universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Enhancing Language-Model Pre-training Datasets: Data Augmentation Inspired by Human Language Acquisition

verfasst von | submitted by

Alexander Tampier BSc

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 926
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Wirtschaftsinformatik
Betreut von | Supervisor:	Univ.-Prof. Dr. -Ing. Benjamin Roth B.Sc. M.Sc.

Acknowledgements

I want to express my deepest gratitude to my supervisor, Univ.-Prof. Dr.-Ing. Benjamin Roth, B.Sc. M.Sc. His insightful ideas and valuable suggestions have contributed significantly to the development and completion of this thesis. I would also like to extend my special thanks to Lukas Thoma, BA, MA, for our regular exchanges and constructive feedback that took place between us, as well as for the numerous opportunities provided to present the findings of this thesis to the Research Group. I would also like to thank you for providing me with the necessary computational resources that were needed to execute the experiments. I want to emphasize that this thesis originated as part of a research challenge and that the initial idea stemmed from the dedicated members of the Research Group Natural Language Processing, which is part of the Research Group Data Mining and Machine Learning at the University of Vienna. Their collective creativity and expertise were instrumental in shaping the foundation of this project. I would particularly like to highlight my beloved wife, to whom I express my sincere gratitude for her patience, encouragement, and understanding while I wrote this thesis. Her faith in me has been a constant source of strength. Furthermore, I would also like to thank my family, without whose support an academic path would not have been possible, and who have accompanied me through the challenges of this educational journey, always believing in me. Finally, I would like to thank all my colleagues, friends, and everyone who has supported me during the time I spent writing this thesis. Your contributions have shaped me into who I am today, no matter how big or small they were. I acknowledge the use of translation and artificial intelligence tools for grammatical improvements and paraphrasing texts to enhance clarity and readability.

Abstract

Large language models often rely on hundreds of billions of words of available high-quality training data. In contrast, human language acquisition is significantly more efficient and utilizes patterns inherent to the language. Motivated by these discrepancies, RecombiText Augmentation is introduced. RecombiText Augmentation is a novel non-neural corpus-dependent compositional data augmentation method for expanding language model training datasets in low-resource scenarios using only corpus statistics. Inspired by human language acquisition and genetic algorithms, it generates synthetic sentences through compositional recombination of the available text. The method identifies similar sentences within a corpus based on a query sentence using common words and meanings through a hybrid search process. It then applies, with the help of a semantic context window, a recombination between the two sentences to generate novel augmented sentences. Two transformer language model architectures are trained on a low-resource domain-specific dataset of 10 million words. A baseline is defined with the original training data and six mixed variants, each with a sample portion of the original training data and a generated augmented portion. The effectiveness of the proposed method is evaluated in two primary ways. First, the data quality is assessed by quantifying the predictability and diversity of the generated text. Second, the language model performance is evaluated on linguistic zero-shot and fine-tuning tasks. The experiments demonstrate improvements in morphological generalization, entity state tracking, reading, and grammatical understanding benchmarks. The available training data can be doubled or even quadrupled without compromising the performance of the language model. In some cases, this even achieves better results in benchmarks than with the original data alone. Overall, RecombiText Augmentation is capable of augmenting language model training datasets in low-resource scenarios.

Kurzfassung

Große Sprachmodelle basieren oft auf Hunderten von Milliarden Wörtern aus hochwertigen Trainingsdaten. Im Gegensatz dazu ist der Spracherwerb beim Menschen wesentlich effizienter und nutzt sprachspezifische Muster. Angesichts dieser Diskrepanzen wurde RecombiText Augmentation eingeführt. RecombiText Augmentation ist eine neuartige, nicht-neuronale, korpusabhängige Methode zur kompositorischen Datenanreicherung, mit der Trainingsdatensätze für Sprachmodelle in ressourcenarmen Szenarien ausschließlich anhand von Korpusstatistiken erweitert werden können. Inspiriert vom menschlichen Spracherwerb und genetischen Algorithmen generiert es synthetische Sätze durch kompositorische Rekombination des verfügbaren Textes. Die Methode identifiziert ähnliche Sätze innerhalb eines Korpus auf der Grundlage eines Suchsatzes unter Verwendung gemeinsamer Wörter und Bedeutungen durch einen hybriden Suchprozess. Anschließend wendet sie mit Hilfe eines semantischen Kontextfensters eine Rekombination zwischen den beiden Sätzen an, um neuartige augmentierte Sätze zu generieren. Zwei Transformer-Sprachmodellarchitekturen werden auf einem ressourcenarmen domänenspezifischen Datensatz von 10 Millionen Wörtern trainiert. Eine Basislinie wird mit den ursprünglichen Trainingsdaten und sechs gemischten Varianten definiert, die jeweils einen Teil der ursprünglichen Trainingsdaten und einen generierten erweiterten Teil enthalten. Die Wirksamkeit der vorgeschlagenen Methode wird auf zwei Arten bewertet. Erstens wird die Datenqualität durch Quantifizierung der Vorhersagbarkeit und Vielfalt des generierten Textes bewertet. Zweitens wird die Leistung des Sprachmodells anhand von linguistischen Zero-Shot- und Fine-Tuning-Aufgaben bewertet. Die Experimente zeigen Verbesserungen bei den Benchmarks für morphologische Generalisierung, im Tracking von Entitätszuständen, Lesen und grammatikalischem Verständnis. Die verfügbaren Trainingsdaten können verdoppelt oder sogar vervierfacht werden, ohne die Leistung des Sprachmodells zu beeinträchtigen. In einigen Fällen werden damit sogar bessere Ergebnisse in Benchmarks erzielt als mit den Originaldaten allein. Insgesamt ist RecombiText Augmentation in der Lage, Trainingsdatensätze für Sprachmodelle in Szenarien mit geringen Ressourcen zu erweitern.

Contents

List of Tables

List of Figures

1. Introduction

Enormous efforts have been made in recent years to optimize the training of large language models (LLMs) on an ever-increasing scale (Warstadt et al., 2023). However, success is based on extensive available high-quality training data, often hundreds of billions of words. This usually results in high computational costs, limiting the applicability to low-resource languages or domains (Warstadt & Bowman, 2022). The first successful generative LLMs were trained with up to 570 GB of text after filtering (Brown et al., 2020). In contrast, human language acquisition is far more efficient. For Example, children can fully learn a language by the time they reach puberty, even though they only hear 3 to 11 million words per year (Warstadt & Bowman, 2022; Warstadt et al., 2025). They utilize studied pattern recognition mechanisms within the language (Tomasello, 2010; A. E. Goldberg, 2006; Gentner, 1983; Gleitman, 1990; Weir, 1962), which drives the intuition to use those mechanisms for efficient language modeling as well. Figure 1.1 shows the discrepancies between humans and machines when learning a language.

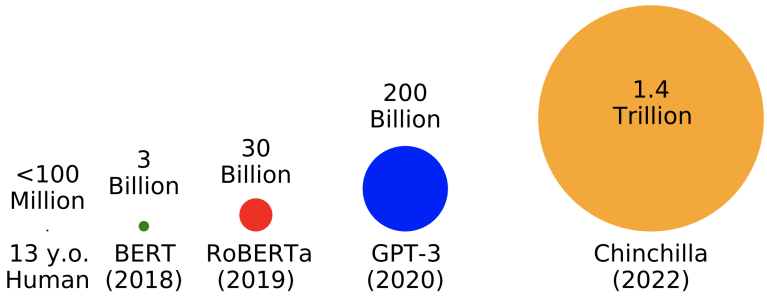

Figure 1.1.: Large Language models are trained with an extensive amount of words than what a human child is confronted with to effectively learn a language, image by Warstadt et al. (2025).

This discrepancy highlights the ineffectiveness of current LLM pre-training and emphasizes the need for data-efficient pre-training in low-resource scenarios. Data augmentation (DA) offers a promising solution for the efficient use of the available data by generating synthetic examples from existing datasets (Jurafsky & Martin, 2019). While DA is well-explored in computer vision and downstream NLP tasks (Feng et al., 2021), its application to LLM pre-training in low-resource scenarios without the use of models trained on additional text remains underexplored (Warstadt et al., 2023; Hu et al., 2024). Recent research initiatives (Warstadt et al., 2025; Choshen et al., 2024; L. Charpentier et al., 2025) show that DA methods can potentially improve language model performance (Warstadt et al., 2025; Hu et al., 2024). However, many rely on resources that were trained

with additional text in addition to the available data, thus violating the limitations of truly low-resource scenarios (Lyman & Hepner, 2024; Haga, Fukatsu, Oba, Bisazza, & Oseki, 2024; Edman, Bylinina, Ghorbanpour, & Fraser, 2024; Z. Zhang, Yang, Ma, Rügamer, & Nie, 2023; Theodoropoulos, Filandrianos, Lyberatos, Lymperaiou, & Stamou, 2024). To address this gap, this thesis proposes a novel statistical NLP method for compositional DA, inspired by human language acquisition, to efficiently utilize available data in low-resource scenarios for training language models. The following gives a structural overview of the thesis.

- Chapter 1 introduces the motivation behind this work and explains why this problem is relevant.
- Chapter 2 provides the theoretical background for the work.
- Chapter 3 reviews existing low-resource DA methods in NLP and highlights work that, at first glance, appears similar to the present work but differs fundamentally in the use of dependencies.
- Chapter 4 outlines the intuition behind the RecombiText Augmentation (RTA) method as well as its algorithmic formulation.
- Chapter 5 describes the experimental setup, which is roughly divided into four steps to generate the data, measure data quality, train the language models, and evaluate the language models' performance that were trained with the presented method.
- Chapter 6 presents the results and discussion of the zero-shot and fine-tuning evaluation tasks for the different language model architectures and the training data variants, as well as the key findings from the experiments, the limitations of the RTA algorithm, and an outlook on potential future work.
- Chapter 7 briefly summarizes the entire thesis.

2. Theoretical Background

This chapter lays the theoretical foundation for Data Augmentation (DA) methods that promote efficient data use in low-resource language modeling. First, the basics of modern large language models (LLMs) are introduced. Section 2.1 presents the transformer architecture and further focuses on two different types of training strategies for language modeling, as well as evaluation metrics that are used in this thesis for assessing the language model performance. Section 2.2 then explains the concept of semantic clustering. Section 2.3 outlines what an information retrieval system is and describes the underlying search algorithms. Section 2.4 provides the necessary theoretical background for the term "low-resource". Finally, the summary gives a brief overview of the chapter.

2.1. Large Language Models

This section introduces large language models (LLMs) and their underlying transformer architecture from Vaswani et al. (2017). First, the most essential transformer components are described. Next, transformer-based language models are discussed, with a focus on pre-training and fine-tuning strategies. Furthermore, the two architectures that are important for this study are described.

2.1.1. Transformer

The transformer architecture is the foundational structure used to build modern LLMs. It is a type of neural network characterized by its unique framework, featuring a mechanism known as self-attention or multi-head attention (Jurafsky & Martin, 2019; Vaswani et al., 2017). Figure 2.1 shows a transformer architecture from Vaswani et al. (2017).

Attention Attention is characterized as the process of forming contextual representations for a token, which is a unit of text, by focusing on and incorporating information from adjacent tokens. This helps the model understand the relationships between tokens across extensive sequences (Jurafsky & Martin, 2019). Attention is described with two concepts, namely scaled dot-product attention and multi-head Attention (Vaswani et al., 2017). Equation 2.1 defines scaled dot-product attention.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

where Q, K represent matrices for the set of queries and keys, and V the values. d_k denotes the dimension of both the input queries and keys, and $\frac{1}{\sqrt{d_k}}$ is the scaling factor

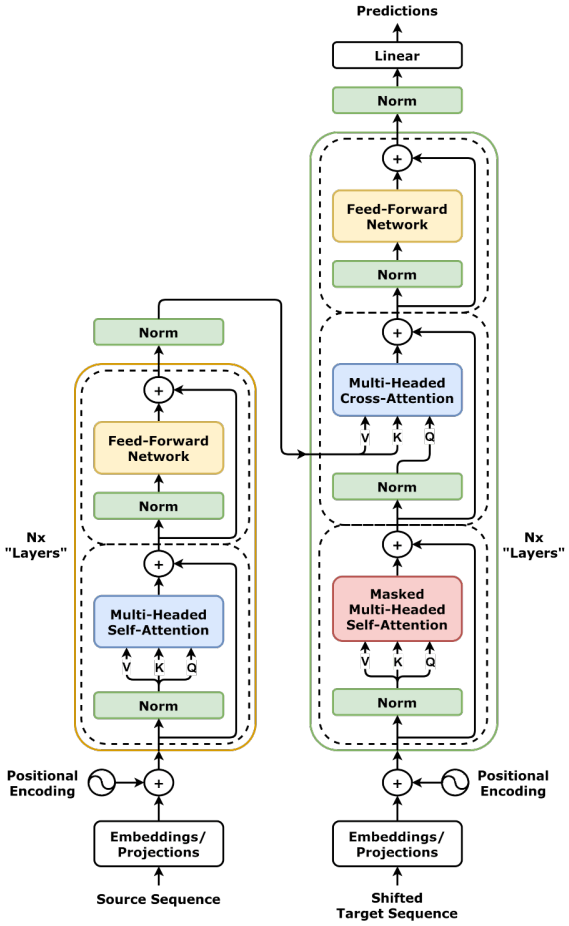

Figure 2.1.: Transformer architecture from Vaswani et al. (2017) and image by Godoy (2021).

Vaswani et al. (2017). Figure 2.2 shows a scaled dot-product attention.

The transformer architecture uses multiple "heads" denoted as $head_i$, where i refers to the i^{th} attention head within the multi-head attention mechanism to perform the attention function in parallel. Equation 2.2 shows the calculation for the multi-head attention (Vaswani et al., 2017).

$$\begin{aligned} \text{MultiHead}(Q,K,V) &= \text{Concat}\left(\text{head}_1,\ldots,\text{head}_\text{h}\right)W^O \\ \text{where}\ \ \text{head}_i &= \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right) \end{aligned} \tag{2.2}$$

where the weight matrices W_i^Q, W_i^K, W_i^V are learnable parameter matrices $W_i^Q \in \mathbb{R}^{d_{model}\times d_k}$, $W_i^K \in \mathbb{R}^{d_{model}\times d_k}$, $W_i^V \in \mathbb{R}^{d_{model}\times d_v}$ to project Q, K, V into lower-dimensional subspaces for each head (A. Zhang, Lipton, Li, & Smola, 2023; Vaswani et al., 2017). Figure 2.3 shows a Multi-head attention.

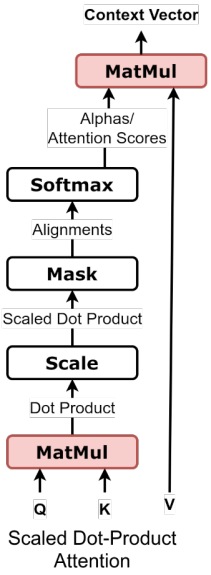

Figure 2.2.: Scaled dot-product attention image by Godoy (2021).

Position-Wise Feed-Forward Network Every layer present in both the encoder and decoder includes a fully connected feed-forward network. The model comprises two linear transformations, with a $\mathrm{ReLU}(x) = \max(0, x)$ activation function applied between them (Vaswani et al., 2017). Equation 2.3 defines the fully connected feed-forward network.

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2.3}$$

where x is the input and W the weights and b the bias term (Vaswani et al., 2017).

Embeddings and Softmax The transformer uses learned embeddings of dimension d_{model} for the source and target tokens. The learned embeddings are then used to convert both the input and output tokens into d_{model}-dimensional vectors. A learned linear transformation and a softmax function then generate predicted next-token probabilities (Vaswani et al., 2017).

Positional Encoding Positional encodings serve to enable the model to maintain the order of the sequences. They store information about the relative or absolute position of the tokens within a sequence (Vaswani et al., 2017). Equation 2.4 defines the positional encoding.

$$\begin{aligned} PE_{(pos,2i)} &= \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \\ PE_{(pos,2i+1)} &= \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \end{aligned} \tag{2.4}$$

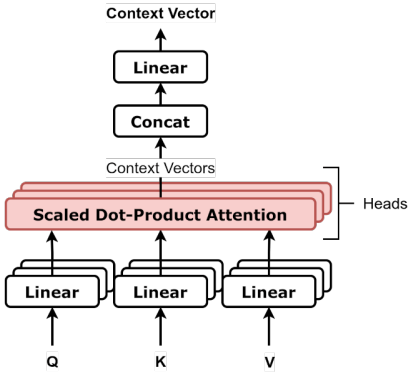

Figure 2.3.: Multi-head attention image by Godoy (2021).

where *pos* is the position of a token and i is the dimension, and d_{model} is the dimension for embedding vectors (Vaswani et al., 2017).

Transformer Blocks Figure 2.1 shows that the original transformer model includes both an encoder (left side) and a decoder (right side) block. Input sequence embeddings for both the source and target are merged with positional encodings prior to being processed by the encoder and decoder. These components are made of stacked modules that employ self-attention mechanisms. The encoder block on the left is composed of multiple identical layers, each comprising two sub-layers: a multi-head self-attention mechanism and a position-wise feed-forward network. In contrast, the decoder block on the right adds a third sublayer, namely the multi-headed cross-attention, where queries are from the decoder self-attention sublayer and the keys and values are from the transformer encoder outputs. Furthermore, the self-attention mechanism differs significantly from the encoder block. Each position within the decoder is allowed to concentrate exclusively on all prior positions. This use of masked attention ensures the autoregressive nature. This ensures that each prediction is based solely on the previously generated output tokens (Vaswani et al., 2017; A. Zhang et al., 2023).

2.1.2. Transformer-based language modeling

When training a language model using the transformer architecture from Vaswani et al. (2017), it is essential to distinguish between pre-training and fine-tuning. Pre-training involves understanding linguistic and real-world information from vast amounts of text data, leading to the development of pre-trained language models, commonly referred to as LLMs. However, LLMs can not learn everything about the environment just by pre-training on text from many domains. It is common to desire the application of the model to a different domain or task. For example, a language model can be adapted specifically for legal or medical documents, or adjusted to suit a particular task. As a result, fine-tuning is employed to train the model further using relevant data from the new

domain or language. In summary, fine-tuning involves continuing to train a pre-trained model on new knowledge without losing the original knowledge on which the model was trained. (Jurafsky & Martin, 2019).

Pre-training on extensive datasets has become popular for developing models with more substantial generalization that can handle various tasks, regardless of the adaptation. Language modeling with transformers is mainly done with two different types of architecture: (i) encoder-only or (ii) decoder-only, which will be briefly explained below. The third architecture (iii), encoder-decoder, is omitted from here, as the thesis focuses solely on two different types of language model architectures, specifically encoder-based and decoder-based. Figure 2.4 highlights the encoder-only and decoder-only pre-training objectives, which will be explained below.

(a) Decoder-only (b) Encoder-only

Figure 2.4.: Language model pre-training objectives adapted from Stanford CS224N WS2025 Slides.

Decoder-Only Subfigure 2.4a illustrates the autoregressive training strategy for text generation using transformer-based language models. Each generated token is appended as a prefix for predicting the next (Jurafsky & Martin, 2019).

Decoder-only architectures have become the standard for large-scale language models. The entire encoder and cross-attention in the decoder are removed from the original encoder-decoder architecture (see Figure 2.1). This leaves a stack of decoder layers with causal self-attention that ensures each token attends only to preceding tokens to prevent access to future tokens during prediction (A. Zhang et al., 2023).

This architecture enables autoregressive training, also known as causal language modeling (CLM), where the model is pre-trained on massive unlabeled text corpora to minimize cross-entropy loss for next-token prediction. Within the target sequence, the input is shifted by one token at a time. The model can access all previous tokens, but not future ones. This is to ensure that the model learns contextual relationships. (A. Zhang et al., 2023; Jurafsky & Martin, 2019). Specifically, the model minimizes the cross-entropy loss L_{CE} to predict the next word in a sequence. Equation 2.5 shows the cross-entropy loss for CLM.

$$L_{CE} = -\sum_{w \in V} \mathbf{y}_t[w] \log \hat{\mathbf{y}}_t[w] \tag{2.5}$$

where w is a word from the vocabulary V and $\mathbf{y}_t$ is the true probability distribution at time step t and $\hat{\mathbf{y}}_t$ is the predicted probability distribution at time step t. The loss then simplifies to $L_{CE}\left(\hat{\mathbf{y}}_t, \mathbf{y}_t\right) = -\log \hat{\mathbf{y}}_t\left[w_{t+1}\right]$ (Jurafsky & Martin, 2019).

An example of such a language model is GPT-2 (Radford et al., 2019) that achieved excellent results in numerous benchmarks without requiring further adjustments or updates to its parameters or architecture. This reinforces the approach that a pure decoder model with CLM can be used for generative capabilities (A. Zhang et al., 2023; Jurafsky & Martin, 2019), which is used later on for the experiments (see Chapter 5).

Encoder-Only Subfigure 2.4b illustrates the bidirectional training strategy for transformer-based language models. Each token is randomly masked and predicted based on bidirectional context (Jurafsky & Martin, 2019).

Architectures that exclusively use encoders utilize bidirectional self-attention to process input sequences. This allows each input token to be considered simultaneously with all other tokens within a sequence. This, in turn, leads to richer contextual representations. Architectures pre-trained with this strategy are particularly well-suited for tasks such as classification or question answering (A. Zhang et al., 2023).

This architecture enables masked language modeling (MLM), in which the model randomly masks a portion of the tokens (usually 15% (Devlin, Chang, Lee, & Toutanova, 2019)). The transformer encoder is trained to predict these masked tokens using bidirectional context (Jurafsky & Martin, 2019). In particular, the model minimizes the cross-entropy loss L_{MLM} over the masked tokens M. Equation 2.5 shows the cross-entropy loss with the MLM training objective.

$$L_{MLM} = -\frac{1}{|M|} \sum_{i \in M} \log P(x_i | h_i^L) \tag{2.6}$$

where x_i is the original masked token, and h_i^L is the final layer output vector for position i, with P computed via softmax over the vocabulary (Jurafsky & Martin, 2019).

One example of this is the transformer architecture $BERT_{base}$ (Devlin et al., 2019), which comprises a total of 12 layers and 768 dimensions, as well as around 110 million parameters. The model was pre-trained using books and Wikipedia texts. Advanced variants such as RoBERTa (Liu et al., 2019) further optimize the architecture of BERT (Devlin et al., 2019). This has led to improved performance in specific language model benchmarks. Encoder-only models pre-trained with MLM are particularly known for their contextual representation capabilities and efficiency in downstream tasks (A. Zhang et al., 2023; Jurafsky & Martin, 2019). This architecture form will be used later for the experiments (see chapter 5).

