Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων
Εργαστηριο Συστηματων Τεχνητης Νοημοσυνης και Μαθησης

Leveraging Synthetic Story Data for Sample-Efficient Language Pre-training

Diploma Thesis

by

Nikitas Theodoropoulos

Επιβλέπων: Γεώργιος Στάμου
Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2024

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Συστημάτων Τεχνητής Νοημοσύνης και Μάθησης

Leveraging Synthetic Story Data for Sample-Efficient Language Pre-training

Diploma Thesis

by

Nikitas Theodoropoulos

Επιβλέπων: Γεώργιος Στάμου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 1η Νοεμβρίου, 2024.

.....	.....	.....
Γεώργιος Στάμου	Αθανάσιος Βουλόδημος	Α.-Γ. Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.	Επ. Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2024

.....

Νικητας Θεοδωροπουλος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός
και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Τα σύγχρονα γλωσσικά μοντέλα (ΓΜ) έχουν κατορθώσει αξιοσημείωτα επιτεύγματα γλωσσικής επάρκειας και κατανόησης, ξεπερνώντας και τις επιδόσεις των ανθρώπων σε ορισμένες εργασίες. Για να επιτύχουν επιδόσεις αιχμής, τα ΓΜ απαιτούν όλο και μεγαλύτερή αύξηση των υπολογιστικών πόρων και του μεγέθους των δεδομένων εκπαίδευσης τους, κάνοντας συχνά χρήση χιλιάδων καρτών γραφικών, και τρισεκατομμυρίων λέξεων. Εξετάζοντας τον τρόπου που μαθαίνουν να μιλάνε οι άνθρωποι, διαπιστώνουμε μια ξεκάθαρη αντίθεση. Τα παιδιά, για να αναπτύξου την ικανότητα της ομιλίας, εκτίθενται το πολύ σε εκατό εκατομμύρια λέξεις έως την ηλικία των 13 ετών, επιδεικνύοντας αξιοθαύμαστη αποτελεσματικότητα όσον αφορά τον αριθμό των απαιτούμενων δειγμάτων γλώσσας. Με κίνητρο αυτή την ασυμφωνία, διερευνούμε την προεκπαίδευση ΓΜ με περιορισμένο αριθμό λέξεων, προσομοιάζοντας το αναπτυξιακό περιβάλλον των παιδιών. Για τα πειράματά μας υιοθετούμε δύο σενάρια, ένα περιορισμένο σενάριο με μέγιστο τις *10 εκατομμύρια* λέξεις, και ένα μεγαλύτερο σενάριο με μέγιστο αριθμό τις *100 εκατομμύρια* λέξεις.

Η μέθοδός μας επικεντρώνεται στην *Επαύξηση Δεδομένων*. Χρησιμοποιούμε πρώτα το *TinyStories* – ένα σύνολο δεδομένων με απλές και σύντομες ιστορίες, κατανοητές από παιδιά 3-4 ετών – για την εκπαίδευση μοντέλων μετασχηματιστών αποκωδικοποιητών, ικανών να παράξουν κείμενο. Στη συνέχεια, εξετάζουμε τη γλωσσική επάρκεια των μοντέλων, καθώς και την ποιότητα και την ποικιλομορφία των παραγωγών τους, ενώσω μεταβάλλουμε το μέγεθος των δεδομένων στο σύνολο εκπαίδευσης τους. Η αξιολόγησή μας δείχνει ότι αυτά τα μοντέλα μπορούν να δημιουργήσουν πρωτότυπες και γραμματικά σωστές ιστορίες ακόμη και με μερικά εκατομμύρια δεδομένα εκπαίδευσης. Στη συνέχεια χρησιμοποιούμε τους αποκωδικοποιητές για να δημιουργήσουμε ένα σύνολο δεδομένων *συνθετικών* ιστοριών, το οποίο συνενώνουμε με ένα ευρύ σύνολο κειμένων για να εκπαιδεύσουμε μοντέλα με αρχιτεκτονική κωδικοποιητή μετασχηματιστή. Για να αξιολογήσουμε την επίδραση της τεχνικής επαύξησης δεδομένων, μετράμε τη γενική κατανόηση της γλώσσας, τη γραμματική και τη γνώση για τον κόσμο που διαθέτουν οι κωδικοποιητές και συγκρίνουμε με μια ευρεία συλλογή από μοντέλα, εκπαιδευμένα με ή χωρίς συνθετικά δεδομένα. Μέσω της μελέτης μας, διαπιστώνουμε ότι ενώ η επαύξηση συνθετικών δεδομένων μπορεί να είναι ωφέλιμη σε μερικές περιπτώσεις, συνολικά έχει αρνητική επίδραση στην τελική απόδοση. Τέλος, σημειώνουμε τους περιορισμούς της προσέγγισής μας και επισημαίνουμε ενδιαφέρουσες οδούς για μελλοντική εργασία.

Λέξεις-κλειδιά — Γλωσσικά Μοντέλα, Επαύξηση Δεδομένων, Γλωσσική Προεκπαίδευση, Μικρά Γλωσσικά Μοντέλα, Γνωστική Μοντελοποίηση, Παραγωγικά Μοντέλα, Παραγωγή Ιστοριών

Abstract

Modern Language Models (LMs) have shown remarkable feats of linguistic proficiency and language understanding, often surpassing the abilities of humans in certain tasks. To achieve state-of-the-art performance, LMs require ever-increasing computational resources and training data, up to thousands of GPUs and trillions of tokens. In stark contrast, children can acquire a language natively by being exposed to no more than a hundred million words by 13 years of age, showing incredible sample efficiency. Motivated by this discrepancy, we investigate LM pre-training with a limited word budget, similar in size to the input human children receive during their development. For our experiments we adopt two configurations, a reduced setting with a maximum of *10 million* words, and a larger setting with a maximum of *100 million* words.

Our method is rooted in *Data Augmentation*. We first use *TinyStories* – a dataset of sort and simple stories, typically understood by 3-4 year old children – to train generative decoder transformer models. We vary the amount of data in the training set of the decoders, and evaluate extensively their language proficiency as well as the quality and diversity of their generations. Our evaluation demonstrates that these models can generate novel and grammatically correct stories even with a few million words of pre-training data. Next, we use the generative models to create a *synthetic* story dataset, which we combine with a diverse corpus of texts to train encoder transformer models. To assess the effect of our data augmentation methodology we measure the general language understanding, grammatical knowledge and world knowledge of the encoder models, and compare with a variety of baselines, with and without the use of synthetic data. We find that while synthetic data augmentation can offer some modest gains, overall it has a negative effect in final linguistic performance. Finally, we underscore the limitations of our proposed approach, and highlight interesting avenues for future work.

Keywords — Language Models, Data Augmentation, Language Pre-training, Small Language Models, Cognitive Modelling, BabyLM Challenge, Generative Models, Story Generation

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, Γεώργιο Στάμου, για όλη την καθοδήγηση του στην πορεία αυτής της διπλωματικής. Ευχαριστώ επιπλέον τη Μαρία Λυμπεραίου, τον Γιώργο Φιλανδριανό, και τον Βασίλη Λυμπεράτο για την εξαιρετική συνεργασία, καθώς και όλη τη βοήθεια, τεχνική και ψυχολογική, που μου παρείχαν στη διάρκεια αυτής της προσπάθειας. Θα ήθελα επίσης να ευχαριστήσω τους καθηγητές μου, Αλέξανδρο Ποταμιάνο, Δημήτρη Φωτάκη, και Νίκο Παπασπύρου, για τις γνώσεις που μου μετάδωσαν στα χρόνια των σπουδών μου. Ευχαριστώ τον Ευθύμη Γεωργίου, και τον Γιώργο Παρασκευόπουλο, που με βοήθησαν να κάνω τα πρώτα βήματα στον χώρο του machine learning, καθώς και τους Νίκο Λούκα, και Δημήτρη Σωτηρίου, που ήταν μαζί μου στις δύσκολες στιγμές. Ευχαριστώ επίσης τον φίλο μου Στυλιανό Κασουρίδη καθώς και τους Ορέστη Γεώργιο Χαρδούβελη, Βασιλική Βασιλείου, Δημήτρη Καλτσογιάννη, Πέτρο Τζάθα, Λυκούργο Μαστόρου, Ιωάννη Αναγνωστίδη και πολλούς άλλους συμφοιτητές οι οποίοι ήταν συνοδοιπόροι στα φοιτητικά μου χρόνια και με βοηθήσαν να αντιμετωπίσω τις προκλήσεις των ακαδημαϊκών σπουδών. Ευχαριστώ ακόμα τον καθηγητή μου Adish Singla, για όσα με δίδαξε για την έρευνα, και τον Ευθύμιο Κάκουρο, που με βοήθησε να ξεπεράσω κάθε πρόβλημα. Ευχαριστώ τη σύντροφο μου Σάρα, που πάντα πιστεύει σε εμένα και στέκεται δίπλα μου – χωρίς την οποία η ζωή μου θα ήταν πολύ διαφορετική. Θα ήθελα να ευχαριστήσω τέλος τους γονείς μου, τον πατέρα μου Κώστα – που μου έμαθε να αγαπάω τα μαθηματικά και την επιστήμη – και τη μητέρα μου Βάσω – που μου ενέπνευσε την αγάπη για τις γλώσσες και την τέχνη – για την απεριόριστη αγάπη και στήριξη τους σε όλα τα χρόνια της ζωής μου. Κλείνω αυτήν την παράγραφο παραθέτοντας ένα απόσπασμα από την αυτοβιογραφία του Bertrand Russell [115]:

> *"... With equal passion I have sought knowledge. I have wished to understand the hearts of men. I have wished to know why the stars shine. And I have tried to apprehend the Pythagorean power by which number holds sway above the flux. ..."*

Νικήτας Θεοδωρόπουλος, Νοέμβριος 2024

Contents

List of Figures

List of Tables

Εκτεταμένη Περίληψη στα Ελληνικά

0.1 Εισαγωγή

Τα σύγχρονα Γλωσσικά Μοντέλα (Language Models), έχουν καταφέρει αξιοσημείωτα επιτεύγματα γλωσσικής κατανόησης και συλλογισμού, ξεπερνώντας πολλές φορές τις ικανότητες του ανθρώπου για ορισμένες εργασίες. Η επιτυχία αυτή οφείλεται σε μεγάλο βαθμό στη συνεχή αύξηση της κλίμακας δύο κρίσιμων μεταβλητών: του αριθμού παραμέτρων των μοντέλων, και του αριθμού των δεδομένων εκπαίδευσης. Πρόσφατες μελέτες [50, 65]. έχουν δείξει ότι όσο αυξάνονται οι τιμές αυτών των μεταβλητών, συνεχίζει να βελτιώνεται η επίδοση και οι γλωσσικές ικανότητες αυτών των μοντέλων. Για τον λόγο αυτό, τα σύγχρονα Μεγάλα Γλωσσικά Μοντέλα φτάνουν σε μέγεθος δισεκατομμύρια παραμέτρους, χρησιμοποιώντας τρισεκατομμύρια λεκτικά μέρη (tokens). Ταυτόχρονα οι υπολογιστικοί πόροι που απαιτούνται για την εκπαίδευση αυτών των μοντέλων αυξάνονται σημαντικά. Για παράδειγμα, το μοντέλο Llama 3 [30], που κυκλοφόρησε πρόσφατα, διαθέτει 405 δισεκατομμύρια παραμέτρους και έχει προεκπαιδευτεί σε 15.6 τρισεκατομμύρια tokens, κάνοντας χρήση 6 χιλιάδων H100 κάρτες γραφικών (GPU). Αυτές οι συνθήκες δημιουργούν πλήθος προβλημάτων, οικολογικής και οικονομικής φύσεως, αλλά και έλλειψη δημοκρατίας στην πρόσβαση στην ερεύνα, αφού στον τομέα της γλωσσικής μοντελοποίησης αυτή είναι εφικτή μόνο για μεγάλους οργανισμούς της βιομηχανίας της τεχνητής νοημοσύνης.

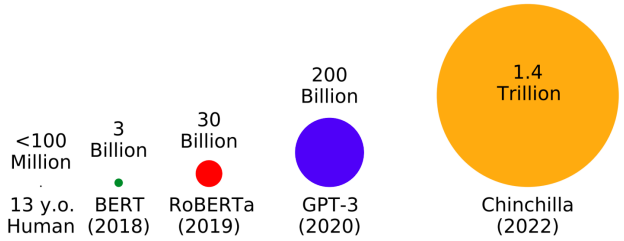

Figure 0.1.1: Μέγεθος των συνόλων δεδομένων που χρησιμοποιούν τα σύγχρονα γλωσσικά μοντέλα σε λέξεις, σε σύγκριση με των αριθμό των λέξεων στον οποίο έχει εκτεθεί ένα παιδί 13 ετών. Εικόνα: [145, 141].

Παρατηρώντας τον τρόπο που οι άνθρωποι μαθαίνουν να μιλάνε και συγκρίνοντας τον με τις σύγχρονες τεχνικές εκπαίδευσης γλωσσικών μοντέλων, βρίσκουμε σημαντικές διαφορές. Τα παιδιά κατά την ανάπτυξη τους εκτίθενται σε 2-7 εκατομμύρια (M) λέξεις την ημέρα [42], το οποίο συνολικά ισοδυναμεί σε λιγότερο από 100M λέξεις σε ηλικία 13 ετών. Αυτή η αντίθεση γίνεται αντιληπτή στην Εικόνα 0.1.1. Προκύπτουν λοιπόν αμφιβολίες, για το κατά πόσο τα γλωσσικά μοντέλα, ακολουθούν γνωστικούς μηχανισμούς παραπλήσιους με αυτούς του ανθρώπινου εγκεφάλου.

Για να αντιμετωπίσουμε κάποια από τα παραπάνω προβλήματα της σύγχρονης γλωσσικής μοντελοποίησης, στην παρούσα εργασία εξετάζουμε την εκπαίδευση γλωσσικών μοντέλων **με περιορισμένο αριθμό λέξεων**. Η έρευνα μας εντάσσεται στα πλαίσια προσπαθειών για την εκπαίδευση *αποδοτικών* μοντέλων με λίγα δεδομένα, σε

συνθήκες που προσομοιάζουν το γλωσσικό περιβάλλον του ανθρώπου όταν μαθαίνει να μιλάει. Συγκεκριμένα συμμετέχουμε στη 2η διεξαγωγή του διαγωνισμού **BabyLM** [21, 145, 146][1]. Στόχος του διαγωνισμού είναι: ① να αναβαθμιστεί η σημασία των γλωσσικών μοντέλων ως γνωστικά μοντέλα του μηχανισμού απόκτησης γλώσσας του ανθρώπου, ② η εύρεσή πιο αποτελεσματικών ως προς τα δεδομένα διαδικασιών εκπαίδευσης και ③ ο εκδημοκρατισμός της έρευνας στον τομέα. Στα πλαίσια του διαγωνισμού καλούμαστε να εκπαιδεύσουμε γλωσσικά μοντέλα σε **δύο σενάρια**, που διαφοροποιούνται σε σχέση με τον μέγιστο αριθμό λέξεων των δεδομένων εκπαίδευσης. Το πρώτο με μέγιστο αριθμό λέξεων τα **10M** – συμβολίζεται ως `"Strict-Small "`, το δεύτερο με μέγιστο αριθμό λέξεων τα **100M** – συμβολίζεται ως `"Strict "`. Στις επόμενες παραγράφους περιγράφουμε τη συνεισφορά μας.

0.1.1 Η προσέγγιση μας

Η έρευνά μας αντλεί έμπνευση από πρόσφατες εξελίξεις στα Μικρά Γλωσσικά Μοντέλα (SLMs) για παραγωγή κειμένου, όπως προτάθηκε στο **TinyStories** (Μικρές Ιστορίες) [31]. Σε αυτήν τη σημαντική μελέτη, οι συγγραφείς έδειξαν ότι η εκπαίδευση σε ένα συνθετικό σύνολο δεδομένων απλών ιστοριών μπορεί να επιτρέψει στα SLM να παράγουν κείμενο υψηλής ποιότητας, που δεν εμπεριέχεται στο αρχικό σύνολο δεδομένων εκπαίδευσης. Κάνουμε την υπόθεση ότι για το σενάριο περιορισμένων δεδομένων που εξετάζουμε, η επαύξησή του αρχικού εκπαιδευτικού συνόλου με συνθετικά δεδομένα υψηλής ποιότητας θα προσφέρει στα μοντέλα τη δυνατότητά να δουν λέξεις με νέα συμφραζόμενα, σε διαφορετικά γλωσσικά πλαίσια. Αυτό με τη σειρά του θα επιτρέψει καλύτερη γλωσσική κατανόηση, σε σύγκριση με την εκπαίδευση μονό στο αρχικό, μη επαυξημένο σύνολο.

Για να εξετάσουμε την υπόθεση μας, αρχικά επεκτείνουμε τη δουλειά στο *TinyStories* [31], μελετώντας τη δημιουργική ικανότητα των μοντέλων με λίγα δεδομένα εκπαίδευσης. Σε τεχνικούς όρους, προτείνουμε την εκπαίδευση ενός γλωσσικού μοντέλου με αρχιτεκτονική αποκωδικοποιητή μετασχηματιστή (decoder transformer) GPT-Neo [12] στο σύνολο ιστοριών *TinyStories*. Πειραματιζόμαστε με την εκπαίδευση μοντέλων GPT-Neo, ενώ μεταβάλλουμε τον *αριθμό των διαθέσιμων δεδομένων* και αξιολογούμε τις ικανότητες τους στην παραγωγή κειμένου, διατηρώντας σταθερή την αρχιτεκτονική του μοντέλου. Η έρευνά μας είναι συμπληρωματική της αρχικής δουλειάς [31], η οποία μετρά τις επιπτώσεις της μεταβολής του βάθους και του μεγέθους της αρχιτεκτονικής στην επίδοση των μοντέλων. Τα ευρήματά μας δείχνουν ότι ακόμη και σε καθεστώτα χαμηλών δεδομένων ($\leq$ 100 εκατομμύρια λέξεις), τα μοντέλα GPT-Neo μπορούν να αποκτήσουν εντυπωσιακή κατανόηση της γραμματικής και να δημιουργήσουν ιστορίες *υψηλής ποιότητας* και *ποικιλομορφίας*, συγκρίσιμες με μοντέλα που έχουν εκπαιδευτεί στο πλήρες σύνολο δεδομένων (περίπου 440 εκατομμύρια λέξεις).

