923.9253	3033.856	3818.332	3650.337	3948.062	2993.07	4274.857	3396.03
2000	3024.5503	4015.455	3462.04	3686.353	4419.2	3353.64	3111.692	3499.381
2100	4984.682	5106.585	3339.451	3206.319	6102.428	3008.452	6180.235	3381.545
2200	3904.0237	4004.407	3499.03	3180.93	5009.597	4163.797	5390.852	3248.532
2300	3326.3901	4438.087	3309.986	3827.221	4498.915	5244.173	4708.613	3237.241
2400	3617.3625	3705.34	3321.961	3443.669	7043.6	5192.322	5087.759	3612.678
2500	3426.1863	3493.156	3101.16	3278.573	5294.611	3262.285	5959.082	3029.704
2600	3929.536	2989.818	3169.209	3348.961	4787.855	4064.311	7771.3	3839.325
2700	3318.4084	2877.347	3041.625	3465.139	3686.567	3402.472	5554.846	3434.443
2800	2798.2139	2729.84	3553.2	3015.829	6963.724	3806.571	5296.702	3893.353
2900	2653.2427	3578.958	3330.19	3079.564	6497.482	3760.58	3342.895	3861.117
3000	2671.6992	3000.713	3196.162	3112.699	3264.097	4493.842	5113.974	4389.064
3100	2956.447	3357.973	3479.73	3169.381	3712.669	3669.551	5344.856	4664.851
3200	3287.3135	3001.875	3196.72	2764.877	4382.15	3182.293	3989.385	3054.209
3300	2486.855	2674.501	3894.077	3264.268	3460.631	4561.381	5568.212	2944.834
3400	3681.2234	3264.878	3909.351	2989.094	3445.657	2945.227	3972.854	4048.585
3500	3042.2605	2669.779	3652.555	2723.573	4546.983	4400.884	5114.647	3656.315
3600	3042.0315	3091.007	4027.494	3650.473	5930.231	3982.463	6214.663	4501.413
3700	3043.7842	3014.716	3912.738	3008.312	3147.093	3602.845	6376.496	2952.597
3800	3102.6042	3313.633	3447.961	3930.166	3914.254	3298.151	5062.238	3494.175
3900	3013.213	2299.637	4141.134	3733.276	5935.788	4126.57	5946.967	3836.997
4000	3759.0522	2979.182	3887.857	3650.793	4862.929	4629.568	6099.234	2529.98
4100	3668.0715	3541.252	3407.547	3026.772	4674.399	3848.662	5993.873	2789.232
4200	2730.591	3756.239	4172.849	3658.614	3783.238	3288.383	4222.746	3993.321
4300	3949.8096	2681.376	4080.428	3128.34	4029.699	3868.773	3971.199	3006.097
4400	3323.0835	2296.129	2628.662	2761.743	4746.249	4753.029	4173.327	4290.566
4500	3658.6946	3549.945	2943.47	3554.22	4932.861	3854.719	4499.232	4120.625
4600	3498.2964	2664.737	2869.677	2947.416	5012.292	5464.019	5441.119	3875.83
4700	3218.3953	2993.687	2914.067	2931.284	5009.84	5457.208	4473.262	3604.246
4800	3705.17	3623.094	3179.02	3024.836	4351.454	7997.697	7150.244	2385.476
4900	3371.4956	3825.662	3449.096	2820.489	4956.879	5906.345	4408.281	3551.005
5000	3406.368	3478.585	3437.806	3316.647	5550.663	5213.405	7000.541	3251.957
5100	3658.293	3001.878	2934.582	3339.919	4153.177	5386.484	4569.976	4214.997
5200	4194.59	3349.578	2663.698	4014.084	4672.335	4513.1	6939.784	3541.404
5300	3139.5151	3182.787	3781.348	2353.679	4483.329	8713.253	7396.936	3471.748
5400	3214.4783	3224.414	4276.697	4426.463	3902.706	7097.28	5437.933	3852.584
5500	3932.4114	3201.333	2812.856	3057.271	4108.582	5330.817	7914.916	2754.497

Table A.7: Validation perplexity of each curriculum trained on sentence based wikitext-103 measured every 100 batches. First 5500 batches.

