Utrecht University

Finetuning LLM's with Subtractive Curriculum Learning through Dataset Cartography

Pim Versteeg

(3804488)

Supervisor: Yupei Du, PhD Candidate
Second reader: Dr. Michael Behrisch

February 12, 2024

Submitted as a 7.5 EC thesis for the UU degree of Bachelor of Science in Artificial Intelligence

Contents

Finetuning LLM's with Subtractive Curriculum Learning through Dataset Cartography

Pim Versteeg
Utrecht University
Utrecht, the Netherlands
p.v.versteeg@students.uu.nl

ABSTRACT

Transformer-based pretrained language models (PLM's) like BERT and GPT are generally trained without attending to variation in example difficulty, instead training with the entire dataset. Swayamdipta et al. showed dataset cartography can be use to discriminate between samples within a dataset and reduce dataset size, while increasing generalization, especially for out-of-distribution data.[1] This approach was found to be ineffective for a smaller PLM like DISTILBERT. Curriculum learning (CL) using dataset cartography is proposed as an alternative. CL is a training strategy that mimics human learning: start with easy material and progress in difficulty according to student ability. CL methods require data to be ordered according to a difficulty metric. While earlier approaches in Natural Language Understanding have used linguistic heuristics to calculate difficulty, I propose using training dynamics. Training dynamics, specifically confidence and variability of confidence, are derived using dataset cartography. These provide a strong metric for ordering the dataset according to *difficulty*(confidence) and *ambiguity*(variability) and are relatively cheap to calculate. Subtractive curricula were created which adapt to model performance at every epoch. I show that using subtractive curricula can reduce training costs while maintaining and in some cases improving both in- and out-of-distribution performance.

1 INTRODUCTION

The understanding and use of natural language has long been a cornerstone of human intelligence. As such, the ability of intelligent systems to interact with humans through natural language can be considered a criterion of being considered an Artificial Intelligence(AI). In his 1637 *Discourse on the Method* Descartes already considered the development of machines capable of reproducing human speech.[1] Where Descartes considered it impossible for machines to master natural language, Turing considered it a benchmark of intelligence. Natural language capability was to be tested through his now famous Turing test. Any intelligent system that could produce responses that were indistinguishable from its human counterpart was considered by Turing to exhibit intelligent behaviour.[2]

[1]For we can easily understand a machine's being constituted so that it can utter words, and even emit some responses to action on it of a corporeal kind, which brings about a change in its organs; for instance, if touched in a particular part it may ask what we wish to say to it; if in another part it may exclaim that it is being hurt, and so on. But it never happens that it arranges its speech in various ways, in order to reply appropriately to everything that may be said in its presence, as even the lowest type of man can do.

- Discourse on the Method, René Descartes, 1637, p.34-35

In recent years the dawn of transformer-based models and large labeled datasets has driven strong advances in Natural Language Processing (NLP).[3, 4] Large language models (LLM's) performance scales with data quantity.[5] In spite of the general optimism towards this approach,[6] out-of-distribution generalization remains difficult.[7, 8] In-distribution (ID) data is data drawn from the same distribution as the training data, whereas out-of-distribution (OOD) data is data from outside of the training distribution. In practice, this often means data from domains or sources not seen during training, or with different characteristics. Model robustness is the model's ability to generalize to OOD data. A shift from data quantity to data quality could provide multiple of advantages, including OOD performance[1], training cost reduction and decrease in environmental impact.

The aim of my research is to explore curriculum methods using dataset cartography. This is motivated by the observation that the methods for data selection described by Swayamdipta et al.[1] are not applicable to smaller models, in this case DISTILBERT. Table 1 shows that while ROBERTA-LARGE performance is similar or better when using the 33% hardest or most ambiguous data, DISTILBERT performance is strongly diminished. Even when selecting 50% of data using the respective metric, performance is still poor compared to the baseline (*100% random*). Curriculum learning methods could be an alternative method of reducing data used in training that does not diminish performance.

The research question for this thesis is: can curriculum learning using training dynamics as difficulty metrics be used to reduce training costs while maintaining performance in small pretrained language models? If so, what type of curricula are effective and what hyperparameters capture these curricula.

A method for curriculum learning was designed and implemented to answer the research question. Curricula are defined in terms of a difficulty metric and a scheduler. The curricula were designed to address the short-comings of the data selection method as described by Swayamdipta et al.[1] for smaller models.

This method was tested on two datasets for Natural Language Inference(NLI): MULTINLI[9] and SNLI[3]. The scope of the research was limited to these datasets due to time and computational restraints. Datasets for Natural Lanuage Inference (NLI) were chosen as inference of entailment and contradiction requires a deep semantic understanding[3] and is a benchmark for research on NLU.[9]

Performances of models trained using this method were analyzed across different method parameters by validating on both in-domain and out-of-domain data unseen during training. These results are compared against a baseline model.

		MNLI			SNLI		
		$MNLI_M$ (ID)	$MNLI_{MM}$ (OOD*)	NLI_{diag} (OOD)	$SNLI_{val}$ (ID)	$SNLI_{test}$ (ID)	NLI_{diag} (OOD)
distilBERT	Random 100%	**80.0**	**80.2**	**54.1**	**87.9**	**87.9**	**48.3**
	Hardest 33%	38.6	38.5	32.7	65.7	65.7	42.3
	Hardest 50%	70.0	70.3	46.2	86.0	85.3	48.3
	Ambiguous 33%	61.3	61.5	46.2	84.8	84.3	45.0
	Ambiguous 50%	77.1	76.6	53.6	87.7	87.1	**48.3**
RoBERTa-large	Random 100%	**90.2**	**90.1**	65.0	**93.1**	92.0	61.8
	Hardest 33%	89.5	89.7	65.3	92.6	91.8	62.0
	Ambiguous 33%	90.1	89.3	**66.9**	92.9	**92.2**	**63.5**

Table 1: Accuracies for distilBERT and RoBERTa-large models trained on MNLI and SNLI datasets. Models were trained over 3 random seeds and tested on ID and OOD datasets. $MNLI_M$/$MNLI_{MM}$ and $SNLI_{VAL}$/$SNLI_{TEST}$ are part of the their respective datasets. textsc$MNLI_{MM}$ is described as a OOD dataset, but in practice accuracy is similar to $MNLI_M$.[9] NLI diagnostics can be considered a true OOD dataset for NLI.[10] Results for RoBERTa models are as reported by Swayamdipta et al.[1]

2 BACKGROUND

A background is given to the concepts relevant to this thesis. Language models, language representations, transformers, curriculum learning and dataset cartography are explained to give an understanding of the field within which the research was conducted.

2.1 Language Models

The currently popular pretrain-finetune style transformers, also known as *foundation models*, have a long lineage of predecessors tracing all the way back to simple statistical models. Early NLP approaches utilised simplistic representations of text, such as bag-of-words (BoW) models, which treated words as stand-alone features without considering the sequence or context of words. Instead BoW constructs a vocabulary of words, and then uses the frequency at which words appear as features.