Στη συνέχεια, κάνουμε χρήση συνθετικών δεδομένων μικρών ιστοριών κατά την προεκπαίδευση, και διερευνούμε την επίδραση τους στις γλωσσικές ικανότητες των μοντέλων. Επιλέγουμε ένα μικρό υποσύνολο του *TinyStories*, εκπαιδεύουμε σε αυτό ένα μοντέλο GPT-Neo, και το χρησιμοποιούμε για να δημιουργήσουμε *εναλλακτικές συμπληρώσεις* στις ιστορίες του εκπαιδευτικού συνόλου. Στη συνέχεια δημιουργούμε ένα συνδυασμό σύνολων δεδομένων που αποτελείται από: ① το υποσύνολο του *TinyStories* που χρησιμοποιείται για την εκπαίδευση του GPT-Neo ② τα δεδομένα που δημιουργούνται, ③ ένα υποσύνολο του συνόλου *BabyLM* (Ενότητα 0.3.1). Με τον συνδυασμό συνόλων εκπαιδεύουμε ένα μοντέλο κωδικοποιητή μετασχηματιστή (encoder transformer) με την αρχιτεκτονική του LTG-BERT [120] – η οποία είναι βελτιστοποιημένη για μικρά σύνολα δεδομένων. Συγκρίνουμε την απόδοση των μοντέλων μας με μια ποικιλία βασικών μεθόδων (baselines), εκπαιδευμένων με και χωρίς τη χρήση συνθετικών δεδομένων. Τα αποτελέσματά μας δείχνουν ότι μια απλή εφαρμογή δεδομένων συνθετικής ιστορίας για προεκπαίδευση γλωσσικών μοντέλων, έχει ως αποτελέσματα μικρά ή αρνητικά κέρδη. Ωστόσο, δεδομένης της υψηλής ικανότητας των μοντέλων GPT-Neo στην παραγωγή κειμένου, πιστεύουμε ότι απαιτείται περισσότερη έρευνα για την πλήρη αξιοποίηση των δυνατοτήτων τους.

Η δουλειάς μας δημοσιεύθηκε [132] στα πλαίσια του 2ου διαγωνισμού BabyLM [21], στο συνέδριο CoNLL 2024. Για να ενθαρρύνουμε την έρευνα στον τομέα της επαύξησης δεδομένων σε συνθήκες περιορισμένων δεδομένων, δημοσιεύουμε την υλοποίηση και τα μοντέλα που εκπαιδεύσαμε [2].

[1] https://babylm.github.io/
[2] https://github.com/nikitas-theo/BERTtimeStories

Στις επόμενες παραγράφους, περιγράφουμε συνοπτικά κάποια σχετική βιβλιογραφία (Ενότητα 0.2), προχωρούμε σε λεπτομερή περιγραφή της μεθόδου μας (Ενότητα 0.3) και των πειραμάτων που εκτελέστηκαν (Ενότητα 0.4), και αναλύουμε τα αποτελέσματα της έρευνας μας (Ενότητα 0.5) Τέλος, συζητάμε τα συμπεράσματα της μελέτης μας, καθώς και μελλοντικές κατευθύνσεις (Ενότητα 0.6).

0.2 Σχετική Βιβλιογραφία

Διάφορες προσπάθεις για την αποδοτική εκπαίδευση μοντέλων έχουν αναπτυχθεί [56, 106, 36] – μειώνοντας σημαντικά τους υπολογιστικούς πόρους, και τον χρόνο που απαιτούνται για την εκπαίδευση γλωσσικών μοντέλων, όπως το BERT [27]. Παρ' όλα αυτά, Ο χώρος της γλωσσικής μοντελοποίησης με περιορισμό στον αριθμό των δεδομένων εκπαίδευσης έχει μείνει σχετικά ανεξερεύνητος. Πρόσφατα υπήρξε πρόοδος σε αυτήν την κατεύθυνσή, με την εισαγωγή μοντέλων βελτιστοποιημένων για μικρά σύνολα δεδομένων όπως το LTG-BERT, [120], και εκδηλώσεις προσανατολισμένες στην κοινότητα, που επικεντρώνονται στην εκπαίδευση με περιορισμένα δεδομένα όπως το Εργαστήριο για την Εκμάθηση από Λίγα Δεδομένα [14] και ο Διαγωνισμός MiniPile [64]. Στη συνέχεια, παρουσιάζουμε συνοπτικά πρόσφατη βιβλιογραφία σχετική με τη προσέγγιση μας.

1ος Διαγωνισμός BabyLM Οι τεχνικές αύξησης δεδομένων αποδείχθηκαν ωφέλιμες στον διαγωνισμό BabyLM για το προηγούμενο έτος [146]. Συγκεκριμένα, το ChapGPT [61] χρησιμοποιεί Regex μοτίβα για εξαγωγή κοινών φράσεων από εργασίες του GLUE[138, 140] και στη συνέχεια αξιοποίησε αυτά τα μοτίβα για να δημιουργήσει επακόλουθες ερωτήσεις που χρησίμευσαν ως πρόσθετα δεδομένα εκπαίδευσης. Στην προσέγγιση του Contextualizer [153], δημιουργούνται επιπλέον δείγματα με δυναμικό συνδυασμό τμημάτων κειμένων από διαφορετικές πηγές κατά τη διάρκεια της εκπαίδευσης. Μια άλλη προσέγγιση που ονομάζεται Baby's CoThought [161] χρησιμοποιεί ένα Large Language Model (LLM) για να διαμορφώσει άσχετες προτάσεις από το σώμα σε συνεκτικές παραγράφους, το οποίο οδήγησε σε αύξηση της επίδοσης. Σημειώνουμε όμως ότι είναι ενάντια στους περιορισμούς δεδομένων του διαγωνισμού.

Γλωσσικά Μοντέλα για Επαύξηση Δεδομένων Τα τελευταία χρόνια, τα LLMs έχουν χρησιμοποιηθεί όλο και περισσότερο για την αύξηση δεδομένων σε διάφορους τομείς [28]. Συγκεκριμένα, το [26] εισήγαγε το ChatGPT ως εργαλείο για τη δημιουργία ρεαλιστικών δειγμάτων κειμένου από συνδυασμό πραγματικών και τεχνητών δεδομένων, βελτιώνοντας τα σύνολα δεδομένων εκπαίδευσης. Ομοίως, οι αρχιτεκτονικές μετασχηματιστών όλων των ειδών, όπως: αποκωδικοποιητές (GPT-2,[110]), κωδικοποιητές (BERT, [27]) και μετασχηματιστές από ακολουθία σε ακολουθία (BART, [73]) έχουν διερευνηθεί για επαύξηση δεδομένων [70]. Στο έργο του [156], το GPT-3 [15] χρησιμοποιήθηκε για τη μίξη δειγμάτων πραγματικών και συνθετικών κειμένων για ισχυρή αύξηση δεδομένων. Επιπλέον, μοντέλα αποκωδικοποιητών έχουν χρησιμοποιηθεί επιτυχώς για τη δημιουργία δεδομένων εκπαίδευσης για κωδικοποιητές, αποδίδοντας βελτιώσεις στη μάθηση με μηδενικά δείγματα [86].

Μικρά Γλωσσικά Μοντέλα Η πρόσφατη μελέτη στο *TinyStories* [31] έδειξε ότι τα Μικρά Γλωσσικά Μοντέλα (SLMs) μπορούν να ξεπεράσουν τα μεγαλύτερα μοντέλα αξιοποιώντας υψηλής ποιότητας συνθετικά δεδομένα εκπαίδευσης, καταφέροντας να παράξουν κείμενο με ευχέρεια, συνοχή και δημιουργικότητα παρά το γεγονός ότι έχουν λιγότερες παραμέτρους. Αυτή η τάση υποστηρίζεται περαιτέρω από την εργασία σε διαδοχικές συστάσεις, όπου μικρά μοντέλα χρησιμοποιούνται αποτελεσματικά για συγκεκριμένες εργασίες [154]. Επιπλέον, το [10] χρησιμοποιεί ένα προεκπαιδευμένο LLM για την κωδικοποίηση προτροπών, χρησιμοποιώντας αυτές τις αναπαραστάσεις για να καθοδηγήσει ένα μικρότερο LM για πιο αποτελεσματική παραγωγή αποκρίσεων.

0.3 Μέθοδοι

Η μέθοδος μας περιγράφεται συνοπτικά στην Εικόνα 0.1.1. Χρησιμοποιούμε ένα υποσύνολο του συνόλου δεδομένων *TinyStories* ($\mathcal{D}_{\text{tiny}}$) [31] για να εκπαιδεύσουμε έναν μετασχηματιστή αποκωδικοποιητή (decoder transformer) με σκοπό την επαύξηση δεδομένων. Χρησιμοποιούμε ως είσοδο (prompt) του decoder τις σύντομες ιστορίες από το $\mathcal{D}_{\text{tiny}}$, για να δημιουργήσουμε ένα σύνολο από παραγόμενα δεδομένα ($\mathcal{D}_{\text{gen}}$): κάθε ιστορία (πράσινο) περικόπτεται και η αρχή της (κίτρινο) χρησιμοποιείται ως είσοδος στο μοντέλο, το οποίο και παράγει μια εναλλακτική ολοκλήρωση της ιστορίας (μπλε). Συμπληρώνουμε τα δύο αυτά σύνολα δεδομένων με ένα υποσύνολο του *BabyLM* ($\mathcal{D}_{\text{baby}}$) και εκπαιδεύουμε έναν μετασχηματιστή κωδικοποιητή (encoder transformer) σε όλα τα δεδομένα. Στη συνέχεια περιγράφουμε αναλυτικά τη μέθοδο μας ξεκινώντας από τα σύνολα δεδομένων.

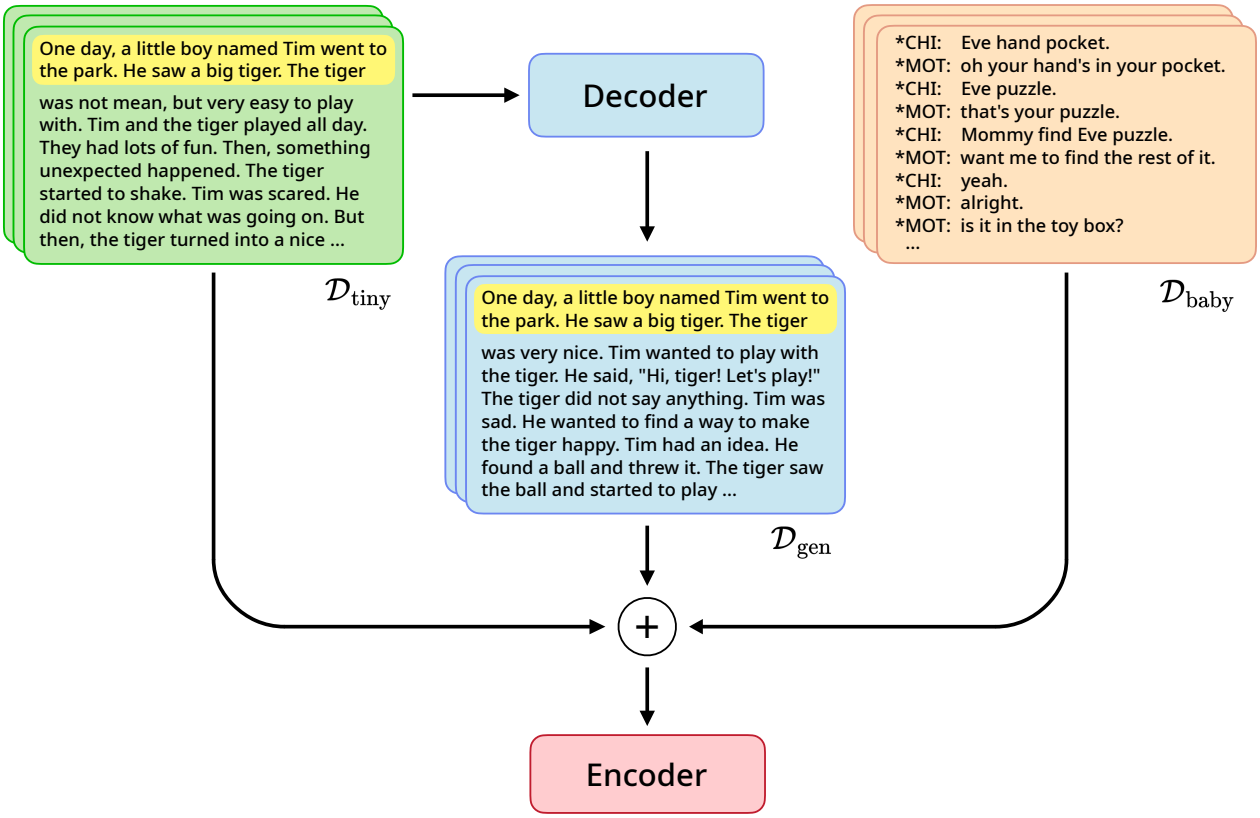

Figure 0.3.1: Σχηματική απεικόνισή της προτεινόμενης μεθόδου. Εικόνα τροποποιημένη απο [132].

0.3.1 Σύνολα Δεδομένων

Η μέθοδος μας βασίζεται σε δύο σύνολα δεδομένων: ① το σύνολο *TinyStories* – το οποίο συμβολίζουμε ως $\mathcal{D}_{\text{tiny}}$, μια συλλογή από μικρές ιστορίες σε απλή γλώσσα, ② το σύνολο *BabyLM* – το οποίο συμβολίζουμε ως $\mathcal{D}_{\text{baby}}$, που δημιουργήθηκε ως ένα αναπτυξιακά πιθανό σύνολο προεκπαίδευσης γλωσσικών μοντέλων. Για ένα σύνολο δεδομένων $\mathcal{D}_{\text{data}}$ συμβολίζουμε επίσης ένα υποσύνολο του, με m εκατομμύρια λέξεις ως $\mathcal{D}_{\text{data-m}}$. Παρακάτω δίνουμε μια σύντομη περιγραφή των συνόλων δεδομένων που χρησιμοποιήσαμε.

BabyLM Το σύνολο δεδομένων *BabyLM* ($\mathcal{D}_{\text{baby}}$), δημοσιεύθηκε στα πλαίσια του ομωνύμου διαγωνισμού από τους διοργανωτές [145, 21], και εμπεριέχει ένα ευρύ σύνολο από κείμενα. Δημιουργήθηκε με σκοπό να προσομοιάσει τα γλωσσικά ερεθίσματα που δέχεται ένα παιδί κατά την ανάπτυξη του, καθώς μαθαίνει να μιλάει. Για αυτόν τον λόγο περιλαμβάνει μεγάλο ποσοστό προφορικής γλώσσας και εμπεριέχει, μεταξύ άλλων, αποσπάσματα από παιδικά βιβλία, διαλόγους, ομιλία που απευθύνεται σε παιδιά, και άρθρα της Wikipedia. Δημοσιεύθηκαν δύο εκδόσεις του συνόλου: μία με μέγιστο αριθμό λέξεων τα 10M, και μία με μέγιστο αριθμό τα 100M. Αυτές αντιστοιχούν στα δύο σενάρια του διαγωνισμού, το `Strict-Small` και το `Strict` , αντίστοιχα.

***Tinystories* (Μικρές Ιστορίες)** Το *TinyStories* ($\mathcal{D}_{\text{tiny}}$) δημοσιεύθηκε στο [31], και αποτελεί ένα συνθετικό σύνολο δεδομένων. Περιλαμβάνει μια συλλογή μικρών ιστοριών που έχουν δημιουργηθεί με την προτροπή των μεγάλων γλωσσικών μοντέλων GPT3.5 και GPT-4 [95]. Το σύνολο δεδομένων δημιουργήθηκε για να διατηρήσει όλα τα βασικά στοιχεία της φυσικής γλώσσας, όπως τη γραμματική, το λεξιλόγιο, την ύπαρξη γεγονότων, και τον συλλογισμό, ενώ έχει περιορισμένη ποικιλομορφία και μέγεθος. Πιο συγκεκριμένα, οι ιστορίες είναι 2-3 παραγράφων και ακολουθούν απλές πλοκές και κεντρικά θέματα. Επιπλέον, το σύνολο δεδομένων έχει περιορισμένο λεξιλόγιο και γενικά προορίζεται να είναι στο επίπεδο κατανόησης των παιδιών 3-4 ετών. Η αρχική έκδοση του συνόλου δεδομένων (V1) που δημιουργήθηκε από τα GPT3.5 και GPT-4 περιέχει περίπου 373 εκατομμύρια λέξεις. Κυκλοφόρησε μια δεύτερη έκδοση (V2), με μόνο ιστορίες που δημιουργήθηκαν από το GPT-4 και περιέχει 440 εκατομμύρια λέξεις. Χρησιμοποιούμε αυτήν την έκδοση σε όλα τα πειράματά μας.