2.1.3. Evaluation Metrics

Classification metrics quantify model performance by attempting to predict the actual truth (prediction labels). These predicted truths are then compared with the so-called gold labels, which represent the exact values. A confusion matrix is used for this purpose, which

is a table that visualizes the predictions of the system or models in two dimensions based on the actual gold labels. One dimension represents the predictions of the system/model, while the other represents the gold labels. Each cell shows a range of possible results (Jurafsky & Martin, 2019). The individual components of such a confusion matrix are briefly presented below.

- True Positive (TP): indicates the count of correctly predicted positive cases by the model
- True Negative (TN): indicates the count of correctly predicted negative cases by the model
- False Positive (FP): indicates the count of false positive predictions by the model
- False Negative (FN): indicates the count of false negative predictions by the model

The definitions of the confusion matrix components are adapted from Reddy and Aggarwal (2015); Savarimuthu, Subramani, and Raj (2024); Jurafsky and Martin (2019). Table 2.1 illustrates such a confusion matrix.

		gold labels	
		Positive	Negative
pred. labels	Positive	TP	FP
	Negative	FN	TN

Table 2.1.: Confusion Matrix with comparison of gold labels and prediction labels (Jurafsky & Martin, 2019).

The confusion matrix helps measure different metrics to evaluate the system's predictions.

Precision The precision helps measure the percentage of elements that the system marked as positive and are actually positive (Jurafsky & Martin, 2019). Equation 2.7 defines the precision.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.7}$$

Recall Recall is commonly known as the true positive rate (TPR). It quantifies the proportion of elements that are positive in the gold labels and have been correctly identified by the system as positive (Jurafsky & Martin, 2019). Equation 2.8 defines the recall.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.8}$$

Accuracy The accuracy indicates the percentage of all observations that the system has correctly identified (Jurafsky & Martin, 2019). Equation 2.9 defines the accuracy.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{2.9}$$

F1 The F-Measure is the most straightforward way to combine precision and recall. It is derived from a weighted harmonic mean. A collection of numbers corresponds to the reciprocal of the arithmetic mean of their reciprocals. (Jurafsky & Martin, 2019). Equation 2.10 defines the harmonic mean.

$$\text{HarmonicMean} = \frac{N}{\sum_{i=1}^{N} \frac{1}{x_i}} \tag{2.10}$$

The F1 measure is a specific F-measure metric. Equation 2.11 defines the F1 measure.

$$F1 = \frac{2 \times \text{ Precision } \times \text{ Recall}}{\text{Precision } + \text{ Recall}} \tag{2.11}$$

Pearson Correlation The Pearson correlation coefficient (PCC) r_{xy} quantifies the linear relationship between the two variables x and y (De Winter, Gosling, & Potter, 2016). It ranges from -1 to 1, indicating the respective correlation (negative/positive), and 0 indicates no linear correlation. This metric is used later on to measure the correlation between human and model language acquisition with a p-value for statistical significance. Equation 2.12 shows the calculation for PCC.

$$r_{xy} = \frac{\sum_{i=1}^{N} x_i y_i}{\sqrt{\sum_{i=1}^{N} x_i^2 \sum_{i=1}^{N} y_i^2}} \tag{2.12}$$

Spearman Correlation The Spearman correlation coefficient r_s, also often denoted as ρ in literature, measures the monotonic relationship between two variables. It is computed like Pearson but rank-transformed to values between 1 and N, followed by mean centering via subtracting $(N+1)/2$ (De Winter et al., 2016). It ranges from -1 to 1, indicating the respective correlation (negative/positive), and 0 indicates no correlation. Equation 2.13 shows the calculation for Spearman ρ.

$$r_s = \rho = \frac{\sum_{i=1}^{N} x_{i,r} y_{i,r}}{\sqrt{\sum_{i=1}^{N} x_{i,r}^2 \sum_{i=1}^{N} y_{i,r}^2}} = 1 - \frac{6 \sum_{i=1}^{N} (x_{i,r} - y_{i,r})^2}{N\,(N^2 - 1)} \tag{2.13}$$

Coefficient of determination The coefficient of determination R^2, also known as the multiple correlation coefficient, is frequently used in classical regression analysis. The coefficient quantifies the proportion of variance explained by the regression model. A generalization is shown in Equation 2.14.

$$R^2 = 1 - \{L(0)/L(\hat{\beta})\}^{2/n}, \tag{2.14}$$

where $L(\hat{\beta})$ and $L(0)$ are likelihoods of the fitted and null models and n is the sample size (Nagelkerke et al., 1991).

Perplexity Mainly employed as a measure to assess the capability of language models in forecasting the subsequent tokens in a sequence. This determines how well the models predict unseen text and whether they are "surprised" by the text at hand. Lower perplexity values indicate better model prediction. Equation 2.15 shows the calculation for perplexity.

$$\text{Perplexity}_\theta\left(w_{1:n}\right) = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{P_\theta\left(w_i \mid w_{<i}\right)}} \tag{2.15}$$

where θ represents the parameterized model and $w_{1:n}$ a sequence of n tokens such as words or subwords $w_1, w_2, \ldots, w_n$ and $P_\theta(w_i \mid w_{<i})$ the conditional probability that the parameterized model assigns to the i-th token w_i, given all previous tokens (Jurafsky & Martin, 2019).

Self-Bilingual Evaluation Understudy To explain Self-Bilingual Evaluation Understudy (Self-BLEU), one must first explain Bilingual Evaluation Understudy (BLEU). The Bilingual Evaluation Understudy (BLEU) score (Papineni, Roukos, Ward, & Zhu, 2002) was initially used for evaluating machine translations. Over time, however, this evaluation metric has also been used for assessing similarities between a reference and a candidate sentence from text generations. In this process, a candidate sentence is compared with a reference sentence for lexical similarity within a sequence, known as n-grams. The metric varies from 0 to 1, where 1 indicates a perfect match between the candidate sentence and the reference sentence. A so-called brevity penalty takes into account the different lengths of the sentences (Papineni et al., 2002). Equation 2.16 shows the brevity penalty (BP).

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}. \tag{2.16}$$

where c is the candidate translation length, r is the effective reference corpus length. Equation 2.16 shows the BLEU (Papineni et al., 2002) score.

$$\log \text{BLEU} = \min\left(1 - \frac{r}{c}, 0\right) + \sum_{n=1}^{N} w_n \log p_n \tag{2.17}$$

where p_n is the modified n-gram precision and w_n are the positive weights summing to one. An extension of the traditional BLEU (Papineni et al., 2002) is Self-BLEU (Zhu et al., 2018). Each sample text is compared with the other samples, and the overall diversity of the samples is measured. A lower Self-BLEU score means the outputs are more diverse.

2.2. Semantic Clustering

The following presents the vector models employed in this work for word and sentence embeddings, as well as the method that is used in combination with these models to create semantic clusters. Furthermore, it is described how these models were evaluated to be relevant.

2.2.1. Word Embeddings: Global Vectors for Word Representation

Global Vectors for Word Representation (GloVe) (Pennington, Socher, & Manning, 2014), uses statistics of the respective words that occur in a corpus. Whereas the Efficient Estimation of Word Representations in Vector Space (word2vec) (Mikolov, Chen, Corrado, & Dean, 2013) uses neural networks to learn the semantics of the words. This thesis uses GloVe as the representation for word embeddings. Consequently, the core idea behind GloVe is shortly outlined, while an explanation of word2vec (Mikolov et al., 2013) is omitted. GloVe is a model for semantic word representation in a multidimensional vector space that captures global corpus statistics. The model uses a word-word cooccurrence matrix to derive semantically rich word representations as d-dimensional vectors (A. Zhang et al., 2023; Pennington et al., 2014). Those d-dimensional vectors are often called word embeddings in literature. Equation 2.18 shows the definition of the main idea based on a weighted least squares regression model.

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right) \left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2 \tag{2.18}$$

where X is the matrix of word-word co-occurrence counts, w the respective word in a vocabulary V from the text and b a bias term and $f\left(X_{ij}\right)$ a weighting function.

2.2.2. Sentence Embeddings: Smooth inverse frequency and unsupervised smoothed inverse frequency

Arora, Liang, and Ma (2017) introduce Smooth Inverse Frequency (SIF). A method for sentence embeddings that outperforms other approaches. A weighted average of pre-trained word embeddings is used for the calculation and refined with principal component analysis (PCA) to discover a low-dimensional subspace that accounts for the majority of the data's variability (Deisenroth, Faisal, & Ong, 2020) and Singular value decomposition (SVD), which is a decomposition method in linear algebra for a specific matrix (Deisenroth et al., 2020). In certain environments, such simple methods outperform approaches based on neural networks when only limited data is available. (Arora et al., 2017). This makes the proposed method an effective method for creating sentence embeddings from statistical calculations.

The main idea behind SIF is that the weighted average of the word vectors in the sentence is calculated. Then the projections of the average vectors on their first singular vector are removed ("common component removal"). The weight of a word w is $a/(a + p(w))$,

where a is a parameter and $p(w)$ represents the estimated word frequency in the corpus (Arora et al., 2017).

The paper by Ethayarajh (2018a) represents an extension and improvement of SIF, incorporating a random walk model to enhance the initial method. The method is referred to as unsupervised smoothed inverse frequency (uSIF) and outperforms that of Arora et al. (2017) by up to 44.4% in text similarity tasks (Ethayarajh, 2018a).

2.2.3. k-nearest neighbors

k-nearest neighbors (k-NN) (Cover & Hart, 1967) is a classification method that assigns an unclassified sample point to one of the k nearest previously classified points, also known as neighbors, in a metric space. This rule functions independently of the data's underlying probability distribution and solely depends on a chosen distance metric to assess proximity. A distance metric $d(x, y)$ is determined as described in Cunningham and Delany (2020) if it meets the following criteria:

1. $d(x, y) \geq 0$; non-negativity
2. $d(x, y) = 0$ if and only if $x = y$; identity
3. $d(x, y) = d(y, x)$; symmetry
4. $d(x, z) \leq d(x, y) + d(y, z)$; triangle inequality

In the context of semantic clustering, k-NN can be used to group embedding vectors by identifying clusters based on the density of nearest neighbors in high-dimensional spaces. This ensures capturing semantic relationships.

2.2.4. Embedding Evaluation

This section briefly describes the various evaluation methods for vector models. Preliminary studies have already compared the relevance of different embedding models using the evaluation methods presented below. The Semantic Textual Similarity Benchmark (STS-B) from GLUE (Wang et al., 2018) is often used to measure the prediction of models of semantic similarity between sentences. The STS-B dataset consists of sentence pairs from many different sources. Humans evaluate each sentence pair and set a similarity rating between 1 and 5. The goal is for the model to predict these annotated ratings as accurately as possible (Jurafsky & Martin, 2019; Iwamoto, Yoshida, Kanayama, Ohko, & Muraoka, 2023). The prediction of semantic text similarity is evaluated using Pearson and Spearman correlations (Wang et al., 2018). Higher Pearson and Spearman correlations indicate that the similarity values predicted by the model correspond to the human ratings (Jurafsky & Martin, 2019). WordSim-353 (Finkelstein et al., 2002) and SimLex-999 (Hill, Reichart, & Korhonen, 2015) are often used to measure the prediction of semantic similarities between words by models. The principle works in the same way as in STS-B (Wang et al., 2018), but at the word pair level (Jurafsky & Martin, 2019).

2.3. Information Retrieval

Information retrieval (IR) is a process in which media are searched for based on the information needs of users or systems. Such a system is often referred to as a search engine. When a specific query is sent to the query system and an ordered list of documents from a collection is returned, the system is also referred to as ad hoc IR. A document is any unit of text that is indexed and retrieved. A collection refers to a set of documents that are used to satisfy the queries. A term refers to a word in a collection, but can also include phrases. A query represents the information needs of a user or system, expressed as a set of terms that describe the user's or system's requirements. The basic IR architecture employs a vector space model. Queries and documents can be represented as vectors using unique word counts. Potential documents are then evaluated, for example, based on the cosine similarity between these vectors (Jurafsky & Martin, 2019). Figure 2.5 shows such an ad hoc IR system.

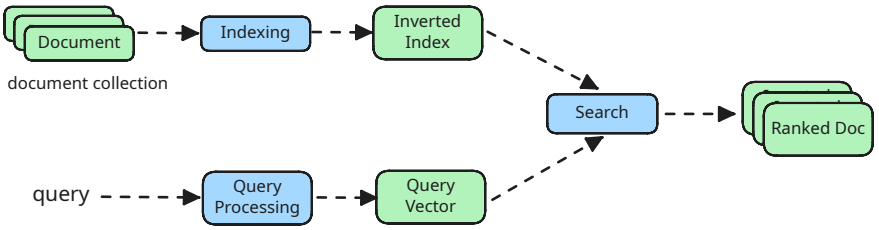

Figure 2.5.: Ad-hoc information retrieval system, image adapted from Jurafsky and Martin (2019)

An Example of such an ad hoc IR system is Elasticsearch[1]. In an ad hoc IR system, different methods are often used for term weighting to evaluate the relevance of a document corresponding to a search query. To determine lexical relevance for a specific search query, the two most relevant variants are TF-IDF (Jurafsky & Martin, 2019) and BM25 (Robertson & Zaragoza, 2009) weighting, where BM25 is a probabilistic enhancement of TF-IDF for ranking documents. For semantic relevance, a key approach is semantic search using k-NN within a high-dimensional vector space, leveraging cosine similarity as the distance metric to identify documents with embeddings most similar to the query's embedding (see Subsection 2.2.3). Hybrid search is an effective method that combines both types of relevance (lexical and semantic relevance). The results of the respective search processes are then combined. The methods for lexical search and hybrid search are explained below.

2.3.1. Term frequency–inverse document frequency

The following refers to the explanation from Jurafsky and Martin (2019). TF-IDF helps to distinguish the importance of documents d within an IR and covers two concepts: term

[1] https://www.elastic.co/elasticsearch

frequency (TF) and inverse document frequency (IDF). Term frequency indicates how often certain words appear in a document. Equation 2.19 shows how term frequency is calculated.

$$\mathrm{tf}_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{count}(t,d) & \text{if } \mathrm{count}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.19}$$

where t represents a specific term and d a specific document. Rare terms are better at distinguishing between documents d, while common terms are less helpful. Equation 2.20 shows how the inverse document frequency is calculated.

$$\mathrm{idf}_t = \log_{10} \frac{N}{\mathrm{df}_t} \tag{2.20}$$

where N represents the total number of documents d within the collection (often denoted as D) and df_t is the number of documents (e.g., sentences, phrases, any piece of text) in which the term t occurs. It is essential to know that rare terms have high IDF weights, while frequent terms have low or no weights.

The $tf - idf$ value for the word t in document d is therefore the product of the term frequency $tf_{t,d}$ and the IDF. Equation 2.21 shows the TF-IDF calculation.

$$\text{tf-idf}(t,d) = \mathrm{tf}_{t,d} \times \mathrm{idf}_t \tag{2.21}$$

To obtain relevant search results for a query, document scoring is used. This involves calculating a document's cosine similarity to another document using a query vector q and a document vector d. Therefore, we need first to define how cosine similarity between two vectors is calculated. Equation 2.22 shows the cosine similarity between two vectors.

$$\cos(\mathbf{v},\mathbf{w}) = \frac{\mathbf{v}\cdot\mathbf{w}}{|\mathbf{v}||\mathbf{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}} \tag{2.22}$$

which measures the angle between two vectors and therefore the similarity. To assess how similarly the two documents align with a query vector q and a document vector d the idea can be expanded by using the tf-idf vectors and calculating $\cos(q,d)$, which results in Equation 2.23.

$$\text{score}(q,d) = \sum_{t\in\mathbf{q}} \frac{\text{tf-idf}(t,q)}{\sqrt{\sum_{q_i\in q} \text{tf-idf}^2(q_i,q)}} \cdot \frac{\text{tf-idf}(t,d)}{\sqrt{\sum_{d_i\in d} \text{tf-idf}^2(d_i,d)}} \tag{2.23}$$

where t is the term in the query vector q and document vector d.

2.3.2. Best Matching 25

The following refers to the explanation from Jurafsky and Martin (2019). Best Matching 25 (BM25) is an extended TF-IDF variant with two additional parameters. k controls the ratio of term frequency to IDF, and b regulates the normalization of document length.

The BM25 score measures the relevance of a document d for a query q. Equation 2.24 shows how documents are scored using BM25.

$$\sum_{t \in q} \underbrace{\log\left(\frac{N}{df_t}\right)}_{\text{IDF}} \underbrace{\frac{tf_{t,d}}{k\left(1-b+b\left(\frac{|d|}{|d_{\text{avg}}|}\right)\right)+tf_{t,d}}}_{\text{weighted tf}} \tag{2.24}$$

where $|d_{avg}|$ is the average document length (Robertson & Zaragoza, 2009; Jurafsky & Martin, 2019).

2.3.3. Reciprocal Rank Fusion

Reciprocal Rank Fusion (RRF) (Cormack, Clarke, & Buettcher, 2009) is a method for combining different document rankings from multiple IR systems. For each document d, a score is calculated as the sum of the input rankings, where k_{RRF} is a constant (default 60) to reduce the influence of outliers. Equation 2.25 presents the RRF score for document $d \in D$, where D is the set of corpus documents.

$$\text{RRFscore}(d \in D) = \sum_{r \in R} \frac{1}{k_{RRF} + r(d)} \tag{2.25}$$

2.4. Low-resource Scenarios

A clear definition for further focus of the present study is provided by defining "low-resource" scenarios in Natural Language Processing (NLP). The term is defined to address the challenges of limited data availability, which is central for developing effective augmentation methods for underrepresented languages and domains. Hedderich, Lange, Adel, Strötgen, and Klakow (2020) outlines that a low-resource scenario in NLP is characterized by limited data availability and quality. Low-resource languages, such as African languages, are underrepresented in comparison to English. Low-resource languages are essentially divided into three dimensions: (i) availability of task-specific labels, (ii) availability of unlabeled language- or domain-specific text, (iii) availability of auxiliary data. For example, in Part-of-speech (POS) tagging, which annotates the grammar of a text, 1–2,000 annotated tokens may qualify as "low-resource", whereas other studies set thresholds of 10,000, 40,000–60,000, or even 350,000 for more complex tasks. Generally, there is no fixed limit for defining resources as "too low". What is considered "low-resource" depends heavily on the respective NLP task, the language, and the amount of data required. With minimal amounts of data, classic non-neural methods often perform better than modern neural methods (Hedderich et al., 2020; Warstadt & Bowman, 2022; Hu et al., 2024).

2.5. Summary

This chapter provided the theoretical basis for the following chapters. It explained the current transformer architecture from Vaswani et al. (2017) and described the two most important training objectives between causal language modeling and masked language modeling. In addition, the difference between pre-training and fine-tuning was explained. To evaluate the performance of language models and text generation systems, the necessary key metrics for evaluation were presented. Furthermore, the essential theoretical background of semantic clustering and information retrieval systems was explained. Semantic clustering deals with word and sentence embeddings to find semantic similarities within a text. Information retrieval systems include search and ranking strategies based on the information needs of users or systems and are also referred to as ad hoc information retrieval systems. Finally, the term "low-resource" in NLP, which is essential for this work, was explained based on previous surveys. The next chapter 3 provides an overview of methods for data augmentation with low resources in NLP, with a focus on identifying strategies that minimize external text dependencies.

3. Low-Resource Data Augmentation Methods in Natural Language Processing

This chapter surveys data augmentation (DA) methods that enable efficient use of available data in low-resource Natural Language Processing (NLP). Based on previous surveys (Feng et al., 2021; Şahin, 2022; Hedderich et al., 2020; Chen, Tam, Raffel, Bansal, & Yang, 2023), this study focuses on examining the different DA methods in terms of their dependencies. Section 3.1 builds a taxonomy based on the dependencies of DA methods and categorizes them into (i) non-neural, (ii) feature-space, (iii) generative-based, and (iv) external resource-based methods. This distinction highlights which methods are potentially viable in truly low-resource scenarios with minimal or no dependencies on additional text. Section 3.2 transitions to more recent methods like human language acquisition-inspired DA methods from recent challenges (Warstadt et al., 2025; Choshen et al., 2024; L. Charpentier et al., 2025). Finally, the summary provides a brief overview of the chapter. The focus is consistently on identifying methods that minimize additional text dependencies.

3.1. Taxonomy of data augmentation methods

Building on the low-resource definition (see Chapter 2), this section categorizes DA methods by their dependencies, revealing which are most adaptable to constrained environments. The taxonomy divides DA methods into four broad categories. The categories are distinguished based on their dependencies: non-neural (deterministic transformations with no external dependencies), Feature-Space (manipulating internal representations), Generative (relying on pre-trained language models), and External Resource-based (such as treebanks[1]). Figure 3.1 presents the framework of the proposed taxonomy and prioritizes low-dependency methods (non-neural and feature-space) as ideal for low-resource NLP.

Non-neural Non-neural methods are defined as those that rely on deterministic approaches, ensuring the outcome is predictable and reproducible. Such methods include statistics or probabilistic models for text transformations that do not utilize pre-trained language models or external knowledge. Easy Data Augmentation (EDA) (Wei & Zou, 2019) presents four simple methods for data augmentation. Two of these are classified

[1]Framework for annotations of grammar such as the Universal Dependencies

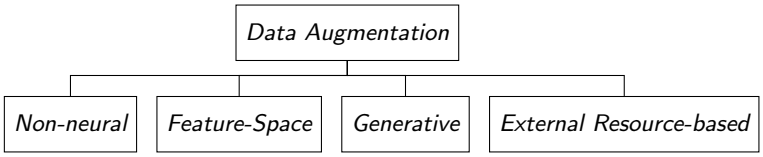

Figure 3.1.: Taxonomy of Data Augmentation Methods

as non-neural operations for DA, which will be briefly outlined. Random Swap (RS) randomly swaps two selected words. Random deletion (RD) removes words within a sentence with a probability of p. This transformation is repeated n times. The method often yields substantial gains on small data sets. Compositional DA generates augmented examples by combining fragments or phrases across sentences to create more comprehensive and diverse examples. Good-Enough Compositional Data Augmentation (GECA) (Andreas, 2020) swaps interchangeable sentence fragments that occur in similar linguistic contexts to promote compositionality. The method is based on the linguistic principle that fragments observed in identical or similar environments are likely to be syntactically and semantically compatible, and therefore interchangeable. These methods minimize external dependencies, relying only on the input data itself, making them ideal for low-resource scenarios.