Batches	Baseline	Random	Dependency	Part of Speech	Unigram	Bigram	Trigram	Length
5600	2864.2917	3627.737	3003.349	2815.709	4386.842	4327.014	5232.348	2702.415
5700	3538.355	3176.168	3046.171	3060.973	3772.631	5686.305	7596.855	3421.795
5800	3252.9683	3272.469	2486.868	4196.891	5201.184	5386.043	6068.738	3089.486
5900	3131.4595	3539.543	2635.31	2772.455	5381.719	7309.636	6318.746	3442.756
6000	2794.038	3587.836	2825.41	3244.816	5596.753	6640.867	7775.148	3509.842
6100	3251.644	3263.527	2918.197	3100.223	3635.773	5764.608	6505.952	3940.546
6200	3002.94	3363.021	2487.748	3734.48	4609.224	6966.229	6210.936	3022.155
6300	3218.558	3161.412	2762.742	3631.77	4138.99	6753.394	7590.026	2781.639
6400	3168.3445	2745.09	2539.634	2845.859	3643.194	5790.03	5593.007	3362.367
6500	3563.9138	2631.605	2615.582	3017.273	5498.478	8534.298	6150.414	2755.562
6600	3336.5125	3152.491	2045.112	3510.307	6806.122	6820.456	8015.877	1892.766
6700	3095.4402	3280.077	2363.671	2707.773	4330.916	7460.527	6372.921	2842.231
6800	3707.2769	3402.757	2401.957	3146.946	5093.779	5133.348	5852.205	3127.991
6900	2563.3862	2491.408	3224.506	2928.696	5140.863	5176.373	6588.458	2814.376
7000	2950.2212	2558.31	3351.246	2230.343	5639.729	6032.242	6812.246	2852.699
7100	3118.5103	2508.693	2510.393	3488.994	7993.922	5985.55	7232.887	3199.032
7200	2862.5986	3386.939	3137.043	3367.331	4575.148	5486.356	7562.751	2948.376
7300	3581.2903	3661.634	3901.34	3907.249	5122.223	6055.366	10125.68	2535.578
7400	2722.6355	3448.971	3190.647	3167.26	5884.57	6804.227	5294.717	2535.351
7500	3889.6006	3437.576	3360.645	2520.448	5903.479	8383.122	6737.35	2751.326
7600	3435.7764	3023.607	3594.426	2703.097	6273.879	4891.422	6465.556	2662.755
7700	2684.486	3649.864	3326.815	2410.613	6438.52	4077.892	8025.103	2668.974
7800	4623.788	3301.87	3548.683	3013.871	7893.651	4928.719	6681.804	2631.33
7900	4340.479	3161.765	3719.331	2770.811	4452.123	3993.706	7188.245	2779.878
8000	3961.5264	3807.014	3452.459	2907.043	3954.853	4021.901	7929.308	2643.695
8100	3549.708	3812.812	3715.364	2949.986	4943.123	5520.156	5313.925	2967.196
8200	2721.3855	4309.146	3804.868	3194.562	4426.138	4493.306	4023.474	2833.751
8300	3770.7717	4154.175	3602.226	2562.071	4908.361	4478.966	4814.494	2852.929
8400	2819.1184	3395.214	2647.107	2739.021	4440.539	3912.589	4741.123	2516.4
8500	2855.0764	4537.215	2495.408	3064.236	5412.395	3348.591	5236.132	2208.325
8600	3505.9644	5123.063	2870.41	2727.767	5053.71	3948.653	4403.252	2785.527
8700	2836.03	2766.154	2849.205	3117.907	5864.273	3649.038	5827.049	2355.334
8800	2745.462	4155.193	2720.012	3277.113	5727.051	4311.576	4987.207	2098.654
8900	2791.9512	3517.63	2800.61	2664.944	6971.113	3963.825	5239.959	2229.124
9000	3544.8262	2923.488	3053.754	3133.222	6793.178	3378.838	6722.614	2413.132
9100	3718.3665	4568.477	2841.7	3127.884	4710.665	4798.744	5396.799	2281.898
9200	2808.6433	2312.481	1903.869	2207.872	4649.204	3642.391	5320.887	2199.175
9300	2165.4578	3975.026	2209.777	2472.435	3676.473	5090.67	5474.493	2135.631
9400	2339.3792	5000.876	2221.992	2454.856	5797.444	4136.993	5279.44	2281.424
9500	2696.5264	4750.741	2413.11	2918.894	6209.71	5272.023	6008.77	2130.816
9600	3324.4998	6199.148	1773.672	2194.82	4300.943	7148.329	4094.777	2508.34
9700	2773.0059	8854.468	2118.811	2509.108	4839.8	3611.947	5180.166	2075.662
9800	3280.1736	4203.143	1835.579	2932.314	4556.438	3910.895	5137.227	2268.739
9900	3473.049	3118.263	2283.341	2728.226	4778.486	4880.072	4650.047	2173.566
10000	3186.8086	5426.308	2254.678	2943.919	4426.129	2952.908	5637.761	2230.848
10100	2909.2644	4874.029	2301.296	2268.375	4064.935	4881.747	5865.94	2422.808
10200	2970.3079	4136.014	1920.374	2401.191	4241.271	4340.136	4660.822	2176.211
10300	4404.0874	3743.304	2076.806	2608.362	2282.178	4047.018	5157.879	2288.977
10400	2650.5618	5259.514	1997.169	2377.451	4587.451	3908.919	4345.876	2457.326
10500	3188.861	4863.026	1921.195	3030.305	3993.9	4020.728	5001.028	2137.973
10600	2928.7676	5619.987	2021.615	1937.735	6627.106	3641.662	5715.097	2114.58
10700	2990.6763	4632.581	1891.141	2465.603	5435.766	3554.868	4419.398	2176.237
10800	3106.0146	4674.662	2052.121	2789.199	3919.651	4020.114	4256.856	1980.449
10900	3359.8572	4426.294	2156.759	2719.192	4010.09	3295.575	4190.228	2470.908