These features could be used for statistical machine learning approaches, such as Support Vector Machines (SVM's). [11] Despite their simplicity, BoW models have been widely used in various NLP tasks, including text classification, sentiment analysis, and information retrieval. However, BoW models were always understood to be of limited usefulness due to their inability to capture both word order and semantic relations. [12]

The introduction of Recurrent Neural Networks (RNNs) marked a significant advancement in sequence modeling, allowing models to capture temporal dependencies in sequential data. Long Short-Term Memory (LSTM) networks, introduced by Hochreiter and Schmidhuber in 1997, addressed the vanishing gradient problem associated with traditional RNNs, enabling more effective modeling of long-range dependencies.[13] Gated Recurrent Units (GRUs) introduced by Cho et al. in 2014 offered a simpler alternative to LSTMs while achieving comparable performance in sequence modeling tasks.[14]

The seq2seq model can be considered the modern, direct ancestor of currently popular transformers, as it introduces the *encoder-decoder* architecture. The 380M parameter model proposed by Sutskever et al. in 2014 consisted of a pair of LSTM's. The first LSTM *encodes* a sequence of tokens into a vector. The second LSTM *decodes* the vector in a sequence of tokens.[15] seq2seq models also notably introduced *attention mechanisms*, albeit in a primitive form.[16] This allowed the decoder to decide which parts of the source sentence received from the encoder to attend to, and which parts to ignore. While this laid the groundwork for later attention mechanisms which would revolutionise NLP, these early forms of attention were additive and thus did not parallelize well.[17]

2.2 Language Representations

Finding a better way to encode words proved vital to understanding natural language. The Elman network marks the first appearance of what would later be known as word embeddings. This early approach calculated a real-valued vector for every word in its vocabulary, meaning it lacked the capability to deal with phenomena like homonymy and polysemy.[18]

Later approaches like `word2vec` [19][20] and `GloVe`[21] represent significant milestones by utilizing local context information, thereby capturing semantic and syntactic similarities between words. The `fastText` model[22] provided a way to handle *out-of-vocabulary* words, as well as morphologically rich languages. `fastText` represents words as the sum of the word embeddings of the character n-grams that make up the word, thereby encoding subword information in the word embedding.

Modern approaches like `ELMo`[23] and `BERT`[24] use a deep bidirectional LSTM/transformer architecture to create word embeddings that are capable of contextual information. In contrast to previous approaches which used static word embeddings, modern approach generate embeddings at a token level. This allows words to have different meanings in depending on their context. Both models have shown a strong understanding of underlying semantic and syntactic structures that previous models failed to capture. By encoding words within their context these approaches provide a means to deal with linguistic phenomenons like homonymy, polysemy and deixis.

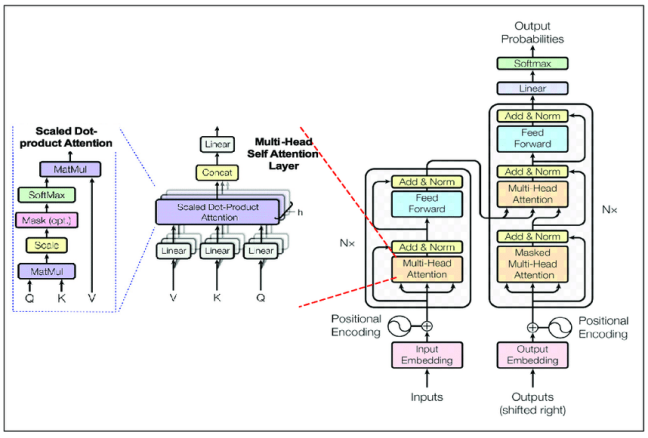


Figure 1: Diagram of the transformer architecture proposed by Vaswani et al. highlighting its attention mechanisms.[17]

2.3 Transformers

While earlier recurrent model architectures, notably long short-term memory (LSTM)[13] and Gated Recurrent Units (GRUs)[14] achieved strong in a variety of NLP tasks, they were limited due to their recurrent nature. Recurrent models proces inputs as a sequence of hidden states, where the computation of each hidden state h_t depends on the current token as well as the previous hidden state h_{t-1}.[13] This approach does not parallelize within training examples, as any hidden state h_t cannot be calculated before h_{t-1}. One of the reasons transformers revolutionized NLP is the fact that their novel attention mechanism allowed them to not rely on recurrence. This allows for significantly more parallelization while also reaching a new state of the art in NLP.[17]

Transformers use a form of attention called *Scaled Dot-Product Attention*. The attention function calculates three vectors from each token embedding within the sequence: *query*, *key* and *value*. Query vectors represent the token the head is currently processing. Key vectors represent all tokens in the sequence. Compatibility or similarity is then calculated by taking the scaled dot product of the query and key vectors. The resulting weights are then scale the value vector, which like the key vector encodes the features of all tokens in the sequence.

$$Attention(Q, K, V) = softmax(\frac{Q \cdot K^T}{\sqrt{d_k}})V$$

The weighted sum of the value vectors is the new embedding for the current token, encoding both the token in itself and how it relates to other tokens in the sequence. The attention mechanism allows transformer models to focus on different tokens by assigning weights to tokens in the same sequence as the token currently being processed.[17]

Multi-head attention calculates attention multiple times: each head uses the query, key and value vectors linearly projected using a learned linear transformation W. Different projections allow each head to attend to different aspects of the input sequence in parallel. The outputs from all heads is then concatenated and again projected. The original transformer model used eight attention heads. Multi-head attention is used in three distinct ways within transformer model.[17]

$$Multihead(Q, K, V) = Concat(head_1, ..., head_h)W^O$$
$$\text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Every encoder layer uses multi-head attention where the queries, keys and values come from the previous layer. Each layer allows the encoder to capture dependencies between tokens, resulting in strongly contextualized representations of the sequence. Attention in decoder layers similarly depend on queries, keys and values from the previous layer, but in contrast to encoder layers only have access to tokens up to the token currently being attended in order to preserve the auto-regressive property.[17]

Finally encoder-decoder layers, also known as cross-attention layers, allow the decoder to attend to all tokens in the input sequence. In encoder-decoder attention the keys and values come from the output of the encoder, while queries come from the previous decoder layer. The inclusion of encoder-decoder attention enables to model to attend to encoder output, allowing for tasks such as machine translation.[17]

In RNN's and CNN's the ordering of a sequence is implicitly captured in the model architecture. However, since transformers models utilize neither recurrence nor convolution, a different method named *positional encoding* is used. Instead of directly adding the word embeddings to either the encoder or decoder stack, the input embeddings are first summed with a positional encoding. A positional encoding is a sinusoidal function is a function of token position and embedding dimension.

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

Positional encodings are unique for each position and can be used to compare relative distance to other positions. Positional encodings are computated in advance for all positions and stored in a positional encoding matrix.[17]

2.4 BERT

BERT (*Bidirectional Encoder Representations from Transformers*)[24] is an encoder-only transformer model introduced by Devlin et al. in 2018. BERT's distinguishing feature is its bidirectionality. In contrast with RNN's and GPT[25], which are unidirectional, BERT attends to tokens both left and right of the current token. This, combined with its encoder-only architecture allows it to calculate word embeddings that capture contextual information as well as long-range dependencies.