Feature-Space Feature-space methods generate augmented data by modifying lower-dimensional representations (H. Zhang, Cisse, Dauphin, & Lopez-Paz, 2017; Guo, Mao, & Zhang, 2019; R. Zhang, Yu, & Zhang, 2020; Jindal et al., 2020; Chen, Yang, & Yang, 2020). In contrast to text transformations, which target words or sentence fragments, feature space methods do not modify the text itself but rather modify the hidden representations. Adversarial DA creates augmented examples by adding adversarial perturbations to the original data (Chen et al., 2023). This is achieved by introducing small yet targeted perturbations into the original text, which helps the models generalize more effectively against adversarial manipulations (Jia & Liang, 2017). Interpolation is a method of DA for NLP that creates new examples and labels by combining existing data-label pairs through linear combinations of their representations, such as word or sentence embeddings or hidden states in a neural network (Chen et al., 2023). This method has demonstrated its effectiveness in mitigating overfitting and enhancing training data diversity (Chen et al., 2020). Like non-neural models, these depend only on existing data, supporting generalization in low-resource scenarios and aligning with the survey's focus on minimal to no dependency.

Generative Generative methods employ pre-trained language models, such as decoder-based language models like GPT (Radford et al., 2019) or neural networks, to create new synthetic text samples (Anaby-Tavor et al., 2020; Chen et al., 2023). They introduce higher dependencies via using pre-trained models (often trained on additional text) for synthetic samples, which limits their applicability in low-resource scenarios when not enough text is available for reliable training of these models. Contextual synonym

replacement uses pre-trained language models to generate context-appropriate word substitutions (Chen et al., 2023). Specifically, (Kobayashi, 2018) employs bidirectional recurrent neural networks (RNNs) to predict contextually relevant synonyms based on surrounding words. Back-translation (Chaudhary, n.d.; Sennrich, Haddow, & Birch, 2016) translates and back-translates sentences from one language to another and then translates them back again to obtain a new representation. This is referred to as single-target language back-translation. The method can also be used with multiple target languages, such as Japanese, Arabic, and Russian, to create diverse variations of the augmentation. This is referred to as multiple-target language back-translation. Language-model-based data augmentation (LAMBADA) (Anaby-Tavor et al., 2020) introduces a pipeline that fine-tunes language models to create new synthetic samples. While effective, their reliance on pre-trained language models contrasts with lower-dependency categories.

External Resource-based Highest in dependencies, these methods utilize external resources such as thesauri or treebanks, which are often unavailable in low-resource contexts (Chen et al., 2023; Şahin, 2022; Hedderich et al., 2020). EDA's synonym replacement (SR) and random insertion (RI) (Wei & Zou, 2019) swap/insert synonyms from lexical databases such as WordNet (Miller, 1995). Dependency tree morphing (Şahin & Steedman, 2018) crops/rotates sentence structures using treebanks (Jurafsky & Martin, 2019).

This category's dependencies underscore the need for alternatives. Table 3.1 summarises the above overviews of various widely used DA methods and analyzes their dependence on text transformation based on predictable and reproducible outcomes (non-neural), modifying lower-dimensional representations (Feature-Space), using pre-trained language models (Generative), and external knowledge (External Resource-based).

DA Method	Non-neural	Feature-Space	Generative	Ext. Resource-based
EDA (Wei & Zou, 2019)	✓			✓
CONTEXTUALAUG (Kobayashi, 2018)			✓	
DTREEMORPH (Şahin & Steedman, 2018)				✓
GECA (Andreas, 2020)	✓			
BACKTRANSLATION (Sennrich et al., 2016)			✓	
MIXTEXT (Chen et al., 2020)		✓		
ADVERSARIAL (Jia & Liang, 2017)		✓		
LAMBADA (Anaby-Tavor et al., 2020)			✓	

Table 3.1.: Overview of data augmentation methods. The check mark (✓) indicates the respective dependency of the suggested methods.

Relevant works of DA methods in low-resource scenarios were presented, and the limitations were highlighted. As illustrated in Table 3.1, the categorization by the presented taxonomy in Figure 3.1 forms a hierarchy where non-neural and Feature-Space methods have the fewest dependencies. Other relevant works (Hedderich et al., 2020) observed that with minimal amounts of data, classic non-neural methods often outperform

modern neural methods in truly low-resource scenarios, making non-neural and Feature-Space methods particularly suitable for such scenarios. In recent years, interest in the pre-training of language models with limited datasets has increased. In this regard, the BabyLM challenge was introduced (Warstadt et al., 2025; Choshen et al., 2024; L. Charpentier et al., 2025). Some participants in this challenge investigated the role and potential of DA methods inspired by human language acquisition to improve the performance of low-resource language modeling. The following Section 3.2 provides a closer examination of the submitted methods.

3.2. Human Language Acquisition-Inspired Data Augmentation

Human language acquisition is remarkably efficient, with children having fully mastered language by puberty. Yet they only hear between 3 and 11 million words per year. This makes conventional human language acquisition significantly more efficient than current language model training (Warstadt & Bowman, 2022). Therefore, research initiatives such as the BabyLM Challenge were launched to bridge the gap between the huge datasets used in modern LLM pre-training and efficient language acquisition in humans. (Warstadt et al., 2025; Choshen et al., 2024; L. Charpentier et al., 2025). To close this efficiency gap, data augmentation (DA) of available texts in low-resource scenarios has a relevant importance for data efficiency. By synthetically expanding limited corpora, a certain degree of data efficiency can be achieved, allowing models to learn more robust representations without relying on extensive external datasets. The BabyLM Challenge has driven innovation in such DA methods. The following section presents the most important approaches of recent years within the framework of the challenge.

Lyman and Hepner (2024) trains a neural network (Mikolov et al., 2013) to replace semantically similar words within the corpus and uses grammatical annotations for correctness. The method is inspired by children's language acquisition and "what if" questioning for reading comprehension. Jumelet et al. (2023) introduces Automatic Task Formation (ATF), a task-specific text generation approach from training data using regular expression patterns and templates, without relying on external resources. (Haga et al., 2024) proposes variation sets (VSs) that use only child-directed speech and include utterances with similar intent but different variations (e.g., "What would you like?" vs. "What do you need?"). These variations are created using a pre-trained GPT-2 language model (Radford et al., 2019). The results showed positive impacts. However, the author suggests exploring non-neural alternatives for greater efficiency. (Edman et al., 2024) drew from second-language (L2) learning, generating paraphrases and hard negatives with pre-trained language models. Similarly, (Z. Zhang et al., 2023) presents the "CoThought" pipeline, which uses Chain of Thought prompting in large language models. The pipeline restructures small datasets into task-oriented, human-readable texts similar to school texts designed for language learners. (Theodoropoulos et al., 2024) developed "BERTtime Stories," training a decoder-based language model on TinyStories (Eldan & Li, 2023) subsets to generate new stories and then pre-trains an encoder model on a mix of the

synthetically generated story completions and a subset of the BabyLM dataset.

Table 3.2 presents various DA methods inspired by human language acquisition, analyzing their dependence on text transformation based on predictable and reproducible outcomes (non-neural), modifying lower-dimensional representations (Feature-Space), using pre-trained language models (Generative), and external knowledge (External Resource-based).

DA Method	Non-neural	Feature-Space	Generative	Ext. Resource-based
WhatIf (Lyman & Hepner, 2024)				✓
ATF (Jumelet et al., 2023)	✓			
VSs (Haga et al., 2024)			✓	
L2 (Edman et al., 2024)			✓	
Baby's CoThought (Z. Zhang et al., 2023)			✓	
BERTtime Stories (Theodoropoulos et al., 2024)			✓	

Table 3.2.: Overview of the BabyLM challenge data augmentation submissions from the years 2023-2024. The check mark (✓) indicates the respective dependency of the suggested methods.

3.3. Summary

This chapter introduced a taxonomy for DA methods classified by dependencies. Different DA methods were categorized into non-neural, feature-space, generative, and external resource-based. Furthermore, more modern works inspired by human language acquisition, drawn from recent research initiatives, were outlined. While these methods enhance language model performance in low-resource scenarios, many still rely on models that are trained with additional text (Lyman & Hepner, 2024; Haga et al., 2024; Edman et al., 2024; Z. Zhang et al., 2023; Theodoropoulos et al., 2024). To address this gap, the following Chapter 4 introduces RecombiText Augmentation (RTA).

4. RecombiText Augmentation

Building on the research gap identified in the previous chapters, the research goal is to develop a non-neural Data Augmentation (DA) method that does not rely on models trained with additional text for low-resource language modeling. The chapter introduces RecombiText Augmentation (RTA), a compositional DA method for efficient utilization of available data toward data efficiency. Section 4.1 introduces the intuition. Section 4.2 presents the algorithmic formulation. Finally, the summary provides a brief overview of the chapter. The source code is publicly available [1].

4.1. Intuition

The idea behind the proposed method stems from human language acquisition, where language is learned through a collection of patterns, which Tomasello (2010) refers to as usage-based linguistics in his work. The method is based on the idea that similarities occur across sentences and that these similar sentences can be interchanged to a certain extent to obtain new sentences. The process of reusing text fragments is understood as compositional DA in Natural Language Processing (NLP). To apply the data-driven ideas of human language acquisition (Tomasello, 2010; A. E. Goldberg, 2006; Gentner, 1983; Gleitman, 1990; Weir, 1962) in computer science, the concept of a one-point crossover from genetic algorithms (D. E. Goldberg, 1989) is borrowed. The idea is to create a new sentence pair from a query and a matching sentence based on lexical and semantic similarity. To identify the pivot elements where sentences can be cut, a sliding context window is utilized that searches for similar fragments within both sentences and selects the words with the highest similar information content for a crossover. Figure 4.1 shows the intuition behind the RecombiText Augmentation (RTA) algorithm to create new sentence pairs.

4.2. Algorithmic Formulation

The following section provides a detailed description of the RTA algorithm, which generates synthetic sentences using non-neural methods to augment language model pre-training datasets in low-resource scenarios. The algorithm only accesses the existing data from the training corpus. It combines techniques from Information Retrieval (IR) to find matching results for a given query sentence. In addition, statistical methods such as GloVe (Pennington et al., 2014) and uSIF (Ethayarajh, 2018b) are used for word and

[1] https://github.com/luciendgolden/RTA

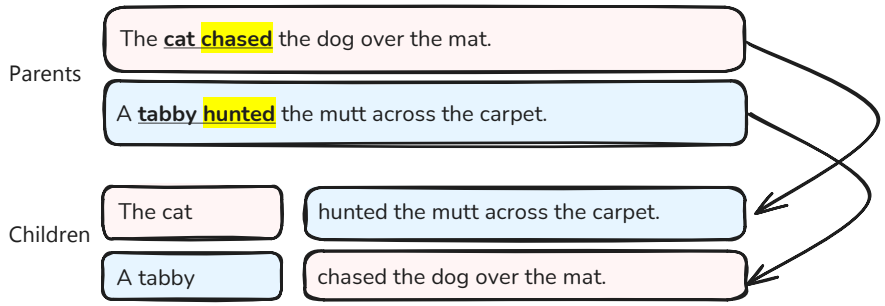

Figure 4.1.: The idea is to create new synthetic sentences based on lexical and semantic similarity from parts of sentences using a one-point crossover. This involves searching for a semantic context window that maximizes the IDF-weighted cosine similarity between a reference and candidate sentence, to determine the pivot elements at which the sentences can be cut and swapped. In this example, the semantic context window size is $W = 2$

sentence embeddings to measure semantic similarities within words and sentences. With the help of the crossover operation inspired by genetic algorithms (D. E. Goldberg, 1989), novel sentence pairs are generated. The process is divided into four main phases.

1. **Word and Sentence Embeddings:** Generate word and sentence embeddings suitable for low-resource scenarios to determine the semantics of the respective words/sentences and save them beforehand for the entire corpus.

2. **Candidate Selection:** Perform a hybrid search for suitable lexical/semantic candidates from the ad hoc IR system, rerank the candidates by combining both results, and select a relevant matching candidate.

3. **Context Window:** Identify a suitable semantic context window where both sentences can be divided into two parts, and determine the relevant pivot elements.

4. **Crossover Operation:** Execute the one-point crossover operation at the pivot elements.

4.2.1. Notation

Table 4.1 shows the notation that is used to describe the algorithm.

4.2.2. Word and Sentence Embeddings

Global Vectors for Word Representation (GloVe) (Pennington et al., 2014) is employed for word embeddings, and unsupervised smoothed inverse frequency (uSIF) (Ethayarajh, 2018a) for sentence embeddings. Both are calculated directly from the available training data. The word embeddings are passed to the algorithm as an input file and cached.

Expression	Definition
d	Document
D	Set of documents
q	Query sentence
$\|q\|, \|m\|$	Sentence lengths
$\tilde{q}, \tilde{m}$	Augmented sentence pairs
$\mathcal{R}_{BM25}$	Lexical search results (BM25)
$\mathcal{R}_{kNN}$	Semantic search results (k-NN)
$\mathcal{R}_{RRF}$	Fused ranked list from Reciprocal Rank Fusion
k_{RRF}	RRF constant for ranking fusion
τ_{window}	Threshold for context window similarity
k_{top}	Number of top-k candidates for sampling
$p(x_i)$	Softmax probability
T	Temperature parameter for softmax
m	Candidate sentence
W	Fixed size of the context window
$\mathbf{v}_t \in \mathbb{R}^d$	Word embedding of token t in d-dimensional space
$\mathbf{v}_{uSIF} \in \mathbb{R}^d$	Sentence embeddings in d-dimensional space
$\text{idf}(t, D)$	IDF value of token t in corpus D
D	Corpus (document collection)
$S_{\text{ctx}}(i, j)$	Context window similarity
i, j	Positions within the sliding context windows
k^*	Pivot index within the context window maximizing similarity
i^*, j^*	Pivot elements for q, m
$\|\|$	Concatenation operator

Table 4.1.: Mathematical notation for the RecombiText Augmentation (RTA) algorithm.

The sentence embeddings are calculated by the algorithm based on the passed word embeddings and stored in the index of the respective document d in the IR system.

4.2.3. Candidate Selection

The following describes how the algorithm retrieves a pool of candidate sentences and selects a suitable matching sentence. Both the lexical and semantic relevance to the search sentence are taken into account.

Hybrid Search For each selected query sentence q, an efficient search is performed in the IR system (see Section 2.3). Two search processes are then performed. First, a lexical search with BM25 (Robertson & Zaragoza, 2009; Jurafsky & Martin, 2019) (see Subsection 2.3.2) is performed to obtain a ranking $\mathcal{R}_{BM25}$ of lexically similar sentences with their score ratings. Then, a semantic search with k-nearest neighbors (k-NN) (Cover & Hart, 1967) (see Subsection 2.2.3) with cosine similarity on the uSIF sentence

embeddings (Ethayarajh, 2018a) is performed, returning a ranking $\mathcal{R}_{kNN}$ of semantically similar sentences (see Subsection 2.2.3).

Reranking The results from the lexical and semantic searches are combined using Reciprocal Rank Fusion (RRF) (Cormack et al., 2009) to produce a fused ranked list $\mathcal{R}_{RRF}$ from $\mathcal{R}_{BM25}$ and $\mathcal{R}_{kNN}$ (see Subsection 2.3.3).

Top-k Sampling After obtaining the reranked list $\mathcal{R}_{RRF}$, a single candidate is selected as the result sentence for the one-point crossover operation. Instead of deterministically selecting the candidate with the highest RRF score, a top-k sampling strategy is employed to maintain the diversity of the augmentation. First, the candidates in $\mathcal{R}_{RRF}$ are sorted in descending order of their respective RRF scores. The best k_{top} candidates are then selected from this list, where k_{top} is a predefined parameter. A single candidate is then chosen from these best k_{top} candidates based on a probability distribution. Specifically, a softmax function is applied to the RRF scores, and a temperature parameter T is defined. Equation 4.1 shows the softmax function used for top-k sampling.

$$p(x_i) = \frac{e^{\frac{x_i}{T}}}{\sum_{j=1}^{k_{top}} e^{\frac{x_j}{T}}} \tag{4.1}$$

where x_i is the RRF score of the i-th candidate. The temperature T modulates the randomness of the selection. A lower T value favors the selection of candidates with higher scores, whereas a higher T value increases the probability of selecting candidates with lower scores.

- As $T \to 0$, the selection becomes almost deterministic and significantly less diverse.
- As $T \to \infty$, the distribution probabilities are evenly distributed, thereby maximizing diversity.

The selected candidate m is then used for the subsequent one-point crossover operation. If the subsequent crossover operation does not generate valid synthetic sets (see subsection 4.2.5), the algorithm repeats the selection process with another candidate from $\mathcal{R}_{RRF}$. The process can be repeated as long as there are candidates available.

4.2.4. Context Window

Before proceeding, the tokens are normalized to lowercase and stripped of whitespace before comparison. To ensure a useful degree of linguistic value for the generated sentences after the crossover operation, so-called pivot elements are identified where the query sentence q and the selected matching candidate m exhibit high semantic similarity in terms of context. This is achieved using a sliding context window where sequences of the two sentences are compared based on the importance and semantic equivalence. To avoid placing too much importance on words that are often insignificant, the respective words within the context window are weighted by their IDF values.

The context window W has a fixed size and specifies the number of consecutive words within the window, which are referred to as tokens. The window moves dynamically over the window tokens in the query sentence q and the top-k candidate sentence m with word embeddings $\mathbf{v}_t \in \mathbb{R}^d$ and IDF values $\text{idf}(t, D)$ where t represents the term and D the set of documents.

For each possible starting position in both sentences, a window of size W is defined, which starts at positions i in the tokenized query sentence q and j in the tokenized candidate sentence m. A context window similarity is defined as $S_{\text{ctx}}(i, j)$ between the corresponding windows of size W. Equation 4.2 shows the similarity calculation for the context window.

$$S_{\text{ctx}}(i,j) = \frac{\sum_{k=0}^{W-1} \cos\left(\mathbf{v}_{q_{i+k}}, \mathbf{v}_{m_{j+k}}\right) \cdot \left(\text{idf}(q_{i+k}, D) + \text{idf}(m_{j+k}, D)\right)}{\sum_{k=0}^{W-1} \left(\text{idf}(q_{i+k}, D) + \text{idf}(m_{j+k}, D)\right)} \tag{4.2}$$

where $\cos(\mathbf{v}_a, \mathbf{v}_b)$ is the respective cosine similarity between the token embeddings in query q and match m sentence. If no valid pairs exist $S_{\text{ctx}}(i, j) = 0$. The optimal context window is determined by maximizing the similarity $S_{\text{ctx}}(i, j)$. Within this window, the pivot elements are the indices $(i^*, j^*) = (i + k^*, j + k^*)$, where k^* represents the maximum similarity pivot index and is defined by Equation 4.3

$$k^* = \arg\max_k \cos(\mathbf{v}_{q_{i+k}}, \mathbf{v}_{m_{j+k}}) \cdot \left(\text{idf}(q_{i+k}, D) + \text{idf}(m_{j+k}, D)\right) \tag{4.3}$$

The crossover then splits before these tokens at position k^* in the respective documents d. A crossover is then performed only if $S_{\text{ctx}}(i, j)$ exceeds a predefined threshold τ_{window}.

4.2.5. Crossover Operation

The pivot elements determined in the context window are now used for the one-point crossover operation. This involves creating two new synthetic sentences by exchanging the respective intersections in query sentence q and matching sentence m. The respective pivot elements i^*, j^* are obtained from the previous step in 4.2.4. Therefore let $q = \langle q_0, \ldots, q_{|q|-1} \rangle$ be the query sentence tokens and $m = \langle m_0, \ldots, m_{|m|-1} \rangle$ the selected match sentence tokens. The two pivot positions returned by the context–window search are denoted as $i^* \in \{0, \ldots, |q|\}$ in q and $j^* \in \{0, \ldots, |m|\}$ in m. The one-point crossover then produces two new synthetic sentences. Equation 4.4 shows the one-point crossover operation.

$$\begin{aligned} \tilde{q} &= \langle q_0, \ldots, q_{i^*-1} \mid\mid m_{j^*}, \ldots, m_{|m|-1} \rangle \\ \tilde{m} &= \langle m_0, \ldots, m_{j^*-1} \mid\mid q_{i^*}, \ldots, q_{|q|-1} \rangle \end{aligned} \tag{4.4}$$

4.2.6. Pseudo Code

Algorithm 1 explains the algorithmic formulation as pseudo code.

Algorithm 1: RecombiText Augmentation (RTA) Algorithm

Data: Dataset D, $q \in D$, $\mathbf{v}_t \in \mathbb{R}^d$, $\mathbf{v}_{uSIF} \in \mathbb{R}^d$, $\text{idf}(t, D)$
Result: $\tilde{q}, \tilde{m}$
Retrieve $\mathcal{R}_{BM25}$ for q according to Equation 2.24;
Retrieve $\mathcal{R}_{kNN}$ for q according to Equation 2.22 on $\mathbf{v}_{uSIF} \in \mathbb{R}^d$;
$\mathcal{R}_{RRF} \leftarrow empty$;
for d *in* $\mathcal{R}_{BM25} \cup \mathcal{R}_{kNN}$ **do**
    Compute RRFscore according to Equation 2.25;
    Add d to $\mathcal{R}_{RRF}$ with its RRF score;
end
Sort $\mathcal{R}_{RRF}$ in descending order of RRF scores;
$retry \leftarrow 1$;
for $retry \leq |\mathcal{R}_{RRF}|$ **do**
    Select k_{top} candidates from $\mathcal{R}_{RRF}$;
    Compute $p(x_i)$ on k_{top} candidates according to Equation 4.1;
    Sample candidate sentence m according to probabilities p;
    Tokenize $q = \langle q_0, \ldots, q_{|q|-1} \rangle$ and $m = \langle m_0, \ldots, m_{|m|-1} \rangle$;
    Normalize q, m;
    $S_{\max} \leftarrow 0$, $i^* \leftarrow 0$, $j^* = 0$;
    for $i = 0$ **to** $|q| - W + 1$ **do**
        for $j = 0$ **to** $|m| - W + 1$ **do**
            Compute $S_{\text{ctx}}(i, j)$ according to Equation 4.2;
            if $S_{ctx}(i, j) > S_{\max}$ **then**
                $S_{\max} \leftarrow S_{\text{ctx}}(i, j)$;
                Compute k^* according to Equation 4.3;
                $i^* \leftarrow i + k^*$;
                $j^* \leftarrow j + k^*$;
            end
        end
    end
    if $S_{\max} > \tau_{window}$ **then**
        $\tilde{q}, \tilde{m}$ according to Equation 4.4;
        return $\tilde{q}, \tilde{m}$;
    else
        $retry \leftarrow retry + 1$;
        Remove candidate m from $\mathcal{R}_{RRF}$;
        continue;
    end
end

4.3. Summary

This chapter introduced RecombiText Augmentation (RTA), a non-neural compositional DA method inspired by human language acquisition. The algorithm utilizes corpus-based statistics and employs a hybrid search approach to identify relevant results for a selected query sentence. A one-point crossover from genetic algorithms (D. E. Goldberg, 1989) is then performed to produce an augmented sentence pair. RTA generates augmented sentences without using external dependencies that were trained on additional text, thereby addressing the inefficiencies and gaps identified in previous chapters. The next Chapter 5 describes the experimental setup to evaluate the method.