Table A.8: Validation perplexity of each curriculum trained on sentence based wikitext-103 measured every 100 batches. Batches 5600-10900.

Batches	Baseline	Random	Dependency	Part of Speech	Unigram	Bigram	Trigram	Length
11000	262300.6	34450.24	6840.316	10874.94	19338.29	4964016	11612.11	4559.554
11100	2600.6765	4998.616	2287.749	2902.101	3135.488	5292.259	4984.297	2497.218
11200	2697.7224	5610.622	2237.043	3115.647	2486.967	3883.755	4174.605	2403.552
11300	3249.2195	3319.064	2291.24	3233.927	3354.398	3823.685	3968.204	1961.928
11400	2947.6704	3926.981	1995.444	3182.323	4094.028	3279.589	3376.812	1822.583
11500	2266.9912	3061.715	1997.625	2737.797	4591.544	2559.695	3850.975	2051.783
11600	1910.9951	3633.076	2017.666	2339.021	4046.791	4028.116	3389.805	1911.797
11700	2703.9133	3730.606	2161.455	2292.595	4232.192	3396.06	3526.608	1887.096
11800	1997.3059	3413.994	2171.736	2602.599	6137.459	3386.59	3847.157	2001.94
11900	2472.8403	3623.55	2011.956	2686.452	3209.035	3684.933	3470.721	2055.052
12000	2580.0857	3236.756	2047.223	2287.82	4252.911	2692.896	3619.992	2211.584
12100	2462.5713	3107.209	2061.792	2409.757	4442.029	2022.006	3572.707	1738.626
12200	2386.6958	3270.702	1742.09	2218.87	3714.259	3032.846	3149.919	2231.156
12300	2010.5862	3375.562	1971.577	2276.157	5888.235	3010.039	4748.44	2376.187
12400	3649.7207	3523.468	1807.192	2140.037	2789.801	2627.215	3046.564	2197.488
12500	3616.9211	2527.415	2066.436	2079.851	4106.784	3907.022	3353.525	2315.153
12600	3447.208	3352.406	2161.191	2616.749	2996.938	4944.613	3134.28	2356.557
12700	3567.5286	2887.539	1850.102	2555.376	4026.019	5691.963	4392.883	1837.156
12800	3496.0151	3070.08	2051.071	2613.339	3699.5	4996.399	3595.804	1779.349
12900	2957.0505	2200.042	2029.144	2694.993	2655.597	4044.406	3979.509	1772.636
13000	2889.5964	2821.793	1962.677	2848.288	4612.958	4555.686	3244.529	1971.573
13100	2884.492	3042.716	2148.672	2278.382	3911.167	4696.947	3596.171	1991.141
13200	3089.082	2329.917	2051.666	2415.661	4331.807	4364.996	3576.048	2060.151
13300	3780.2158	2874.229	2148.193	2687.167	3905.77	4439.62	3330.12	2088.202
13400	2475.67	3946.725	1880.399	2624.047	4216.894	4043.237	3692.884	1956.24
13500	2391.5706	4227.497	1702.377	2825.495	3614.787	2928.353	3596.696	2102.339
13600	2178.5073	3532.708	2186.421	2787.64	2999.277	3202.496	3107.828	1650.679
13700	3008.851	2929.372	1758.434	2765.942	3707.263	4550.124	3509.052	1763.421
13800	2215.242	3245.225	1685.312	2763.713	4098.856	2509.298	3393.489	1802.992
13900	2460.976	3058.251	1958.757	2428.149	5447.126	3144.837	3197.692	1417.536
14000	2944.5322	3279.467	1856.194	2358.343	4116.395	3639.197	3004.733	1386.126
14100	2243.2942	3078.41	1875.342	2476.079	4168.629	4346.108	2838.907	1855.109
14200	2629.6426	3068.33	1520.55	2496.886	4298.392	3075.109	3678.563	1891.769
14300	2368.2566	3311.473	2274.057	2409.711	3852.69	2250.046	3635.187	1708.987
14400	2550.2156	2957.337	2546.271	2676.356	3802.065	2900.622	3531.421	1751.901
14500	3179.2258	2918.599	3736.432	2196.153	4384.977	3041.994	3737.262	1747.746
14600	2960.2913	2671.386	1978.412	2186.668	3883.192	3858.036	3759.196	1799.298
14700	2556.041	2622.209	2180.598	2400.547	3716.956	2847.02	3890.97	1868.855
14800	2944.977	2724.473	1908.646	2003.582	2287.438	2827.39	3519.482	1935.015
14900	3058.9365	3159.76	1791.704	2110.232	4458.922	4601.517	3644.684	1814.54
15000	3373.5347	3268.794	1731.719	2115.681	4234.21	2655.245	3602.088	1910.209
15100	3619.1846	3525.784	2091.911	2336.762	3716.094	2779.542	3231.95	1780.052
15200	3959.872	3215.453	1916.56	2132.752	3879.117	3125.468	3430.319	1701.954
15300	4391.342	3927.056	1975.358	2348.073	5354.714	2415.163	3068.784	1756.15
15400	4431.6123	3377.846	1849.331	2047.901	4445.123	2385.826	3000.101	1726.074
15500	3709.352	3503.648	2244.007	2412.537	3740.15	2438.061	3197.201	1730.452
15600	4916.862	3641.964	1946.571	2569.364	5624.819	2265.686	3404.546	1756.93
15700	4265.769	3650.713	1639.628	2648.778	6393.979	2677.095	2602.288	1705.534
15800	3012.003	2658.754	2292.661	2545.123	4097.075	2777.115	3035.369	1756.578
15900	4748.132	2635.553	2167.567	1949.89	7333.727	2663.09	3760.357	1672.691
16000	3591.87	2848.635	1961.523	1914.378	8042.142	4309.771	2692.323	1816.02
16100	5728.612	2913.521	2280.985	1848.195	5667.022	2690.273	2760.998	1578.525
16200	3709.089	3093.82	2017.133	1913.861	5655.845	3693.841	2730.397	1663.372
16300	5435.975	2588.199	2200.272	1741.35	4931.408	2095.966	2910.535	1691.521
16400	3899.814	2816.65	2143.566	1520.613	5432.34	3398.078	2736.456	1738.191
16500	5760.409	2858.464	2046.358	1679.481	4213.457	5952.215	2700.678	1779.349

Table A.9: Validation perplexity of each curriculum trained on sentence based wikitext-103 measured every 100 batches. Batches 11100-16500.