0.3.2 Δημιουργία Δεδομένων

Περιγράφουμε τη δημιουργία του συνθετικού συνόλου ιστοριών $\mathcal{D}_{\text{gen}}$. Για να δημιουργήσουμε τα δεδομένα πρώτα εκπαιδεύουμε ένα μοντέλο αποκωδικοποιητή μετασχηματιστή (GPT-Neo) σε ένα υποσύνολο του *TinyStories*, το οποίο συμβολίζουμε ως $\mathcal{D}_{\text{tiny-m}}$. Χρησιμοποιούμε την αρχή των ιστοριών από το $\mathcal{D}_{\text{tiny-m}}$ για να δημιουργήσουμε προτροπές (prompts) και να παράξουμε εναλλακτικές συμπληρώσεις χρησιμοποιώντας το μοντέλο μας.

Ξεκινάμε επιλέγοντας το μέγεθος m του υποσυνόλου, λαμβάνοντας υπόψη δύο παράγοντες: την ανάγκη για αρκετή **ποικιλομορφία** στο τελικό σύνολο και την ανάγκη διασφάλισης **υψηλής ποιότητας** στο παραγμένο κείμενο. Δεδομένης της υπόθεσης ότι η ποιότητα των παραγμένων κειμένων αυξάνεται αναλογικά με το μέγεθος των δεδομένων εκπαίδευσης, θέλουμε να επιλέξουμε ένα αρκετά μεγάλο μέγεθος m για το $\mathcal{D}_{\text{tiny-m}}$, ώστε να διασφαλίσουμε την απαιτούμενη ποιότητα από τα παραγωγικά μοντέλα μας (αποκωδικοποιητές). Ταυτόχρονα, θέλουμε να αφήσουμε αρκετό αριθμό λέξεων διαθέσιμο, για να συμπεριλάβουμε ένα αρκετά μεγάλο μέρος του συνόλου δεδομένων *BabyLM* στην τελική εκπαίδευση. Αυτό θα διασφαλίσει ότι τα μοντέλα κωδικοποιητές εκτίθενται σε ένα πλούσιο λεξιλόγιο και μια ποικιλία γλωσσικών πλαισίων. Διαισθητικά, θέλουμε τα δεδομένα προεκπαίδευσης να έχουν μεγάλη ποικίλα, καθώς τα παιδιά μαθαίνουν από πολλές διαφορετικές γλωσσικές πηγές.

Για να ισορροπήσουμε αυτές τις αντικρουόμενες απαιτήσεις, θα βρούμε την ελάχιστη ποσότητα δεδομένων TinyStories που θα επιτρέψει στα μοντέλα μας να παράξουν ποιοτικές ιστορίες. Δειγματοληπτούμε από το *TinyStories* δημιουργώντας μια συλλογή υποσυνόλων με διαφορετικά μεγέθη, $\mathcal{D}_{\text{tiny-m}} : m \in \{5, 10, 25, 50, 75, 100\}$M (εκατομμύρια λέξεις). Για κάθε υποσύνολο, εκπαιδεύουμε ένα μοντέλο GPT-Neo και αξιολογούμε τις παραγωγικές και γλωσσικές του ικανότητες. Στην αξιολόγησή μας, συμπεριλαμβάνουμε μετρικές γραμματικής κατανόησης, ποικιλομορφίας και ποιότητας των παραγωγών. Οι μετρικές μας παρουσιάζονται στην Ενότητα 0.3.4. Για κάθε ένα από τα σενάρια `Strict` και `Strict-Small` , επιλέγουμε ένα υποσύνολο $\mathcal{D}_{\text{tiny-m}}$ και ένα αντίστοιχο μοντέλο GPT-Neo που έχει εκπαιδευτεί σε αυτό, με βάση την αξιολόγηση και τα παραπάνω κριτήρια, και τα χρησιμοποιούμε για την επάυξηση δεδομένων. Για να δημιουργήσουμε το $\mathcal{D}_{\text{gen}}$, για κάθε ιστορία $s \in \mathcal{D}_{\text{tiny-m}}$, περικόπτουμε την ιστορία στο 15%-30% του μεγέθους της και τη χρησιμοποιούμε για να προτρέψουμε το μοντέλο να παράξει μια εναλλακτική συμπλήρωση. Επιλέξαμε να χρησιμοποιήσουμε μικρό ποσοστό της αρχικής ιστορίας για να αποφύγουμε την επανάληψη, δεδομένου ότι οι ιστορίες στο $\mathcal{D}_{\text{tiny-m}}$ θα βρίσκονται ήδη στον συνδυασμό συνόλων για την τελική εκπαίδευση του κωδικοποιητή.

Για την παραγωγή δεδομένων πειραματιζόμαστε με δύο μεθόδους: άπληστη αποκωδικοποίηση (greedy decoding) και δειγματοληψία πυρήνα (nucleus sampling) [51]. Κατά τη δειγματοληψία, για κάθε ξεκίνημα ιστορίας παράγουμε k συμπληρώσεις με τα μοντέλα μας. Για να περιορίσουμε την επανάληψη μεταξύ των k διαφορετικών παραγωγών, και να αποφύγουμε τη σπατάλη υπολογιστικών πόρων (FLOPs), υπολογίζουμε τη μετρική Self-BLEU (Ενότητα 0.3.4) για ένα σύνολο τιμών k και επιλέγουμε αυτές με την καλύτερη ισορροπία μεταξύ: της ποικιλομορφίας, και του συνολικού αριθμού πρόσθετων δεδομένων εκπαίδευσης.

0.3.3 Δημιουργία Τελικού Συνόλου

Για καθένα ένα από τα σενάρια `Strict` και `Strict-Small` , δημιουργήσαμε τα σύνολα $\mathcal{D}_{\text{tiny-m}}$ και $\mathcal{D}_{\text{gen}}$ όπως περιγράφηκε προηγουμένως. Τώρα δημιουργούμε τον συνδυασμό συνόλων δεδομένων – συμβολίζεται με $\mathcal{D}_{\text{comb}}$, που χρησιμοποιείται για την εκπαίδευση του μετασχηματιστή κωδικοποιητή. Κατανέμουμε τις διαθέσιμες λέξεις που απομένουν στον προυπολογισμό μας, σε ένα υποσύνολο του *BabyLM* ($\mathcal{D}_{\text{baby-b}}$), που δημιουργούμε με τυχαία δειγματοληψία από το *BabyLM* σε επίπεδο κειμένων. Αφήνουμε μεθόδους δειγματοληψίας που λαμβάνουν υπόψιν το περιεχόμενο των κειμένων για μελλοντική εργασία. Για τα `Strict` / `Strict-Small` σενάρια, το μέγεθος b του $\mathcal{D}_{\text{baby-b}}$ επιλέγεται έτσι ώστε: $b + m \leq 100\text{M}/10\text{M}$. Κατασκευάζουμε τώρα το $\mathcal{D}_{\text{comb}}$ συνδυάζοντας όλα τα σύνολα δεδομένων $\mathcal{D}_{\text{comb}} = (\mathcal{D}_{\text{tiny-m}}, \mathcal{D}_{\text{baby-b}}, \mathcal{D}_{\text{gen}})$. Χρησιμοποιούμε έναν στόχο Μοντελοποίησης Γλώσσας με Μάσκα (Masked Language Modeling) για να εκπαιδεύσουμε έναν μετασχηματιστή κωδικοποιητή στο $\mathcal{D}_{\text{comb}}$, χρησιμοποιώντας την αρχιτεκτονική LTG-BERT [120].

0.3.4 Μετρικές Αξιολόγησης

Για την αξιολόγηση των μετασχηματιστών κωδικοποιητών (encoder transformers) χρησιμοποιούμε τη σουίτα αξιολόγησης του διαγωνισμού, που αποτελείται από τρία σύνολα δείκτών αναφοράς (benchmarks): το BLiMP,

το (Super)GLUE και το EWoK, καθένα από τα οποία αξιολογεί, σε γενικές γραμμές, τη γλωσσική επάρκεια, τη γενική κατανόηση της γλώσσας, και τη γνώση για τον κόσμο, αντίστοιχα. Σημειώνουμε ότι τα benchmarks της πρόκλησης είναι φιλτραρισμένες εκδόσεις [146], καθιστώντας τα αποτελέσματά μας μη συγκρίσιμα με αξιολογήσεις στα πλήρη δεδομένα. Για τους αποκωδικοποιητές μετασχηματιστές, χρησιμοποιούμε τα EWoK και BLiMP, και επίσης εισάγουμε ορισμένες επιπλέον διαδικασίες αξιολόγησης: το Self-BLEU για τη μέτρηση της ποικιλομορφίας, καθώς και μια αξιολόγηση με τη βοήθεια ενός Μεγάλου Γλωσσικού Μοντέλου (LLM) για τη μέτρηση της ποιότητας των παραγωγών. Εξηγούμε κάθε μία από τις μεθόδους αξιολόγησης παρακάτω.

BLiMP Το BLiMP (Benchmark of Linguistic Minimal Pairs) [143], είναι ένα σύνολο από εργασίες που έχουν σχεδιαστεί για την αξιολόγηση της γραμματικής και συντακτικής γνώσης των γλωσσικών μοντέλων. Αποτελείται από ζεύγη ελάχιστα διαφορετικών προτάσεων που καλύπτουν διάφορα γραμματικά φαινόμενα και αφορούν τη σύνταξη, τη μορφολογία και τη σημασιολογία. Το μοντέλο πρέπει να αντιστοιχήσει μεγαλύτερη πιθανότητα στη γραμματικά σωστή πρόταση σε κάθε ζευγάρι. Αξιολογούμε επίσης στο συμπλήρωμα του BLiMP (Supp.), που κυκλοφόρησε στην προηγούμενη έκδοση του διαγωνισμού [145], και περιλαμβάνει πρόσθετα γραμματικά φαινόμενα.

(Super)GLUE Το GLUE (General Language Understanding Evaluation) [138] δημιουργήθηκε για να αξιολογήσει τις ικανότητες γενικής κατανόηση γλώσσας (NLU) των γλωσσικών μοντέλων, σε ένα ευρύ φάσμα εργασιών κατανόησης φυσικής γλώσσας. Σύντομα μετά την εισαγωγή του, τα γλωσσικά μοντέλα κατάφεραν να ξεπεράσουν την ανθρώπινη επίδοση στο GLUE (για μη ειδικούς). Αυτό είχε σαν συνέπεια τη δημιουργία του SuperGLUE [140], που παρουσιάστηκε για να προσφέρει ένα πιο απαιτητικό σύνολο εργασιών. Χρησιμοποιούμε συνολικά 10 εργασίες και από τα δύο benchmakrs, που καλύπτουν: την απάντηση ερωτήσεων (BoolQ, MultiRC), την ταξινόμηση συναισθημάτων (SST-2), την παράφραση (MRPC, QQP), τη γραμματική αποδοχή (CoLA), τους συλλογισμούς "κοινής λογικής" (WSC) και τη σχέση συμπεράσματος (MNLI, QNLI, RTE).

EWoK Το EWoK (Elements of World Knowledge) [55], αξιολογεί την ικανότητα ενός γλωσσικού μοντέλου να κατανοεί και να μοντελοποιεί γνώση για τον κόσμο. Συγκεκριμένα, μετράει πόσο καλά ένα μοντέλο μπορεί να συνδέσει ένα κείμενο στόχο είτε με ένα κατάλληλο είτε με ένα αταίριαστο γλωσσικό πλαίσιο, δίνοντας έμφαση σε βασικές έννοιες όπως οι κοινωνικές δυναμικές και οι χωρικές σχέσεις. Τόσο τα κείμενα πλαίσια όσο και οι στόχοι δημιουργούνται ως ζεύγη ελάχιστης αντίθεσης, με προσαρμόσιμα στοιχεία όπως αντικείμενα, πράκτορες (agents) και τοποθεσίες. Τα δείγματα είναι ελάχιστα ζεύγη ζευγών, και παρόμοια με το BLiMP, ένα γλωσσικό μοντέλο πρέπει να αντιστοιχίσει μεγαλύτερη πιθανότητα στο ζεύγος σωστού πλαισίου και κειμένου.

Self-BLEU Για να μετρήσουμε την ποικιλομορφία των παραγόμενων ιστοριών, χρησιμοποιήσαμε τη μετρική Self-BLEU [163]. Δεδομένης μιας συλλογής παραγμένων κειμένων, υπολογίζουμε τη μετρική BLEU με μια παραγωγή ως υπόθεση και τις άλλες ως αναφορές, αξιολογώντας πόσο παρόμοια είναι με τις υπόλοιπες. Ορίζουμε το Self-BLEU ως τον μέσο όρο όλων των βαθμολογιών BLEU στο σώμα όλων των παραγωγών. Η μετρική ορίζεται σε συνεχή κλίμακα στο $[0, 1]$, με τις υψηλότερες τιμές να αντιστοιχούν σε λιγότερη ποικιλομορφία.

LLM Evaluation Για να παρέχουμε μια ολοκληρωμένη αξιολόγηση των παραγωγικών ικανοτήτων των μετασχηματιστών αποκωδικοποιητών (GPT-Neo) ως προς την *ποιότητα*, χρησιμοποιήσαμε το Claude-3.5 Sonnet [2]. Ακολουθώντας την προσέγγιση στο [31], δίνουμε τις ιστορίες που δημιουργούνται στο Μεγάλο Γλωσσικό Μοντέλο (LLM), και του ζητάμε να τις αξιολογήσει με βάση τη *Γραμματική*, τη *Δημιουργικότητα* και τη *Συνέπεια* της ολοκλήρωσης σε σχέση με την αρχή της ιστορίας, σε κλίμακα από το 1 έως το 10. Λεπτομέρειες για την αξιολόγηση δίνονται στην Ενότητα 0.4.1. Ένα παράδειγμα της αλληλεπίδρασης με το LLM δίνεται στο Παράρτημα B.

0.4 Πειραματικό Μέρος

Διεξήγαμε τα πειράματά μας σε ένα κοινό υπολογιστικό σύστημα (μοιρασμένο σε πλήθος χρηστών) με 8 κάρτες γραφικών (GPU) Nvidia V100 με 16 GB, επιπλέον, για την αξιολόγηση των μοντέλων μας έγινε χρήση μιας GPU Nvidia RTX-3090 με 24 GB. Χρησιμοποιήσαμε τις βιβλιοθήκες PyTorch [99] και HuggingFace [151] για να εκπαιδεύσουμε όλα τα μοντέλα μας, και για τις αξιολογήσεις μας για τα BLiMP, EWoK και (Super)GLUE

χρησιμοποιήσαμε την επίσημη βιβλιοθήκη αξιολόγησης που δημοσιεύθηκε στα πλαίσια του διαγωνισμού [35, 21]. Τέλος, για τον υπολογισμό του Self-BLEU χρησιμοποιήθηκε η βιβλιοθήκη NLTK [11].

0.4.1 *TintStories* (Μικρές Ιστορίες) – Ανάλυση και Εκπαίδευση

Για τον μετασχηματιστή αποκωδικοποιητή που χρησιμοποιούμε για την επαύξηση δεδομένων, επιλέγουμε μία από τις αρχιτεκτονικές GPT-Neo με τις καλύτερες επιδόσεις στην έρευνα του TinyStories [31] [3]. Όλα τα μοντέλα GPT-Neo που εκπαιδεύουμε χρησιμοποιούν τις ίδιες υπερπαραμέτρους, εκτός από το weight decay, το dropout και το μέγεθος του λεξιλογίου (vocabulary size), που ρυθμίζονται για κάθε συγκεκριμένο μέγεθος δεδομένων εκπαίδευσης. Χρησιμοποιούμε μια διαδικασία εκπαίδευσης παρόμοια με αυτήν στο [31], με πρόσθετο regularization για τα σενάρια χαμηλών δεδομένων που εξετάζουμε. Οι υπερπαράμετροι και οι λεπτομέρειες σχετικά με την αρχιτεκτονική περιλαμβάνονται στο Παράρτημα C. Επιλέξαμε να εκπαιδεύσουμε τα μοντέλα στην πιο πρόσφατη έκδοση των δεδομένων *TinyStories* (V2), που δημιουργήθηκε προτρέποντας μόνο το GPT-4. Το πλήρες σύνολο δεδομένων χωρίς δειγματοληψία περιέχει περίπου 440M λέξεις. Κατά τη διάρκεια της αξιολόγησής μας, αναφέρουμε επίσης αποτελέσματα για το αρχικό μοντέλο που κυκλοφόρησε από τους συγγραφείς, το οποίο έχει εκπαιδευτεί στην πρώτη έκδοση του συνόλου δεδομένων (V1) με 373M περίπου λέξεις.

Στις ακόλουθες παραγράφους, διεξάγουμε μια ενδελεχή ανάλυση της σχέσης μεταξύ: της γλωσσικής ικανότητας των μοντέλων GPT-Neo που εκπαιδεύονται σε υποσύνολα του *TinyStories* και του μεγέθους των συνόλων εκπαίδευσης $|\mathcal{D}_{\text{tiny-m}}|$. Πειραματιζόμαστε με διάφορα μεγέθη υποσυνόλων *TinyStories* $\mathcal{D}_{\text{tiny-m}}$: $m \in \{5, 10, 25, 50, 75, 100\}$M (εκατομμύρια λέξεις). Από τα πειράματά μας, αντλούμε πληροφορίες σχετικά με τις ικανότητες των γλωσσικών μοντέλων σε καθεστώτα χαμηλών δεδομένων. Αυτή η αξιολόγηση θα ορίσει επίσης την επιλογή του υποσυνόλου *TinyStories* $\mathcal{D}_{\text{tiny}}$ που χρησιμοποιείται για τη δημιουργία του συνόλου δεδομένων $\mathcal{D}_{\text{gen}}$ και για την εκπαίδευση του τελικού μοντέλου μετασχηματιστή κωδικοποιητή.