5. Experiments

This chapter describes the experimental setup for the RecombiText Augmentation (RTA) method presented in Chapter 4. The study is intended as a contribution to the BabyLM Challenge 2025 (L. Charpentier et al., 2025) and evaluates the potential of RTA in low-resource language modeling. The experiments are designed to strictly adhere to the challenge guidelines on data restrictions and resource usage. The total number of words is defined by the number of whitespace-separated input words from the dataset, which must not exceed 100M words in total by all models trained on text. No external data sources other than those provided by the challenge were used. All necessary steps were designed to ensure that the word budget was adhered to. The experiments are carried out in four stages. Firstly, the generation of the augmented training data in Section 5.1. Secondly, the evaluation of the quality of augmented training data in Section 5.2. Thirdly, the pre-training of the language model in Section 5.3. Fourthly, the evaluation of the language models using zero-shot and fine-tuning tasks is presented in Section 5.4. Finally, the summary provides a brief overview of the chapter.

5.1. Data Generation

The experiments use data from the official BabyLM Challenge 2025 (L. Charpentier et al., 2025). The data is publicly available[1] and simulates a domain-specific, low-resource corpus. The dataset was explicitly developed as an experimental dataset for language model training research in low-resource scenarios. The `strict-small` version is used, containing approximately 10 million words of English texts from various sources. Table 5.1 shows the different data sources from the used dataset.

Dataset	Description	Citation	# Words
British National Corpus (BNC)	Dialogue	(Consortium et al., 2007)	0.93M
CHILDES	Child-directed speech	(MacWhinney, 2000)	2.84M
Project Gutenberg (children's stories)	Written English	(Gerlach & Font-Clos, 2020)	2.54M
OpenSubtitles	Movie subtitles	(Lison & Tiedemann, 2016)	2.04M
Simple English Wikipedia	Written Simple English	-	1.45M
Switchboard Dialog Act Corpus	Dialogue	(Godfrey, Holliman, & McDaniel, 1992; Stolcke et al., 2000)	0.15M
Total			9.95M

Table 5.1.: Datasets for the experiments simulating a domain-specific low-resource scenario by L. Charpentier et al. (2025) after filtering.

[1] `https://osf.io/ryjfm/`

5.1.1. Sampling

To create the different proportions from the baseline dataset, the sample chunks and split script from Warstadt et al. (2025) is used. The sampling process randomly assigns entire chunks of text from the input file to a training, development, or test file to achieve approximately the desired number of words or the desired percentage. Context integrity is ensured by not splitting the chunks. The probabilities are calculated based on user-defined percentages. Each finished chunk is randomly assigned to the output files for training, test, or development, or discarded based on the specified probabilities. This results in a stratified random sampling at the chunk level rather than per line or word.

5.1.2. Preprocessing

The data sets were preprocessed to remove specific metadata. The custom Python script from Timiryasov and Tastet (2023) was used for this purpose. Corpus-specific cleaning functions are applied to normalize the text for model training. Additional spaces, tabs, and unnecessary line breaks were removed from the respective datasets. Additional cleanups vary depending on the corpus. In OpenSubtitles (Lison & Tiedemann, 2016), for example, subtitle credits are removed. The cleaned files are then saved under the same name in the output directory. The division into train/dev/test is retained.

5.1.3. Data Augmentation

$\mathcal{D}_{base}$ is defined as the $\approx 10M$ words baseline dataset from Table 5.1 and $\mathcal{D}_{baseX}$ as the proportion that is used from $\mathcal{D}_{base}$, where X is the proportion number of words in millions sampled. Furthermore, $\mathcal{D}_{aug}$ is the dataset generated by RTA, forming the augmented datasets $\mathcal{D}_{augY}$, where Y is the number of words in millions generated. In the mixed dataset $\mathcal{D}_{baseX} + \mathcal{D}_{augY}$, the augmented sentence $\tilde{m}$ is added for each search term q, and for each matching sentence m, the augmented sentence $\tilde{q}$ is added. This type of augmentation is referred to as adjacent data augmentation. Table 5.2 shows the different proportion samples taken from $\mathcal{D}_{base}$ along with the generated text $\mathcal{D}_{aug}$ that form the basis for the mixed training data.

For the 10%–90% proportions, the sampled baseline data is extended by the augmented data. For the 100% proportion, the whole baseline is extended by augmented sentences, which results in an ≈20M word dataset.

5.1.4. Models

The following describes the architectures, configurations, and training setups for the different models used in the data generation.

GloVe Global Vectors for Word Representation (GloVe) (Pennington et al., 2014) is used before running the algorithm to generate word embeddings, enabling the determination of semantic relationships between words and within subsequent sentence sequences in the RTA algorithm. The embeddings capture the semantic relationships between words

Variant	Proportion	# Total Words	# Base Words	# Aug Words	Ratio r
$\mathcal{D}_{base10M}$	0%	9.95M	9.95M	0.00M	0.00
$\mathcal{D}_{base1M} + \mathcal{D}_{aug9M}$	10%	10.55M	1.06M	9.49M	9.00
$\mathcal{D}_{base2.5M} + \mathcal{D}_{aug7.5M}$	25%	9.68M	2.43M	7.25M	3.00
$\mathcal{D}_{base5M} + \mathcal{D}_{aug5M}$	50%	9.72M	4.96M	4.76M	1.00
$\mathcal{D}_{base7.5M} + \mathcal{D}_{aug2.5M}$	75%	9.99M	7.48M	2.52M	0.33
$\mathcal{D}_{base9M} + \mathcal{D}_{aug1M}$	90%	9.96M	8.95M	1.01M	0.11
$\mathcal{D}_{base10M} + \mathcal{D}_{aug10M}$	100%	19.49M	9.95M	9.54M	1.00

Table 5.2.: Sampled proportions from the `strict-small` low-resource training data by L. Charpentier et al. (2025), total words (in millions) in the resulting training dataset after data augmentation, actual number of words sampled from the baseline dataset $\mathcal{D}_{base10M}$, augmented words generated as $\mathcal{D}_{aug}$ with RTA and the nominal augmentation ratio $r = \frac{|\mathcal{D}_{augY}|}{|\mathcal{D}_{baseX}|}$. The actual ratios may vary slightly due to sampling and data preprocessing.

based on their co-occurrence statistics in the corpus, forming a key foundation of the data augmentation process. The model was trained on the different training data variations outlined in Subsection 5.1.3. The embeddings are passed to the data augmentation algorithm as an input file. The training was conducted locally on a 2023 MacBook Pro with 18 GB of memory and an Apple M3 Pro chip. Table 5.3 describes the hyperparameters used for the GloVe models.

Hyperparameter	Value
Vector Size	50
Window Size	10
Minimum Vocabulary Count	5
Maximum Co-occurrence Weight (`x_max`)	10
Maximum Iterations	25
Number of Threads	4
Memory	4.0 GB

Table 5.3.: Hyperparameters for the GloVe model used in the experiments.

uSIF Unsupervised smoothed inverse frequency (uSIF), introduced by Ethayarajh (2018b), is used when running the algorithm to create sentence embeddings to determine the semantics between sentences. uSIF (Ethayarajh, 2018b) takes two parameters: n, the expected random walk length, which is the average sentence length from the input file, and m, the number of common discourse vectors. For n, the respective average sentence

length of the dataset is used, which is currently being processed. For m, the default value of `5` is used. The sentence embeddings are stored in the Information Retrieval (IR) system as vectors in the respective IR document at runtime. The calculation of the sentence embeddings was conducted locally on a 2023 MacBook Pro with 18 GB of memory and an Apple M3 Pro chip.

RTA The augmentation ratio r controls the amount of synthetic text generated for $\mathcal{D}_{augY}$ relative to the proportional baseline dataset $\mathcal{D}_{baseX}$, where the target augmented words are $\approx \mathcal{D}_{baseX} \times r$. The process estimates and runs multiple rounds to achieve this, with each round producing a portion of the total. The goal of each round is to generate uniform augmentations. The sentences are selected by weighted random selection, with less frequently used sentences being preferred. The candidates m are weighted with a usage penalization and chosen from the top-k selection with a temperature T. RTA is then used to form two new augmented sentences $\tilde{q}$ and $\tilde{m}$. The sentences are inserted next to the originals, forming the dataset $\mathcal{D}_{aug}$. The frequency of use for q, m is limited to `max(1, int(`r`))` per sentence to prevent excessive use and reduce bias. This results in an expanded corpus that mixes originals with synthetic examples for training. Table 5.4 lists the selected hyperparameters for each variant. Each hyperparameter was chosen to strike a balance between semantic similarity and sentence diversity.

Variant	**RRF** k	τ_{window}	W	**top-**k	T	**Ratio** r
$\mathcal{D}_{base1M} + \mathcal{D}_{aug9M}$	60	60%	3	50	1.3	9.00
$\mathcal{D}_{base2.5M} + \mathcal{D}_{aug7.5M}$	60	60%	3	30	1.1	3.00
$\mathcal{D}_{base5M} + \mathcal{D}_{aug5M}$	60	60%	3	20	1.0	1.00
$\mathcal{D}_{base7.5M} + \mathcal{D}_{aug2.5M}$	60	60%	3	15	0.9	0.33
$\mathcal{D}_{base9M} + \mathcal{D}_{aug1M}$	60	60%	3	10	0.8	0.11
$\mathcal{D}_{base10M} + \mathcal{D}_{aug10M}$	60	60%	3	20	1.0	1.00

Table 5.4.: Hyperparameters for the RTA.

Implementation used `Python 3.13` with `Elasticsearch 8.0.0` as the ad hoc IR system on a 2023 MacBook Pro with 18 GB of memory and an Apple M3 Pro chip.

5.2. Data Quality

This section introduces the evaluation of the quality for the augmented datasets $\mathcal{D}_{aug}$, focusing on perplexity for language model predictability and Self-BLEU for diversity. The data quality evaluation is independent of data generation and serves solely to evaluate the quality of the generated texts afterwards.

Perplexity Perplexity is used for measuring fluency and applied separately to $\mathcal{D}_{baseX}$ and the augmented dataset $\mathcal{D}_{augY}$. The official OpenAI GPT-2 language model (Radford

et al., 2019) from Hugging Face[2] is utilized, as it has been trained on large amounts of data. This enables the data quality evaluation of how surprised the model is by the augmented examples.

Self-BLEU To evaluate the diversity of the synthetic sentences generated with RTA, the Self-BLEU is calculated separately for $\mathcal{D}_{baseX}$ and $\mathcal{D}_{augY}$. Each sentence is treated as a hypothesis, and the remaining augmented corpus as a reference. The experiments include the determination of BLEU scores over three runs. A subsample of $1,000$ sentences is used per run, and an average is calculated across all three runs at the end.

5.3. Language Model Pre-training

The training strategy for language models (LM) adheres to the BabyLM restrictions mentioned above, where all models trained on text are not allowed to see more than 100M whitespace-separated words. This budget is inspired by estimates of human language acquisition, where children hear roughly 100M words by puberty to fully master a language (see Chapter 1) (Warstadt et al., 2025). This strict limit necessitates a more complex and sometimes unintuitive experimental design but ensures a fair comparison between the different variants. The following section briefly explains the assumptions for complying with the restrictions, followed by the checkpointing strategy and the considerations that were made to evaluate RTA as an isolated method.

Setup The total whitespace-separated words indicate the number of words for the training dataset $|\mathcal{D}_{baseX} + \mathcal{D}_{augY}|$ which is referred to as # Words. The Avg. Splits/Word is derived from byte-pair encoding (BPE) tokenization using the script by L. G. G. Charpentier and Samuel (2024). # Words GloVe equals the size of the sampled base dataset. For each variant, the GloVe (Pennington et al., 2014) model is trained solely on the respective dataset portion $|\mathcal{D}_{baseX}|$. This word count is deducted from the 100M word budget to allocate the remaining words to the LM training in # Words LM. Tokens/Step is fixed at 8192 since the batch size of 16 and sequence length of 512 is chosen. Multiplying the batch size and the sequence length gives Tokens/Step $= 16 \times 512 = 8192$. Words/Step approximates the words processed per step by dividing Tokens/Step by Avg. Splits/Word, for example, yielding approximately 5599 Words/Step for the $\mathcal{D}_{base1M} + \mathcal{D}_{aug9M}$ variant via Words/Step $= \frac{\text{Tokens/Step}}{\text{Avg. Splits/Word}}$. Max Steps gives the total training steps needed to exhaust the language model word budget which is calculated as the maximum number of words the LM should see divided by Words/Step via Max Steps $= \frac{\text{\# Words LM}}{\text{Words/Step}}$. Epochs is the number the LM iterates over the dataset and is calculated as Epoch $= \frac{\text{\# Words LM}}{\text{\# Words}}$ to ensure that the LM processes its exact word allocation over multiple dataset passes without exceeding the budget. Table 5.5 shows the training setup for the LMs based on BPE tokenizer.

[2] https://huggingface.co/openai-community/gpt2

Checkpointing Furthermore, the models follow a specific checkpointing strategy where checkpoints are created every 1M words during the initial 10M words, and every 10M words thereafter until the final checkpoint at 100M words.

Assumptions For the 10%–90% proportions, the training data $\mathcal{D}$ are trained on ≈ 10 epochs ($\pm$ number words GloVe has seen). For the 100% proportion, the training data $\mathcal{D}$ are trained on ≈ 5 epochs ($\pm$ number words GloVe has seen). This shall ensure a fair comparison between the different training scenarios. Let $|\mathcal{D}|$ be the total number of words from the final training dataset and E be the number of epochs, then the total words processed by the language model $\mathcal{P}$ is calculated as shown in Equation 5.1.

$$\begin{aligned} \mathcal{P} &= |\mathcal{D}| \times E \\ \mathcal{P}_1 &= (10 \times 10^6) \times 10 = 100 \times 10^6 \\ \mathcal{P}_2 &= (20 \times 10^6) \times 5 = 100 \times 10^6 \end{aligned} \tag{5.1}$$

and since $\mathcal{P}_1 = \mathcal{P}_2$ both scenarios process the same total volume of approximately 100 million words ($\pm$ number words GloVe has seen).

Variant	# Words	Avg. Splits/Word	# Words GloVe	# Words LM	Tokens/Step	Words/Step	Max Steps	Epochs
$\mathcal{D}_{base10M}$	9.95M	1.61	0.00M	100.00M	8192.00	5095.00	19,627.00	10.05
$\mathcal{D}_{base1M} + \mathcal{D}_{aug9M}$	10.55M	1.46	1.06M	98.94M	8192.00	5599.00	17,672.00	9.38
$\mathcal{D}_{base2.5M} + \mathcal{D}_{aug7.5M}$	9.68M	1.52	2.43M	97.57M	8192.00	5397.00	18,079.00	10.08
$\mathcal{D}_{base5M} + \mathcal{D}_{aug5M}$	9.72M	1.53	4.96M	95.04M	8192.00	5358.00	17,738.00	9.77
$\mathcal{D}_{base7.5M} + \mathcal{D}_{aug2.5M}$	9.99M	1.56	7.48M	92.52M	8192.00	5238.00	17,664.00	9.26
$\mathcal{D}_{base9M} + \mathcal{D}_{aug1M}$	9.96M	1.59	8.95M	91.05M	8192.00	5152.00	17,672.00	9.14
$\mathcal{D}_{base10M} + \mathcal{D}_{aug10M}$	19.49M	1.53	9.95M	90.05M	8192.00	5365.00	16,785.00	4.62

Table 5.5.: Language Model training setup for each data augmentation variant, where # Words represents the total whitespace-separated words in the preprocessed training dataset, Avg. Splits/Word indicates the average subword tokens per word from BPE tokenizer, # Words GloVe is the size of the base dataset that the model has seen, # Words LM reflects the budgeted words for the language model pre-training, Tokens/Step is the number of tokens per training step the LM sees, Words/Step approximates the number of words per training step the LM sees, Max Steps gives the total training steps for the LM, Epochs is the number the LM iterates over the dataset.

The language models were trained on 1x NVIDIA H100 Tensor Core GPU with `Python 3.12.9`, `Transformers 4.50.3`, and `PyTorch 2.7.1+cu126`.

5.3.1. GPT-2

Table 5.6 shows the hyperparameters for the causal language modeling (CLM) with GPT-2 (Radford et al., 2019) language model pre-training (see Chapter 2). These settings were

applied to both the baseline model and the models trained on the augmented data. For models trained on augmented data variants, the maximum training steps are reduced by the number of words the GloVe model (Pennington et al., 2014) has seen.

Hyperparameter	**Value**
Model Type	openai-community/gpt2
Tokenizer	BPE
Learning Rate	5×10^{-5}
Maximum Gradient Norm	1.0
Adam β_1	0.9
Adam β_2	0.999
Adam ϵ	1×10^{-8}
Block Size	512
Batch Size	16
Save Strategy	Steps
Save Total Limit	20
Logging Steps	100
Evaluation Strategy	Steps
Seed	42

Table 5.6.: Hyperparameters for the GPT-2 model training used in the experiments.

5.3.2. RoBERTa

Table 5.7 shows the hyperparameters for the masked language modeling (MLM) with RoBERTa (Liu et al., 2019) language model pre-training (see Chapter 2). These settings were applied to both the baseline model and the models trained on the augmented data. For models trained on augmented data variants, the maximum training steps are reduced by the number of words the GloVe model (Pennington et al., 2014) has seen.

5.4. Language Model Evaluation

This section describes the evaluation tasks used to assess the performance of the language models that were trained with the augmented datasets. First, the classification metrics are outlined, then the (fast) zero-shot evaluation tasks are explained, and lastly the finetuning evaluation tasks. The official 2025 BabyLM Challenge Evaluation Pipeline[3] by L. Charpentier et al. (2025) is used.

5.4.1. Zero-shot Evaluation

Zero-shot tasks evaluate the ability of language models without requiring prior task-specific training. The aim is to test generalization and adaptability.

[3] https://github.com/babylm/evaluation-pipeline-2025

Hyperparameter	Value
Model Type	FacebookAI/roberta-base
Tokenizer	BPE
Learning Rate	5×10^{-5}
Maximum Gradient Norm	1.0
Adam β_1	0.9
Adam β_2	0.999
Adam ϵ	1×10^{-8}
Sequence Length	512
Batch Size	32
MLM Probability	15%
Save Strategy	Steps
Save Total Limit	20
Logging Steps	100
Evaluation Strategy	Steps
Seed	42

Table 5.7.: Hyperparameters for the RoBERTa model training used in the experiments.

- BLiMP: Benchmark of Linguistic Minimal Pairs (BLiMP) describes a method for evaluating language models based on their ability to adapt to the structure of the English language. Models are tested with minimal sentence pairs. A pair of sentences is presented. A grammatically correct sentence and an incorrect sentence that differ as little as possible. Assigning a higher probability to the correct sentence means higher accuracy (Warstadt et al., 2020).

- BLiMP Supplement: A supplement to BLiMP, which consists of five test suites with BLiMP-like minimal pairs that focus on untested linguistic areas such as Dialogs and questions. The correct sequence is expected to have a higher probability than the unacceptable sequence. (Warstadt et al., 2023)

- EWoK: Elements of World Knowledge (EWoK) evaluates the ability of world modeling for language models. It tests the ability to distinguish plausible from implausible context-target pairs across 11 cognition-inspired domains, such as social interactions and spatial relations. It uses a minimal-pair design (Ivanova et al., 2024).

- Eye Tracking and Self-paced Reading: Are behavioral paradigms used to measure word-by-word language processing. Eye tracking tracks where and for how long words are viewed on a screen, thus showing how quickly the words are understood or whether they need to be re-read. Self-paced reading measures how quickly words are read by pressing a button to see more words. Both methods help to recognize how easily or with difficulty words are processed (de Varda, Marelli, & Amenta, 2024).

- Entity Tracking: Entity Tracking (ENT) tests how well language models track changes to objects such as a book or an apple within a story or conversation. The model is given an initial situation and then asked a question about it (Kim & Schuster, 2023). The task has been adapted accordingly. Language models must assign a certain probability to the respective continuation of a story or conversation (L. Charpentier et al., 2025).

- WUGs: Tests the morphological generalization for language models, such as the conversion of adjectives into nouns, e.g., "happy" to "happiness". The question of whether language models apply strict rules or learn by comparison with previously seen examples is investigated. This is tested with invented adjectives, some of which follow clear patterns and others do not. The task is split into WUG Past, where the model is asked to generate past tense forms, and WUG Adj. for adjective nominalization (Weissweiler et al., 2023; Hofmann, Weissweiler, Mortensen, Schütze, & Pierrehumbert, 2025; L. Charpentier et al., 2025).

- COMPS: Introduces a task designed to assess how language models understand property knowledge, where properties of broader categories are passed down to more specific ones. The dataset consists of minimal pair sentences (Misra, Rayz, & Ettinger, 2022).

- AoA: Age of Acquisition (AoA) tracks the word surprisal over training checkpoints to derive learning curves and determine ages of acquisition for the vocabulary items related to human AoA data. (Chang & Bergen, 2022; L. Charpentier et al., 2025).

Zero-shot tasks are evaluated based on accuracy in predicting the correct completion/sentence for BLiMP (Warstadt et al., 2025), BLiMP Supplement (Warstadt et al., 2025), EWoK (Ivanova et al., 2024), ENT (Kim & Schuster, 2023; L. Charpentier et al., 2025), and COMPS (Misra et al., 2022). WUGs (Hofmann et al., 2025) is evaluated with Spearman ρ. The change in the prediction from the baseline is denoted as R^2, which is used for Eye Tracking and for Self-paced Reading (de Varda et al., 2024; L. Charpentier et al., 2025). AoA (Chang & Bergen, 2022) is evaluated by the Pearson correlation coefficient r_{xy} and the statistical significance p-value.

5.4.2. Checkpoint Evaluation

The learning dynamics during the pre-training of language models are evaluated using fast zero-shot evaluation at intermediate checkpoints (see Section 5.3). The fast zero-shot evaluation utilizes a smaller set of evaluation examples, which enables rapid testing of the models to serve as the basis for performance reports at intermediate checkpoints. Performance is evaluated every 1 million words up to 10 million words, and then every 10 million words up to 100 million words. In this way, it is tracked how quickly the models acquire the respective skills for the zero-shot evaluation tasks (L. Charpentier et al., 2025).