Batches	Baseline	Random	Dependency	Part of Speech	Unigram	Bigram	Trigram	Length
16600	5474.128	2884.826	1884.325	1786.089	5583.255	3653.356	2235.502	1701.01
16700	5394.362	2830.033	1613.511	1855.339	5332.088	3604.594	2763.783	1802.436
16800	2142.807	3056.592	1726.021	1743.585	4894.997	3832.360	3168.774	1777.959
16900	3653.911	2477.513	1876.478	1717.887	4702.398	3547.463	3506.964	1538.599
17000	2623.699	2864.975	1814.027	1837.478	5624.287	2961.518	3777.023	1642.067
17100	3946.361	2881.194	1573.796	1626.424	5477.412	5801.621	4522.428	1686.476
17200	3931.53	2960.243	1825.012	1729.626	3872.457	5079.338	3204.623	1665.452
17300	2483.086	3014.486	1674.596	1863.654	4254.981	4786.290	2864.236	1625.513
17400	2201.756	2845.112	1661.79	1918.791	4660.876	3303.216	3083.025	1623.081
17500	3762.044	3017.391	1963.165	1948.196	4514.555	3630.837	2700.909	1610.702
17600	1916.478	3334.092	1713.173	1566.398	6403.468	5796.097	2827.983	1589.058
17700	2133.296	3893.108	1615.585	1793.384	6750.477	2867.517	2806.61	1594.783
17800	3443.813	4011.528	1590.357	1629.844	7231.735	5932.110	2400.474	1675.2
17900	3704.21	3190.787	1469.256	1593.947	6466.648	3567.406	2875.86	1626.964
18000	5203.719	3181.392	1228.67	1533.493	6644.731	3736.544	2382.266	1669.715
18100	2668.049	2297.651	1099.461	1610.487	3970.642	5157.464	2630.59	1716.483
18200	4370.502	2572.062	1273.885	1482.608	3078.424	5397.142	2872.213	1694.746
18300	5140.358	2410.223	1444.652	1588.948	3883.507	4110.709	2288.334	1652.913
18400	4342.102	2519.764	1502.864	1594.955	2383.756	4526.784	2733.952	1639.476
18500	4314.982	2582.652	1734.776	1653.818	4777.05	4260.981	2298.712	1679.498
18600	3823.823	2107.931	1651.656	1625.005	3894.389	4925.690	2919.162	1657.156
18700	3579.227	2221.069	1889.513	1596.393	6128.305	2497.559	2705.167	1640.252
18800	4485.336	2294.088	1851.501	1382.937	7034.37	4672.204	2903.462	1601.016
18900	3450.497	2346.806	1466.698	1575.33	6894.88	3873.480	2945.7	1504.928

Table A.10: Validation perplexity of each curriculum trained on sentence based wikitext-103 measured every 100 batches. Batches 16600-18900.

Figure A.1: Validation perplexity of each curriculum trained on sentence based wikitext-103 measured every 100 batches.