Για να αντλήσουμε κάποιες αρχικές πληροφορίες για τη γλωσσική ικανότητα των μοντέλων αποκωδικοποιητή GPT-Neo, μετράμε την απόδοση στο BLiMP, το συμπλήρωμά του (Supp.) και το EWoK. Τα αποτελέσματα παρουσιάζονται στον Πίνακα 0.4.1. Είναι σημαντικό να σημειώσουμε ότι οι 50M λέξεις, φαίνεται να είναι ένα σημείο αποκοπής, με αξιοσημείωτες πτώσεις στην απόδοση για μοντέλα εκπαιδευμένα με λιγότερες λέξεις. Με βάση αυτό, για την επαύξηση δεδομένων, επιλέγουμε $\mathcal{D}_{\text{tiny-50M}}$ για το `Strict` σενάριο, και το $\mathcal{D}_{\text{tiny-5M}}$ για το `Strict-Small` σενάριο. Είναι σημαντικό ότι δε συμπεριλαμβάνουμε την αξιολόγηση με χρήση LLM (που παρουσιάζεται παρακάτω) στη λήψη αυτής της απόφασης, καθώς αυτό θα παραβίαζε τους περιορισμούς δεδομένων της δουλειάς μας. Αφήνουμε περαιτέρω έρευνα σχετικά με το μέγεθος των δειγμάτων για το `Strict-Small` σενάριο για μελλοντική εργασία.

Train Data	BLiMP ↑	Supp. ↑	EWoK ↑
5M	55.5	53.8	51.1
10M	58.4	51.6	51.9
25M	59.9	55.1	52.4
50M	62.8	52.8	53.0
75M	64.0	54.8	53.4
100M	**64.8**	50.8	53.1
440M (V2)	64.6	55.0	53.9
373M (V1)	**64.8**	**60.9**	**54.0**

Table 0.4.1: Αποτελέσματα αξιολόγησης στο BLiMP, BLiMP Supplement (Supp.) και EWoK, για μοντέλα GPT-Neo που έχουν εκπαιδευτεί στο *TinyStories* με διαφορετικά μεγέθη συνόλων εκπαίδευσης. Παρατηρούμε ότι καθώς ο όγκος των δεδομένων μειώνεται, οι βαθμολογίες στα BLiMP και EWoK γενικά μειώνονται επίσης. Αντίθετα, η βαθμολογία στο BLiMP Supp. δείχνει μεγαλύτερη διακύμανση.

[3] https://huggingface.co/roneneldan/TinyStories-33M

Οι παραπάνω βαθμολογίες μας δίνουν στοιχεία σχετικά με τη γραμματική κατανόηση (BLiMP) και τη γνώση για τον κόσμο (EWoK) που έχουν τα μοντέλων μας, αλλά παραλείπουν δύο σημαντικούς άξονες αξιολόγησης της παραγωγικής απόδοσης: ① την **ποικιλομορφία** (diversity) των παραγωγών και ② την **ποιότητα** (quality) των παραγωγών. Εστιάζουμε σε αυτούς τους δύο άξονες στις ακόλουθες παραγράφους. Εκτός από τις ποσοτικές βαθμολογίες, συμπεριλαμβάνουμε επίσης δείγματα παραγωγών απο τα μοντέλα GPT-Neo στο Παράρτημα A.

Αξιολόγηση της Ποιότητας των Παραγωγών Η αξιολόγηση της παραγωγικής ικανότητας ενός μοντέλου σε εργασίες (tasks) που έχουν να κάνουν με παραγωγή κειμένου ανοιχτού τύπου (open-ended) αποτελεί πρόκληση, καθώς οι περισότερες μέθοδοι αξιολόγησης είναι φτιαγμένες για δομημένα αποτελέσματα, και συνήθως μετρούν την πιστότητας σε ένα σύνολο κειμένων αναφοράς. Για να αντιμετωπίσουμε αυτο το πρόβλημα, υιοθετούμε τη μέθοδο αξιολόγησης που προτάθηκε πρόσφατα από τους [31], και χρησιμοποιούμε ένα Μεγάλο Γλωσσικό Μοντέλο (LLM) για να αξιολογήσουμε τις ιστορίες των μοντέλων μας. Οι αρχικοί συγγραφείς χρησιμοποίησαν το GPT-4, εμείς χρησιμοποιούμε αντ' αυτού Claude-3.5 Sonnet [2].

Χρησιμοποιούμε ένα σύνολο 44 χειροκίνητα κατασκευασμένων προτροπών (prompts) [4] που περιέχουν την αρχή μιας ιστορίας και παράγουμε 10 συμπληρώσεις της ιστορίας με κάθε μοντέλο μας. Στη συνέχεια παρέχουμε στο LLM την αρχή της ιστορίας και την ολοκλήρωση του μοντέλου και του ζητάμε να αξιολογήσει την απόκριση του μοντέλου σε τρεις άξονες: (α) *Γραμματική*, (β) *Δημιουργικότητα* και (γ) *Συνέπεια* με την αρχή της ιστορίας. Επιπλέον, του ζητάμε να ταξινομήσει την ιστορία σε διαφορετικές ηλικιακές ομάδες, που κυμαίνονται από τα 3 έως τα 16 έτη. Οι βαθμολογίες δίνονται σε μια κλίμακα από το 1 έως το 10. Υπολογίζουμε τον μέσο όρο για όλες τις ιστορίες και όλες τις συμπληρώσεις για κάθε μοντέλο στον Πίνακα 0.4.2. Παρατηρούμε ότι ο περιορισμός των δεδομένων εκπαίδευσης, έως και 25M λέξεις, έχει ως αποτέλεσμα μικρές μειώσεις στην απόδοση. Αυτό δηλώνει ότι η ποιότητα των παραγομένων των μοντέλων διατηρείται και στο καθεστώς λίγων δεδομένων.

Αξιολόγηση της Ποικιλομορφίας των Παραγωγών Για τη μέτρηση της ποικιλομορφίας χρησιμοποιούμε τη μετρική Self-BLEU [163] – όσο υψηλότερη είναι η μετρική, τόσο μικρότερη είναι η ποικιλομορφία των παραγωγών του μοντέλου. Για κάθε μοντέλο, δειγματοληπτούμε 100 ιστορίες από το σύνολο εκπαίδευσης και τις περικόπτουμε στο 15%-30% του μήκους τους, για να χρησιμοποιηθούν ως προτροπές (prompts). Στη συνέχεια, τα μοντέλα μας ολοκληρώνουν την ιστορία, κάνοντας χρήση άπληστης αποκωδικοποίησης (greedy decoding). Αναφέρουμε βαθμολογίες Self-BLEU, κλιμακωμένες στο $[0, 100]$, για το σύνολο των 100 ολοκληρώσεων στον Πίνακα 0.4.2. Τα αποτελέσματά μας δείχνουν ότι τα μοντέλα με περιορισμένα δεδομένα εκπαίδευσης μπορούν να επιτύχουν υψηλή ποικιλομορφία, διατηρώντας ταυτόχρονα την ποιότητα παραγωγής, όπως αποδεικνύεται από τις βαθμολογίες των μοντέλων GPT-Neo που έχουν εκπαιδευτεί σε 25M και 50M λέξεις.

Train Data	Gr. ↑	Cr. ↑	Cons. ↑	SB ↓
5M	4.56	4.99	3.37	38.6
10M	5.31	5.34	3.98	38.3
25M	6.00	5.65	4.55	34.6
50M	6.01	5.53	4.54	33.0
75M	6.08	5.50	4.49	37.1
100M	6.17	5.57	**4.78**	39.8
440M (V2)	5.88	5.53	4.49	37.3
373M (V1)	**6.24**	**5.73**	4.70	**29.6**

Table 0.4.2: Αποτελέσματα της αξιολόγησης των μοντέλων GPT-Neo με χρήση Μεγάλου Γλωσσικού Μοντέλου (LLM) και της μετρικής Self-BLEU. Δίνουμε οδηγίες στο Claude-3.5 Sonnet να αξιολογήσει την παραγωγική απόδοση σε τρεις κατηγορίες: Γραμματική (Gr), Δημιουργικότητα (Cr.), Συνέπεια (Cons.) με την αρχή της ιστορίας. Συμπεριλαμβάνουμε επίσης το Self-Blue (SB), που μετράει την ποικιλομορφία των παραγωγών.

[4] https://huggingface.co/datasets/roneneldan/TinyStories

0.4.2 Δημιουργία Δεδομένων

Τώρα περιγράφουμε τη δημιουργία του συνόλου δεδομένων $\mathcal{D}_{\text{comb}} = (\mathcal{D}_{\text{tiny-m}}, \mathcal{D}_{\text{baby-b}}, \mathcal{D}_{\text{gen}})$. Όπως συζητήθηκε στην Ενότητα 0.4.1, επιλέγουμε τα υποσύνολα $\mathcal{D}_{\text{tiny-50M}}$ για το `Strict` σενάριο και $\mathcal{D}_{\text{tiny-5M}}$ για το `Strict-Small` σενάριο. Για συντομία, παρακάτω αναλύουμε τη διαδικασία μόνο για το `Strict-Small` σενάριο, ακολουθούμε όμως τα ίδια βήματα και για το `Strict` σενάριο. Εκπαιδεύουμε ένα μοντέλο GPT-Neo στο $\mathcal{D}_{\text{tiny-5M}}$ και το χρησιμοποιούμε για να δημιουργήσουμε το σύνολο δεδομένων $\mathcal{D}_{\text{gen}}$. Χρησιμοποιούμε την αρχή κάθε ιστορίας από το σύνολο εκπαίδευσης $\mathcal{D}_{\text{tiny-5M}}$ (15%-30% του μεγέθους της), για να προτρέψουμε τον αποκωδικοποιητή να δημιουργήσει εναλλακτικές συμπληρώσεις. Πειραματιζόμαστε με διαφορετικές τεχνικές παραγωγής, συμπεριλαμβανομένης της άπληστης δημιουργίας – $\mathcal{D}_{\text{gen-greedy}}$ και της δειγματοληψίας πυρήνα – $\mathcal{D}_{\text{gen-nucleus-k}}$, όπου k είναι ο αριθμός των παραγωγών για κάθε ιστορία. Συμπληρώνουμε τα δύο αυτά σύνολα, με δεδομένα απο το *BabyLM* ($\mathcal{D}_{\text{baby-5M}}$), χωρίς να ξεπεράσουμε τον περιορισμό των 10M, για να σχηματίσουμε το $\mathcal{D}^{\text{10M}}_{\text{comb}} = (\mathcal{D}_{\text{tiny-5M}}, \mathcal{D}_{\text{baby-5M}}, \mathcal{D}_{\text{gen}})$.

Για να επιλέξουμε κατάλληλες τιμές για την παράμετρο k για τη δειγματοληψία πυρήνα, χρησιμοποιούμε τη μετρική Self-BLEU. Δειγματοληπτούμε 100 ιστορίες από το $\mathcal{D}_{\text{tiny-5M}}$ και χρησιμοποιούμε την αρχή τους (15%-30% του μήκους) για προτρέψουμε το μοντέλο GPT-Neo, και να δημιουργήσουμε 50 συμπληρώσεις για κάθε προτροπή με $p = 0,95$. Για κάθε τιμή $k \in \{2, 3, ..., 50\}$ υπολογίζουμε το Self-BLEU μεταξύ της ομάδας των παραγωγών S_k. Στόχος μας είναι να δούμε πόσο διαφορετικές είναι οι διάφορες παραγωγές για την ίδια προτροπή, καθώς αυξάνεται ο αριθμός των παραγωγών (k). Το μέσο Self-BLEU για όλα τες ιστορίες και παραγωγές απεικονίζεται στην Εικόνα 0.4.1. Επιλέγουμε να πειραματιστούμε με $k = 5$ και $k = 10$, ως μια καλή ισορροπία μεταξύ της ποικιλομορφίας και του μεγέθους των επιπλέον δεδομένων, καθώς παρατηρούμε ότι σε αυτό το εύρος, το Self-BLEU παραμένει σταθερό.

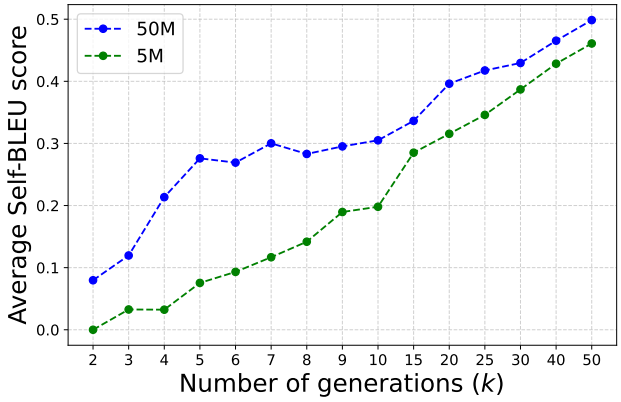

Figure 0.4.1: Μέση βαθμολογία Self-BLEU για τα μοντέλα GPT-Neo εκπαιδευμένα 5 και 50 εκατομμύρια λέξεις απο το *TinyStories*. Για κάθε μοντέλο, χρησιμοποιούμε 100 ιστορίες και παράγουμε 50 ολοκληρώσεις. Σχεδιάζουμε το μέσο Self-BLEU όλων των ιστοριών, καθώς ο αριθμός των παραγωγών ανα ιστορία (k) αυξάνεται.

0.4.3 Εκπαίδευση του Κωδικοποιητή Μετασχηματιστή

Στη συνέχεια, χρησιμοποιούμε τον συνδυασμό συνόλων δεδομένων $\mathcal{D}_{\text{comb}}$ για να εκπαιδεύσουμε ένα μοντέλο μετασχηματιστή κωδικοποιητή (encoder transformer) LTG-BERT [120]. Η διαδικασία εκπαίδευσής που χρησιμοποιούμε βασίζεται στον πηγαίο κώδικα που δημοσιεύθηκε από τους συγγραφείς[5]. Χρησιμοποιούμε παρόμοιες υπερπαραμέτρους (Παράρτημα C), προσαρμοσμένες για τις υποδομές μας, και εκτελούμε ελάχιστη βελτιστοποίηση υπερπαραμέτρων. Προκειμένου να αξιολογήσουμε την επίδραση της επαύξησης δεδομένων στην τελική απόδοση, εκπαιδεύουμε μια ποικιλία από βασικά μοντέλα (baselines), διαφοροποιώντας το σύνολο δεδομένων προεκπαίδευσης και διατηρώντας σταθερές όλες τις άλλες συνθήκες εκπαίδευσης. Συγκεκριμένα, για κάθε σενάριο,

[5] https://github.com/ltgoslo/ltg-bert

όλα τα μοντέλα χρησιμοποιούν τις ίδιες υπερπαραμέτρους και τον ίδιο αριθμό απο FLOPs, διασφαλίζοντας μια δίκαιη σύγκριση.

Βασικά Μοντέλα Για το σενάριο `Strict-Small` , καθιερώνουμε βασικά μοντέλα LTG-BERT για σύγκριση με τη μέθοδο μας, χρησιμοποιώντας 10M λέξεις από τα σύνολα δεδομένων *BabyLM* – $\mathcal{D}_{\text{baby-10M}}$ και *Tinystories* – $\mathcal{D}_{\text{tiny-10M}}$. Επιπλέον, εκπαιδεύσαμε έναν κωδικοποιητή χρησιμοποιώντας έναν συνδυασμό 5M λέξεων από κάθε ένα από τα δύο σύνολα δεδομένων – $\mathcal{D}_{\text{baby-5M}} + \mathcal{D}_{\text{tiny-5M}}$. Αυτά τα μοντέλα χρησιμεύουν ως σημεία αναφοράς βάσει των οποίων αξιολογούμε την απόδοση μοντέλων που έχουν εκπαιδευτεί με διάφορους συνδυασμούς παραγόμενων δεδομένων, με στόχο την αξιολόγηση της αποτελεσματικότητας της επαύξησης δεδομένων. Η ίδια μεθοδολογία εφαρμόστηκε με συνέπεια και στο `Strict` σενάριο. Εδώ, εκπαιδεύσαμε κωδικοποιητές με 100M λέξεις από κάθε σύνολο δεδομένων, άλλα και συνδυαστικά, χρησιμοποιώντας 50M λέξεις από κάθε σύνολο. Συμπεριλαμβάνουμε επίσης αποτελέσματα για τα βασικά μοντέλα του διαγωνισμού BabyLM [21] – συγκεκριμένα το LTG-BERT [120] και το BabyLlama [134]. Σημειώνουμε ότι αυτά τα μοντέλα εκπαιδεύτηκαν με διαφορετικές συνθήκες από τα δικά μας. Μια σημαντική παράμετρος που διαφέρει είναι ο αριθμός τον εποχών εκπαίδευσης: για το βασικό μοντέλο του διαγωνισμού LTG-BERT χρησιμοποιήθηκαν 20 περίπου εποχές, ενώ για τα δικά μας μοντέλα 27 περίπου εποχές (20 χιλιάδες βήματα).

Ισορροπημένη Εκπαίδευση Καθώς αυξάνουμε τον αριθμό των παραγόμενων κειμένων στο εκπαιδευτικό σύνολο LTG-BERT ($\mathcal{D}_{\text{comb}}$), τροποποιούμε επίσης την κατανομή των δειγμάτων *TinyStories* και *BabyLM* που συναντά το μοντέλο κατά τη διάρκεια της εκπαίδευσης. Αυτό θα μπορούσε να επηρεάσει την απόδοση του μοντέλου, καθώς γίνεται πιο συντονισμένο στο *TinyStories*. Για να αντιμετωπίσουμε αυτό το πρόβλημα, πειραματιζόμαστε με μια παραλλαγή εκπαίδευσης όπου εξισορροπούμε τον αριθμό των δειγμάτων και από τα δύο σύνολα δεδομένων. Συγκεκριμένα, τα δείγματα σε κάθε παρτίδα (batch) λαμβάνονται με ίση πιθανότητα από το *TinyStories* – το οποίο περιλαμβάνει τόσο πρωτότυπα όσο και παραγμένα κείμενα – και το *BabyLM*. Αυτή η μέθοδος διασφαλίζει ότι το μοντέλο εκτίθεται σε ίσο αριθμό δειγμάτων από κάθε σύνολο δεδομένων κατά τη διάρκεια της εκπαίδευσης. Το σύμβολο στιλέτο † στα αποτελέσματα δηλώνει τη χρήση αυτής της στρατηγικής.