5.4.3. Fine-tuning Evaluation

Fine-tuning tasks evaluate the performance of language models after task-specific training. The fine-tuning tasks use selected tasks from (Super)GLUE (Wang et al., 2018, 2019). The tasks are filtered and reduced to a maximum of 10,000 training examples (L. Charpentier et al., 2025). Table 5.8 shows the hyperparameters used for the finetuning tasks.

Hyperparameter	**MultiNLI, RTE, QQP, MRPC**	**BoolQ, MultiRC**	**WSC**
Learning Rate	3×10^{-5}	3×10^{-5}	3×10^{-5}
Batch Size	32	16	32
Epochs	10	10	30
Weight Decay	0.01	0.01	0.01
Optimizer	AdamW	AdamW	AdamW
Scheduler	cosine	cosine	cosine
Warmup Percentage	6%	6%	6%
Dropout	0.1	0.1	0.1

Table 5.8.: Hyperparameters for fine-tuning the language models used in the experiments across different datasets with default settings from the official evaluation pipeline by L. Charpentier et al. (2025).

The individual tasks for which the language models are fine-tuned are explained briefly.

- BoolQ: Boolean Questions (BoolQ) presents a collection of naturally occurring yes/no questions paired with Wikipedia paragraphs containing the answers. The questions are challenging because they require complex thinking. For Example, implicit information must be used to decide whether something is true or false, not just facts (Clark et al., 2019).

- MNLI: Multi-Genre Natural Language Inference (MNLI) tests how well language models understand sentences. The models must decide whether a sentence supports, contradicts, or is neutral to another. Ten different styles of written and spoken English are covered, including books, speeches, and conversations. MNLI tests models for their ability to adapt to new styles (Williams, Nangia, & Bowman, 2017).

- MRPC: The Microsoft Research Paraphrase Corpus (MRPC) contains pairs of sentences from news articles, each labeled by humans as semantically similar or not similar. The corpus was selected from a large set of matching paraphrase pairs from news data. Humans then checked these for semantic similarity. The dataset helps to determine how well language models can recognize sentences with similar meanings (Dolan & Brockett, 2005).

- QQP2: As with MRPC, the Quora Question Pairs (QQP2) dataset evaluates a language model's ability to recognize whether two questions are semantically similar

based on questions from Quora [4].

- MultiRC: Multi-Sentence Reading Comprehension (MultiRC) is a reading comprehension challenge. Each question requires the combination of information from various sentences to answer. Unlike other datasets, MultiRC allows multiple correct answers and does not require exact text answers from the paragraph (Khashabi, Chaturvedi, Roth, Upadhyay, & Roth, 2018).

- RTE: Recognizing Text Entailment (RTE) tests how well systems can recognize whether one text can be inferred from another. Evaluations, such as implications, neutrals, or contradictions, must be assessed. The language models must identify sentences in documents that support a specific statement (Dagan, Glickman, & Magnini, 2005; Giampiccolo, Magnini, Dagan, & Dolan, 2007; Bentivogli, Clark, Dagan, & Giampiccolo, 2009).

- WSC: Winograd Schema Challenge (WSC) uses pairs of sentences that have only minimal differences, differing only by one or two words. A word or phrase can have multiple interpretations. In this challenge, the computer is given a sentence and must correctly identify the meaning of an ambiguous word. The sentences are formulated in such a way that they are easy for humans to identify. However, they are more difficult for language models to determine, as they cannot rely solely on the patterns they have learned but must also draw on world knowledge or reasoning, for example. The test evaluates whether language models can understand and interpret language in the same way as humans. The test is an alternative to the Turing test, which tests whether a machine can appear human in conversation (Levesque, Davis, & Morgenstern, 2012).

The evaluation metrics for the fine-tuning tasks were tailored to the specific objectives of the respective datasets (see Table 5.1) from L. Charpentier et al. (2025). For MNLI (Williams et al., 2017), a multi-classification accuracy with three classes is used. Binary accuracy is used for BoolQ (Clark et al., 2019), MultiRC (Khashabi et al., 2018), and WSC (Levesque et al., 2012). For paraphrase recognition tasks such as MRPC (Dolan & Brockett, 2005) and QQP, the F1 score is used (L. Charpentier et al., 2025).

5.5. Summary

This chapter outlined the experiments for evaluating the RecombiText Augmentation (RTA) method in low-resource language modeling. The experiments are intended as a contribution to the BabyLM Challenge 2025 (L. Charpentier et al., 2025). The setup introduced the 100M word budget restriction and uses the strict-small dataset from L. Charpentier et al. (2025) of approximately 10M words. The experiments involve four stages. First, data generation includes proportional data sampling, preprocessing, and the data augmentation strategy. Second, the data quality is evaluated via perplexity and

[4] https://www.quora.com/profile/Ricky-Riche-2/First-Quora-Dataset-Release-Question-Pairs

Self-BLEU. Third, GPT-2 (Radford et al., 2019) and RoBERTa (Robertson & Zaragoza, 2009) are pre-trained with BPE tokenization and a custom checkpointing strategy. Fourth, the language model performance evaluation covers zero-shot tasks, model correlation with child language development, and fine-tuning on (Super)GLUE (Wang et al., 2018, 2019) subsets. The following Chapter 6 reports the quantitative and qualitative results.

6. Results and Discussion

The following chapter presents the results of the experiments described in Chapter 5 to evaluate the RecombiText Augmentation (RTA) method defined in Chapter 4. The quantitative results will be presented in Section 6.1, followed by the qualitative results in Section 6.2. Finally, the key findings from the results will be presented in Section 6.3, along with the limitations of the method in Section 6.4 and future work that could be investigated in Section 6.5. Finally, the summary provides a brief overview of the chapter.

6.1. Quantitative Results

The following discusses the quantitative results of the experiments described in Chapter 5. First, the data quality evaluation results are outlined, specifically the Self-BLEU scores (Papineni et al., 2002; Zhu et al., 2018) and perplexity (PPL). Next, the results of the language models for the zero-shot, fast zero-shot at different checkpoints, and fine-tuning tasks are presented.

6.1.1. Data Quality Results

Table 6.1 shows the results for PPL and Self-BLEU (Papineni et al., 2002; Zhu et al., 2018) for the different dataset variants. The results obtained from the dataset $\mathcal{D}_{baseX}$ are compared to those derived from the dataset $\mathcal{D}_{augY}$.

Perplexity The PPL results for the base datasets show a mean μ of 17.83 and a standard deviation σ of 0.21. The PPL results for the augmented datasets show a mean μ of 53.09 and a standard deviation σ of 4.11. The augmented datasets yield a value approximately three times higher than the base datasets, indicating that the augmented sentences do not always adhere to the predictable underlying linguistic patterns. However, it is worth noting that augmented data always comes with a certain degree of higher PPL, and the results show that the augmented data still ensures a certain level of predictability for the language models (≤ 100).

Self-BLEU The Self-BLEU (Papineni et al., 2002; Zhu et al., 2018) for the base datasets show a mean μ of 8.41% and a standard deviation σ of 0.31%. The Self-BLEU (Papineni et al., 2002; Zhu et al., 2018) for the augmented datasets show a mean μ of 14.57% and a standard deviation σ of 4.45%. A slight increase is observed in the augmented datasets, indicating that the data retains a certain level of diversity and accurately reflects the original data. Only in the $\mathcal{D}_{aug1M}$ dataset, an outlier of 22.67% is seen, which suggests

that the hyperparameters for top-k and temperature T (see Chapter 5) were chosen too strictly within this dataset to generate diverse examples.

	PPL		Self-BLEU	
Variant	D_{baseX}	D_{augY}	D_{baseX}	D_{augY}
$\mathcal{D}_{base1M} + \mathcal{D}_{aug9M}$	18.23	47.50	8.83	12.65
$\mathcal{D}_{base2.5M} + \mathcal{D}_{aug7.5M}$	17.69	49.84	8.67	12.56
$\mathcal{D}_{base5M} + \mathcal{D}_{aug5M}$	17.83	54.21	8.50	12.33
$\mathcal{D}_{base7.5M} + \mathcal{D}_{aug2.5M}$	17.81	55.05	8.02	18.33
$\mathcal{D}_{base9M} + \mathcal{D}_{aug1M}$	17.66	59.21	8.30	22.67
$\mathcal{D}_{base10M} + \mathcal{D}_{aug10M}$	17.75	53.51	8.19	11.74

Table 6.1.: Data quality evaluation for the proportion datasets $\mathcal{D}_{baseX}$ and the augmented datasets $\mathcal{D}_{augY}$ with their respective PPL and Self-BLEU (in %) values.

6.1.2. Zero-shot Evaluation Results

The following discusses the quantitative zero-shot and combined fine-tuning results of the experiments described in Chapter 5. First, the results from the decoder-based causal language model (CLM) will be presented, followed by those from the encoder-based masked language model (MLM).

GPT-2 The evaluation of the GPT-2 (Radford et al., 2019) variants shows subtle differences in performance trends across multiple benchmarks. Table 6.2 shows the zero-shot evaluation results combined with the fine-tuning results from the macro-average accuracy. The variant $\mathcal{D}_{base5M} + \mathcal{D}_{aug5M}$ achieves the highest average performance of 31.64 and performs well compared to the baseline on BLiMP Supplement 61.08 (Warstadt et al., 2025), EWoK 50.80 (Ivanova et al., 2024), Self-paced Reading 4.74 Eye Tracking 12.05 (de Varda et al., 2024), while achieving the same GLUE score of 57.67 (Wang et al., 2018, 2019) as the baseline in the combined fine-tuning results. The variant $\mathcal{D}_{base7.5M} + \mathcal{D}_{aug2.5M}$ performs best in BLiMP with 62.91, and the variant $\mathcal{D}_{base9M} + \mathcal{D}_{aug1M}$ performs best in WUG Adj. (Weissweiler et al., 2023; Hofmann et al., 2025) with 0.71. The baseline remains competitive in COMPS (Misra et al., 2022) with 51.48 and is not outperformed by any of the variants here. It is seen that a better average baseline performance is achieved with only 50% of the original data. A slightly augmented-heavy (25/75) or balanced (50/50) data mixture yields the best results across most metrics.

RoBERTa The evaluation of RoBERTa variants reveals distinct performance trends across multiple benchmarks. Table 6.3 shows the zero-shot evaluation results combined with the fine-tuning results from the macro-average accuracy. The variant $\mathcal{D}_{base2.5M} + \mathcal{D}_{aug7.5M}$ achieves the highest average performance 32.29, excelling in Entity Tracking 38.89 (Kim & Schuster, 2023; L. Charpentier et al., 2025) and COMPS 51.25 (Misra et

		BLiMP				WUGs			Reading			
Variant	**Prop.**	**BLiMP**	**Suppl.**	**EWoK**	**ENT**	**Adj.**	**Past**	**COMPS**	**SPR**	**ET**	**GLUE**	**Avg.**
Baseline	0%	61.62	58.17	50.01	12.95	0.69	-0.06	**51.48**	4.01	11.83	**57.67**	30.84
$\mathcal{D}_{base1M} + \mathcal{D}_{aug9M}$	10%	58.30	57.42	49.81	16.24	0.47	-0.17	50.38	3.92	10.07	**57.67**	30.41
$\mathcal{D}_{base2.5M} + \mathcal{D}_{aug7.5M}$	25%	60.18	60.48	50.41	**19.65**	0.54	**0.24**	50.49	3.97	11.65	**57.67**	31.53
$\mathcal{D}_{base5M} + \mathcal{D}_{aug5M}$	50%	61.87	**61.08**	**50.80**	16.51	0.56	**0.24**	50.83	**4.74**	**12.05**	**57.67**	**31.64**
$\mathcal{D}_{base7.5M} + \mathcal{D}_{aug2.5M}$	75%	**62.91**	59.02	49.74	17.50	0.56	0.03	51.42	4.13	11.64	**57.67**	31.46
$\mathcal{D}_{base9M} + \mathcal{D}_{aug1M}$	90%	61.27	59.80	50.24	16.53	**0.71**	-0.05	50.84	4.39	11.80	**57.67**	31.32
$\mathcal{D}_{base10M} + \mathcal{D}_{aug10M}$	100%	61.99	59.11	50.20	16.19	0.54	0.07	51.22	4.32	12.02	**57.67**	31.33

Table 6.2.: GPT-2 evaluation results with downsampled size from baseline dataset (prop.), Blimp, Bimp Supplement (Suppl.), EWoK, Entity Tracking (ENT), COMPS as accuracy, GLUE subset as macro-average accuracy (in %), WUGs with Spearman's rank correlation coefficient ρ, Self-paced Reading (SPR), Eye Tracking (ET) with change in R^2. The best results are printed in bold for each task.

al., 2022), while also scoring competitively in WUGs Past 0.23 (Weissweiler et al., 2023; Hofmann et al., 2025). The variant $\mathcal{D}_{base1M} + \mathcal{D}_{aug9M}$ with 10% base proportion leads in BLiMP 57.06 (Warstadt et al., 2020) and Self-paced Reading 3.53 (de Varda et al., 2024). In contrast, the variant $\mathcal{D}_{base10M} + \mathcal{D}_{aug10M}$ with 100% proportion and trained with half of the epochs compared to the baseline ($\approx$ 5 epochs) performs best in BLiMP Supplement 53.10 (Warstadt et al., 2025) and WUG Adj. 0.72 (Weissweiler et al., 2023; Hofmann et al., 2025). The variant $\mathcal{D}_{base5M} + \mathcal{D}_{aug5M}$ achieves the highest GLUE score 62.29 (Wang et al., 2018, 2019) and strong Eye Tracking 11.02 (de Varda et al., 2024). It matches closely with the variant $\mathcal{D}_{base7.5M} + \mathcal{D}_{aug2.5M}$ in Eye Tracking 11.03 (de Varda et al., 2024). The baseline remains best in EWoK 51.01 (Ivanova et al., 2024). It is seen that a better baseline performance is achieved with only 25% of the original data. A slightly augmented-heavy (25/75) or balanced (50/50) data mixture yields the best results across most metrics.

		BLiMP				WUGs			Reading			
Variant	**Prop.**	**BLiMP**	**Suppl.**	**EWoK**	**ENT**	**Adj.**	**Past**	**COMPS**	**SPR**	**ET**	**GLUE**	**Avg.**
Baseline	0%	54.44	50.98	**51.01**	32.27	0.58	-0.09	50.34	3.08	10.06	61.27	31.39
$\mathcal{D}_{base1M} + \mathcal{D}_{aug9M}$	10%	**57.06**	52.33	50.17	28.41	0.52	-0.03	50.66	**3.53**	10.17	61.51	31.43
$\mathcal{D}_{base2.5M} + \mathcal{D}_{aug7.5M}$	25%	55.15	51.77	49.38	**38.89**	0.48	**0.23**	**51.25**	3.52	11.01	61.19	**32.29**
$\mathcal{D}_{base5M} + \mathcal{D}_{aug5M}$	50%	55.56	51.55	49.89	33.65	0.67	-0.03	50.93	3.50	**11.02**	**62.29**	31.90
$\mathcal{D}_{base7.5M} + \mathcal{D}_{aug2.5M}$	75%	55.76	52.72	49.92	29.02	0.68	-0.20	50.90	3.42	**11.03**	61.65	31.49
$\mathcal{D}_{base9M} + \mathcal{D}_{aug1M}$	90%	54.30	51.67	49.62	33.83	0.57	-0.15	50.85	2.93	10.25	62.11	31.60
$\mathcal{D}_{base10M} + \mathcal{D}_{aug10M}$	100%	54.39	**53.10**	49.46	29.70	**0.72**	-0.03	50.49	2.81	10.48	61.98	31.31

Table 6.3.: RoBERTa evaluation results with downsampled size from baseline dataset (prop.), Blimp, Bimp Supplement (Suppl.), EWoK, Entity Tracking (ENT), COMPS as accuracy, GLUE subset as macro-average accuracy (in %), WUGs with Spearman's rank correlation coefficient ρ, Self-paced Reading (SPR), Eye Tracking (ET) with change in R^2. The best results are printed in bold for each task.

6.1.3. Checkpoint Evaluation Results

The following describes the quantitative checkpoint results from the experiments in Chapter 5. First, the results on the Pearson correlation between model training and child language development will be presented, followed by the fast zero-shot results in the various training steps of the language models.

Age of Acquisition Table 6.4 shows the language model and child language development Age of Acquisition (AoA) (Chang & Bergen, 2022) for GPT-2 (Radford et al., 2019) and RoBERTa (Liu et al., 2019) along the training checkpoints. For GPT-2 (Radford et al., 2019), the baseline showed negligible correlation, while mixed datasets yield improvements. The variant $\mathcal{D}_{base7.5M} + \mathcal{D}_{aug2.5M}$ achieved the highest correlation of 0.265 ($p = 0.182$) and for the variant $\mathcal{D}_{base2.5M} + \mathcal{D}_{aug7.5M}$ a correlation of 0.233 ($p = 0.404$). Though none reached statistical significance $p < 0.05$. The RoBERTa (Liu et al., 2019) variants consistently show poor correlations and are at or near zero in most variants. One exception is the variant $\mathcal{D}_{base1M} + \mathcal{D}_{aug9M}$ with a negative value of -0.274 ($p = 0.656$). This indicates a limited correlation in the MLM language model and child language development. Overall, the baseline-heavier variant (75/25) shows some correlation in the CLM, suggesting the approach partially aligns with child language development.

	AoA	
Variant	**GPT-2**	**RoBERTa**
Baseline	-0.001 (p=0.997)	0.00
$\mathcal{D}_{base1M} + \mathcal{D}_{aug9M}$	0.120 (p=0.670)	-0.274 (p=0.656)
$\mathcal{D}_{base2.5M} + \mathcal{D}_{aug7.5M}$	0.233 (p=0.404)	0.00
$\mathcal{D}_{base5M} + \mathcal{D}_{aug5M}$	0.031 (p=0.888)	0.00
$\mathcal{D}_{base7.5M} + \mathcal{D}_{aug2.5M}$	0.265 (p=0.182)	0.00
$\mathcal{D}_{base9M} + \mathcal{D}_{aug1M}$	0.089 (p=0.718)	0.00
$\mathcal{D}_{base10M} + \mathcal{D}_{aug10M}$	0.204 (p=0.388)	0.00

Table 6.4.: Age of Acquisition results measured by Pearson correlation coefficients r_{xy} and p-values across the different GPT-2 and RoBERTa model variants.

GPT-2 fast zero-shot Figure 6.1 shows the fast zero-shot results of the individual training checkpoints as a line chart, with each point in the graph representing an evaluated training checkpoint. The different variants are overlaid within the tasks and compared with each other. There is an improvement compared to the baseline in the grammatical tasks such as Blimp (Warstadt et al., 2020) and Blimp Supplement (Warstadt et al., 2025), as well as in the reading comprehension tasks Eye Tracking and Self-paced Reading (de Varda et al., 2024) and WUG Past (Weissweiler et al., 2023; Hofmann et al., 2025). There is no significant improvement compared to the baseline in the EWOK task (Ivanova et al., 2024). For ENT (Kim & Schuster, 2023; L. Charpentier et al., 2025), all variants

are slightly above the baseline. For WUG Adj. (Weissweiler et al., 2023; Hofmann et al., 2025), only the $\mathcal{D}_{base9M} + \mathcal{D}_{aug1M}$ variant manages to end up slightly above the baseline. A balanced mix of baseline and synthetic data (50/50) often achieves the best overall performance in the results, which is consistent with the observations from the zero-shot evaluation results for CLMs.

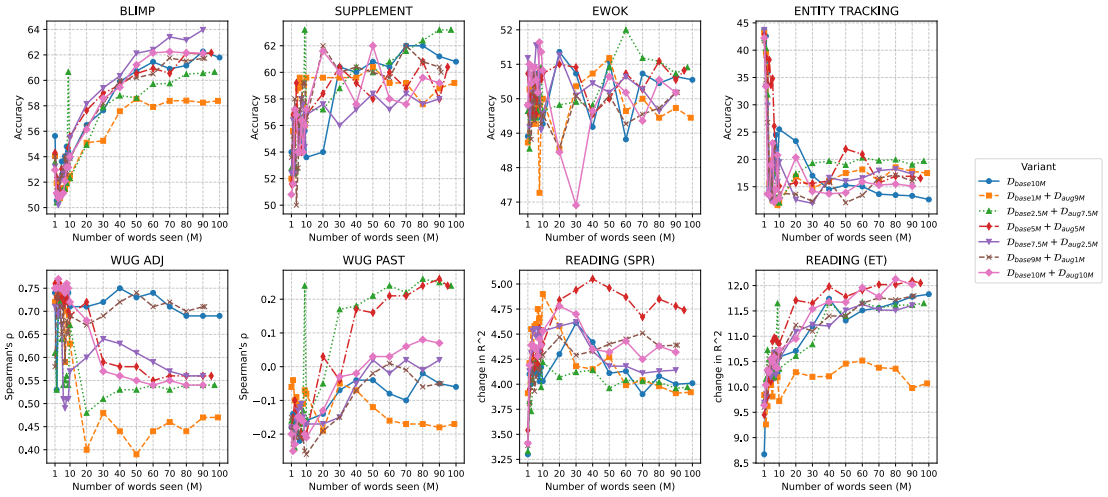

Figure 6.1.: GPT-2 fast zero-shot results for the different variants across various training checkpoints.

RoBERTa fast zero-shot Figure 6.2 shows the fast zero-shot results of the individual training checkpoints as a line chart, with each point in the graph representing an evaluated training checkpoint. The different variants are overlaid within the tasks and compared with each other. In the Blimp task (Warstadt et al., 2020), variant $\mathcal{D}_{base1M} + \mathcal{D}_{aug9M}$ shows the most significant improvement, with training peaks already reached at around 50M words. For Blimp Supplement (Warstadt et al., 2025), heavy-augmented variants dominate over the baseline, but show more fluctuations and moderate end values. ENT (Kim & Schuster, 2023; L. Charpentier et al., 2025) starts relatively high but drops sharply at around 20M words and then stabilizes again at a lower level. Most variants are below the baseline in ENT, but the $\mathcal{D}_{base2.5M} + \mathcal{D}_{aug7.5M}$ variant is the only one that is significantly above it at the end. For WUG Adj. (Weissweiler et al., 2023; Hofmann et al., 2025), the mixed variants with balanced (50/50) and higher baseline proportions (75/25) are considerably above the baseline towards the end. In WUG Past (Weissweiler et al., 2023; Hofmann et al., 2025), all variants correlate with the baseline except for the $\mathcal{D}_{base2.5M} + \mathcal{D}_{aug7.5M}$ variant, which shows significant improvements. In the reading tasks (de Varda et al., 2024), the variants with a heavy-augmented share are significantly above the baseline towards the end. The results show that an encoder-based MLM with a high proportion of augmented data (25/75) performs better than the baseline in entity tracking as well as grammatical and morphological understanding.