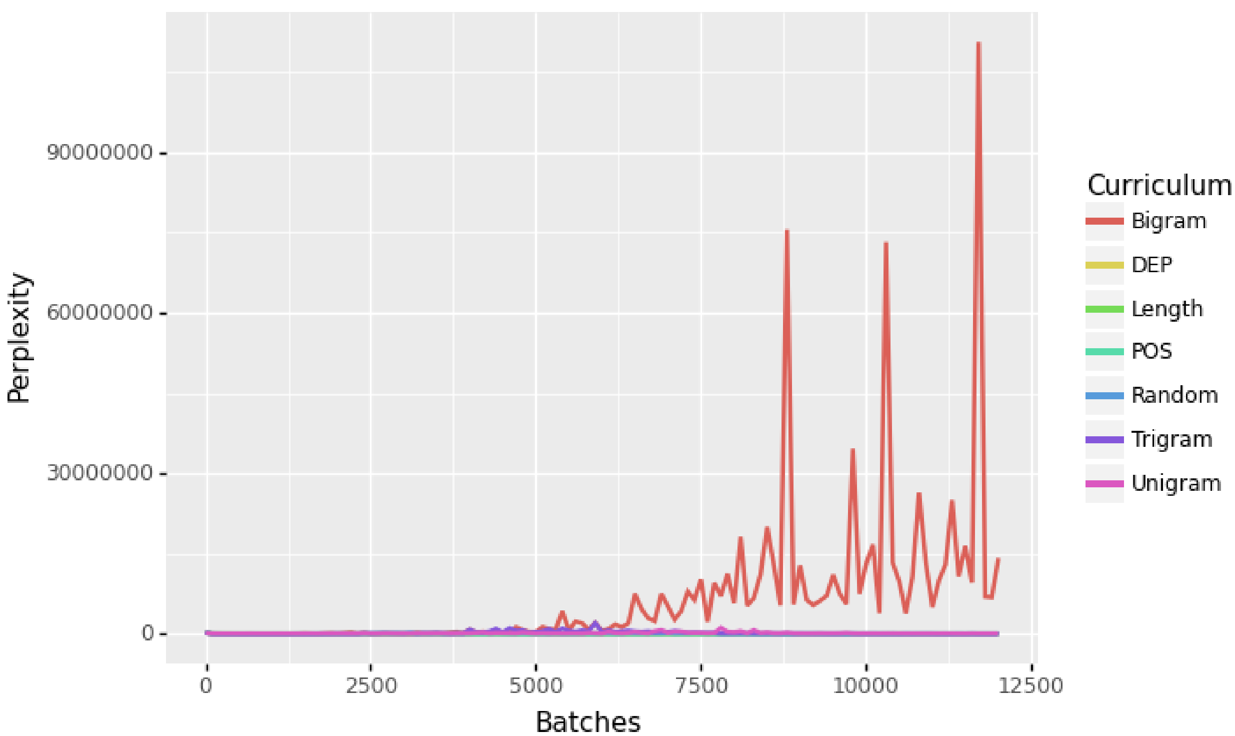

Figure A.2: Validation perplexity of each curriculum trained on line based wikitext-103 measured every 100 batches. Bigram performance is removed for ease of interpretation.