0.5 Αποτελέσματα

Παρουσιάζουμε τα τελικά αποτελέσματα της αξιολόγησης μας για τα σενάρια `Strict-Small` και `Strict` στον Πίνακα 0.5.3 και τον Πίνακα 0.5.4, αντίστοιχα. Το σύμβολο $\star$ δηλώνει το μοντέλο που υποβλήθηκε στον διαγωνισμό.

0.5.1 `Strict-Small` – Μικρό Σενάριο

Στο `Strict-Small` σενάριο, η σύγκριση της επίδοσης των εκπαιδευμένων μοντέλων για τα σύνολα εκπαίδευσης $\mathcal{D}_{\text{baby-10M}}$ και $\mathcal{D}_{\text{tiny-10M}}$ αποκαλύπτει ότι το σύνολο δεδομένων *BabyLM* είναι πιο ωφέλιμο για την προεκπαίδευση σε γλωσσικά δεδομένα σε σύγκριση με το *TinyStories*. Οι μετρικές απόδοσης για το *TinyStories* είναι σταθερά χαμηλότερες, εκτός από την περίπτωση του EWoK. Ενδιαφέρον επίσης προκαλεί ότι η αντικατάσταση του μισού συνόλου δεδομένων *BabyLM* με δεδομένα από το *TinyStories* επηρεάζει ελάχιστα την απόδοση του μοντέλου. Ωστόσο, καθώς προσθέτουμε περισσότερα δείγματα δεδομένων συνθετικών ιστοριών, ο θετικός αντίκτυπος του συνόλου δεδομένων *BabyLM* αρχίζει να μειώνεται, με αποτέλεσμα η απόδοση να πλησιάζει αυτή του $\mathcal{D}_{\text{tiny-10M}}$ όπου το *BabyLM* δε χρησιμοποιήθηκε καθόλου. Αυτό υποδηλώνει ότι η εκπαίδευση επηρεάζεται υπερβολικά από τον αυξημένο όγκο δεδομένων του *TinyStories*. Για να μετριαστεί αυτό το αποτέλεσμα, ισορροπήσαμε την κατανομή των δειγμάτων από τα δύο σύνολα δεδομένων σε μια παρτίδα (batch). Αυτή η προσέγγιση επηρεάζει θετικά την απόδοση του μοντέλου. Συγκεκριμένα, για το BLiMP, ξεπερνά ελαφρώς την απόδοση του μοντέλου που εκπαιδεύτηκε αποκλειστικά στο $\mathcal{D}_{\text{baby-10M}}$, πετυχαίνοντας την καλύτερη επίδοση συνολικά. Επιπλέον, σε σύγκριση με άλλα σενάρια επαύξησης, η απόδοση στο GLUE αυξάνεται.

Μια ακόμα ενδιαφέρουσα παρατήρηση αφορά την τεχνική δειγματοληψίας που χρησιμοποιείται για την αύξηση των δεδομένων. Η αλλαγή της στρατηγικής δειγματοληψίας από άπληστη κωδικοποίηση σε δειγματοληψία πυρήνα επηρεάζει θετικά την απόδοση του μοντέλου στο BLiMP και το BLiMP Supp., ενώ επηρεάζει αρνητικά την απόδοση στο EWoK και στο GLUE. Αυτό το φαινόμενο είναι πιθανό να οφείλεται στη φύση των ίδιων των μετρικών αξιολόγησης. Το BLiMP εστιάζει στην αξιολόγηση της κατανόησης της γραμματικής και του συντακτικού, και

η αυξημένη ποικιλομορφία που προκαλεί η δειγματοληψία πυρήνα, εκθέτει το μοντέλο σε ένα ευρύτερο φάσμα γλωσσικών πλαισίων και συντακτικών παραλλαγών. Αυτό βελτιώνει την επίδοση στο BLiMP. Αντίθετα, το EWoK και το GLUE απαιτούν σημασιολογική συνοχή και συνέπεια στα γεγονότα, και η αυξημένη ποικιλομορφία από τη δειγματοληψία πυρήνα μπορεί να εισάγει θόρυβο και να οδηγήσει σε λιγότερο συνεκτικές αφηγήσεις. Αυτό δυνητικά μπερδεύει το μοντέλο και μειώνει την απόδοση. Επομένως, ενώ η μεγαλύτερη ποικιλία στις ιστορίες ωφελεί μετρικές γραμματικής κατανόησης όπως το BLiMP, ενδέχεται να μην είναι τόσο χρήσιμη για σημασιολογικές μετρικές ή μετρικές βασισμένες σε γεγονότα, όπως το EWoK και το GLUE.

Model	Training Data	Total	BLiMP	Supp.	EWoK	GLUE	Avg.
LTG-BERT	$\mathcal{D}_{\text{baby-10M}}$	10M	60.6	60.8	48.9/63.1 [6]	60.3	57.7/61.2
BabyLlama	$\mathcal{D}_{\text{baby-10M}}$	10M	69.8	59.5	50.7	63.3	60.8
LTG-BERT (**ours**)	$\mathcal{D}_{\text{baby-10M}}$	10M	62.8	**63.7**	66.2	71.0	**65.9**
	$\mathcal{D}_{\text{tiny-10M}}$	10M	59.8	54.2	**67.0**	67.0	62.0
	$\mathcal{D}_{\text{tiny-10M}} + \mathcal{D}_{\text{gen-greedy}}$	20M	58.7	57.8	63.8	67.1	61.9
	$\mathcal{D}_{\text{baby-5M}} + \mathcal{D}_{\text{tiny-5M}}$	10M	62.6	60.7	66.6	**71.2**	65.3
	$\mathcal{D}_{\text{baby-5M}} + \mathcal{D}_{\text{tiny-5M}} + \mathcal{D}_{\text{gen-greedy}}$	15M	62.1	60.2	65.5	70.6	64.6
	$\mathcal{D}_{\text{baby-5M}} + \mathcal{D}_{\text{tiny-5M}} + \mathcal{D}_{\text{gen-nucleus-1}}$	15M	62.5	62.3	63.9	69.5	64.6
	$\mathcal{D}_{\text{baby-5M}} + \mathcal{D}_{\text{tiny-5M}} + \mathcal{D}_{\text{gen-nucleus-1}}$ † ⋆	15M	**63.2**	59.3	65.5	71.1	64.8
	$\mathcal{D}_{\text{baby-5M}} + \mathcal{D}_{\text{tiny-5M}} + \mathcal{D}_{\text{gen-nucleus-5}}$	33M	62.4	60.1	65.8	69.4	64.4
	$\mathcal{D}_{\text{baby-5M}} + \mathcal{D}_{\text{tiny-5M}} + \mathcal{D}_{\text{gen-nucleus-10}}$	56M	61.0	58.4	65.3	69.5	63.6

Table 0.5.3: Επίδοση των μοντέλων για το `Strict-Small` σενάριο με όριο 10 εκατομμύρια λέξεις.

0.5.2 `Strict` – Μεγάλο Σενάριο

Είναι ενδιαφέρον ότι για το `Strict` σενάριο παρατηρούμε ότι η αύξηση δεδομένων έχει θετική επίδραση στις μετρικές BLiMP και EWoK. Συγκεκριμένα, η προσθήκη του συνόλου δεδομένων $\mathcal{D}_{\text{gen-greedy}}$, έχει ως αποτέλεσμα αυξημένη απόδοση σε σύγκριση με τα βασικά μοντέλα που εκπαιδεύτηκαν στα $\mathcal{D}_{\text{tiny-100M}}$ και $\mathcal{D}_{\text{baby-100M}}$, καθώς και ένα μείγμα των δύο ($\mathcal{D}_{\text{tiny-50M}} + \mathcal{D}_{\text{baby-50M}}$). Επιπλέον, ο συνδυασμός $\mathcal{D}_{\text{tiny-50M}} + \mathcal{D}_{\text{baby-50M}}$ έχει χειρότερη απόδοση και από τα δύο μοντέλα με επαύξηση δεδομένων, $\mathcal{D}_{\text{gen-greedy}}$ και $\mathcal{D}_{\text{gen-nucleus-1}}$, γεγονός που υποδηλώνει ότι τα συνθετικά δεδομένα μπορούν να προσφέρουν κάποιο κέρδος στο μεγαλύτερο σενάριο.

Όπως και με το `Strict-Small` σενάριο, η αύξηση του μεγέθους του συνόλου δεδομένων *TinyStories* επηρεάζει αρνητικά την απόδοση των μοντέλων, πλησιάζοντας την απόδοση του μοντέλου που έχει εκπαιδευτεί αποκλειστικά στο $\mathcal{D}_{\text{tiny-100M}}$. Ωστόσο, σε αυτήν την περίπτωση, η εξισορρόπηση των συνόλων δεδομένων κατά την εκπαίδευση δε βελτιώνει την απόδοση του μοντέλου. Στο μεγαλύτερο σύνολο δεδομένων των 100M λέξεων, ακόμη και με εξισορρόπηση, ο τεράστιος όγκος των δεδομένων *TinyStories* μπορεί να εξουδετερώσει την επίδραση των δεδομένων *BabyLM*. Το μοντέλο εκτίθεται σε πολύ μεγαλύτερη ποσότητα περιεχομένου *TinyStories*, το οποίο μπορεί να κυριαρχήσει στη μάθηση και να μειώσει την αποτελεσματικότητα της εξισορρόπησης. Επιπλέον, ενώ η στρατηγική δειγματοληψίας πυρήνα βελτιώνει για άλλη μια φορά την απόδοση στο BLiMP Supp., δε βοηθά με το BLiMP όπως έκανε στο `Strict-Small` σενάριο.

0.6 Συμπέρασμα

Σε αυτή την εργασία, διερευνήσαμε μεθόδους επαύξησης δεδομένων για γλωσσική μοντελοποίηση σε συνθήκες περιορισμένων δεδομένων. Αρχικά, χρησιμοποιώντας το σύνολο δεδομένων *TinyStories*, εκπαιδεύσαμε αποκωδικοποιητές μετασχηματιστές GPT-Neo, και διερευνήσαμε τη σχέση μεταξύ της παραγωγικής ικανότητας, και του μεγέθους του συνόλου εκπαίδευσης. Τα ευρήματα μας φανερώνουν την ικανότητα αυτών των μοντέλων να παράγουν κείμενα υψηλής ποιότητας και ποικιλομορφίας, ακόμα και σε συνθήκες λίγων δεδομένων.

[6] Τα αποτελέσματα στο EWoK για τα μοντέλα LTG-BERT διαφέρουν από αυτά που δημοσιεύθηκαν από τους διοργανωτές – για παράδειγμα η αξιολόγηση μας για το 10M σενάριο είναι 63.1 αντί για το παρεχόμενο 48.9. Πιστεύουμε ότι αυτό οφείλεται σε αστάθεια της συγκεκριμένης αρχιτεκτονικής, με διαφορετικά αποτελέσματα να αναφέρονται περιστασιακά σε διαφορετικά μηχανήματα.

Model	Training Data	Total	BLiMP	Supp.	Ewok	GLUE	Avg
LTG-BERT	$\mathcal{D}_{\text{baby-100M}}$	100M	69.2	66.5	51.9/65.7	68.4	64.0/67.5
BabyLlama	$\mathcal{D}_{\text{baby-100M}}$	100M	73.1	60.6	52.1	69.0	63.7
LTG-BERT (**ours**)	$\mathcal{D}_{\text{baby-100M}}$	100M	64.0	**67.6**	62.8	**74.0**	**67.1**
	$\mathcal{D}_{\text{tiny-100M}}$	100M	61.2	63.2	63.1	70.6	64.5
	$\mathcal{D}_{\text{tiny-100M}} + \mathcal{D}_{\text{gen-greedy}}$	200M	61.1	59.6	63.8	69.1	63.4
	$\mathcal{D}_{\text{tiny-50M}} + \mathcal{D}_{\text{baby-50M}}$	100M	65.5	65.6	62.5	71.0	66.2
	$\mathcal{D}_{\text{tiny-50M}} + \mathcal{D}_{\text{baby-50M}} + \mathcal{D}_{\text{gen-greedy}}$	150M	**66.6**	63.3	**65.0**	71.8	66.7
	$\mathcal{D}_{\text{tiny-50M}} + \mathcal{D}_{\text{baby-50M}} + \mathcal{D}_{\text{gen-nucleus-1}}\star$	150M	65.6	65.0	64.6	72.7	67.0
	$\mathcal{D}_{\text{tiny-50M}} + \mathcal{D}_{\text{baby-50M}} + \mathcal{D}_{\text{gen-nucleus-1}}\dagger$	150M	65.2	63.5	64.3	72.6	66.4
	$\mathcal{D}_{\text{tiny-50M}} + \mathcal{D}_{\text{baby-50M}} + \mathcal{D}_{\text{gen-nucleus-5}}$	350M	65.4	64.4	61.2	69.8	65.2
	$\mathcal{D}_{\text{tiny-50M}} + \mathcal{D}_{\text{baby-50M}} + \mathcal{D}_{\text{gen-nucleus-10}}$	600M	63.7	63.3	64.5	69.5	65.3

Table 0.5.4: Επίδοση των μοντέλων για το `Strict` σενάριο με όριο 100 εκατομμύρια λέξεις.

Στη συνέχεια, για να μετρήσουμε την επίδραση της επαύξησης με συνθετικά δεδομένα, χρησιμοποιήσαμε τα μοντέλα GPT-Neo για να παράξουμε σύνολα συνθετικών ιστοριών, και εκπαιδεύσαμε μοντέλα LTG-BERT σε εναν ευρό συνδυασμό από σύνολα δεδομένων. Από την αξιολόγηση μας συμπεραίνουμε ότι ενώ η σύνθεση δεδομένων υψηλής ποιότητας είναι δυνατή σε συνθήκες περιορισμένων δεδομένων, η αποτελεσματική χρήση τους για προεκπαίδευση δημιουργεί διάφορες προκλήσεις. Για το `Strict` σενάριο, η χρήση συνθετικών δεδομένων προκάλεσε μικρή βελτίωση σε ορισμένες μετρικές, ενώ η προσεκτική εξισορρόπηση των συνόλων δεδομένων ήταν ευεργετική για το σενάριο `Strict-Small` . Εν κατακλείδι, οι προσπάθειες μας υπογραμμίζουν την περίπλοκη ισορροπία που απαιτείται μεταξύ της ποσότητας, της ποιότητας και της ενσωμάτωσης των συνθετικών δεδομένων για την αποτελεσματική χρήση τους για προεκπαίδευση σε γλωσσικά δεδομένα. Ελπίζουμε οι εργασίες μας να αποτελέσουν έναυσμα για περισσότερη έρευνα στον τομέα της γλωσσικής μοντελοποίησης με λίγα δεδομένα, και σημειώνουμε κάποιες ιδέες για μελλοντικές προεκτάσεις στην επόμενη παράγραφο.

0.6.1 Μελλοντικές Κατευθύνσεις

Στη μελέτη μας χρησιμοποιήθηκε μια μόνο αρχιτεκτονική για κάθε ένα απο τα μοντέλα αποκωδικοποιητή και κωδικοποιητή. Επιπλέον, περιοριστήκαμε σε συγκεκριμένα σύνολα δεδομένων, χρησιμοποιώντας μόνο το *TinyStories* για τη δημιουργία συνθετικών δεδομένων και συνδυασμό των *TinyStories* και *BabyLM* για την εκπαίδευση του κωδικοποιητή. Αυτές οι επιλογές έγιναν για να εξασφαλίσουμε ελέγξιμες συνθήκες στα πειράματα μας, και να εξαχθούν σταθερά συμπεράσματα. Παρ'ολ'αυτά, η εισαγωγή ενός ευρέος φάσματος μοντέλων και δεδομένων εκπαίδευσης σε μελλοντική εργασία θα οδηγήσουν σε μεγαλύτερη γενίκευση των αποτελεσμάτων μας.

Μια άλλη υποσχόμενη κατεύθυνση για μελλοντική έρευνα είναι ο πειραματισμός με τη διαδικασία παραγωγής δεδομένων και συνδυασμού των συνόλων δεδομένων. Ερευνήσαμε μόνο ένα μείγμα των δύο συνόλων δεδομένων – τη συμπερίληψη τους σε ίση αναλογία – κατά τη δημιουργία υποσυνόλων, ενώ τα κείμενα μέσα σε ένα υποσύνολο δεδομένων δειγματίστηκαν τυχαία. Θεωρούμε ότι ένας καλύτερος έλεγχος του μείγματος των συνόλων δεδομένων θα μπορούσε να ενισχύσει περαιτέρω τα οφέλη της τεχνικής επαύξησης δεδομένων που προτείνουμε. Επιπλέον, όσον αφορά την παραγωγή, η στρατηγική προτροπής και το ποσοστό περικοπής των ιστοριών θα μπορούσαν να ρυθμιστούν με μεγαλύτερη ακρίβεια, προκειμένου να βελτιωθεί η ισορροπία μεταξύ της ποιότητας και του πλεονασμού των δεδομένων.