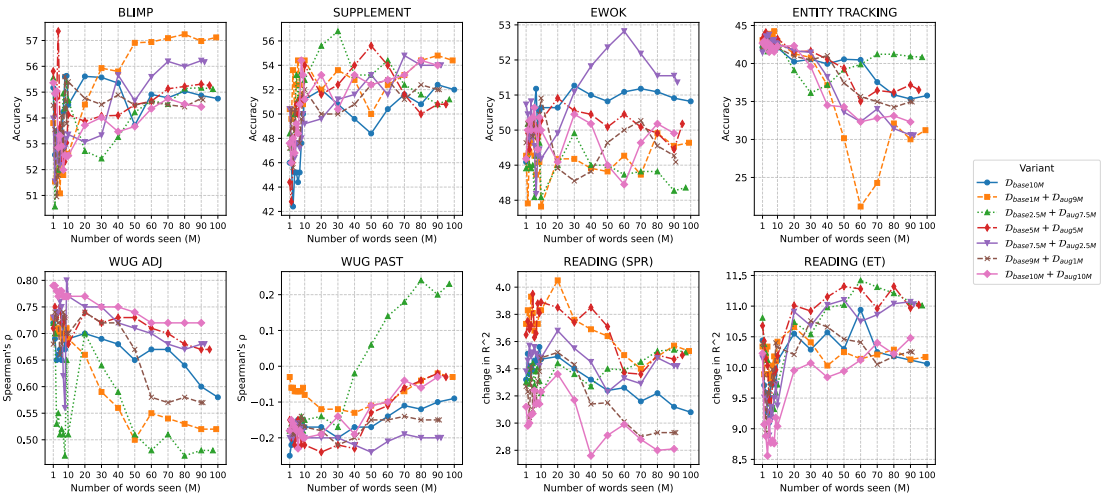

Figure 6.2.: RoBERTa fast zero-shot results for the different variants across various training checkpoints.

6.1.4. Fine-tuning Evaluation Results

The following describes the quantitative fine-tuning results on a subset of (Super)GLUE tasks (L. Charpentier et al., 2025; Wang et al., 2018, 2019) from the experiments in Chapter 5.

GPT-2 Table 6.5 shows the fine-tuning evaluation results for the CLM. The identical performance metrics across all dataset variants for the CLM within the official evaluation pipeline from L. Charpentier et al. (2025) show that the fixed seed values ensure deterministic behavior for CLM training.

Variant	Prop.	BoolQ	MNLI	MRPC	QQP	MultiRC	RTE	WSC	Avg.
Baseline	0%	64.04	35.70	81.05	62.78	57.55	53.96	61.54	59.51
$\mathcal{D}_{base1M} + \mathcal{D}_{aug9M}$	10%	64.04	35.70	81.05	62.78	57.55	53.96	61.54	59.51
$\mathcal{D}_{base2.5M} + \mathcal{D}_{aug7.5M}$	25%	64.04	35.70	81.05	62.78	57.55	53.96	61.54	59.51
$\mathcal{D}_{base5M} + \mathcal{D}_{aug5M}$	50%	64.04	35.70	81.05	62.78	57.55	53.96	61.54	59.51
$\mathcal{D}_{base7.5M} + \mathcal{D}_{aug2.5M}$	75%	64.04	35.70	81.05	62.78	57.55	53.96	61.54	59.51
$\mathcal{D}_{base9M} + \mathcal{D}_{aug1M}$	90%	64.04	35.70	81.05	62.78	57.55	53.96	61.54	59.51
$\mathcal{D}_{base10M} + \mathcal{D}_{aug10M}$	100%	64.04	35.70	81.05	62.78	57.55	53.96	61.54	59.51

Table 6.5.: GPT-2 fine-tuning results with downsampled size from baseline dataset (prop.). MNLI (Williams et al., 2017) uses multi-classification accuracy with three classes. BoolQ (Clark et al., 2019), MultiRC (Khashabi et al., 2018), RTE (Wang et al., 2018), and WSC (Levesque et al., 2012) use binary accuracy. MRPC (Dolan & Brockett, 2005) and QQP use F1 score (L. Charpentier et al., 2025) (in %).

RoBERTa Table 6.6 shows the fine-tuning evaluation results for the MLM. The baseline model achieves an average performance of 61.10 across seven tasks. Incorporating augmented data generally improved performance with the variant $\mathcal{D}_{base9M} + \mathcal{D}_{aug1M}$, yielding the highest average score of 62.30 with substantial improvements in RTE 58.99 (Dagan et al., 2005; Giampiccolo et al., 2007; Bentivogli et al., 2009) and WSC 67.31 (Levesque et al., 2012), which suggests that a small amount of augmented data enhances performance on specific tasks. The balanced variant $\mathcal{D}_{base5M} + \mathcal{D}_{aug5M}$ also performed better than the baseline with 62.08, especially in MultiRC 60.23 (Khashabi et al., 2018) and MNLI 43.42 (Williams et al., 2017). The augmented variant on the full dataset with ≈ 5 epochs achieved an average score of 61.63, which surpasses the baseline but shows declining stats in MRPC 80.75 (Dolan & Brockett, 2005) and QQP 58.47, but better results in all other tasks. Overall, a slight upward trend is seen in the variants trained with augmented data compared to the baseline in MNLI (Williams et al., 2017), MultiRC (Khashabi et al., 2018), RTE (Dagan et al., 2005; Giampiccolo et al., 2007; Bentivogli et al., 2009), and WSC (Levesque et al., 2012).

Variant	Prop.	BoolQ	MNLI	MRPC	QQP	MultiRC	RTE	WSC	Avg.
Baseline	0%	66.24	40.99	**83.39**	59.46	57.55	54.68	65.38	61.10
$\mathcal{D}_{base1M} + \mathcal{D}_{aug9M}$	10%	66.36	42.32	81.23	**61.18**	57.96	56.12	65.38	61.51
$\mathcal{D}_{base2.5M} + \mathcal{D}_{aug7.5M}$	25%	66.12	41.95	81.55	60.33	57.55	56.83	61.54	60.84
$\mathcal{D}_{base5M} + \mathcal{D}_{aug5M}$	50%	66.54	**43.42**	82.47	61.08	**60.23**	55.40	65.38	62.08
$\mathcal{D}_{base7.5M} + \mathcal{D}_{aug2.5M}$	75%	**67.22**	43.13	82.28	60.31	57.96	56.83	65.38	61.88
$\mathcal{D}_{base9M} + \mathcal{D}_{aug1M}$	90%	66.67	42.46	82.13	60.59	57.96	**58.99**	**67.31**	**62.30**
$\mathcal{D}_{base10M} + \mathcal{D}_{aug10M}$	100%	65.87	42.14	80.75	58.47	59.32	57.55	**67.31**	61.63

Table 6.6.: RoBERTa fine-tuning results with downsampled size from baseline dataset (prop.). MNLI (Williams et al., 2017) uses multi-classification accuracy with three classes. BoolQ (Clark et al., 2019), MultiRC (Khashabi et al., 2018), RTE (Wang et al., 2018), and WSC (Levesque et al., 2012) use binary accuracy. MRPC (Dolan & Brockett, 2005) and QQP use F1 score (L. Charpentier et al., 2025) (in %).

6.1.5. Training Effort

Table 6.7 shows the language model pre-training efforts. These were relatively low. The GPT-2 (Radford et al., 2019) models required approximately 1 hour per variant. RoBERTa (Liu et al., 2019), on the other hand, needed about 1.5 hours per variant. A total of 17.83 hours of training was performed for all experiments. RoBERTa's (Liu et al., 2019) slightly higher efforts reflect the bidirectional masking objective, which required more iterations for convergence compared to the autoregressive model GPT-2 (Radford et al., 2019). The language models were trained on 1x NVIDIA H100 Tensor Core GPU.

Variant	Training Effort		
	GPT-2	**RoBERTa**	**Total**
$\mathcal{D}_{base10M}$	0.93h	1.59h	2.52h
$\mathcal{D}_{base1M} + \mathcal{D}_{aug9M}$	1.08h	1.63h	2.72h
$\mathcal{D}_{base2.5M} + \mathcal{D}_{aug7.5M}$	1.06h	1.61h	2.68h
$\mathcal{D}_{base5M} + \mathcal{D}_{aug5M}$	0.91h	1.59h	2.50h
$\mathcal{D}_{base7.5M} + \mathcal{D}_{aug2.5M}$	1.03h	1.56h	2.59h
$\mathcal{D}_{base9M} + \mathcal{D}_{aug1M}$	0.89h	1.54h	2.44h
$\mathcal{D}_{base10M} + \mathcal{D}_{aug10M}$	0.88h	1.51h	2.39h

Table 6.7.: Language Model pre-training effort in hours.

6.2. Qualitative Results

The following discusses the qualitative results of the data generation experiments described in Chapter 5. Table 6.8 shows qualitative examples created using the RTA method, with the "Reference Sentences" column showing the two sentence pairs selected for augmentation. The first row represents the query sentence q, and the second the selected matching candidate sentence m. The "Augmented Sentences" column displays the two augmented variants $\tilde{q}, \tilde{m}$ resulting from the one-point crossover. We can see that the method is capable of generating synthetic variations by exchanging contextual elements between lexically and semantically similar sentences. For example, in the first example, the context window `"spring deepened into"` is taken from the query sentence and compared with the context window `"Days ran into"`, with the algorithm considering the two words `"deepened"` and `"ran"` to be the most valuable. The algorithm could also set the intersection at the lexically similar point `"into"`, but rejects the element here because the IDF of the pivot elements `"deepened"` and `"ran"` has a higher value. Augmented sentences such as `"My Missionary life has, on the whole, was very happy."` is a result of the algorithm not relying on grammatical annotations and presents issues in linguistic quality.

6.3. Key Findings

The findings of this study suggest that the proposed method is capable of augmenting language model pre-training datasets by using only a small portion of the original data. However, it does not necessarily enhance the quality of the data. It was shown that using augmented datasets does not harm the performance of language models, but rather partially improves it across various benchmarks. Augmented datasets exhibit a higher degree of PPL (53.09±4.11) and Self-BLEU (14.57%±4.45%) compared to the proportional baseline datasets $\mathcal{D}_{baseX}$, which indicates less predictability by OpenAI's GPT-2 (Radford et al., 2019) but retains linguistic diversity. For increasing the performance of the language

Reference Sentences	Augmented Sentences
Two months passed, and spring deepened into summer.	Two months passed, and spring ran into weeks, weeks into months, but the expected agent of deliverance was not forthcoming.
Days ran into weeks, weeks into months, but the expected agent of deliverance was not forthcoming.	Days deepened into summer.
Life on the whole was very happy.	Life on the whole, been a very happy one....'
My Missionary life has, on the whole, been a very happy one....'	My Missionary life has, on the whole was very happy.
I've already started something.	I've been spectating
Cos I've been spectating	Cos I've already started something.
It was very early on a hot December morning.	it was not very cold she took them out in the carriage.
On nice sunny days when it was not very cold she took them out in the carriage.	On nice sunny days when It was very early on a hot December morning.

Table 6.8.: Examples of RTA-generated augmented sentences based on a picked reference sentence and matching candidates with context window $W = 3$ and a minimum similarity threshold $\tau_{window} = 0.6$.

models, mixtures with a high proportion of augmented data (25/75) or a balanced ratio (50/50) achieve the best zero-shot and fine-tuning results. Checkpoint analyses show moderate correlations with child language development according to the AoA (Chang & Bergen, 2022) benchmark, but improve compared to the baseline. Overall, causal language models improve the strongest with the augmented data in morphological generalization (WUG Past), entity state tracking (ENT), reading (SPR), and grammatical understanding (BLiMP Suppl.) benchmarks. This underscores the potential of RTA for efficient data augmentation despite occasional issues in linguistic quality.

6.4. Limitations

In this study, the quality of the word and sentence embeddings was not evaluated across different dataset variants. The effectiveness of semantic search, which relies on these embeddings, directly affects the quality of the context window and the resulting augmented sentences. Poor or suboptimal embeddings can lead to less coherent or less contextually appropriate augmentations.

6.5. Future Work

Future work could focus on improving the quality of word and sentence embeddings when using RTA to ensure robust semantic alignment. Furthermore, a separate evaluation of the effects of lexical versus semantic augmentations and the presentation of the mixed training data (sequential, adjacent, and shuffled) to the language model could provide deeper insights into the individual contributions. A separate evaluation for lexical versus semantic augmentations would further contribute to the optimization of the RTA method by showing which type of augmentation leads to improvements. Investigating the effects of the used RTA hyperparameters (hyperparameter optimization) could enable more targeted data augmentation strategies for low-resource scenarios.

6.6. Summary

The RTA method was evaluated on data quality and low-resource language modeling using GPT-2 (Radford, Narasimhan, Salimans, Sutskever, et al., 2018) and RoBERTa (Liu et al., 2019). Quantitative results showed higher perplexity and Self-BLEU for augmented datasets, indicating diversity but reduced predictability by OpenAI's GPT-2 (Radford et al., 2018). Zero-shot results favored 25/75 or 50/50 mixtures for improving in morphological generalization (WUG Past), entity state tracking (ENT), reading (SPR), and grammatical understanding (BLiMP Suppl.) benchmarks. Checkpoint analyses revealed modest correlations with child language development. Fine-tuning on (Super)GLUE (Wang et al., 2018, 2019) subsets yielded no changes for GPT-2 (Radford et al., 2018) and consistent gains for RoBERTa (Liu et al., 2019). Training efforts were relatively low. Qualitative results generated coherent augmentations but occasional issues in linguistic quality. RTA doubles or quadruples data without language model performance losses. Limitations include unevaluated embeddings. Future work could explore data presentation methods and tune RTA hyperparameters. The following Chapter 7 briefly summarizes the whole study.

7. Conclusion

The RecombiText Augmentation (RTA) method, a fully non-neural compositional data augmentation method inspired by human language acquisition (Gentner, 1983; A. E. Goldberg, 2006; Tomasello, 2010; Gleitman, 1990), demonstrates that the proposed compositional data augmentation is capable of expanding language model pre-training datasets, thus increasing the language model performance in low-resource scenarios. The method addresses the identified research gap for data augmentation methods that don't use models trained with additional text resources. Mixtures with a high proportion of augmented data (25/75) or a balanced ratio (50/50) achieve the best language model performances for morphological generalization (WUG Past up to Δ 500%)[1], entity state tracking (ENT up to Δ 51.74%), reading (SPR up to Δ 18.20%), and grammatical understanding (BLiMP Suppl. up to Δ 5%) benchmarks. The available training data can be doubled or even quadrupled without compromising the performance of the language model. As shown, this even achieves better results for the benchmarks than with the original data alone. The results adhere to the core objective of the thesis to further close the data-efficiency gap between efficient human language acquisition and inefficient use of data for pre-training language models. The augmented data have a higher PPL and Self-BLEU value than the original data. Qualitative results show that RTA generates diverse but less predictable outputs by OpenAI's GPT-2 (Radford et al., 2019) that are nevertheless beneficial for language model training. Due to the omission of external resources, such as the annotation of grammar, the augmentations sometimes show issues in linguistic quality. While this study showcases the advantages of the method in low-resource language modeling, opportunities exist to refine embeddings for better semantic alignment and augmentation quality. Furthermore, future research could explore language model data presentation methods and tune RTA hyperparameters. Overall, the RTA method is capable of augmenting language model pre-training datasets in low-resource scenarios and leaves room for further improvement.

[1] Percentage change for best LM performance over the baseline (Δ)

Bibliography

Anaby-Tavor, A., Carmeli, B., Goldbraich, E., Kantor, A., Kour, G., Shlomov, S., ... Zwerdling, N. (2020, April). Do Not Have Enough Data? Deep Learning to the Rescue! *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(05), 7383–7390. Retrieved 2025-02-28, from https://ojs.aaai.org/index.php/AAAI/article/view/6233 doi: 10.1609/aaai.v34i05.6233

Andreas, J. (2020). Good-Enough Compositional Data Augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 7556–7566). Online: Association for Computational Linguistics. Retrieved 2025-02-28, from https://www.aclweb.org/anthology/2020.acl-main.676 doi: 10.18653/v1/2020.acl-main.676

Arora, S., Liang, Y., & Ma, T. (2017). A simple but tough-to-beat baseline for sentence embeddings. In *International conference on learning representations*.

Bentivogli, L., Clark, P., Dagan, I., & Giampiccolo, D. (2009). The fifth pascal recognizing textual entailment challenge. *TAC*, *7*(8), 1.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... others (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877–1901.

Chang, T. A., & Bergen, B. K. (2022, 01). Word acquisition in neural language models. *Transactions of the Association for Computational Linguistics*, *10*, 1-16. Retrieved from https://doi.org/10.1162/tacl_a_00444 doi: 10.1162/tacl_a_00444

Charpentier, L., Choshen, L., Cotterell, R., Gul, M. O., Hu, M., Jumelet, J., ... others (2025). Babylm turns 3: Call for papers for the 2025 babylm workshop. *arXiv preprint arXiv:2502.10645*.

Charpentier, L. G. G., & Samuel, D. (2024). Gpt or bert: why not both? *arXiv preprint arXiv:2410.24159*.

Chaudhary, A. (n.d.). *A Visual Survey of Data Augmentation in NLP*. Retrieved from https://amitness.com/posts/data-augmentation-for-nlp.html

Chen, J., Tam, D., Raffel, C., Bansal, M., & Yang, D. (2023). An empirical survey of data augmentation for limited data learning in nlp. *Transactions of the Association for Computational Linguistics*, *11*, 191–211.

Chen, J., Yang, Z., & Yang, D. (2020). MixText: Linguistically-Informed Interpolation of Hidden Space for Semi-Supervised Text Classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 2147–2157). Online: Association for Computational Linguistics. Retrieved 2025-02-28, from https://www.aclweb.org/anthology/2020.acl-main.194 doi: 10.18653/v1/2020.acl-main.194

Choshen, L., Cotterell, R., Hu, M. Y., Linzen, T., Mueller, A., Ross, C., ... Zhuang, C. (2024). [call for papers] the 2nd babylm challenge: Sample-efficient pretraining on a developmentally plausible corpus. *arXiv preprint arXiv:2404.06214*.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., & Toutanova, K. (2019). Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Naacl*.

Consortium, B., et al. (2007). British national corpus. *Oxford Text Archive Core Collection*.

Cormack, G. V., Clarke, C. L., & Buettcher, S. (2009). Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international acm sigir conference on research and development in information retrieval* (pp. 758–759).

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory*, *13*, 21-27. Retrieved from https://api.semanticscholar.org/CorpusID:5246200

Cunningham, P., & Delany, S. J. (2020). k-nearest neighbour classifiers: (with python examples). *arXiv preprint arXiv:2004.04523*.

Dagan, I., Glickman, O., & Magnini, B. (2005). The pascal recognising textual entailment challenge. In *Machine learning challenges workshop* (pp. 177–190).

Deisenroth, M. P., Faisal, A. A., & Ong, C. S. (2020). *Mathematics for machine learning*. Cambridge University Press.

de Varda, A. G., Marelli, M., & Amenta, S. (2024). Cloze probability, predictability ratings, and computational estimates for 205 english sentences, aligned with existing eeg and reading time data. *Behavior Research Methods*, *56*(5), 5190–5213.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)* (pp. 4171–4186).

De Winter, J. C., Gosling, S. D., & Potter, J. (2016). Comparing the pearson and spearman correlation coefficients across distributions and sample sizes: A tutorial using simulations and empirical data. *Psychological methods*, *21*(3), 273.

Dolan, B., & Brockett, C. (2005). Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (iwp2005)*.

Edman, L., Bylinina, L., Ghorbanpour, F., & Fraser, A. (2024, October). *Are BabyLMs Second Language Learners?* arXiv. Retrieved 2025-02-11, from http://arxiv.org/abs/2410.21254 (arXiv:2410.21254 [cs]) doi: 10.48550/arXiv.2410.21254

Eldan, R., & Li, Y. (2023). Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*.

Ethayarajh, K. (2018a). Unsupervised random walk sentence embeddings: A strong but simple baseline. In *Proceedings of the third workshop on representation learning for nlp* (pp. 91–100).

Ethayarajh, K. (2018b). Unsupervised random walk sentence embeddings: A strong but simple baseline. In *Proceedings of the third workshop on representation learning for*

nlp (pp. 91–100).

Feng, S. Y., Gangal, V., Wei, J., Chandar, S., Vosoughi, S., Mitamura, T., & Hovy, E. (2021). A survey of data augmentation approaches for nlp. *arXiv preprint arXiv:2105.03075*.

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., & Ruppin, E. (2002). Placing search in context: The concept revisited. *ACM Transactions on Information Systems (TOIS)*, *20*(1), 116–131.

Gentner, D. (1983, April). Structure-Mapping: A Theoretical Framework for Analogy*. *Cognitive Science*, *7*(2), 155–170. Retrieved 2025-02-17, from https://onlinelibrary.wiley.com/doi/10.1207/s15516709cog0702_3 doi: 10.1207/s15516709cog0702_3

Gerlach, M., & Font-Clos, F. (2020). A standardized project gutenberg corpus for statistical analysis of natural language and quantitative linguistics. *Entropy*, *22*(1), 126.

Giampiccolo, D., Magnini, B., Dagan, I., & Dolan, W. B. (2007). The third pascal recognizing textual entailment challenge. In *Proceedings of the acl-pascal workshop on textual entailment and paraphrasing* (pp. 1–9).

Gleitman, L. (1990). The structural sources of verb meanings. *Language acquisition*, *1*(1), 3–55.

Godfrey, J. J., Holliman, E. C., & McDaniel, J. (1992). Switchboard: Telephone speech corpus for research and development. In *Acoustics, speech, and signal processing, ieee international conference on* (Vol. 1, pp. 517–520).

Godoy, D. V. (2021). *dvgodoy/dl-visuals*. Retrieved from https://github.com/dvgodoy/dl-visuals

Goldberg, A. E. (2006). *Constructions at work: the nature of generalization in language*. Oxford: Oxford University Press. doi: 10.1093/acprof:oso/9780199268511.001.0001

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning* (1st ed.). USA: Addison-Wesley Longman Publishing Co., Inc.

Guo, H., Mao, Y., & Zhang, R. (2019, May). *Augmenting Data with Mixup for Sentence Classification: An Empirical Study*. arXiv. Retrieved 2025-02-28, from http://arxiv.org/abs/1905.08941 (arXiv:1905.08941 [cs]) doi: 10.48550/arXiv.1905.08941

Haga, A., Fukatsu, A., Oba, M., Bisazza, A., & Oseki, Y. (2024, November). *BabyLM Challenge: Exploring the Effect of Variation Sets on Language Model Training Efficiency*. arXiv. Retrieved 2025-02-10, from http://arxiv.org/abs/2411.09587 (arXiv:2411.09587 [cs]) doi: 10.48550/arXiv.2411.09587

Hedderich, M. A., Lange, L., Adel, H., Strötgen, J., & Klakow, D. (2020). A survey on recent approaches for natural language processing in low-resource scenarios. *arXiv preprint arXiv:2010.12309*.