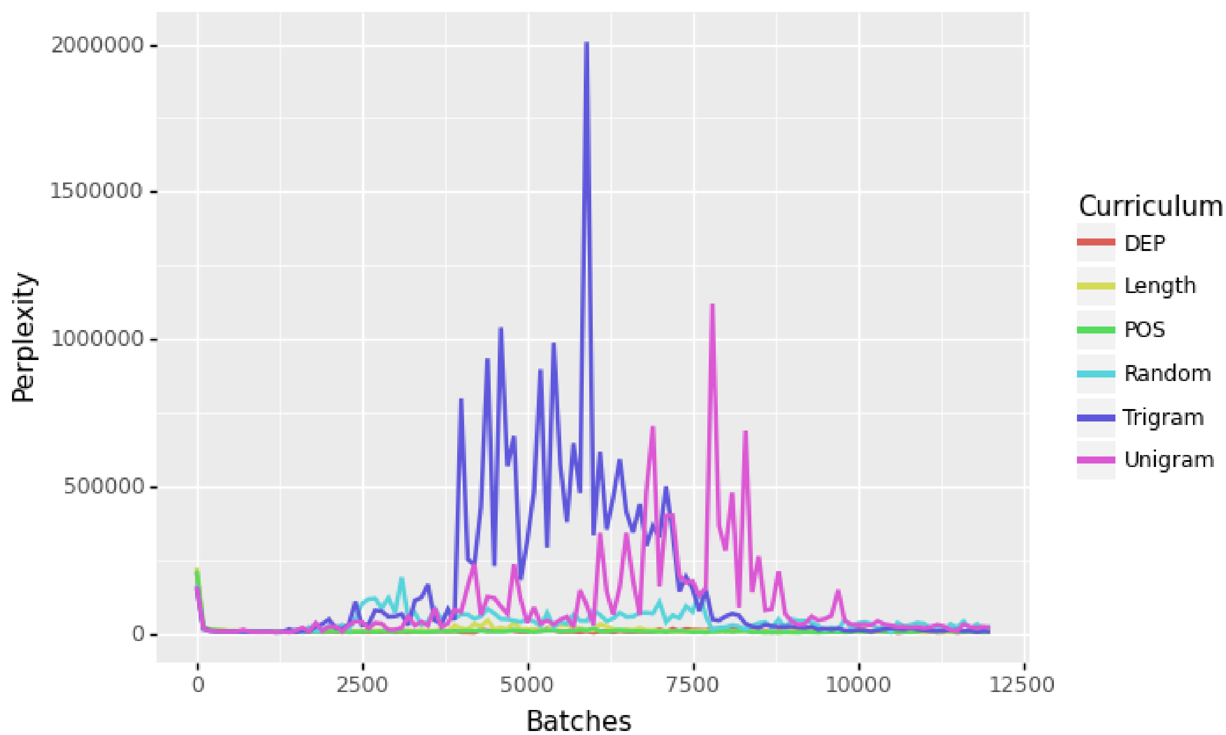

Figure A.3: Validation perplexity of each curriculum trained on line based wikitext-103 measured every 100 batches. Bigram and Baseline performance is removed for ease of interpretation.