Διαγράφοντας κάποιες μελλοντικές κατευθύνσεις στοχεύουμε να ενθαρρύνουμε περαιτέρω έρευνα σε αυτόν τον τομέα, εστιάζοντας στον αντίκτυπο της αύξησης δεδομένων στη γλωσσική προεκπαίδευση σε συνθήκες περιορισμένων δεδομένων.

Chapter 1

Introduction

Modern Language Models (LMs), have achieved remarkable feats of language understanding and reasoning [16, 148, 149, 40], often surpassing the abilities of humans in certain tasks. To a large degree, this success is due to the continuous increase in the magnitude of two critical variables: the number of model parameters and the size of pre-training datasets. Recent work [50, 65] has shown that as the available compute and training tokens increase, so does model downstream performance, prompting academia and industry alike to keep scaling up training configurations. As an example, the recently introduced Llama 3 [30] model contains 405B parameters and was pre-trained on 15.6T tokens, on 6K H100 GPUs. This trend is also demonstrated by the domination of Natural Language Processing (NLP) research by Large Language Models (LLMs), which have been successfully used in order to enhance an abundance of tasks [92, 107, 93, 69, 68, 3, 96, 133]. However, this comes with significant shortcomings. The current state of LM pre-training raises a variety of ethical and environmental concerns – such as increased energy usage and carbon emissions [128, 48, 76] – and threatens democratic access to the field of language modeling, as research is only made viable for large industry organizations.

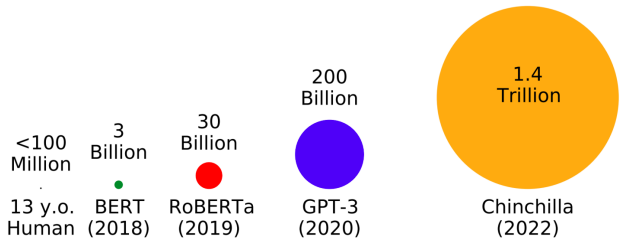

Figure 1.0.1: Training dataset sizes of recently introduced language models compared to the average input that a 13 y. o. child has received during development. Figure reproduced from [145, 141].

Another set of issues relates to the *cognitive plausibility* of current LMs. By observing the human language acquisition process and comparing it with modern LM training regimes, we notice some key differences, specifically concerning the amount of linguistic input required for effectively understanding language. During their development, children are exposed to 2-7M words per day [42], for a total of no more than 100M words by 13 years of age, making them extremely efficient language learners. At the same time, state-of-the-art LMs require billions or even trillions of words. This contrast in sample efficiency, illustrated in Figure 1.0.1, casts doubts on the cognitive plausibility of current LM training procedures. Additionally, the vast scales of data

prohibit cognitive scientists from effectively employing LMs for psycholinguistic research and transferring linguistic insights to human language learning. Indeed, researchers have identified the access to immense amounts of data as the most glaring advantage of modern LMs over humans [77, 141].

The issues with present LM training practices, as outlined above, motivate the exploration of a direction counter to the current trend of scaling up, namely: language pre-training using **data limitations** inspired by human development. Our work is part of recent efforts for training *efficient* language models in a limited data setting, mimicking the linguistic environment that children are exposed to during development. Specifically, we participate in the 2nd iteration of the **BabyLM Challenge**[1] [21, 145, 146]. The goals of the competition are aligned with the benefits of data constrained language modeling, specifically the competition strives to: ① improve the relevance of LMs as cognitive models of the human language acquisition process, ② aid the discovery of more efficient pretraining schemes by experimenting (sandboxing) on small datasets before scaling up promising approaches, ③ make LM research viable for researchers outside well funded industry groups.

The competition features three experimental tracks, the `Strict` track with a maximum of 100M words, the `Strict-Small` track with a maximum of 10M words, and the `Multimodal` track with a word budget of 100M words and unlimited visual input. Participants are allowed to construct their own training data, provided they stay within the track limits. In the following paragraphs, we briefly describe our approach to the `Strict` , and `Strict-Small` tracks of the challenge.

Our work is published in the proceedings of the 2nd iteration of the BabyLM challenge at CoNLL 2024, titled: *"BERTtime Stories: Investigating the Role of Synthetic Story Data in Language Pre-training"* [132]. To encourage and facilitate further research in the area, we make our models and implementation publicly available [2].

1.1 Our Approach

The central idea of our approach is **data augmentation**. We leverage the recently introduced *TinyStories* dataset [31] ($\mathcal{D}_{\text{tiny}}$), a dataset of short stories with simple language and vocabulary, intended to be understood by 3-4 year old children. This dataset was recently used to demonstrate that Small Language Models (SLMs) trained on simple data, can create high quality and novel story completions. Inspired by these results, we posit that harnessing this dataset for augmentation would benefit LM pre-training for the small data regimes (10M and 100M words) of our setting. We use $\mathcal{D}_{\text{tiny}}$ to train decoder transformer GPT-Neo [12] models with limited amounts of story data, and employ them to generate alternate completions to the stories in their training set, adapting the beginning of the stories in $\mathcal{D}_{\text{tiny}}$ as a prompt. This results in the creation of a *synthetic* dataset of sort stories – denoted as $\mathcal{D}_{\text{gen}}$. The $\mathcal{D}_{\text{gen}}$ and $\mathcal{D}_{\text{tiny}}$ datasets are combined with a sample of the *BabyLM* dataset ($\mathcal{D}_{\text{baby}}$) [21], to form a combined corpus ($\mathcal{D}_{\text{comb}}$). Using the combined corpus, we train encoder transformer models with the LTG-BERT architecture [120].

Through our efforts, we aim to uncover the effect of data augmentation with synthetic story data in the linguistic understanding of the encoder models. Specifically, to achieve this we train LTG-BERT models and compare performance with various baselines, trained with and without the generated dataset $\mathcal{D}_{\text{gen}}$. For the proposed models and baselines, we evaluate grammatical proficiency, world knowledge and general language understanding. Some marginal gains are observed from the application of our methodology, for both the `Strict` and `Strict-Small` tracks of the challenge. However, overall, increasing the amount of synthetic story data proved detrimental to final model performance. Our results underscore the potential data augmentation using synthetic story data in improving low resource language modeling.

A second set of investigations concerns the relationship between the size of the training data for GPT-Neo

[1] https://babylm.github.io/
[2] https://github.com/nikitas-theo/BERTtimeStories

models trained on subsets of *TinyStories* and their linguistic abilities. Specifically, we want to understand how the quality of the generations of the models and their general language proficiency change as we vary the amount of words (in millions) that are leveraged for their pretraining. To probe this relationship we construct subsets of *TinyStories* with $\{5, 10, 25, 50, 75, 100\}$M (millions) of words, and utilize a comprehensive evaluation suite including language benchmarks, diversity measures, and an LLM assisted qualitative evaluation.

Our experiments demonstrate that Small Language Models (SLMs), trained with $\leq$ 100M of words, demonstrate remarkable linguistic understanding, and can produce high quality generations with a high degree of grammatical correctness and diversity. These results shed light into the capabilities of SLMs in low data regimes, and give strong evidence that the potential of data augmentation with synthetic story data has not been fully explored. Specifically, given the ability of the GPT-Neo models to produce high quality text, we believe that more research is needed to fully realize and exploit their potential for augmentation in low resource language pretraining.

1.2 Thesis Outline

In the chapters that follow, we expand on the required background knowledge and further motivate the need for sample-efficient language pre-training. Having defined the context of our work and positioned it among current efforts, we begin to illustrate our methodology. Next, we detail our experimental procedure, analyze the results and finally discuss our findings and illustrate future directions.

- We begin with a short introduction to *Language Modeling*, starting with more traditional approaches (e.g., n-grams, static embeddings), and moving to modern developments, with the introduction of Attention, Transformers, and Large Language Models (LLMs) **(Chapter 2)**.

- Next, we summarize recent efforts for *efficient and cognitively plausible* language modeling, aiming to solve the issues raised by current LM training regimes **(Chapter 3)**.

- Having set the context of our work, we describe in detail our *methodology for data augmentation* in low resource training conditions using a synthetic dataset of sort stories **(Chapter 4)**.

- We present our *experimental procedures* and *findings* in detail, and demonstrate evaluation results for GPT-Neo decoders trained on various amounts of *TinyStories* data, and the effect of data augmentation on the linguistic abilities of the LTG-BERT encoder models **(Chapter 5)**.

- We conclude with a *discussion* of our results, and an illustration of promising avenues for *future work* **(Chapter 6)**.

- Details and additional results from our experiments are included in **Appendices A, B, C, D** .

Chapter 2

Language Modeling

Contents

In this chapter, we delve into the subject of **Language Models** or **LM**s. These are models that have learned how to assign a probability to a sequence of words, or given an incomplete sequence, assign a probability to all possible next words. It turns out that learning to predict the next (or missing) word in a sequence is a sufficiently complex task, thaw allows models to learn about fundamental semantic and syntactic properties of language. These properties are expressed in the model parameters, and in the rich representations that can be derived about a word or a sentence. Even modern Large Language Models (LLMs) that show remarkable linguistic abilities and excel in a variety of language reasoning and understanding tasks, at their core, are trained at predicting upcoming words from words in their neighborhood. Language modeling, is at the heart of Natural Language Processing (NLP), as in order to succeed at a language task, some general knowledge about the structure of language and a representation space for linguistic units are extremely helpful.

Language is also the human ability most connected with human though and intelligence. To uncover the secrets of language acquisition, and teach machines to read, write and communicate, has been one of the longstanding research challenges of the field of Artificial Intelligence [136].

Figure 2.0.1 illustrates the evolution of language modeling through the 20th and 21st centuries, up to today – highlighting the drastic increase in the *task solving capabilities* of language models. Initially, *statistical* language models were primarily used, in order to support specific tasks (e.g., information retrieval, speech processing) in which the estimation of probability proved beneficial. With the introduction of neural LMs (e.g., word2vec [91]), the focus moved to learning general linguistic representations, i.e., *features*, which where employed to solve a limited range of typical NLP tasks, and reduced the need for manual feature engineering. The next step was the learning of *context-aware* representations, with the emerging paradigm of unsupervised pre-training on vast amounts of data, and finetuning on a limited set of labels. Models such as ELMO [103], BERT [27] and GPT-2 [111], were applied to a broader range of NLP tasks – such as sentiment classification,

paraphrasing, and entailment. The most recent development is the advent of *Large Language Models* or LLMs [2, 16, 95] which can be considered as *general purpose* task solvers – giving rise to scaling laws [50, 65] and instruction tuning and prompting methods [19, 158]. These models can solve a wide variety of *real-world* tasks, and have effectively transformed modern information processing. During this evolution of language modeling, we witness that the *scope* of the problems that LMs can tackle as well as their *performance* have been greatly advanced. In the following sections, we dive through each of the periods of LM development, and gain insights about the past and future of language modeling.

We begin our exploration of language modeling by first briefly talking about *N-grams* the simplest form of language models (Section 2.1). Afterwards our analysis progresses to static word representations, like word2vec, and neural language models with recurrent architectures (Section 2.2). Next, we arrive at the Transformer, and the modern attention-based LM architectures that revolutionized the NLP field (Section 2.3). Lastly, we introduce the current LLM landscape, and underscore potential challenges (Section 2.4).

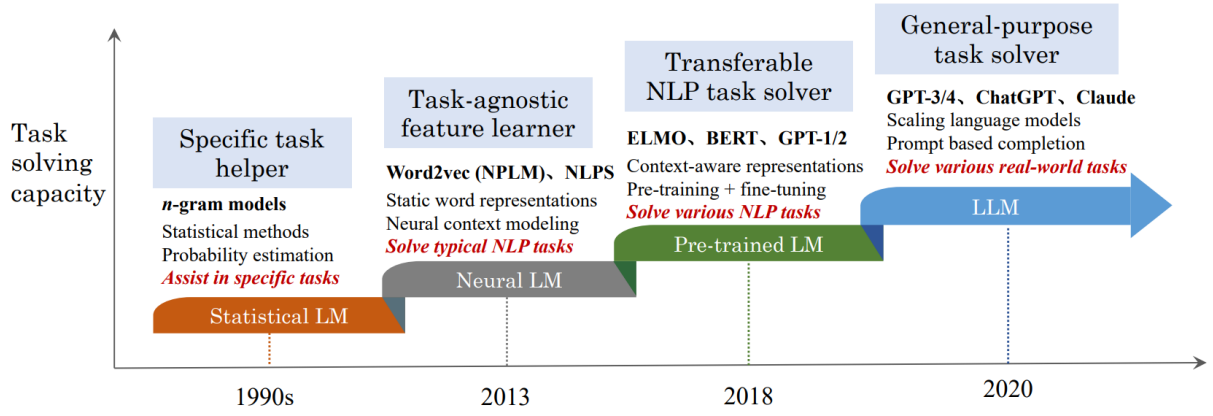

Figure 2.0.1: Evolution of Language Models through four periods. Each period is marked by substantial increases in the task-solving capabilities and scope of the models. Figure from [162].

2.1 Statistical LMs – N-grams

N-gram language models fall into the general category of *statistical LMs* [82, 58] which were some of the earliest attempts to model language. These techniques construct a word prediction model, by relying on the *Markov Assumption* – the usage of only the recent history to predict the next possible word. Given a sequence of n words, denoted as $w_{1:n}$, we can compute the probability $P(w_{1:n})$, using the chain rule of probability. We note here that the term "words" could mean linguistic tokens in general, as we shall see that modern LMs do not process words as individual units. With the chain rule of probability we have: $P(w_{1:n}) = P(w_1)P(w_2|w_1)...P(w_n|w_{1:n-1}) = \prod_{k=1}^{n} P(w_k|w_{1:k-1})$. Calculating the probability $P(w_k|, w_1, ..., w_{k-1})$ however is intractable for large sequences, as both empirical and statistical estimation methods are not applicable without making further assumptions – even when relying on a trillion word corpus like those used by LLMs. This is a consequence of the exponential increase in the number of unique word sequences as the number of words n in the sequence $w_{1:n}$ increases.

Instead of computing the probability of a word given the complete history, we can instead approximate the past context by using a limited number of words. By taking into account only *the previous* word in a sequence, we make the following assumption: $P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-1})$ – the probability of the current word, depends only on the probability of the previous word. This way we formulate a *bi-gram* LM. This general assumption – that the current word prediction depends only on the most recent context – is also called the *Markov Assumption*. The bi-gram model is easily generalized into the tri-gram and the *N-gram*,

which computes the probability by looking $N-1$ words in the past. In general, the probability of a sequence using an N-gram language model is expressed as: $P(w_{1:n}) = \prod_{k=1}^{n} P(w_k|w_{k-1:k-N})$.

To estimate the probabilities for N-gram models, we use *Maximum Likelihood Estimation*. Specifically, the model parameters are computed from co-occurency counts in a large corpus, which are then normalized to lie in the range $[0, 1]$. This is a *frequentist* approach to probability. With $C(w_{1:n})$ we symbolize the number of times the sequence $w_{1:n}$ occurs in the text data, and give the general equation for empirically estimating the probabilities for an N-gram model below:

$$P(w_n|w_{n-N+1:n-1}) = \frac{C(w_{n-N+1:n-1}w_n)}{C(w_{n-N+1:n-1})}$$

Using this simple approximation, N-grams are able to capture basic grammatical and syntactic facts about language, and have been useful in information retrieval, and early natural language processing. However, they often suffer from the *curse of dimensionality*, as it becomes increasingly difficult to estimate the probabilities for higher order models (large N values) – since the number of probabilities that needs to be calculated grows exponentially.

2.2 Neural LMs

Neural Language Models [7, 90] leverage Neural Networks (NNs) to estimate the probability of a sequence of words. Common architectures include Multi Layer Perceptrons (MLPs) and Recurrent Neural Networks (RNNs) e.g., the LSTM [49]. These methods focus on deriving dense *features* for words and sequences that can then be utilized to increase performance in a variety of downstream tasks, by leveraging pre-learned semantic information. As an example of an approach from that time, we detail in the next section the core idea behind *word embeddings*, and specifically word2vec [91] – one of the most influential contributions in modern NLP.

2.2.1 Word Embeddings

The idea of word embeddings, is build around a simple concept, called *the distributional hypothesis*: words that occur in similar contexts, tend to have similar meaning. This hypothesis was formulated by linguists long before the advent of word embeddings and Neural LMs. Vector semantics, refers to learning representations of meaning, in some multidimensional space, that are called embeddings. These representations are learned directly from the distribution in texts, and effectively underlay every natural language processing application that relies on the meaning of the words. Here, we introduce *static* embeddings, where each word has a fixed vector representation. In later sections, we will talk about *contextual* representations, such as those obtained by a bidirectional transformer BERT [27] model.

Next, we introduce one of the most influential word embedding methods, word2vec [91]. Unlike previous embedding approaches which were *sparse*, e.g., like TF-iDF [117] and frequency count based vectors, word2vec embeddings are short and dense, with the number d of dimensions ranging from 50 to 1000 [63]. Their values are real numbers, and the embeddings are vectors $e \in \mathbb{R}^d$. Dense vectors in general are more effective than sparse vectors, in almost every application. One reason for this, is their efficiency, as being dense allows them to pack a lot more information for each used dimension. This benefits classification networks, as far fewer parameters need to be learned and stored when working with these representations – which also combats overfitting. However, one undesirable effect of this, is that the dimensions are not *interpretable* – an issue that is present in general with modern NNs and has sparked its own research field [118].