Hill, F., Reichart, R., & Korhonen, A. (2015, December). SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, *41*(4), 665–695. Retrieved from https://aclanthology.org/J15-4004/ doi: 10.1162/COLI_a_00237

Hofmann, V., Weissweiler, L., Mortensen, D. R., Schütze, H., & Pierrehumbert, J. B.

(2025). Derivational morphology reveals analogical generalization in large language models. *Proceedings of the National Academy of Sciences*, *122*(19), e2423232122.

Hu, M. Y., Mueller, A., Ross, C., Williams, A., Linzen, T., Zhuang, C., ... Wilcox, E. G. (2024). Findings of the second babylm challenge: Sample-efficient pretraining on developmentally plausible corpora. *arXiv preprint arXiv:2412.05149*.

Ivanova, A. A., Sathe, A., Lipkin, B., Kumar, U., Radkani, S., Clark, T. H., ... others (2024). Elements of world knowledge (ewok): A cognition-inspired framework for evaluating basic world knowledge in language models. *arXiv preprint arXiv:2405.09605*.

Iwamoto, R., Yoshida, I., Kanayama, H., Ohko, T., & Muraoka, M. (2023). Incorporating syntactic knowledge into pre-trained language model using optimization for overcoming catastrophic forgetting. In *Findings of the association for computational linguistics: Emnlp 2023* (pp. 10981–10993).

Jia, R., & Liang, P. (2017). Adversarial Examples for Evaluating Reading Comprehension Systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (pp. 2021–2031). Copenhagen, Denmark: Association for Computational Linguistics. Retrieved 2025-02-28, from http://aclweb.org/anthology/D17-1215 doi: 10.18653/v1/D17-1215

Jindal, A., Ghosh Chowdhury, A., Didolkar, A., Jin, D., Sawhney, R., & Shah, R. R. (2020). Augmenting NLP models using Latent Feature Interpolations. In *Proceedings of the 28th International Conference on Computational Linguistics* (pp. 6931–6936). Barcelona, Spain (Online): International Committee on Computational Linguistics. Retrieved 2025-02-28, from https://www.aclweb.org/anthology/2020.coling-main.611 doi: 10.18653/v1/2020.coling-main.611

Jumelet, J., Hanna, M., de Heer Kloots, M., Langedijk, A., Pouw, C., & van der Wal, O. (2023, December). ChapGTP, ILLC's Attempt at Raising a BabyLM: Improving Data Efficiency by Automatic Task Formation. In A. Warstadt et al. (Eds.), *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning* (pp. 74–85). Singapore: Association for Computational Linguistics. Retrieved 2025-02-06, from https://aclanthology.org/2023.conll-babylm.6/ doi: 10.18653/v1/2023.conll-babylm.6

Jurafsky, D., & Martin, J. H. (2019). Speech and language processing 3rd edition draft. *October 2019*.

Khashabi, D., Chaturvedi, S., Roth, M., Upadhyay, S., & Roth, D. (2018). Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 conference of the north american chapter of the association for computational linguistics: Human language technologies, volume 1 (long papers)* (pp. 252–262).

Kim, N., & Schuster, S. (2023). Entity tracking in language models. *arXiv preprint arXiv:2305.02363*.

Kobayashi, S. (2018). Contextual Augmentation: Data Augmentation by Words with Paradigmatic Relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)* (pp. 452–457). New Orleans,

Louisiana: Association for Computational Linguistics. Retrieved 2025-02-28, from http://aclweb.org/anthology/N18-2072 doi: 10.18653/v1/N18-2072

Levesque, H. J., Davis, E., & Morgenstern, L. (2012). The winograd schema challenge. *KR*, *2012*(13th), 3.

Lison, P., & Tiedemann, J. (2016). Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *Proceedings of the tenth international conference on language resources and evaluation (lrec'16)* (pp. 923–929).

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lyman, A., & Hepner, B. (2024). Whatif: Leveraging word vectors for small-scale data augmentation. In *The 2nd babylm challenge at the 28th conference on computational natural language learning* (pp. 229–236).

MacWhinney, B. (2000). *The childes project: The database* (Vol. 2). Psychology Press.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, *38*(11), 39–41.

Misra, K., Rayz, J. T., & Ettinger, A. (2022). Comps: Conceptual minimal pair sentences for testing robust property knowledge and its inheritance in pre-trained language models. *arXiv preprint arXiv:2210.01963*.

Nagelkerke, N. J., et al. (1991). A note on a general definition of the coefficient of determination. *biometrika*, *78*(3), 691–692.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the association for computational linguistics* (pp. 311–318).

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)* (pp. 1532–1543).

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1*(8), 9.

Reddy, C. K., & Aggarwal, C. C. (Eds.). (2015). *Healthcare data analytics* (1st ed.). Chapman and Hall/CRC. doi: 10.1201/b18588

Robertson, S., & Zaragoza, H. (2009). The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends® in Information Retrieval*, *3*(4), 333–389. Retrieved 2025-07-19, from http://www.nowpublishers.com/article/Details/INR-019 (Publisher: Now Publishers) doi: 10.1561/1500000019

Savarimuthu, X., Subramani, S., & Raj, A. N. J. (2024). *Artificial intelligence for multimedia information processing*. Retrieved from https://doi.org/10.1201/9781003405436 doi: 10.1201/9781003405436

Sennrich, R., Haddow, B., & Birch, A. (2016). Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the*

Association for Computational Linguistics (Volume 1: Long Papers) (pp. 86–96). Berlin, Germany: Association for Computational Linguistics. Retrieved 2025-02-28, from http://aclweb.org/anthology/P16-1009 doi: 10.18653/v1/P16-1009

Stolcke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., ... Meteer, M. (2000). Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, *26*(3), 339–373.

Theodoropoulos, N., Filandrianos, G., Lyberatos, V., Lymperaiou, M., & Stamou, G. (2024, December). *BERTtime Stories: Investigating the Role of Synthetic Story Data in Language pre-training.* arXiv. Retrieved 2025-02-10, from http://arxiv.org/abs/2410.15365 (arXiv:2410.15365 [cs]) doi: 10.48550/arXiv.2410.15365

Timiryasov, I., & Tastet, J.-L. (2023). Baby llama: knowledge distillation from an ensemble of teachers trained on a small dataset with no performance penalty. *arXiv preprint arXiv:2308.02019*.

Tomasello, M. (Ed.). (2010). *Constructing a language: a usage-based theory of language acquisition*. Cambridge, Mass: Harvard University Press.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., ... Bowman, S. (2019). Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, *32*.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Warstadt, A., & Bowman, S. R. (2022). What artificial neural networks can tell us about human language acquisition. In *Algebraic structures in natural language* (pp. 17–60). CRC Press.

Warstadt, A., Mueller, A., Choshen, L., Wilcox, E., Zhuang, C., Ciro, J., ... others (2023). Proceedings of the babylm challenge at the 27th conference on computational natural language learning. In *Proceedings of the babylm challenge at the 27th conference on computational natural language learning*.

Warstadt, A., Mueller, A., Choshen, L., Wilcox, E., Zhuang, C., Ciro, J., ... others (2025). Findings of the babylm challenge: Sample-efficient pretraining on developmentally plausible corpora. *arXiv preprint arXiv:2504.08165*.

Warstadt, A., Parrish, A., Liu, H., Mohananey, A., Peng, W., Wang, S.-F., & Bowman, S. R. (2020, December). BLiMP: The Benchmark of Linguistic Minimal Pairs for English. *Transactions of the Association for Computational Linguistics*, *8*, 377–392. Retrieved 2024-12-03, from https://direct.mit.edu/tacl/article/96452 doi: 10.1162/tacl_a_00321

Wei, J., & Zou, K. (2019). EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 6381–

6387). Hong Kong, China: Association for Computational Linguistics. Retrieved 2025-02-28, from https://www.aclweb.org/anthology/D19-1670 doi: 10.18653/v1/D19-1670

Weir, R. (1962). *Language in the crib*. Mouton. Retrieved from https://books.google.at/books?id=sgINAQAAIAAJ

Weissweiler, L., Hofmann, V., Kantharuban, A., Cai, A., Dutt, R., Hengle, A., ... others (2023). Counting the bugs in chatgpt's wugs: A multilingual investigation into the morphological capabilities of a large language model. *arXiv preprint arXiv:2310.15113*.

Williams, A., Nangia, N., & Bowman, S. R. (2017). A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into deep learning*. Cambridge University Press. (https://D2L.ai)

Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Zhang, R., Yu, Y., & Zhang, C. (2020). SeqMix: Augmenting Active Sequence Labeling via Sequence Mixup. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 8566–8579). Online: Association for Computational Linguistics. Retrieved 2025-02-28, from https://www.aclweb.org/anthology/2020.emnlp-main.691 doi: 10.18653/v1/2020.emnlp-main.691

Zhang, Z., Yang, H., Ma, B., Rügamer, D., & Nie, E. (2023, December). Baby's CoThought: Leveraging Large Language Models for Enhanced Reasoning in Compact Models. In A. Warstadt et al. (Eds.), *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning* (pp. 158–170). Singapore: Association for Computational Linguistics. Retrieved 2025-02-06, from https://aclanthology.org/2023.conll-babylm.13/ doi: 10.18653/v1/2023.conll-babylm.13

Zhu, Y., Lu, S., Zheng, L., Guo, J., Zhang, W., Wang, J., & Yu, Y. (2018). Texygen: A benchmarking platform for text generation models. In *The 41st international acm sigir conference on research & development in information retrieval* (pp. 1097–1100).

Şahin, G. G. (2022, March). To Augment or Not to Augment? A Comparative Study on Text Augmentation Techniques for Low-Resource NLP. *Computational Linguistics*, *48*(1), 5–42. Retrieved 2025-02-26, from https://aclanthology.org/2022.cl-1.2/ (Place: Cambridge, MA Publisher: MIT Press) doi: 10.1162/coli_a_00425

Şahin, G. G., & Steedman, M. (2018). Data Augmentation via Dependency Tree Morphing for Low-Resource Languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (pp. 5004–5009). Brussels, Belgium: Association for Computational Linguistics. Retrieved 2025-02-28, from http://aclweb.org/anthology/D18-1545 doi: 10.18653/v1/D18-1545

A. Final Checkpoint Results Details

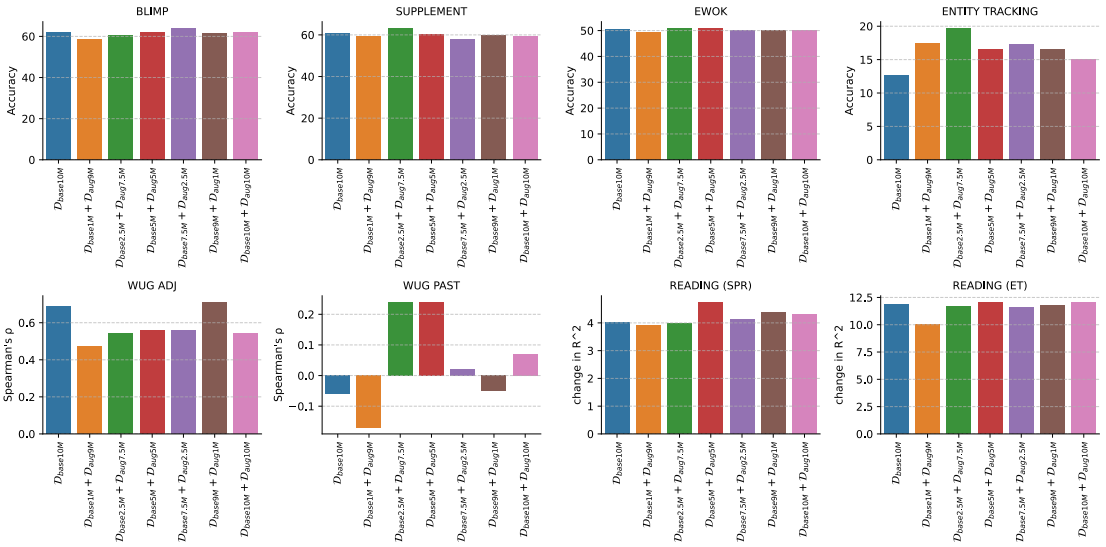

Figure A.1.: GPT-2 final fast zero-shot checkpoint results.

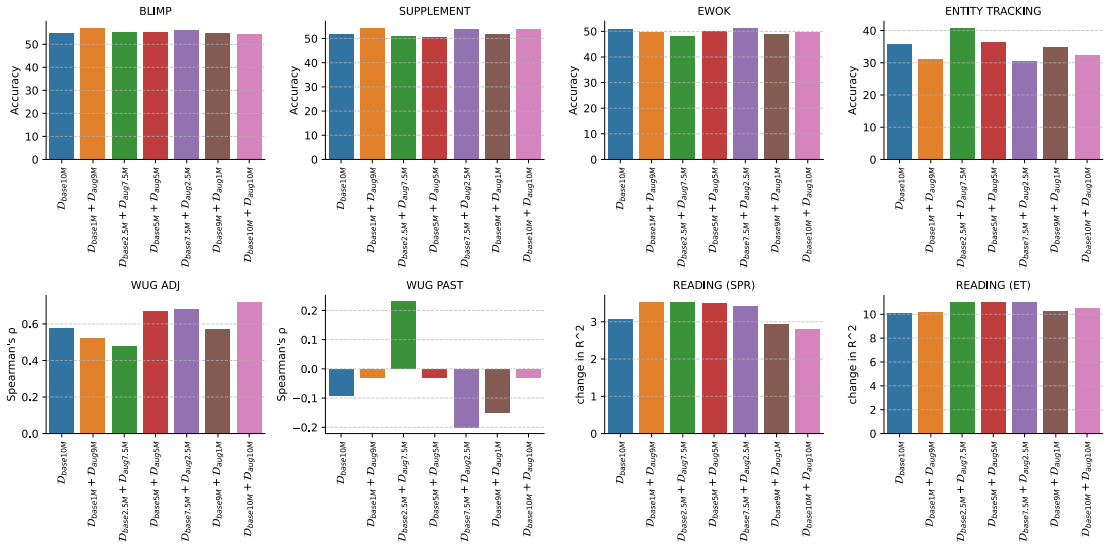

Figure A.2.: RoBERTa final fast zero-shot checkpoint results.

B. GPT-2 Checkpoint Results Details

Revision	BLiMP	BLiMP Sup.	EWoK	Entity Tracking	WUG Adj	WUG Past	Reading (SPR)	Reading (ET)	Avg.
1M	55.63	54.00	48.91	43.73	0.74	−0.18	3.30	8.67	26.85
2M	50.54	51.60	49.73	42.57	0.53	−0.14	4.10	9.81	26.09
3M	51.21	55.60	50.09	35.95	0.72	−0.17	4.00	9.81	25.90
4M	52.19	56.40	49.55	14.40	0.75	−0.19	4.12	10.14	23.42
5M	53.63	56.80	50.73	18.72	0.75	−0.20	4.13	10.12	24.34
6M	53.63	54.80	50.00	16.83	0.73	−0.22	4.12	10.24	23.77
7M	54.03	56.00	50.73	19.72	0.72	−0.21	4.10	10.17	24.41
8M	54.80	55.20	49.82	24.50	0.72	−0.21	4.03	10.30	24.90
9M	54.39	55.60	49.45	19.92	0.74	−0.17	4.17	10.28	24.30
10M	53.92	53.60	49.27	25.52	0.71	−0.16	4.03	10.59	24.69
20M	56.49	54.00	51.36	23.35	0.71	−0.14	4.30	10.71	25.10
30M	57.64	60.40	50.73	17.02	0.72	−0.07	4.61	11.20	25.28
40M	59.90	60.00	49.18	14.50	0.75	−0.04	4.42	11.74	25.06
50M	60.73	60.80	51.09	15.29	0.73	−0.04	4.11	11.31	25.50
60M	61.45	60.40	48.82	15.03	0.74	−0.08	4.13	11.51	25.25
70M	60.91	62.00	50.73	13.64	0.71	−0.10	3.90	11.56	25.42
80M	61.16	62.00	50.45	13.47	0.69	−0.02	4.08	11.66	25.44
90M	62.27	61.20	50.64	13.33	0.69	−0.05	4.00	11.78	25.48
100M	61.79	60.80	50.55	12.66	0.69	−0.06	4.01	11.83	25.28

Table B.1.: Results from fast zero-shot evaluation for the baseline dataset pre-trained with GPT-2 until it has seen 100 million whitespace-separated words.

Revision	BLiMP	BLiMP Sup.	EWoK	Entity Tracking	WUG Adj	WUG Past	Reading (SPR)	Reading (ET)	Avg.
1M	54.07	52.00	48.73	43.21	0.72	−0.06	3.91	9.84	26.55
2M	51.93	55.60	50.27	39.55	0.73	−0.04	4.21	9.26	26.44
3M	51.51	56.80	50.27	36.53	0.74	−0.10	4.55	9.62	26.24
4M	50.74	55.60	50.00	15.60	0.73	−0.09	4.25	10.23	23.38
5M	51.40	58.80	49.27	19.87	0.74	−0.11	4.58	10.04	24.32
6M	51.55	59.60	50.27	12.47	0.71	−0.12	4.60	9.81	23.61
7M	51.46	59.20	49.73	12.53	0.59	−0.15	4.75	10.52	23.58
8M	51.87	57.60	47.27	12.14	0.70	−0.11	4.66	10.19	23.04
9M	52.27	56.00	49.55	11.62	0.67	−0.07	4.54	10.19	23.10
10M	52.48	59.60	50.00	12.13	0.63	−0.08	4.90	9.72	23.67
20M	55.10	59.60	48.55	16.87	0.40	−0.19	4.58	10.29	24.40
30M	55.25	59.60	50.36	14.73	0.48	−0.05	4.18	10.20	24.34
40M	57.57	59.60	50.73	16.07	0.44	−0.07	4.15	10.21	24.84
50M	58.54	60.40	51.18	17.50	0.39	−0.12	4.27	10.46	25.33
60M	57.91	59.20	49.64	18.16	0.44	−0.16	3.99	10.52	24.96
70M	58.39	59.20	50.00	16.08	0.46	−0.17	4.04	10.38	24.80
80M	58.41	57.60	49.45	18.56	0.44	−0.17	3.98	10.36	24.83
90M	58.25	58.80	49.73	17.82	0.47	−0.18	3.91	9.98	24.85
99M	58.39	59.20	49.45	17.50	0.47	−0.17	3.92	10.07	24.85

Table B.2.: Results from fast zero-shot evaluation for the dataset $\mathcal{D}_{base1M} + \mathcal{D}_{aug9M}$ pre-trained with GPT-2 until it has seen 99 million whitespace-separated words.

Revision	BLiMP	BLiMP Sup.	EWoK	Entity Tracking	WUG Adj	WUG Past	Reading (SPR)	Reading (ET)	Avg.
1M	53.57	52.80	49.64	42.17	0.61	−0.16	3.33	9.73	26.46
2M	50.40	52.40	48.55	40.37	0.53	−0.19	3.83	10.15	25.76
3M	50.68	52.40	49.45	31.88	0.69	−0.24	3.73	10.73	24.92
4M	51.13	52.80	49.45	15.52	0.72	−0.18	4.04	10.54	23.00
5M	51.27	56.80	50.00	15.77	0.64	−0.16	4.17	10.28	23.60
6M	51.50	56.80	49.45	12.76	0.54	−0.17	4.24	10.50	23.20
7M	51.49	56.00	49.64	13.71	0.55	−0.19	4.18	10.20	23.20
8M	51.94	55.20	50.18	17.74	0.56	−0.13	4.33	10.70	23.82
9M	60.66	63.20	50.91	19.70	0.54	0.24	3.97	11.65	26.36
10M	52.31	57.20	49.64	12.07	0.67	−0.16	4.22	10.36	23.29
20M	54.93	57.20	49.82	17.35	0.48	−0.05	4.07	10.61	24.30
30M	57.90	58.80	49.91	19.31	0.51	0.17	4.12	10.84	25.20
40M	58.80	60.40	49.82	19.69	0.53	0.18	4.14	11.66	25.65
50M	58.63	60.00	50.91	18.97	0.53	0.21	3.96	11.40	25.58
60M	59.71	60.80	52.00	20.32	0.54	0.24	4.04	11.67	26.17
70M	59.76	61.60	51.18	19.77	0.53	0.22	4.03	11.56	26.08
80M	60.49	62.40	51.09	19.97	0.54	0.26	4.02	11.61	26.30
90M	60.56	63.20	50.73	19.03	0.54	0.25	3.96	11.62	26.24
97M	60.66	63.20	50.91	19.70	0.54	0.24	3.97	11.65	26.36

Table B.3.: Results from fast zero-shot evaluation for the dataset $\mathcal{D}_{base2.5M} + \mathcal{D}_{aug7.5M}$ pre-trained with GPT-2 until it has seen 97 million whitespace-separated words.

Revision	BLiMP	BLiMP Sup.	EWoK	Entity Tracking	WUG Adj	WUG Past	Reading (SPR)	Reading (ET)	Avg.
1M	54.30	51.60	50.73	42.91	0.76	−0.18	3.54	9.45	26.64
2M	52.36	56.80	49.55	38.72	0.73	−0.15	3.95	9.74	26.46
3M	51.42	53.60	50.09	33.81	0.73	−0.10	4.17	9.82	25.44
4M	51.26	59.20	49.55	38.25	0.76	−0.13	4.33	10.18	26.68
5M	53.03	55.60	50.55	33.63	0.73	−0.16	4.16	10.11	25.96
6M	52.63	55.20	49.55	34.81	0.71	−0.15	4.28	10.90	25.99
7M	52.59	54.00	50.36	26.10	0.72	−0.20	4.20	10.97	24.84
8M	53.33	54.40	50.82	18.36	0.73	−0.21	4.56	10.95	24.12
9M	54.69	56.80	50.55	19.39	0.70	−0.20	4.33	10.89	24.64
10M	55.58	56.80	50.73	15.11	0.69	−0.20	4.40	10.85	24.24
20M	57.63	58.40	51.00	15.71	0.72	0.03	4.84	11.71	25.00
30M	58.99	60.40	50.91	15.62	0.59	−0.04	4.94	11.65	25.38
40M	59.62	59.20	49.55	16.06	0.58	0.17	5.05	11.98	25.28
50M	60.54	58.00	50.00	21.91	0.58	0.16	4.96	11.78	25.99
60M	60.94	60.00	50.73	20.94	0.55	0.21	4.87	11.91	26.27
70M	60.55	58.80	50.27	15.69	0.56	0.21	4.67	12.02	25.35
80M	62.15	60.80	51.09	16.92	0.56	0.24	4.85	12.02	26.08
90M	62.07	58.00	50.55	16.90	0.56	0.26	4.78	12.09	25.65
95M	62.16	60.40	50.82	16.54	0.56	0.24	4.74	12.05	25.94

Table B.4.: Results from fast zero-shot evaluation for the dataset $\mathcal{D}_{base5M} + \mathcal{D}_{aug5M}$ pre-trained with GPT-2 until it has seen 95 million whitespace-separated words.