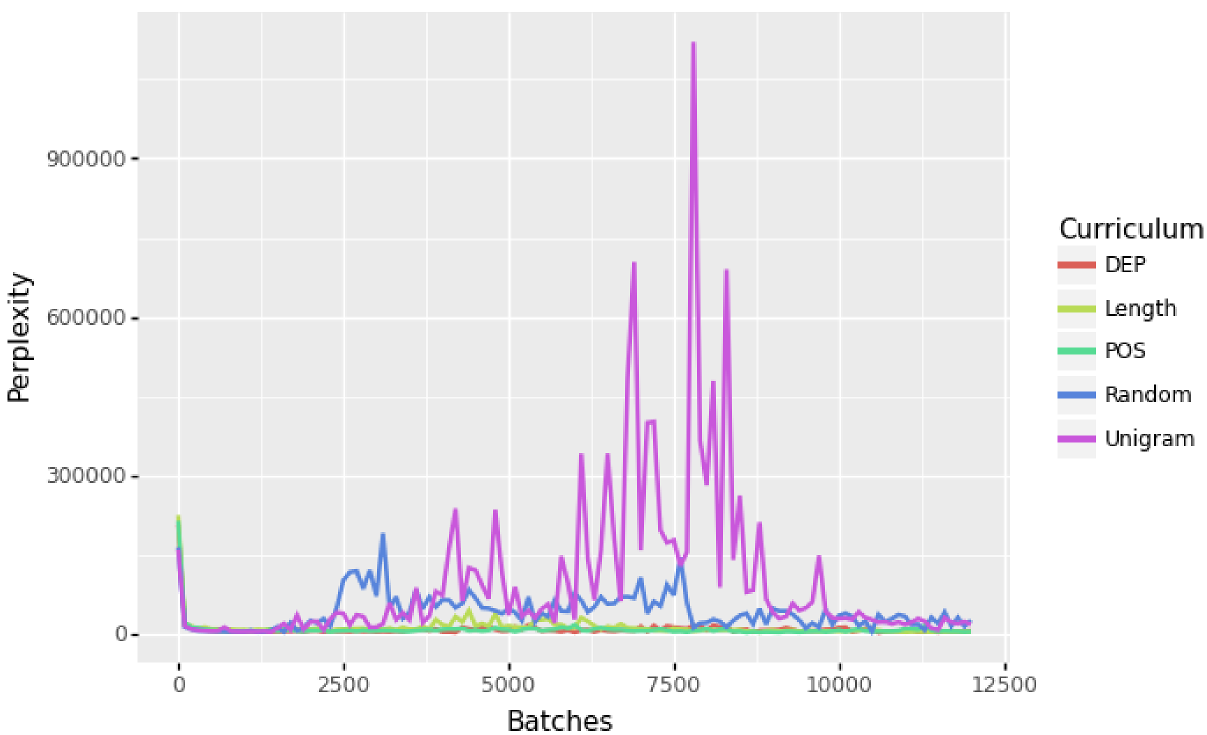

Figure A.4: Validation perplexity of each curriculum trained on line based wikitext-103 measured every 100 batches. Bigram, trigram, and Baseline performance is removed for ease of interpretation.

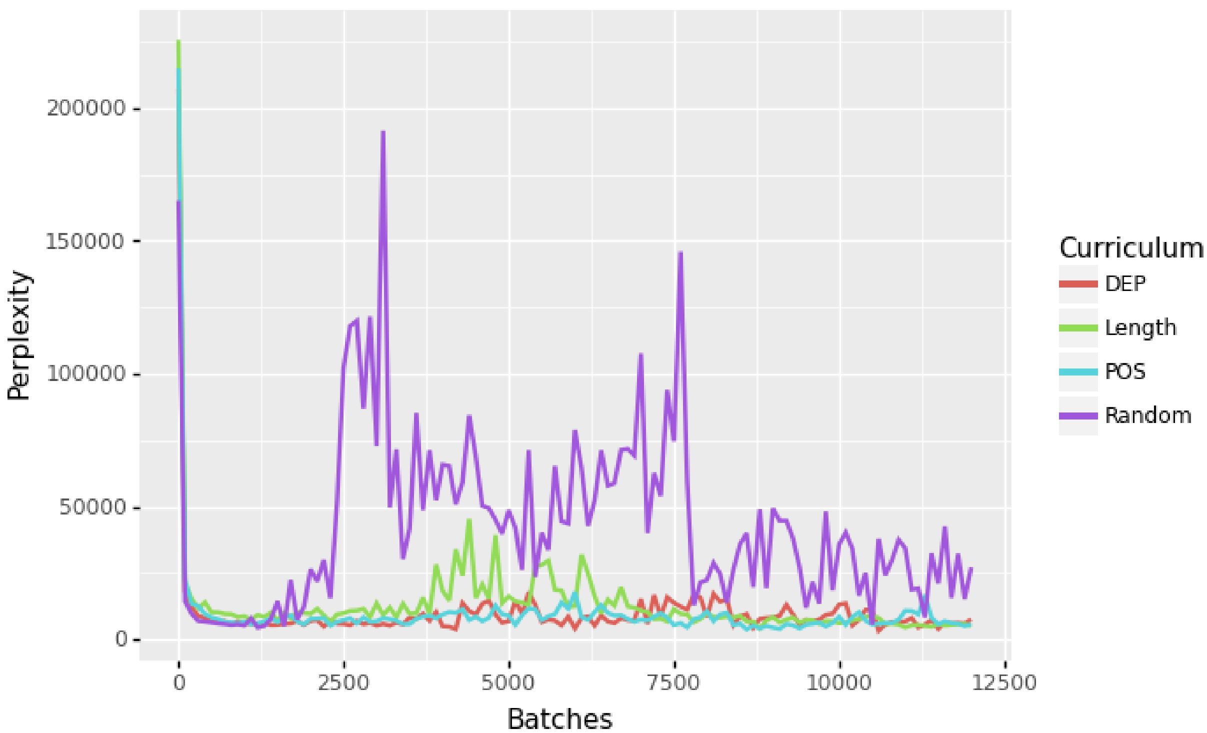

Figure A.5: Validation perplexity of each curriculum trained on line based wikitext-103 measured every 100 batches. Unigram, Bigram, Trigram and Baseline performance is removed for ease of interpretation.

VITA

Daniel Campos is a PhD student at the University of Illinois Urbana-Champaign where he is working on next generation NLP systems focused on chemistry, agriculture, and truly personal search. spacemanidol@gmail.com.