The word2vec embeddings dispose of counting-based co-occurence methods, by training a classifier to predict if a word is correct for a given context. The classifier *weights* are then used as the word's embeddings. This

Figure 2.2.1: T-SNE projection in two dimension, for the embeddings corresponding to some words and phrases. The image demonstrates that words with similar meanings are grouped close together in the representation space. The original embeddings were trained for the task of sentiment analysis, and have 60 dimensions. Figure reproduced from [63, 74].

method relies on *self-supervision*, avoiding any use of a human label signal during training. This is one of the most important ideas in NLP, as it makes it possible to train on vast amounts of unlabeled text. Given a context of n words $W = (w_1, w_2, ..., w_c, ..., w_n)$ and a target word w_c, we perform a *binary classification* task, where we want either (a) to predict if the word w_c is in fact correctly placed in the context (CBOW), or (b) to predict if the context $W_{\setminus c}$ is the correct context for this word (skip-gram). For skip-gram, the real target word and context are treated as *positive* examples, while randomly sampled words are chosen for *negative* examples, and a logistic regression classifier is trained to recognize between the two. The learned weights of the classifier, serve as representations for the words. The model uses the dot product of the embeddings to estimate the similarity between words. During learning, target and context words that are similar are moved closer together, resulting in an embedding space with useful similarity properties. Later, more advanced embedding techniques were introduced, such as fastText (using sub-word models) [13] or GLoVE [102] embeddings (using global statistics).

Embedding spaces can capture interesting semantic relationships and interactions between words. Figure 2.2.1 demonstrates how words that have similar connotations (negative/positive/neutral) are grouped together in the embedding space. Additionally, embeddings spaces can foster semantic relationships between vectors, that e.g., encode similarity and word relations. One notable example is given by [89]: vector("King") - vector("Man") + vector("Queen") $\approx$ vector("Queen").

2.2.2 Recurrent Language Models

We briefly introduce Recurrent Neural Networks (RNNs) that were the dominant paradigm for modelling long-range sequences and dependencies – including language modeling – before the introduction of Transformer models. An RNN is able to account for dependencies across time in a sequence, and can thus model the probability $P(w_i|w_{<i})$ without making the Markov assumption that we observed in N-gram models. RNN language models, have internal mechanisms that can directly capture the inherently sequential nature of human language, and the current word prediction can effectively depend on information from hundreds of words before. The recurrent computation taking present inside an RNN, unfolded through time is illustrated in Figure 2.2.2. At each timestep t, a hidden state h_t is responsible for preserving the contextual information from previous steps $x_{<t}$ that will influence the processing of the current input x_t and the production of an output o_t. Moreover, the state h_t operates as a type of memory, that can be modified in complex ways.

One issue with general RNN architectures is that they are difficult to train, and learning long-term depen-

dencies with these models is hindered by problems such as *vanishing* or *exploding* gradients [6]. This is a result of the continuous application of matrix multiplications – proportional to the sequence length – which can drive gradients to zero. To adress these issues, the Long Sort-Term Memory (LSTM) architecture was introduced [49] (among others), that enables a model to decide what information to keep, by *learning* and *forgetting*. Units with specialized functions, named *gates* control how information is accessed. Namely, the LSTM makes use of the *forget*, *add*, and *output* gates. For a detailed description of the LSTM architecture, we refer to [63]. RNNs are very powerful models, that can be applied to sequence labeling (POS tagging), sequence classification (e.g., sentiment analysis) and language generation. For these uses, a special *encoder-decoder* formulation may be employed, and also an *attention* mechanism may be used. The encoder model produces a contextual representation c_i for each token, which is then attended to be a decoder model.

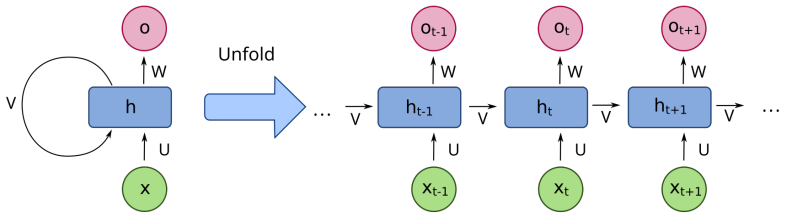

Figure 2.2.2: Illustration of the recurrent computation in an RNN, unfolded through time. Figure by [32].

RNNs can be readily applied to the task of language modeling, as they can be trained to predict the next word in a sequence given the context of the previous words. Specifically, they can model $P(w_i|w_{<i})$, assigning a probability to every possible next word, and defining a *distribution* over the whole vocabulary. Utilizing the chain rule of probability, this also allows them to assign a probability to a sequence: $P(w_{1:n}) = \prod P(w_i|w_{<i})$. Previous LMs, whether N-grams or MLPs, had a restriction in the context to a specific number of words, in contrast RNN LMs [90] can process an unlimited context. They achieve this by breaking computation to one individual word at a time, with the hidden state h_t summarizing necessary information from past words. This allows them to theoretically capture dependencies to the start of a sequence. Figure 2.2.3 illustrates the training of an RNN neural network. RNN LMs are trained with self-supervision to predict the next word given past context. They use the cross-entropy loss and a one-hot encoded label signal, indicating the correct next word according to the training data.

RNNs were overshadowed by the advantages of the Transformer architecture [137], that by introducing *self-attention*, dominated performance in many NLP tasks, where traditionally state-of-the-art approaches utilized recurrent architectures e.g., the LSTM. One of the major advantages of Transformers is that representations can be computed *in parallel* allowing for faster and more efficient computation. This is in contrast to RNNs, which are bound by their linear sequential dependency, i.e., the computation for x_i needs to happen after the computation for $x_{<i}$. However, recently RNN LMs have become relevant, offering competitive performance with comparative transformer architectures by [108, 109, 33]. These methods propose advances in the RNN architecture, that allow parallel computation.

RNNs were also the first architectures utilized to derive *contextual* representations for words. In the landmark work of ELMo [103], a deep bidirectional LSTM language model was training (BiLSTM), that in contrast to static embeddings (e.g., word2vec) could capture the *dynamic* context around a word. Two LSTMs were used, trained to predict the next word from each *direction*, left to right, and right to left. The representations where combined, resulting in a richer overall word embedding. ELMo was also one of the first models used for *transfer learninig* in NLP, offering substantial improvements to many tasks, and initiating the era of pre-trained LMs, a major shift in the field.

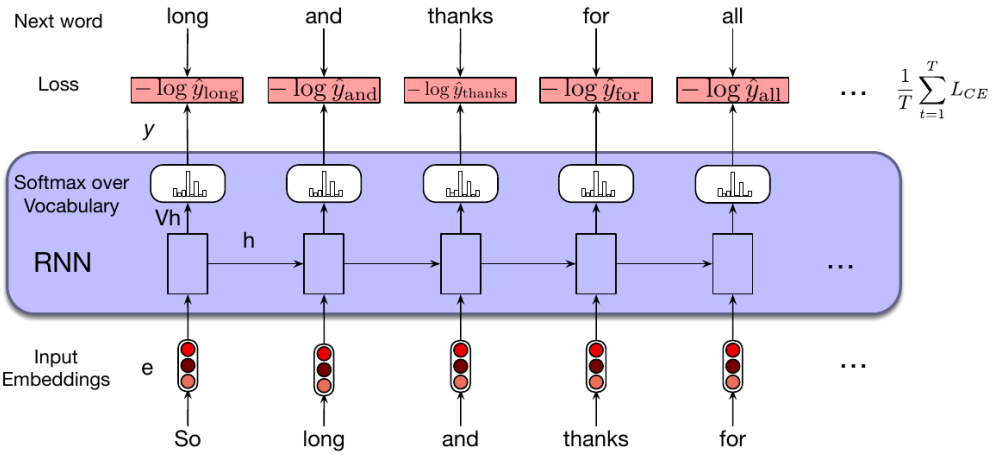

Figure 2.2.3: Training an RNN as a Language Model. Figure reproduced from [63].

2.3 Pre-Trained LMs – Transformers

We now delve into the *Transformer* – which has become the standard architecture for building LMs, and has essentially dominated every NLP application. The original architecture by [137] is illustrated in Figure 2.3.2. The name derives from the way transformers process the representation vector u_i for each token, using a cascade of transformations $u_i \in U \rightarrow V$, where the spaces have the same dimensionality d, but V will have *richer* and generally more useful representations. Originally introduced for NLP [137, 27] they have become ubiquitous in all machine learning applications, outperforming previous CNN and RNN architectures [66, 101]. The transformers, with the introduction of the *self-attention* mechanism, were able to achieve very high parallelization, making them more *efficient* architectures for learning language, compared to RNNs. Transformers are very effective, and can be used as *foundational* models, by being pre-trained on a vast number of training data, and learning rich word representations that are leveraged for state-of-the-art performance in a large array of downstream NLP tasks. This is the now common paradigm of first "pre-training" and later "fine-tuning" that was sparked by the initial employment of contextual representations in ELMo [103], and the later introduction of BERT [27]. In BERT context-aware word representations are learned by self-supervised training on a large corpus, and they then serve as general-purpose features.

The key feature of the transformer is *self-attention*, which we introduce in later paragraphs. Essentially, it is a way to build contextual representations of a token by aggregating the information of surrounding tokens. This form of communication happens at each layer, using as inputs the representations from the previous layer, and can aid the model in learning the relations of tokens across long spans of text. Over the application of many layers, richer and more high-level representations are developed. Below we give a brief description of the different types of transformer architectures and their strengths and weaknesses.

Encoder-only Models These models excel in tasks that require *understanding* of the input, such as sentence classification, and named entity recognition. The use of context-sensitive representations – employing both the left and the right context of a word – gives them the ability to extract meaningful information about a sequence. During training, words are randomly replaced with a special `[MASK]` token, and need to be predicted using the surrounding context, in what is known as the cloze task. Encoder transformers learn the probability distribution $P(w_i|W_{\setminus i})$, where $W_{\setminus i} = (w_1, ..., w_{i-1}, w_{i+1}, ..., w_n)$. These models remain state-of-the-art in Natural Language Understanding (NLU) tasks such as those included in (Super)GLUE [138, 140]. Example models are BERT [27], RoBERTa [80], and DeBERTa [47]. An illustration of the BERT architecture is given in Figure 2.3.1.

Decoder-only Models This family of models demonstrates excellent performance in generative tasks, such as text generation. The models learn to predict words in a causal or autoregressive way, similar to the

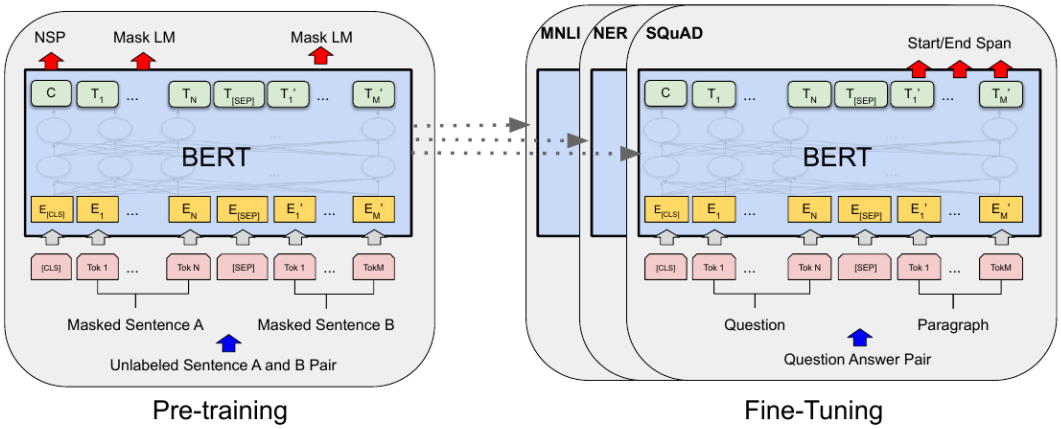

Figure 2.3.1: Illustration of the BERT Architecture. This figure demonstrates the first "pre-training" and later "fine-tuning" paradigm. During pre-training, some tokens are masked and predicted using the context of their surroundings. A special Next Sentence Prediction (NSP) objective is also employed. The contextualized representations are leveraged during finetuning on specific tasks (e.g., MNLI [138], Named Entity Recognition – NER, and SQuAD [113]). Figure reproduced from [27].

training of RNNs – learning the probability distribution $P(w_i|w_{<i})$. The term "autoregressive" stems from the way these models generate text one token at the time, which is then fed back to the model as the next input, in order to generate the following token. They are able to perform all the same tasks as encoder-only transformers, but might show a small decrease in performance. Additionally, this kind of architecture powers the modern LLM revolution, with the development of autoregressive models such as GPT-4 [95] and Claude [2] that dominate the NLP scene and fundamentally transform industry. Example models include the GPT [15, 110] and Llama [30] families of models.

Encoder-Decoder models These models are also known as sequence-to-sequence (**seq2seq**) models, and are good for generative tasks that map an input to an output sequence, e.g., translation and summarization. Rich contextualized representations are computed for the input sequence by the encoder module, which are then passed to the decoder part of the model. The decoder leverages these representations and learns to construct the output sequence by autoregressive next word prediction. The advantage of this type of architecture is that it can combine the benefits of both encoders – that can *understand* a sequence and extract information, and decoders – that can employ this understanding, to *generate* an output sequence. The decoder can use a different language or modality than the encoder, and the input and output in general do not have to be similar – e.g., they can have different lengths, as in summarization. Pre-training can be done with encoder or decoder objectives, but usually is significantly more complex. The encoder-decoder model learns the probability distribution $P(y_i|\text{Encoder}(x), y_{<i})$ where x and y are the input and output sequences. Example architectures include the original transformer by [137], BART [73], T5 [112] and ELECTRA [23].

2.3.1 Self-Attention

We now introduce the mechanism of *self-attention* that sits at the heart of transformer models. Static word embeddings in word2vec [90] can help LMs understand the semantic properties of words and relate them to one another. However, they do not account for the *context* of words, as a word might have multiple senses, and its meaning might change dramatically with the surrounding words. The attention layer in the transformer is a mechanism that helps model compute contextual representations of words, by integrating information from relevant nearby tokens. By stacking multiple attention layers, we can build incrementally more rich and meaningful representations. Each layer uses information from the previous layer, and information from all the other tokens, to derive the representation for the next layer. A visualization of the way that tokens attend with each other in the transformer is illustrated in Figure 2.3.3.

We describe formally the attention mechanism. The attention head of the transformer, allows for the representation of *three distinct roles* for each input embedding $\mathbf{x}_i$ during the attention process. ① The element $\mathbf{x}_i$ can be the one *currently* considered whose representation we are calculating by comparing it to other inputs. This role is referred as **query**. Intuitively, the current token has to define what information from the other tokens is relevant to it. ② It can also act as part of the context, denoted as **key**, when it is being compared to the current input to assess its similarity to it. We can think of this role as the token demonstrating what information it has to offer. ③ Lastly, it can act as a **value**, when it gets weighted and summed in order to form the next representation of the current element, transmitting in this process the information it contains. These roles are captured with three matrix transformations applied to the representation $\mathbf{x}_i$, namely: $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$. By matrix multiplication, the input $\mathbf{x}_i$ projected to a different representation space according to the role it takes as a query, key or value. To enable *parallelization*, we compute these representations for all vectors $\mathbf{x}_i \in \mathbf{X}$ at the same time, forming the following equations:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \mathbf{K} = \mathbf{X}\mathbf{W}_K, \mathbf{v} = \mathbf{X}\mathbf{W}_V$$

Where $\mathbf{Q}, \mathbf{W}, \mathbf{K}$ contain the query, key, and value representations for all the vectors of the input sequence $\mathbf{X}$ in a layer of the transformer. To compute the similarity between two elements $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}$ we use the dot product between the query matrix $\mathbf{Q}$ and key matrix $\mathbf{K}$. This product is also scaled by the dimension of the representations d, to avoid numerical instability issues. We compute now the *attention* product for a token $\mathbf{x}_i$, by summing the values of the other elements $\mathbf{x}_j$ weighted by the similarity of $\mathbf{x}_i$'s query and the other tokens key vector representations. This can be done succinctly by calculating the product $\mathbf{Q}\mathbf{K}^T$. These scores are scaled and a Softmax is applied to produce the final attention score. which is then multiplied with the matrix $\mathbf{V}$ – the collection of all the value vectors, to get a representation for each token. The equation of the matrix $\mathbf{A}$ for the resulting representations of tokens in $\mathbf{X}$ is shown below:

$$\mathbf{A} = \text{Softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V}$$

In practice, the attention between the tokens is *restricted*, according to whether we are training an *encoder* or *decoder* type transformer LM. For encoders, all the tokens are free to attend to all others, but for decoders each token only attends to *previous* tokens. To enhance the representation abilities of transformer models, *multi-headed* attention is used, where multiple attention heads are computed in parallel, and the output of all the heads is concatenated to form the final representation. Intuitively, we can think that each head will *specialize* towards a specific function. Lastly, a matrix $\mathbf{W}_O$ is used to map the concatenated vector of all the heads, into a representation with the correct shape for the next layer. One important observation is that the complexity of the attention layer is $O(N^2)$ where N is the sequence length, thus attention is a computationally *expensive* process. This has resulted in many efforts to decrease the required computation, which are summarized in [131].

In Figure 2.3.2 we see the full architecture of the transformer from [137], which is an *encoder-decoder* originally implement for machine translation. We notice that apart from the *Multi-Head Self-Attention* block that we described in the previous paragraphs, some other critical components are included. LayerNorms are employed to normalize values and keep them in the required range for learning, and residual connections allow the free passing of gradients and information from higher to lower levels – avoiding issues of *vanishing gradients*. An important observation is that feed-forward layers are at the heart of the computation of the transformer. We can think of the transformer architecture as a machine that *"thinks"* using the feed-forward layers, doing the hard computation, and then this information is disseminated and *shared* across tokens in the Multi-Head Self-Attention modules. Lastly, information about the position of tokens is encoded in the transformer by the summation of content and positional vectors. In the original transformer these were a combination of sine and cosine functions at different frequencies, but were later proven to be ineffective, and more advanced

techniques have since emerged [46, 130]. Describing the full complexities and intricacies of the transformer architecture is outside the scope of this thesis, and we refer to comprehensive resources, such as [63].