Revision	BLiMP	BLiMP Sup.	EWoK	Entity Tracking	WUG Adj	WUG Past	Reading (SPR)	Reading (ET)	Avg.
1M	53.22	52.40	51.18	43.72	0.71	−0.18	3.40	9.69	26.77
2M	50.86	56.40	49.18	33.34	0.70	−0.21	4.13	9.81	25.53
3M	50.19	52.40	49.82	29.89	0.69	−0.18	4.31	10.26	24.67
4M	52.10	54.00	50.18	14.51	0.71	−0.16	4.53	10.34	23.28
5M	52.37	54.00	50.91	12.53	0.67	−0.13	4.55	10.19	23.14
6M	52.69	55.60	51.55	19.16	0.51	−0.12	4.47	10.37	24.28
7M	52.66	58.00	51.55	21.46	0.49	−0.11	4.55	10.68	24.91
8M	53.25	57.60	50.73	17.98	0.51	−0.15	4.31	10.23	24.31
9M	54.13	54.40	49.09	15.91	0.51	−0.17	4.48	10.69	23.63
10M	55.51	56.80	49.55	19.48	0.57	−0.17	4.53	10.55	24.60
20M	58.15	57.60	51.27	12.61	0.60	−0.17	4.58	11.09	24.47
30M	59.40	56.00	50.09	11.95	0.64	−0.15	4.62	11.23	24.22
40M	60.35	57.20	50.45	16.64	0.63	−0.05	4.33	11.20	25.09
50M	62.12	58.40	50.18	16.02	0.61	0.02	4.18	11.51	25.38
60M	62.40	57.20	50.64	16.56	0.59	−0.02	4.18	11.63	25.40
70M	63.41	58.40	50.27	17.96	0.57	0.02	4.11	11.52	25.78
80M	63.16	57.60	49.64	18.22	0.56	−0.01	4.13	11.51	25.60
90M	63.97	58.00	50.18	17.34	0.56	0.02	4.14	11.61	25.73

Table B.5.: Results from fast zero-shot evaluation for the dataset $\mathcal{D}_{base7.5M} + \mathcal{D}_{aug2.5M}$ pre-trained with GPT-2 until it has seen 90 million whitespace-separated words.

Revision	BLiMP	BLiMP Sup.	EWoK	Entity Tracking	WUG Adj	WUG Past	Reading (SPR)	Reading (ET)	Avg.
1M	54.01	53.60	49.82	41.72	0.58	−0.15	3.39	9.60	26.57
2M	51.30	55.20	49.00	34.55	0.73	−0.23	3.81	9.87	25.53
3M	50.75	58.00	48.82	26.76	0.75	−0.21	4.04	10.33	24.91
4M	52.24	50.00	50.73	17.76	0.71	−0.21	4.22	10.40	23.23
5M	52.52	52.80	49.36	16.44	0.65	−0.16	3.93	10.51	23.26
6M	53.05	57.20	50.73	12.97	0.71	−0.21	4.07	10.41	23.62
7M	54.04	58.00	50.18	22.73	0.59	−0.20	4.16	10.55	25.01
8M	52.57	59.20	49.82	20.77	0.68	−0.21	4.13	10.67	24.70
9M	53.54	59.20	49.27	17.82	0.65	−0.25	4.28	10.61	24.39
10M	53.87	56.40	49.55	13.78	0.69	−0.26	4.29	10.61	23.62
20M	56.36	62.00	48.55	13.62	0.67	−0.19	4.47	11.22	24.59
30M	58.20	60.00	50.09	12.35	0.69	−0.15	4.29	11.10	24.57
40M	59.99	60.40	49.55	16.01	0.72	−0.07	4.33	11.40	25.29
50M	60.20	60.00	50.09	12.07	0.74	−0.02	4.40	11.40	24.86
60M	60.51	59.60	49.27	13.44	0.71	0.01	4.45	11.65	24.96
70M	61.76	62.00	49.55	16.59	0.72	−0.01	4.51	11.76	25.86
80M	61.51	60.80	49.73	17.15	0.70	−0.06	4.38	11.72	25.74
90M	61.73	60.40	50.18	16.04	0.71	−0.05	4.39	11.79	25.65
91M	61.73	60.00	50.18	16.48	0.71	−0.05	4.39	11.80	25.66

Table B.6.: Results from fast zero-shot evaluation for the dataset $\mathcal{D}_{base9M} + \mathcal{D}_{aug1M}$ pre-trained with GPT-2 until it has seen 91 million whitespace-separated words.

Revision	BLiMP	BLiMP Sup.	EWoK	Entity Tracking	WUG Adj	WUG Past	Reading (SPR)	Reading (ET)	Avg.
1M	52.96	50.80	49.82	42.24	0.75	−0.20	3.41	9.64	26.18
2M	51.14	51.60	51.00	33.36	0.75	−0.25	4.19	10.04	25.23
3M	50.75	57.20	50.45	13.69	0.77	−0.23	4.39	10.33	23.42
4M	50.81	56.80	50.91	13.46	0.75	−0.18	4.39	10.29	23.40
5M	51.00	54.00	50.91	18.96	0.75	−0.15	4.36	10.55	23.80
6M	51.15	54.00	50.55	22.98	0.75	−0.15	4.36	10.49	24.27
7M	52.09	56.40	50.18	12.17	0.74	−0.16	4.28	10.59	23.29
8M	53.17	54.00	51.64	14.50	0.76	−0.16	4.45	10.39	23.59
9M	53.16	56.00	51.36	20.77	0.75	−0.20	4.39	10.37	24.58
10M	53.99	57.20	50.82	12.87	0.72	−0.21	4.53	10.73	23.83
20M	56.14	61.60	48.45	20.35	0.68	−0.13	4.78	10.95	25.35
30M	58.57	60.00	46.91	14.16	0.57	−0.03	4.70	11.54	24.55
40M	59.46	57.60	49.64	13.71	0.56	−0.02	4.35	11.68	24.62
50M	61.22	62.00	50.64	13.89	0.55	0.03	4.32	11.67	25.54
60M	62.16	58.00	50.18	15.91	0.54	0.03	4.43	11.96	25.40
70M	62.25	57.60	49.36	15.31	0.55	0.06	4.25	11.78	25.15
80M	62.16	59.60	50.55	15.49	0.54	0.08	4.38	12.13	25.62
90M	62.13	59.20	50.18	15.12	0.54	0.07	4.32	12.02	25.45

Table B.7.: Results from fast zero-shot evaluation for the dataset $\mathcal{D}_{base10M} + \mathcal{D}_{aug10M}$ pre-trained with GPT-2 until it has seen 90 million whitespace-separated words.

C. RoBERTa Checkpoint Results Details

Revision	BLiMP	BLiMP Sup.	EWoK	Entity Tracking	WUG Adj	WUG Past	Reading (SPR)	Reading (ET)	Avg.
1M	55.17	46.00	49.09	42.11	0.72	−0.25	3.32	10.44	25.83
2M	52.57	46.00	49.27	43.03	0.69	−0.22	3.51	9.71	25.57
3M	53.49	42.40	49.73	42.40	0.65	−0.21	3.36	9.59	25.18
4M	53.21	45.20	49.82	42.52	0.66	−0.20	3.42	9.27	25.49
5M	52.58	45.20	50.09	42.34	0.67	−0.17	3.39	9.10	25.40
6M	52.84	44.40	50.64	42.38	0.65	−0.15	3.46	9.74	25.50
7M	54.24	45.20	51.18	42.46	0.73	−0.17	3.45	9.72	25.85
8M	55.60	47.60	49.91	42.71	0.67	−0.16	3.49	9.76	26.20
9M	55.63	50.00	50.55	42.50	0.68	−0.15	3.56	9.74	26.56
10M	54.51	51.20	50.64	42.08	0.69	−0.17	3.47	10.11	26.57
20M	55.61	52.00	50.64	40.24	0.70	−0.17	3.49	10.55	26.63
30M	55.57	50.80	51.27	40.55	0.69	−0.20	3.40	10.29	26.55
40M	55.35	49.60	51.00	39.95	0.68	−0.17	3.32	10.57	26.29
50M	53.72	48.40	50.82	40.55	0.65	−0.17	3.24	10.30	25.94
60M	54.91	50.40	51.09	40.47	0.67	−0.14	3.26	10.94	26.45
70M	54.78	51.60	51.18	37.55	0.67	−0.11	3.16	10.25	26.14
80M	55.04	50.80	51.09	35.88	0.64	−0.12	3.22	10.18	25.84
90M	54.87	52.40	50.91	35.33	0.60	−0.10	3.12	10.12	25.91
100M	54.75	52.00	50.82	35.78	0.58	−0.09	3.08	10.06	25.87

Table C.1.: Results from fast zero-shot evaluation for the RoBERTa baseline dataset pre-trained with RoBERTa until it has seen 100 million whitespace-separated words.

Revision	BLiMP	BLiMP Sup.	EWoK	Entity Tracking	WUG Adj	WUG Past	Reading (SPR)	Reading (ET)	Avg.
1M	53.80	50.40	49.27	43.08	0.73	−0.03	3.73	10.34	26.42
2M	51.54	49.60	47.91	42.26	0.73	−0.06	3.83	9.89	25.71
3M	51.77	53.60	49.36	43.46	0.70	−0.06	3.73	10.29	26.61
4M	53.04	50.00	49.64	42.83	0.71	−0.07	3.93	10.34	26.30
5M	51.08	52.40	50.18	43.06	0.70	−0.07	3.81	9.89	26.38
6M	52.86	54.40	49.27	43.91	0.70	−0.07	3.68	9.82	26.82
7M	51.79	52.40	48.73	43.73	0.69	−0.07	3.73	10.03	26.38
8M	53.01	51.60	49.82	44.25	0.70	−0.07	3.73	10.18	26.65
9M	52.59	50.80	49.09	42.82	0.71	−0.06	3.83	10.10	26.24
10M	52.61	54.40	47.82	42.91	0.69	−0.08	3.83	10.42	26.58
20M	54.52	52.00	49.18	41.28	0.66	−0.12	4.05	10.66	26.53
30M	55.93	53.60	49.18	40.79	0.59	−0.12	3.76	10.41	26.77
40M	55.80	52.80	48.91	37.15	0.56	−0.13	3.69	10.03	26.10
50M	56.91	50.00	48.82	30.16	0.50	−0.11	3.64	10.25	25.02
60M	56.94	52.40	49.27	21.17	0.55	−0.10	3.50	10.14	24.23
70M	57.09	53.20	48.73	24.28	0.54	−0.07	3.40	10.21	24.67
80M	57.24	54.40	49.91	32.11	0.53	−0.04	3.49	10.29	25.99
90M	56.98	54.80	49.55	30.02	0.52	−0.02	3.57	10.13	25.69
99M	57.12	54.40	49.64	31.22	0.52	−0.03	3.53	10.17	25.82

Table C.2.: Results from fast zero-shot evaluation for the dataset $\mathcal{D}_{base1M} + \mathcal{D}_{aug9M}$ pre-trained with RoBERTa until it has seen 99 million whitespace-separated words.

Revision	BLiMP	BLiMP Sup.	EWoK	Entity Tracking	WUG Adj	WUG Past	Reading (SPR)	Reading (ET)	Avg.
1M	55.56	48.40	48.91	42.48	0.72	−0.18	3.30	10.81	26.25
2M	50.57	47.20	50.18	43.41	0.70	−0.16	3.40	10.38	25.71
3M	53.33	47.20	49.00	42.31	0.53	−0.18	3.29	10.24	25.72
4M	51.99	50.00	48.91	41.59	0.55	−0.16	3.33	10.12	25.79
5M	53.02	53.20	49.00	42.49	0.51	−0.16	3.44	9.41	26.36
6M	53.42	51.20	48.09	42.57	0.52	−0.16	3.28	9.26	26.02
7M	54.95	53.20	49.27	41.99	0.51	−0.15	3.38	9.34	26.56
8M	54.37	51.20	49.82	42.33	0.47	−0.14	3.41	9.32	26.35
9M	55.40	51.20	49.64	42.22	0.65	−0.18	3.31	9.59	26.48
10M	54.70	52.80	48.09	42.84	0.51	−0.15	3.22	9.72	26.47
20M	52.72	55.60	49.00	39.11	0.70	−0.14	3.44	10.74	26.40
30M	52.43	56.80	49.91	36.12	0.64	−0.17	3.36	10.54	26.20
40M	53.26	53.20	49.00	37.18	0.59	−0.02	3.27	10.98	25.93
50M	54.22	53.20	49.00	39.19	0.51	0.06	3.40	11.02	26.33
60M	54.67	54.40	48.73	39.88	0.48	0.14	3.39	11.42	26.64
70M	54.54	52.40	48.82	41.21	0.51	0.18	3.45	11.31	26.55
80M	55.14	51.60	48.82	41.22	0.47	0.24	3.53	11.21	26.53
90M	55.17	50.80	48.27	40.93	0.48	0.20	3.54	11.07	26.31
97M	55.12	51.20	48.36	40.83	0.48	0.23	3.52	11.01	26.34

Table C.3.: Results from fast zero-shot evaluation for the dataset $\mathcal{D}_{base2.5M} + \mathcal{D}_{aug7.5M}$ pre-trained with RoBERTa until it has seen 97 million whitespace-separated words.

Revision	BLiMP	BLiMP Sup.	EWoK	Entity Tracking	WUG Adj	WUG Past	Reading (SPR)	Reading (ET)	Avg.
1M	55.81	44.40	49.18	43.25	0.71	−0.15	3.65	10.68	25.94
2M	54.10	42.80	49.91	43.76	0.75	−0.20	3.75	10.43	25.66
3M	54.75	46.00	49.45	44.11	0.73	−0.18	3.69	10.15	26.09
4M	57.36	48.80	50.00	42.68	0.74	−0.22	3.72	10.03	26.64
5M	53.63	50.80	50.27	42.82	0.75	−0.20	3.95	9.47	26.44
6M	52.37	50.00	50.00	42.49	0.72	−0.15	3.63	9.27	26.04
7M	52.55	51.20	50.09	42.38	0.72	−0.18	3.66	9.74	26.27
8M	52.31	51.20	49.73	42.63	0.73	−0.22	3.88	9.89	26.27
9M	53.04	51.20	49.91	42.58	0.70	−0.20	3.81	9.96	26.38
10M	54.13	54.00	50.27	43.24	0.68	−0.22	3.89	10.19	27.02
20M	53.87	51.60	50.91	41.55	0.74	−0.24	3.85	11.01	26.66
30M	54.08	52.40	50.55	41.63	0.72	−0.22	3.74	10.92	26.73
40M	54.10	54.00	50.45	40.55	0.73	−0.23	3.85	11.15	26.83
50M	54.51	55.60	50.09	39.37	0.73	−0.13	3.71	11.32	26.90
60M	54.65	54.00	50.45	35.02	0.71	−0.11	3.37	11.28	26.17
70M	55.13	51.60	50.09	36.45	0.70	−0.06	3.36	10.96	26.03
80M	55.22	50.00	49.91	36.20	0.68	−0.04	3.50	11.32	25.85
90M	55.31	50.80	49.45	37.18	0.67	−0.02	3.47	10.97	25.98
95M	55.26	50.80	50.18	36.52	0.67	−0.03	3.50	11.02	25.99

Table C.4.: Results from fast zero-shot evaluation for the dataset $\mathcal{D}_{base5M} + \mathcal{D}_{aug5M}$ pre-trained with RoBERTa until it has seen 95 million whitespace-separated words.

Revision	BLiMP	BLiMP Sup.	EWoK	Entity Tracking	WUG Adj	WUG Past	Reading (SPR)	Reading (ET)	Avg.
1M	53.98	50.40	50.73	41.41	0.73	−0.20	3.38	10.15	26.32
2M	51.54	47.60	50.45	42.60	0.73	−0.18	3.46	9.51	25.71
3M	53.29	48.00	49.73	43.58	0.74	−0.20	3.57	9.48	26.02
4M	52.29	46.40	50.82	43.36	0.74	−0.17	3.49	9.28	25.78
5M	52.18	47.60	49.82	43.80	0.76	−0.19	3.52	8.89	25.80
6M	53.34	47.60	49.55	42.94	0.75	−0.16	3.56	9.36	25.87
7M	53.37	47.20	48.18	42.82	0.62	−0.16	3.51	9.18	25.59
8M	52.84	48.00	49.27	43.10	0.56	−0.17	3.49	9.50	25.82
9M	53.20	50.00	49.45	42.66	0.80	−0.18	3.42	9.73	26.14
10M	53.36	49.20	49.18	42.59	0.77	−0.20	3.48	9.38	25.97
20M	53.07	49.60	49.91	41.69	0.75	−0.20	3.68	10.91	26.18
30M	53.33	51.20	51.09	41.45	0.75	−0.20	3.55	10.68	26.48
40M	55.66	51.60	51.82	38.24	0.72	−0.22	3.45	11.02	26.54
50M	54.67	53.20	52.36	33.60	0.71	−0.24	3.23	11.10	26.08
60M	55.59	51.60	52.82	32.33	0.70	−0.21	3.33	10.75	25.86
70M	56.19	54.80	52.18	34.02	0.68	−0.19	3.29	10.86	26.48
80M	55.99	54.00	51.55	31.43	0.67	−0.20	3.48	11.04	26.00
90M	56.22	54.00	51.55	30.59	0.68	−0.20	3.42	11.07	25.92
92M	56.16	54.00	51.36	30.55	0.68	−0.20	3.42	11.03	25.88

Table C.5.: Results from fast zero-shot evaluation for the dataset $\mathcal{D}_{base7.5M} + \mathcal{D}_{aug2.5M}$ pre-trained with RoBERTa until it has seen 92 million whitespace-separated words.

Revision	BLiMP	BLiMP Sup.	EWoK	Entity Tracking	WUG Adj	WUG Past	Reading (SPR)	Reading (ET)	Avg.
1M	54.49	47.20	49.64	42.17	0.68	−0.18	3.27	10.39	25.96
2M	53.55	50.40	50.18	42.27	0.74	−0.16	3.23	9.88	26.26
3M	50.95	45.60	50.09	42.79	0.73	−0.19	3.04	8.93	25.24
4M	51.79	50.40	50.09	43.02	0.72	−0.19	3.42	9.60	26.11
5M	52.97	48.80	49.82	42.50	0.66	−0.18	3.10	9.57	25.91
6M	54.10	48.80	50.27	41.99	0.68	−0.17	3.19	9.71	26.07
7M	52.69	47.60	50.09	42.08	0.74	−0.16	3.14	9.97	25.77
8M	55.60	49.60	50.09	42.11	0.73	−0.14	3.31	9.78	26.39
9M	53.78	51.60	50.45	42.11	0.70	−0.15	3.38	10.40	26.53
10M	55.40	52.00	50.91	42.10	0.70	−0.17	3.48	10.33	26.84
20M	54.76	50.00	48.91	40.94	0.74	−0.18	3.52	10.21	26.11
30M	54.52	50.00	48.55	40.48	0.72	−0.22	3.43	10.76	26.03
40M	54.89	50.80	48.82	41.03	0.72	−0.20	3.14	10.66	26.23
50M	54.50	52.40	49.64	37.43	0.67	−0.15	3.15	10.46	26.01
60M	54.62	52.80	50.00	35.65	0.58	−0.15	3.01	10.41	25.87
70M	54.51	51.20	50.27	34.97	0.57	−0.14	2.90	10.05	25.54
80M	54.43	52.40	49.55	34.23	0.58	−0.15	2.93	10.18	25.52
90M	54.76	52.00	49.27	34.97	0.57	−0.15	2.93	10.26	25.58
91M	54.70	52.00	49.09	34.94	0.57	−0.15	2.93	10.25	25.54

Table C.6.: Results from fast zero-shot evaluation for the dataset $\mathcal{D}_{base9M} + \mathcal{D}_{aug1M}$ pre-trained with RoBERTa until it has seen 91 million whitespace-separated words.

Revision	BLiMP	BLiMP Sup.	EWoK	Entity Tracking	WUG Adj	WUG Past	Reading (SPR)	Reading (ET)	Avg.
1M	55.36	47.60	49.18	42.68	0.79	−0.18	3.12	10.23	26.10
2M	54.94	46.00	50.00	42.94	0.79	−0.15	2.98	9.07	25.82
3M	55.04	48.00	50.27	42.20	0.78	−0.16	3.00	8.88	26.00
4M	52.85	46.80	50.00	42.77	0.77	−0.17	3.06	8.56	25.58
5M	53.31	49.20	50.36	41.51	0.78	−0.18	3.07	9.12	25.90
6M	52.90	48.40	50.64	42.35	0.78	−0.23	3.24	8.77	25.86
7M	51.99	50.80	50.00	41.99	0.77	−0.18	3.14	8.81	25.92
8M	52.56	54.40	49.45	41.54	0.77	−0.19	3.15	8.76	26.31
9M	52.51	53.20	50.36	42.01	0.77	−0.20	3.14	9.18	26.37
10M	52.54	51.20	50.00	42.27	0.77	−0.20	3.23	9.04	26.11
20M	53.72	53.20	49.09	42.32	0.77	−0.19	3.36	9.95	26.53
30M	54.02	50.40	50.45	39.59	0.75	−0.14	3.17	10.07	26.04
40M	53.47	53.20	50.18	34.50	0.75	−0.19	2.76	9.84	25.56
50M	53.67	52.40	49.00	34.29	0.74	−0.11	2.91	9.94	25.36
60M	54.36	52.80	48.45	32.33	0.72	−0.10	2.99	10.12	25.21
70M	54.75	53.20	49.64	32.81	0.72	−0.04	2.88	10.40	25.55
80M	54.54	54.40	50.18	33.10	0.72	−0.06	2.80	10.22	25.74
90M	54.43	54.00	49.91	32.31	0.72	−0.03	2.81	10.48	25.58

Table C.7.: Results from fast zero-shot evaluation for the dataset $\mathcal{D}_{base10M} + \mathcal{D}_{aug10M}$ pre-trained with RoBERTa until it has seen 90 million whitespace-separated words.