While powerful, its encoder-only architecture means it is most suitable to language understanding, though methods have been developed for generation.[26] BERT is pretrained using masked language modelling (MLM) and next sentence prediction (NSP). The MLM task masks a word from the input sequence which is then to be predicted. The NSP task presents two sentences and asks the model to predict whether the second sentence is likely to follow the first sentence.

2.5 distilBERT

DISTILBERT is a lightweight variation of BERT, being 40% smaller, 60% faster while retaining 97% of its NLU capabilities. DISTILBERT is derived from BERT through a process called *knowledge distillation*[27, 28]. A large, complex model (e.g. BERT) called the teacher model is used to teach a smaller and simpler student model (e.g. DISTILBERT). During training, the student model has access to the predictions made by the teacher model, and tries to minimize the difference between these predictions and its own. While DISTILBERT retains the same architecture as BERT, it has fewer and smaller layers, as well as fewer attention heads. The combination of knowledge distillation and a smaller model allows DISTILBERT to represent BERT's knowledge more efficiently.

2.6 Curriculum learning

Curriculum learning was pioneered by Elman [29], and formalized by Bengio et al. [30] in the context machine learning. Curriculum learning involves intelligently designing the learning process of a model to improve performance, reduce training costs and/or promote convergence. Like children start learning about simple shapes and colours over quantum physics, models too can benefit from training on easy data before being presented harder or more complex data.[30] CL methods consist of two key components: the *scheduler* (also known as the pacing function) and the *difficulty metric* (also know as the difficulty scoring function).

The scheduler is an algorithm for deciding which examples are introduced to the model at which point during training. Schedulers may be discrete or continuous. Discrete schedulers group training examples into *buckets*. Depending on the strategy, the scheduler produces a curriculum for a specific training step may be sampled from one or more buckets.

One simple example of a curriculum strategy, known as *One-Pass curriculum*, proposed by Cirik et al.[31] divides the data into K buckets, then for epoch $e \in [1, 2, ..., K]$ trains the model on $\mathbf{X}_k, \mathbf{Y}_k^*$, the data in bucket k. Continuous schedulers forego bucketing, instead viewing the data as an ordered set. An example of a continuous scheduler is the approach proposed by Platanious et al.[32] where a monotonically increasing scheduler function returns a subset of the data to be used in training.

Self-paced learning (SPL) differs from vanilla CL strategies like One-Pass curricula by not calculating difficulty metrics a priori, instead relating the example difficulty to model performance during training. As such, example difficulty may change during training. The original SPL algorithm used likelihood of prediction as a difficulty metric[33]. The underlying concept is that model competence at a certain training step is a good indicator of how difficult the curriculum for the next training step should be.[34]

Self-paced curriculum learning (SPCL) combines vanilla CL and SPL in a single framework. Where for vanilla CL the curriculum is derived from prior knowledge and thus is fixed a priori and for SPL the curriculum is dynamic over training, SPCL utilizes both prior knowledge and model performance during training.[35] SPCL orders the data according to prior knowledge, but during training uses model performance to adapt the curriculum, i.e. which subset of the data is used.[34]

The difficulty metric is defined by difficulty scoring function γ where for examples $(\mathbf{x}_i, y_i^*)$ and $(\mathbf{x}_j, y_j^*)$ if $\gamma(\mathbf{x}_i, y_i^*) > \gamma(\mathbf{x}_j, y_j^*)$, then example $(\mathbf{x}_i, y_i)$ is more difficult than example $(\mathbf{x}_j, y_j^*)$.

$$\gamma : \mathbf{X}, \mathbf{Y}^* \rightarrow [0, 1] \subset \mathbb{R}$$

For typical curriculum learning methods, this means example $(\mathbf{x}_i, y_i)$ would scheduled for training after example $(\mathbf{x}_j, y_j)$. Examples of difficulty metrics are model loss[33] and linguistic heuristics (e.g. sentence length or relative token occurrence)[32].

Teacher-Student Curriculum Learning (TSCL) is a framework where the knowledge of a teacher model is used in curriculum creation.[34] A student model is then trained on the task using this curriculum. The teacher models can function as either a scheduler or difficulty metric. Hacohen and Weinshall showed that a teacher-based difficulty metric increases learning speed and improves performance when training deep neural networks (CNN's specifically).[36] Matiisen et al. show that a teacher model can be used for subtask selection (i.e. scheduling) to improve performance of LSTM's.[37]

CL is mostly known within the domain of reinforcement learning[38], but recently Xu et al. [39] showed a CL approach can achieve significant and universal performance improvements on a wide range of NLU tasks.

2.7 Dataset Cartography

In 2022 Swayamdipta et al. proposed *Data Maps*: a novel model-based approach to characterizing datasets. In dataset cartography *training dynamics* are used to build a data map representing how individual examples behave during training. Examples of data maps can be found in figure 2. These data maps show three distinct regions known as *easy-to-learn*, *hard-to-learn* and *ambiguous*. Data is mapped according to two (or three) metrics derived during training: *confidence* and *variability of confidence*. These metrics are named training dynamics.

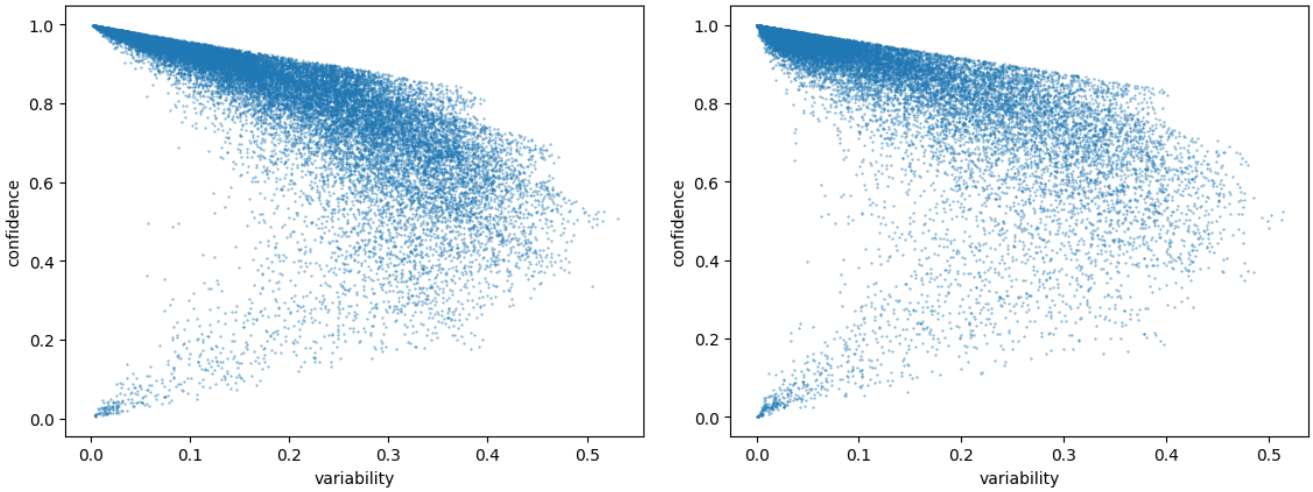
Confidence μ_i is a measure of example difficulty. Confidence is defined as the mean model probability of the golden label over all epochs:

$$\widehat{\mu}_i = \frac{1}{E} \sum_{e=1}^{E} p_{\theta^{(e)}}(y_i^* | \mathbf{x}_i)$$

In this equation $p_{\theta^{(e)}}$ denotes the model's probability with learned parameters at the end of the e^{th} epoch. An example with high confidence (>0.8) can be considered *easy-to-learn*, whereas an example with low confidence (<0.4) *hard-to-learn*. Swayamdipta et al. give no exact demarcations for easy-to-learn and hard-to-learn areas within the data map, so the given values may not correspond to other data maps. To prevent confusion with the concept of *difficulty* in difficulty scoring functions, I refer to this training dynamic as confidence from here on out.[1]

Variability of confidence $\widehat{\sigma}$ is a measure of example ambiguity. Variability is defined as the standard deviation of $p_{\theta^{(e)}}(y_i^* | \mathbf{x}_i)$ over epochs:

$$\widehat{\sigma}_i = \sqrt{\frac{\sum_{e=1}^{E} (p_{\theta^{(e)}}(y_i^* | \mathbf{x}_i) - \widehat{\mu}_i)^2}{E}}$$



(a) *MultiNLI* **(b) SNLI**

Figure 2: ***Data map for the MultiNLI and SNLI datasets. For clarity, 25K samples were plotted.***

Variability denotes how consistent a model is in assigning labels over training. An example with multiple labels over training has a high variability (>0.25) and is considered *ambiguous*. Uncertainty about the meaning of an example corresponds with our intuitive concept of ambiguity: a sentence that could be interpreted in two or more ways.[1]

Correctness is a third, more coarse, measure that is introduced. Correctness is the fraction of times where the label is correctly predicted, i.e. the parameters for the golden label $p_{\theta^{(e)}}(y_i^*|\mathbf{x}_i)$ predict a higher value than any for any other label. For a L-label dataset this would require $p_{\theta^{(e)}}(y_i^*|\mathbf{x}_i) > \frac{1}{L}$.[1] Correctness is not used in this thesis.

Swayamdipta et al. showed that data selection based on confidence or variability can improve both ID and OOD performance while reducing training costs. Performance of models trained on the 33% hardest or most ambiguous data were compared to a 100% baseline. Specifically for MultiNLI and SNLI an OOD increase of performance of respectively 1.9 and 1.7 percent point was reported. ID performance for both datasets was similar to the baseline: for *SNLI* a 0.2 percent point increase in performance was reported, while for MultiNli performance was diminished by 0.1 percent point.[1]

3 RELATED WORK

Platianos et al. propose a curriculum learning framework for *Neural Machine Translation* (NMT) using competence-based scheduler. The difficulty metrics are based on simple linguistic heuristics: *sentence length* and *word rarity*. A competence function $c(t) \in [0,1] \subset \mathbb{R}$ calculates a competence score at time step t. For time step t the curriculum contains all examples for which $\textsc{difficulty}(\mathbf{x}, y^*) \leq \textsc{competence}(t)$. Both linear and non-linear competence functions are explored during experiments. Experiments show reduced training costs and increased performance for curricula with a *word rarity*-based difficulty metric and a non-linear competence function.[32]

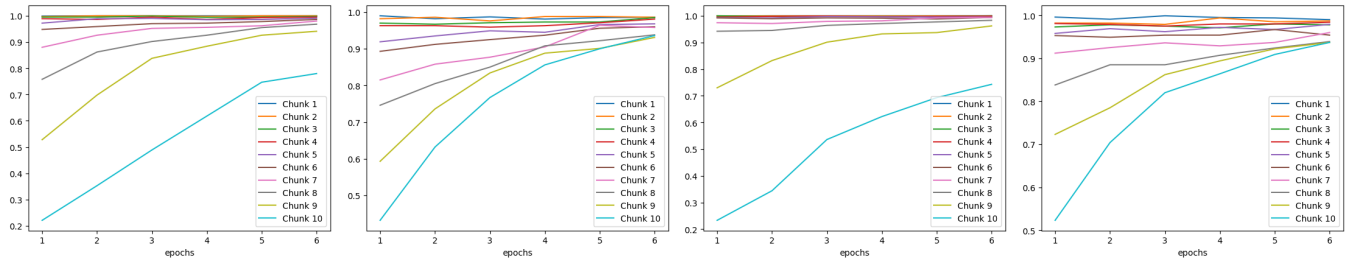
Christopolou et al. propose using training dynamics - confidence, correctness and variability - as difficulty metrics. These are combined with two schedulers: *annealing*[39] and the competence-based scheduler from Platianos et al.[32] we described earlier. The annealing scheduler requires data to be split into discrete buckets and thus is combined with correctness, which has $1 + E$ discrete values.[1] Each epoch is trained on one bucket, but $1/(E+1)$ examples are also randomly sampled from the previous bucket. Competence-based schedulers are combined with either confidence or a combination of confidence and variability.[40]

Maharan et al. propose variability-based difficulty metric as well as two other metrics. Adaptive curriculum learning[41] is used to dynamically update the difficulty score using training loss as a measure of model competence. Adaptive curricula outperform both the baseline and static curricula consistently over a variety of commonsense reasoning tasks. Question Answering Probability (QAP)[42] slightly outperforms variability as a DSF, but differences are very small (0.1 to 0.2 percent point). Additionally, QAP is shown to be correlated with variability, explaining the similar performance.[40]

İnce et al. use dataset cartography in data selection for training models for *compositional generalization* tasks. The approach is similar to Swayamdipta et al.[1] where 33% subsets of the data based on data maps are selected for training. Experiments show the importance of *hard-to-learn* data in these tasks. İnce et al. also explore curricula based on training dynamics, which confirm the importance of hard-to-learn data in curricula. [43]

3.1 Insights

Findings of these papers were considered when designing the CL method presented in this thesis. The results achieved by Platianos et al. show schedulers based on model competence are effective for transformer-based models.[32] Potential for improvement lies in the competence function which is known a priori and thus model-agnostic. While this is advantageous in terms of simplicity due to



(a) MNLI / confidence **(b) MNLI / variability of confidence** **(c) SNLI / confidence** **(d) SNLI / variability of confidence**

Figure 3: Plots showing the validation accuracy of each bucket over epochs. Data was sorted according to either confidence or variability of confidence, where bucket 1 is the *easiest*/*least ambiguous* and bucket 10 is the *hardest*/*most ambiguous* respectively. A random sample of size 1000 was taken from the chunk for validation. Note: buckets are named chunks in the legends. Higher resolution plots are available in the appendix.

lack of tuneable hyperparameters, it also ignores the model as a source of information.

Christopolou et al. and Maharan et al. show CL methods can be effective for learning NLU tasks. These papers also show training dynamics can be effective as an estimate of example difficulty.[40, 44] The adaptive curriculum learning approach proposed by Maharan et al. shows that model state can be used as a measure of model competence by using training loss to update the difficulty scores over training.[44] İnce et al. show the importance of hard-to-learn data in curricula.[43]

4 METHODS

The approach presented in this thesis differs from related work in two main facets: (1) model competence is determined by validating buckets and (2) instead of introducing more difficult data to the curriculum over training, instead easier data is removed from the curriculum. An overview of the training pipeline can be found in figure 4.

The dataset was sorted according to the difficulty metric and split into N equal-sized buckets. For example, when a *confidence*-based difficulty metric is used, the first bucket will be the *easiest* and the last bucket will be the *hardest*. Similarly, if a *variability of confidence*-based difficulty metric is used, the first and last buckets will respectively be the least and most *ambiguous*. These buckets were validated during training by calculating the accuracy of the model on each bucket.

For the set of training examples $\mathbf{X}$, the ordered set of buckets $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^{N}$ and a difficulty scoring function $\gamma : \mathbf{X} \to [0, 1] \subset \mathbb{R}$ a bucket $\mathbf{b}$ is defined as follows:

$$\mathbf{b} \subset \mathbf{X} \qquad \text{where } \gamma(\mathbf{v}) \leq \gamma(\mathbf{w}), \mathbf{v} \in \mathbf{b}_i, \mathbf{w} \in \mathbf{b}_j, \forall i < j \text{ and } \forall i, j\ |\mathbf{b}_i| = |\mathbf{b}_j|$$

The validation accuracy for each chunk with different datasets and difficulty scoring functions can be seen in figure 3. These plots show that there appears to be a strong correlation between chunk difficulty/ambiguity and model accuracy on that bucket. *easier* and less *ambiguous* buckets reach a near 100% accuracy at the end of the first epoch, while accuracy *harder* and more *ambiguous* buckets grow more slowly and plateau at a lower accuracy. This observation is congruent with the idea that training dynamics provide a strong metric for how difficult data is to learn. It also supports the potential of training dynamics as difficulty metrics.

4.1 Scheduler

The observations of bucket behaviour were developed into a variant of Adaptive Curriculum Learning (ACL) called Subtractive Curriculum Learning (SCL). Under the assumption that *easier*/*less ambiguous* buckets can be removed from the curriculum as the model becomes more competent over training[40, 44], the scheduler initially includes all buckets in the curriculum, then excludes buckets as soon as their validation reaches a threshold value θ. At the end of every epoch, the easiest or least ambiguous bucket is evaluated and its accuracy is compared to threshold θ. If the accuracy falls below threshold θ, the bucket is removed from the curriculum for the remaining epochs. Multiple buckets can be evaluated at the end of an epoch. Bucket evaluation stops when accuracy falls below threshold θ, under the assumption that further buckets with a higher difficulty metric will have a lower accuracy and thus will also be below the threshold. The exclusion of buckets results in a reduction of training cost. The subtractive scheduler was chosen based on three considerations.

Firstly, although training with *ambiguous* examples is known to improve OOD performance while maintaining ID performance, ambiguous examples alone are insufficient for effective learning. Swayamdipta et al. show the importance of using a small portion of *easy* examples in training for successful optimization.[1] This is also observed in the baseline experiments we observe: table 1 shows using solely *hard-to-learn* or *ambiguous* examples result in poor performance. This is especially true for the models trained on *hardest* 33% examples of MNLI, which barely perform above a *random* classifier (even slightly under for the GLUE DIAGNOSTICS dataset).

Secondly, when using *variability of confidence* as a difficulty metric the examples with low difficulty come from two distinct regions within the datamap (figure 2). Using only this data for initial training would expose the model to both the *easiest* and *hardest* examples simultaneously, where the *hardest* examples also have a high likelihood of being noisy/mislabeled.[1]

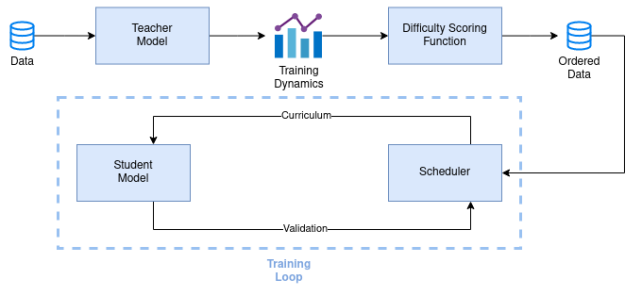


Figure 4: ***Curriculum learning with training dynamics and adaptive scheduling.***

Finally, due to the nature of pretraining PLM's already posses a strong basis of understanding for natural language. Validation accuracy scores for *random* 100% baseline models show that accuracy scores are high after the first epoch of training, then improve only marginally over the following epochs. Baseline validation scores improved for by only 0.7 and 0.5 percent point for MultiNLI and SNLI respectively during experiments.

The idea of subtractive curricula draw parallels to real-life learning. Students first develop a strong basis of knowledge of the entire field of study, before specializing and learning more difficult concepts that may rely on basic knowledge. Similarly, the initial broad curriculum is intended to provide the model with a basic understanding of the task. Curricula during later training become more *specialized* by removing examples deemed too easy and focusing on data proven to be useful in OOD generalization.[1]

4.2 Difficulty scoring function

The difficulty metrics are derived by training a teacher model. The teacher model is a distilBERT model trained on a 100% dataset for a single run. Training dynamics were calculated during this run. The difficulty scoring function assigned difficulty scores to all examples in the training data. The training data was then ordered according to their difficulty score.

4.3 Threshold θ

Threshold θ is a tunable hyperparameter that determines the threshold of accuracy on which buckets are excluded from the curriculum. Lowering θ decreases training cost, but might have effects on model performance. The effects of θ on model performance and training costs are experimentally tested and analyzed later in this thesis.

4.4 Learning Rate

A consideration for this approach is how to adapt the learning rate over training. For adaptive curricula the total number of examples seen during training is not known a priori. The standard approach of creating a learning rate schedule a priori is therefore not feasible. Instead, I chose to linearly decay learning rate at the end of every epoch. Using this method, the learning rate schedule is similar to the batch-based linear decay used in the baseline, except the approximation of linear decay is more discrete as it compromises of as many values as the number of epochs. For the baseline models, learning rate is decayed at the end of every batch resulting in a more continuous decay function.

4.5 Pseudocode

The algorithm for subtractive curriculum learning is described in pseudocode below, where θ is the threshold hyperparameter, E is the number of epochs and N is the number of buckets. The bucketing algorithm is abstracted from for clarity, but is implemented as statistical data binning based on the difficulty metric.

Algorithm 1 Subtractive Curriculum Learning (SCL)

$\theta \leftarrow \{x \in \mathbb{R} : 0 \leq x < 1\}$ ▷ constant
$E \leftarrow \{x \in \mathbb{N} : x \geq 1\}$ ▷ constant
$N \leftarrow \{x \in \mathbb{N} : x > 1\}$ ▷ constant

`data` $\leftarrow \gamma($`data`, `scores`$)$ ▷ ranking
$\mathbf{B} \leftarrow \{\mathbf{b}_i\}_{i=1}^{N}$ ▷ bucketing
`curriculum` $\leftarrow \mathbf{B}$

for e in E **do**
  `model` $\leftarrow train($`model`, `curriculum`$)$
  for b in `curriculum` **do**
    $accuracy \leftarrow evaluate(\mathbf{b})$
    if $accuracy > \theta$ **then**
      `curriculum` $\leftarrow$ `curriculum` $- \mathbf{b}$ ▷ subtraction
    else
      break

5 EXPERIMENTAL SETUP

Experiments consisted of finetuning models, then validating the models on both in-distribution and out-of-distribution datasets. Efforts from other research described earlier were not considered for comparison as they used different model architectures. As these were generally models with a much larger parameter size it was impossible to do a fair comparison. Christopoulou et al. and Maharana et al. used $\texttt{RoBERTa}_{\texttt{BASE}}$(123M parameters) and $\texttt{RoBERTa}_{\texttt{LARGE}}$(354M parameters) respectively[40, 44]. For comparison, `distilBERT` only uses 66M parameters. `distilBERT` was trained on the original 16GB `BERT` dataset, whereas `RoBERTa` was also trained on this dataset as well as an additional 144GB dataset.

The experiments were designed to explore different methods of using training dynamics, as the methods described in Swayamdipta et al.(2019)[1] were found to be ineffective for smaller models, specifically distilBERT.

5.1 Data

Two datasets were selected for finetuning experiments: the Multi-Genre Natural Language Inference (MultiNLI or MNLI) corpus [9] and the Stanford Natural Language Inference (SNLI) corpus [3]. These datasets were selected due to them both being Natural Language Inferences (NLI) corpora as well as popular benchmarks for Natural Language Understanding. Both datasets were also included in Swayamdipta et al. (2019)[1], making it possible to compare results between distilBERT and RoBERTa-large models. Finally, at

393K and 550K training examples, both datasets are of a size where there is a sufficient spread of examples across the two dataset cartography variables used (confidence and variability). This ensures the data maps are well-populated.

Another advantage of selecting corpora for the same task (NLI) and with the same example/label format is that models trained on either dataset can be validated on the GLUE DIAGNOSTICS dataset. GLUE DIAGNOSTICS is a small, handcrafted NLI dataset created to diagnose model weaknesses across common phenomena, including world knowledge logical operators and lexical entailments. It is known to be significantly harder than other GLUE datasets.[10]

The Stanford Natural Language Inference (SNLI) corpus[3] is a benchmark dataset for the task of Natural Language Inference (NLI). The task associated with the dataset consists of determining whether the *hypothesis* sequence logically follows from the *premise* sequence. Possible labels include:

(0) *entailment*: The premise entails the hypothesis ($P \vDash H$).

(1) *neutral*: The premise neither entails nor contradicts the hypothesis ($P \nvDash H$).

(2) *contradiction*: The premise contradicts the hypothesis ($P \vDash \neg H$).

SNLI consists of 570K labeled examples of premise/hypothesis pairs. The premises were sourced from the Flickr30k corpus[45], a corpus of 160K crowd-sourced captions for 30K images. For each premise, three hypotheses (one per label) were manually composed by crowd workers. SNLI contained 785 examples without a gold label. These examples were removed from the dataset during preprocessing.

The Multi-Genre Natural Language Inference (MULTINLI) corpus[9] is also a benchmark dataset for the task of Natural Language Inference (NLI). This corpus contains 433K examples from across a variety of domains and text genres. The format of the data is equal to that of SNLI. Premises for MULTINLI were sourced from ten diverse genres. Nine genres were sourced from the Open American National Corpus (OANC)[46–49], including face-to-face conversations, telephone conversations, travel guides and letters. Premises for the tenth genre were sourced from works of fictions across literary genres from 1912 to 2010. MULTINLI seeks to improve over SNLI by sourcing from multiple genres of text and increasing task difficulty.

5.2 Difficulty scoring functions

The experiments used two different difficulty scoring functions to sort and bucket the data according to difficulty ahead of curriculum generation: confidence (*difficulty*) and variability of confidence (*ambiguity*). Examples were ranked in descending order of confidence and ascending order of confidence of variability; i.e. from *easiest*/*least ambiguous* to *hardest*/*most ambiguous*.

5.3 Threshold θ

Different values of threshold parameter θ were tested over experiments. Experiments for confidence-based curricula used $\theta \in [0.95, 0.98, 0.99, 0.995]$. Experiments for variability-based curricula used $\theta \in [0.93, 0.93, 0.95, 0.98, 0.99, 0.995]$. These parameters were selected through analysis of the bucket behaviour data shown in figure 3.

5.4 Baselines

Five baseline experiments were designed: *random* 100%, *hard* 33%, *hard* 50%, *ambiguous* 33%, and *ambiguous* 50%. As described earlier, the latter 4 all showed reduced performance compared to the *random* 100% baseline, prompting the exploration of curriculum learning methods as an alternative. Baseline experiments were conducted across both datasets, resulting in ten experimental setups.

5.5 Training

The SCL experiment parameters (dataset, difficulty scoring function and threshold) resulted in twenty experiments, in addition to ten baseline experiments, for a total of thirty experiments. For each experiment three models were trained with randomly initialized parameters.[2] All models were trained over 6 epochs using the similar hyperparameters.

The ADAMW optimizer was used with linear decay and no warm-up. Initial learning rate was set at 5e−5. For SCL experiments, the total amount of batches was not known a priori due the adaptive nature. For these experiments, the learning rate was decayed at the end of every epoch, instead of at the end of every batch.

All models were trained using FP16 mixed precision training.[53] Training was done on a single A100 GPU with batch size 128. SCL methods and experiments were implemented in Python using the `Hugging Face` and `PyTorch` libraries. Code is available at Google Colab.

6 RESULTS

The results of the experiments are presented in table 2. The results of dataset cartography are also discussed. For this I refer back to 2a and 2b as presented earlier. The results are analyzed and compared to eachother as well as the baseline. Lastly, the effects and tuning of hyperparameter θ and the effects of the difficulty metric are discussed.

6.1 Dataset cartography

Figures 2a and 2b show data maps for the MULTINLI[9] and SNLI[3] datasets respectively. Both MULTINLI and SNLI data maps show the majority of the data is relatively easy (high confidence) and unambiguous (low variability). Variability is highest for medium confidence examples and lowest for either high or low confidence samples. Both data maps show a distinctive bell curve as observed by Swayamdipta et al. in their research.[1]

Comparison of data maps reveals that SNLI is an *easier* and *less ambiguous* dataset than MULTINLI, as is consistent with the literature.[9] From these data maps, it can be hypothesized that models trained on SNLI will generally perform better on ID validation compared to MULTINLI, while having worse performance on the harder OOD dataset. This hypothesis will be returned to in later results.

[2] Model parameter initialization are commonplace in NLP as well as other machine learning fields. Models are sensitive to parameter initialization and other sources of randomness due to their non-convex optimization landscape, initialization bias and inherent stochastic nature[50–52].

	θ	MNLI Training Cost	MNLI_M (ID)	MNLI_MM (OOD*)	NLI_diag (OOD)	SNLI Training Cost	SNLI_val (ID)	SNLI_test (ID)	NLI_diag (OOD)
Random 100%	n/a	100%	79.7	79.9	54.0	100%	87.9	87.6	48.3
Confidence	0.95	46.7%	65.5	65.1	42.8	36.7%	56.6	56.0	41.1
	0.98	53.3%	73.8	74.1	48.3	43.3%	71.9	71.3	49.2
	0.99	65%	79.3	78.9	54.5	51.7%	81.8	79.7	**50.2**
	0.995	73.3%	79.6	**80.3**	**55.3**	58.3%	87.6	86.7	49.5
Variability	0.93	46.7%	78.9	78.5	54.3	43.3%	87.7	86.9	49.9
	0.94	56.7%	79.0	79.5	54.8	48.3%	87.8	87.3	49.9
	0.95	56.7%	79.3	79.6	55.1	48.3%	87.8	87.5	48.6
	0.98	88.3%	79.5	**80.3**	54.9	68.3%	**88.5**	87.6	**50.2**
	0.99	100%*	79.5	78.9	54.0	88.3%	88.1	**87.9**	49.6
	0.995	100%*	**79.9**	79.9	53.9	100%*	87.8	87.7	48.5

Table 2: Results of experiments using Subtractive Curriculum Learning. Models were trained over 3 random seeds and tested on ID and OOD datasets. MNLI_M/MNLI_MM and SNLI_VAL/SNLI_TEST are part of the their respective datasets. MNLI_MM is described as a OOD dataset, but in practice accuracy is similar to MNLI_M.[9] NLI DIAGNOSTICS can be considered a true OOD dataset for NLI.[10] Experiments marked with an asterix are those where threshold was never reached, i.e. the curricula for all epochs consisted of the full training set (all buckets).

6.2 MultiNLI

Curricula using confidence-based difficulty metrics show slight improvement or near-match in accuracy on both ID sets, as well improvement on the OOD set for $\theta \in [0.99, 0.995]$. In these experiments training costs were reduced by 35% and 26.7% respectively. Higher values of θ correlate with an increase in performance across test sets.

Curricula using variability-based difficulty metrics show improvements in performance for a wider range of θ-values: all $\theta \in [0.93, 0.94, 0.95, 0.98]$ all show improvements in performance on the NLI DIAGNOSTICS set. Notably, the variability-based curriculum shows a slight improvement in performance while reducing training costs by 53.3% compared to the baseline.

For models trained with a variability-based difficulty and $\theta > 0.99$ bucket evaluation never reached the threshold, thus no buckets were removed from the curricula. The experiments are in effect baseline experiments, which is reflected in the results.

6.3 SNLI

Results of experiments performed on the SNLI dataset are similar to those of experiments performed on the MULTINLI dataset, albeit that a stronger ID and a weaker OOD performance is observed. This is consistent with the hypothesis formulated from the data maps as discussed earlier, i.e. models trained on SNLI are more prone to overfitting leading to poor generalization. In this case, overfitting is mainly due to SNLI being an overall *easier* and *less ambiguous* dataset than MULTINLI,[54] although there may be other factors.

Confidence-based curricula with $\theta \in [0.98, 0.99, 0.995]$ show an increase in OOD performance of up to 1.9 percent point. Notably, all of these curricula show a decrease in ID performance, although the models trained with $\theta = 0.995$ show near-baseline performance. All confidence-based curricula are cheaper in terms of training costs, where notably the model trained with $\theta = 0.98$ showed a 56.7% decrease in training costs while increasing performance on the OOD set by 0.9 percent point.

Similarly to the variability-based, high θ models for MULTINLI, models trained with a variability-based difficulty and $\theta = 0.995$ never removed any buckets from the curriculum over training.

6.4 Threshold θ

The results show a sensitivity to the threshold hyperparameter θ. When θ is tuned too low the curricula become too hard too soon, diminishing the performance across all test sets. This can be observed for the confidence-based curriculum with $\theta = 0.95$. By contrast, when θ is tuned too high no buckets are ever removed from the curriculum, as no bucket validation ever surpasses the threshold. This can be observed for variability-based curricula with $\theta \in [0.99, 0.995]$ for MULTINLI and $\theta = 0.995$ for SNLI. Variability-based curricula appear to be less sensitive to undertuning θ as all values that were evaluated show similar performance: ±0.6 percent points from baseline for MULTINLI and +0.2 to +1.9 percent points for SNLI.

6.5 Confidence vs variability

When comparing confidence and variability as difficulty metrics, there are distinct differences. Primarily, the confidence-based difficulty sorts the data regions in order from *easy-to-learn* to *ambiguous* to *hard-to-learn*, whereas the variability-based difficulty metric does not. Instead, data with a lower difficulty metric may come from either the *easy-to-learn* or the *hard-to-learn region*. The effects of this behaviour can be observed in the results:

One effect of the order in which the variability-based difficulty metric orders the data is that very *hard-to-learn* examples are placed in the first bucket. Very *hard-to-learn* examples are those whose

confidence and variability values are both zero or near-zero. These examples are likely to be noisy and mislabeled.[1] This has two effects that can be observed in the results. Firstly, if θ is sufficiently high this bucket will prevent any changes to the curriculum: very *hard-to-learn* examples cannot be learned by the model and thus lower the evaluation score of the bucket.

Secondly, if θ is low enough, then this first bucket containing these unlearnable examples is removed from the curriculum at some point during training. Removing the bucket containing unlearnable and potentially noisy examples is be beneficial to the overall performance as training with noisy labels has been shown to cause overfitting.[55] This may explain why the performance of variability-based models in the experiments is less sensitive to θ than the performance of confidence-based models. Confidence-based models show a sharp decrease in performance for lower values of θ. In these models, the curriculum is diminished to only buckets containing *hard-to-learn* data early on in training, leading to a relatively large influence of unlearnable or noisy examples. Variability-based models do not show this behaviour since the bucket containing these unlearable or noisy examples are removed from the curriculum early in training.

7 CONCLUSION

An empirical analysis of a novel adaptive curriculum learning approach was conducted. Results show that Subtractive Curriculum Learning can maintain or improve both in-domain and out-of-domain performance while reducing training costs. The results show that training costs can be reduced up to 56.7% while maintaining both ID performance and improving OOD performance, and up 51.7% while improving both ID and OOD performance.

While in this thesis improvements of up 1.9 percent points were achieved on OOD test datasets, further research on how certain types of data contribute to model robustness may further improve generalization of PLM's on downstream task. In particular, this thesis shows the importance of *easy-to-learn* data in finetuning smaller PLM's. Data selection methods using the 33% and 50% hardest or most ambiguous showed a strongly diminished performance for a smaller model like DISTILBERT, whereas curricula that include on *easy-to-learn* data for at least epoch were shown to maintain or improve performance while also reducing training costs.

Although both confidence- and variability-based difficulty metrics are shown to effective, variability-base difficulty metrics are shown to be less sensitive the tuning of hyperparameter θ. When using variability-based difficult metrics, using $\theta = 0.98$ is recommended due to strong OOD results.

Validation of buckets is proven to a valid approach for adapting curricula. The similarity of difficulty of examples within a bucket allowed for a randomly selected sample to be representative of the entire bucket, making the approach computationally inexpensive.

7.1 Limitations

The dependence on tunable hyperparameter θ may limit the practical usage of the proposed method. The optimal value of θ is not known a priori, and may vary over tasks, datasets and models. Further research could indicate whether the optimal θ is stable or task/dataset/model-dependent. The latter would require expensive tuning of θ, limiting the practical usefulness of the approach.

While the limitations of data selections led to the development of the presented approach, it is not certain that data selection methods cannot still be a better approach. More sophisticated data selection methods using dataset cartography could still outperform curriculum learning methods in terms of performance and training costs. The problem of optimal data selection is still open, and warrants further research.

7.2 Further research

While this thesis served as a proof-of-concept for substractive curricula using dataset cartography, there are still many open question regarding this line of research. As discussed, the optimal value of θ for different tasks, datasets or models is still unknown. Answering this question is vital to the practical viability of the approach, but lies beyond the scope of this thesis.

Alternative difficulty metrics and schedulers, including continuous schedulers, are all promising lines of research. One suggestion for future research is the exploration of alternative measures of model competence. An interesting example of this would be utilizing a teacher model to classify the premise-hypothesis relation. Examples of these types of metrics exists in the form of NLI test datasets, like GLUE DIAGNOSTICS[**?**] and FANCY[56]. These datasets provide additional information about the relation between the premise-hypothesis pair like the logical, semantical or domain-specific knowledge required to make a correct inference. If these types of metrics could be learned by a teacher model, they may provide valuable information for creating curricula.

8 ACKNOWLEDGEMENTS

I would like to thank Yupei Du for supervising this project. Yupei's advice has been invaluable. I also would like to thank the community of the Hugging Face forums for their help in answering questions about implementational matters.

REFERENCES

[1] Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. Dataset cartography: Mapping and diagnosing datasets with training dynamics. *CoRR*, abs/2009.10795, 2020.

[2] A. M. Turing. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX(236):433–460, 10 1950.

[3] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

[4] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics.

[5] Samuel R. Bowman. Eight things to know about large language models, 2023.

[6] Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, and Dawn Song. Pretrained transformers improve out-of-distribution robustness. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2744–2751, Online, July 2020. Association for Computational Linguistics.

[7] Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomas Kocisky, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. Learning and evaluating general linguistic intelligence, 2019.

[8] Tal Linzen. How can we accelerate progress towards human-like linguistic generalization? In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5210–5217, Online, July 2020. Association for Computational Linguistics.

[9] Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *CoRR*, abs/1704.05426, 2017.

[10] 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. *CoRR*, abs/1804.07461, 2018.

[11] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[12] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[14] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[15] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

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

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

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

[19] Tomás Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.

[20] Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*, 2013, 01 2013.

[21] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[22] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomás Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2016.

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

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

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

[26] Alex Wang and Kyunghyun Cho. BERT has a mouth, and it must speak: BERT as a markov random field language model. *CoRR*, abs/1902.04094, 2019.

[27] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. volume 2006, pages 535–541, 08 2006.

[28] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

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

[30] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pages 1–8, Montreal, Quebec, Canada, 2009. ACM Press.

[31] Volkan Cirik, Eduard Hovy, and Louis-Philippe Morency. Visualizing and understanding curriculum learning for long short-term memory networks, 2016.

[32] Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom M. Mitchell. Competence-based curriculum learning for neural machine translation, 2019.

[33] M. Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.

[34] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey, 2022.

[35] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander Hauptmann. Self-paced curriculum learning. 01 2015.

[36] Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks, 2019.

[37] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher-student curriculum learning, 2017.

[38] Wojciech Zaremba and Ilya Sutskever. Learning to execute, 2015.

[39] Benfeng Xu, Licheng Zhang, Zhendong Mao, Quan Wang, Hongtao Xie, and Yongdong Zhang. Curriculum learning for natural language understanding. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6095–6104, Online, July 2020. Association for Computational Linguistics.

[40] Fenia Christopoulou, Gerasimos Lampouras, and Ignacio Iacobacci. Training dynamics for curriculum learning: A study on monolingual and cross-lingual NLU. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2595–2611, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[41] Yajing Kong, Liu Liu, Jun Wang, and Dacheng Tao. Adaptive curriculum learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5067–5076, October 2021.

[42] Shiyue Zhang and Mohit Bansal. Addressing semantic drift in question generation for semi-supervised question answering, 2019.

[43] Osman Batur İnce, Tanin Zeraati, Semih Yagcioglu, Yadollah Yaghoobzadeh, Erkut Erdem, and Aykut Erdem. Harnessing dataset cartography for improved compositional generalization in transformers, 2023.

[44] Adyasha Maharana and Mohit Bansal. On curriculum learning for commonsense reasoning. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 983–992, Seattle, United States, July 2022. Association for Computational Linguistics.

[45] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.

[46] Charles Fillmore, Daniel Jurafsky, Nancy Ide, and Catherine Macleod. An american national corpus: A proposal. 04 1999.

[47] Catherine Macleod, Nancy Ide, and Ralph Grishman. The American national corpus: A standardized resource for American English. In M. Gavrilidou, G. Carayannis, S. Markantonatou, S. Piperidis, and G. Stainhauer, editors, *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00)*, Athens, Greece, May 2000. European Language Resources Association (ELRA).

[48] Nancy Ide and Catherine Macleod. The american national corpus: A standardized resource for american english. 08 2001.

[49] Nancy Ide and Keith Suderman. Integrating linguistic resources: The American national corpus model. In Nicoletta Calzolari, Khalid Choukri, Aldo Gangemi, Bente Maegaard, Joseph Mariani, Jan Odijk, and Daniel Tapias, editors, *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy, May 2006. European Language Resources Association (ELRA).

[50] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[51] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms, 2012.

[52] Nils Reimers and Iryna Gurevych. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *CoRR*, abs/1707.06799, 2017.

[53] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. *CoRR*, abs/1710.03740, 2017.

[54] Damien Teney, Yong Lin, Seong Joon Oh, and Ehsan Abbasnejad. Id and ood performance are sometimes inversely correlated on real-world datasets, 2023.

[55] Duc Tam Nguyen, Chaithanya Kumar Mummadi, Thi Phuong Nhung Ngo, Thi Hoai Phuong Nguyen, Laura Beggel, and Thomas Brox. Self: Learning to filter noisy labels with self-ensembling, 2019.

[56] Guido Rocchietti, Flavia Achena, Giuseppe Marziano, Sara Salaris, and Alessandro Lenci. *FANCY: A Diagnostic Data-Set for NLI Models*, pages 287–292. 01 2022.