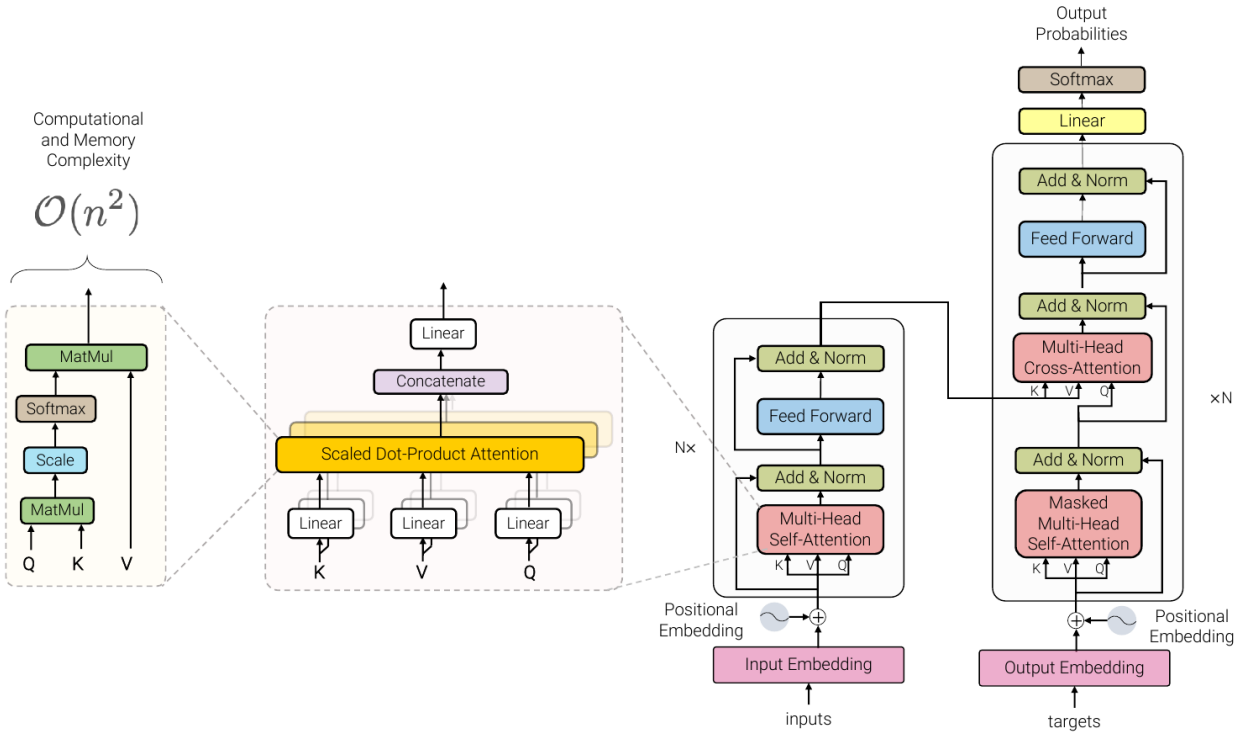

Figure 2.3.2: Illustration of the standard Transformer Architecture [137]. Figure from [131]

2.3.2 Tokenization

We now focus on *tokenization*, as it is a process at the core of modern LM pre-training, that is often overlooked and treated as an implementation detail. Modern tokenization focuses on creating efficient *subwords*, i.e., parts of words that will be used at the input and output representation layers of the LM. Each subword corresponds to an *integer* that indexes into an *Embedding* table (i.e., a row of a dense matrix W) to produce the representation for this subword. This embedding is then inserted into the LM for computation. Conversely, when the LM makes a prediction it outputs a series of sub-word integers which are then converted back into a sequence of words.

Tokenization aims to solve a simple problem that modern LMs face – how best to represent linguistic input for optimal processing. Two extreme ends are either word-level representations, or character-level representations. Both of them suffer from their own set of issues. When dealing with characters, we have the benefit of a very small embedding matrix W and thus a reduction in *model parameters*. Moreover, the relations between words are accurately represented, as no lexical information is lost. The major issue is that this exercises a strong limit on the effective context of the model, as characters easily take up the whole sequence length and are not information dense. On the other end, whole word representations result in a very large embedding matrix and a waste of model parameters, and dealing with out-of-vocabulary (OOV) words becomes problematic. To solve these issues **sub-word** level tokenization emerged – offering a balance between context length, embedding matrix size, and unkwown word representation. Some recent approaches also focus on byte-level representations, trying to eliminate the need for tokenization [157].

We also note that tokenization is the cause for many of the issues that current LLMs face, such as the

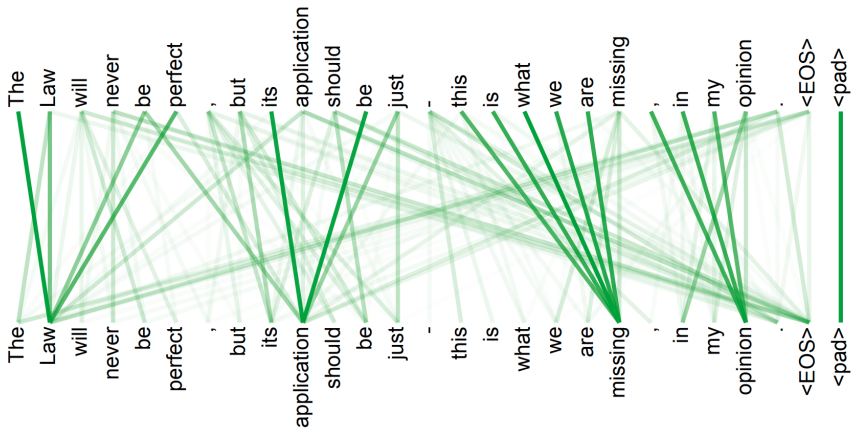

Figure 2.3.3: Attention visualization – words are related according to the *structure* of the sentence. For example, "law" and "application" strongly relate to nearby words, as they are the central words in their context. Figure reproduced from [137].

inability to spell words correctly, count the letters in a word, reverse a string, and also issues with non-English languages. For an interesting study into the possible effects of tokenization that can cause extreme model performance, we point to this blog post [1]. Lastly, we give an example of a tokenization scheme in Figure 2.3.4 – utilizing Tiktokenizer [2].

```
Once·upon·a·time,·there·was·a·sailor·named·Tom.·Tom·had·a·big·boat.·He·liked·to·sail·on·the·sea.·One·day,·Tom·saw·a·little·fish.·The·fish·was·sad.·It·was·lost·and·wanted·to·go·home.\n
Tom·said,·"I·can·help·you,·little·fish.·You·can·fit·in·my·boat,·and·I·will·take·you·home."
```

```
7454, 2402, 257, 640, 11, 612, 373, 257, 43272, 3706,
4186, 13, 4186, 550, 257, 1263, 8848, 13, 679, 8288, 2
84, 14936, 319, 262, 5417, 13, 1881, 1110, 11, 4186, 2
497, 257, 1310, 5916, 13, 383, 5916, 373, 6507, 13, 63
2, 373, 2626, 290, 2227, 284, 467, 1363, 13, 198, 1378
7, 531, 11, 366, 40, 460, 1037, 345, 11, 1310, 5916, 1
3, 921, 460, 4197, 287, 616, 8848, 11, 290, 314, 481,
1011, 345, 1363, 526
```

Figure 2.3.4: Illustration of tokenization using a sample of the TinyStories data [31], and the Tiktokenizer – an online playground for OpenAI tokenizers. We used the GPT-4 tokenizer.

[1] https://www.lesswrong.com/posts/aPeJE8bSo6rAFoLqg/solidgoldmagikarp-plus-prompt-generation
[2] https://tiktokenizer.vercel.app/

2.4 Large Language Models

Another advantage of the transformer architecture is the so-called scaling hypothesis, where simply increasing a model's size, training data, and compute indefinitely can lead to more gains in performance. Indeed, many researchers have attempted to map the optimal ways in which to increase and tune these crucial parameters – discovering interesting *scaling laws* [65, 50], and these efforts have led the community to create models with trillions of training tokens and huge model parameter counts, e.g., the recently introduced Llamma 3 models [30]. An illustration of the relationship between model performance and these scaling variables is illustrated in Figure 2.4.1. These advances led to the era of *Large Language Models* (LLMs), such as Claude Sonnet [2] and GPT-4 [95], that are able to solve various *real-world* tasks. LLMs have shown remarkable understanding and reasoning abilities [16, 148, 149], many of which emerge only at very large scales – often referred to as *emergent abilities*. This has led to some researchers deeming them as indications of General AI [17]. For a recent review of LLMs we point to these studies [92, 107, 93]. LLM's have revolutionized the way that humans interact with and utilize AI models, disrupting virtually every domain of business and intellectual labor. The major approach to interfacing with these models is by the use of *prompting* [20].

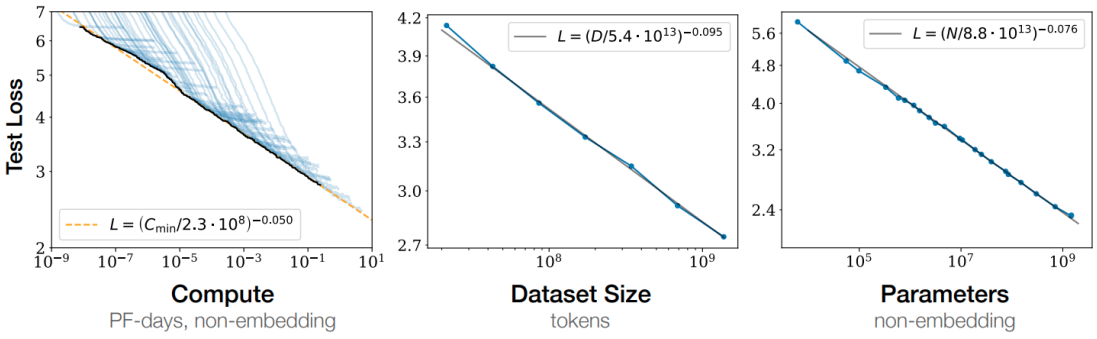

Figure 2.4.1: The performance on Language Modeling (measured by the test loss) improves smoothly, as we increase the model size, dataset size, and amount of compute. To achieve optimal performance, *all three factors* must be scaled up simultaneously. The empirical performance of LMs relates to each parameter with a *power-law* – if we apply no constraints to the other two. Figure reproduced from [65].

Issues with current LLMs However, we note that LLMs also come with significant shortcomings. One emerging issue, recognized by many researchers [128, 48, 76], is the huge *energy demands* for LLM training, due to the massive amount of compute and vast GPU clusters required for their development. This raises concerns about the environmental impact of LLMs as, for example, there are increased CO2 emissions. One related issue is that of *accessibility* as training and operating these models requires high-end computational infrastructures and resources, making research and deployment inaccessible to small academic groups and industry organizations. This keeps current AI centralized only to a select few tech "giants" and results in a lack of democracy [114]. Additionally, LLMs are susceptible to attacks, putting the *privacy* of users at risk – which is especially important with the adoption of AI in everyday workflows by a general audience of all ages [18]. Lastly, LLMs diverge from the way humans process and learn language, and their data demands have been identified as one of the most important ways in which they differ from human learning constraints [77, 141]. Due to this, these models cannot inform psycholinguistic research, and can't be used to learn more about the human language acquisition process. This list of concerns about LLMs is by no means exhaustive, but highlights some of the issues of the current LM training regimes. This section motivates the introduction of the next chapter on *data-efficient* and cognitively inspired LMs, that aim to address some of the aforementioned issues.

Chapter 3

Cognitively Plausible and Sample-Efficient Language Models

Contents

In this chapter, we summarize efforts in *sample-efficient* language modeling that emerged as a response to the trend of continuously increasing model parameters and data sizes in LM training. This direction of research aims to mitigate the diverse range of issues raised by the modern LLM training regime, outlined previously in the introduction – Chapter 1, and our survey of language modeling – Section 2.4. We begin by introducing efficient LMs, focusing on investigations of sample-efficiency and reviewing research in computational linguistics that probes the relationship between the size of training data and model performance (Section 3.1). We then present the BabyLM Challenge [21, 145, 146] – as it has been one of the major community-led efforts for sample-efficient and data constraint language modeling (Section 3.2). We give a brief overview of the different techniques employed as part of the challenge, and touch more upon *data augmentation* methods, as they form the backbone of our proposed methodology – explained in Section 4. Finally, we delve into Small Language Models or SLMs – as opposed to LLMs – and analyze a recent work on *story generation* demonstrated by *TinyStories* [31], that inspired our work (Section 3.3).

3.1 Sample-Efficient Language Modeling

The term "efficient LMs" can have many meanings, depending on the parameter that is being optimized to achieve efficiency. One line of work aims to reduce the computational demands of the Transformer, the base of most modern LMs, as the attention mechanism is an expensive operation with a complexity of $O(N^2)$ where N is the sequence length. These approaches propose new modifications to the Transformer architecture, and we point to a recent survey by [131] for more details. Other areas of improvement include reducing the cost of finetuning and inference (e.g., LoRA [52]) or the introduction of more efficient pre-training techniques (e.g., mixed precision [88]). There is also work on creating LMs with fewer parameters (e.g., by knowledge distillation [121, 59]). Lastly, a range of works seek to optimize the training process by limiting the available compute, time, or budget. Notable approaches include MosaicBERT [105], and BERT models trained on a 24-hour budget [37, 57] using lower grade servers or consumer GPUs. These methods utilize architectural modifications, more efficient training objectives, and hyperparameter tuning.

However, none of the approaches detailed so far focus on training LMs with limitations on the available data, and this area of research remains relatively unexplored. Recently, some approaches emerged towards this direction, with the introduction of LTG-BERT [120] – an architecture optimized for low-data regimes, and community efforts such as the MiniPile Challenge [64]. In a relevant line of work, there has been research in discovering the relationship between LM capabilities and the size of their training data. We detail these efforts and how they connect to human linguistic research in the next paragraph.

Work done in BabyBERTa [54] modifies the RoBERTa [80] architecture to never predict un-masked tokens, and trains a 5M parameter model in a 5M word corpus. They show that this modification allows their model to acquire grammatical knowledge comparable to the original RoBERTa model. The authors also investigate the effect of data domain on performance, and underscore the opportunity that LMs possess for psycholinguistic research. In the study of [144], they evaluate the ability of RoBERTa models to learn linguistic features with small quantities of data, ranging from 1M to 1B words. They find that while linguistic representation is acquired early, preference comes later. In [160], the authors investigate the need for billions of training data, by evaluating different LM abilities as the available data ranges from 1M to 30B words. They find that 10M to 100M words are enough to perform well on most grammatical and semantic evaluations, but commonsense reasoning and understanding require a larger amount of data. Inverse scaling laws were tested in [85], finding that smaller LMs can outperform larger ones in specific problems, and scaling might not be enough for solving some tasks. Early work by [97] evaluated grammar acquisition from limited input data for LSTM models. A survey by [141] outlines the possibilities of using LMs to learn more about the human language acquisition process. The steps towards more human-like linguistic generalization in LMs are discussed in [77]. Lastly, in Figure 3.1.1 we include results from the experiments of [160], that illustrate how model abilities scale with training data.

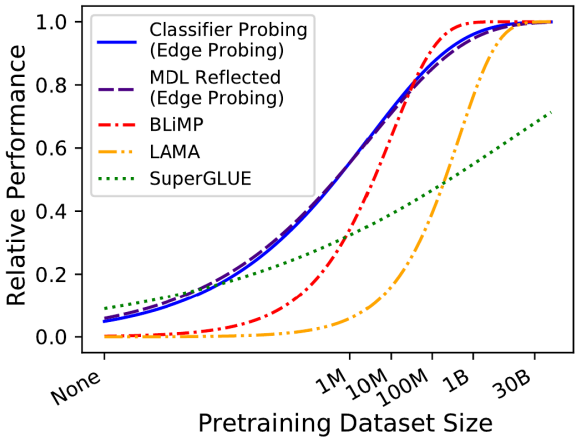

Figure 3.1.1: Effect of the size pre-training data on various linguistic benchmarks and probing methods. Classifier and MDL probing test models' ability to encode linguistic features, BLiMP [143] evaluates grammatical knowledge, SuperGLUE [138, 140] tests natural language understanding, and LAMA [104] evaluates factual knowledge. Image reproduced from [160].

3.2 The BabyLM Challenge

The **BabyLM** Challenge encourages language model pre-training with human-like data limitations. Specifically, the challenge includes two text tracks: ① the `Strict-Small` track, with a 10M word budget, and ② the `Strict` track, with a 100M word budget. The creation of the challenge was motivated by the shortcomings of the current LM training landscape – such as the vast needs in computation and data – as well as a desire to provide wide access to LM research, develop performance advances with small datasets that can be scaled up, and enable collaboration between the cognitive science, psycholinguistic, and AI communities.

The *first* iteration of the challenge has already been conducted, and our efforts in this thesis are part of the *second* iteration. Below, we summarize some key findings of the first iteration that provide useful insights and guidance for future endeavors in data constrained language modeling.

Figure 3.2.1 includes a brief overview of the results of the **first iteration** of the challenge. Some initial observations are that encoder architectures seem to be in general outperforming decoder architectures. Intuitively, we can understand this, as the benchmarks of the BabyLM challenge probe the *understanding* of the models (e.g., grammatical, linguistic understanding), in which encoder models excel. This graph also informs us about the best types of approach to the tracks of the challenge, with architectural modifications and data augmentation being among the best ones. Moreover, the difference in performance between the `Strict` and `Strict-Small` tracks was not significantly large. This is surprising, given that the training data has one magnitude of difference between the two scenarios. Another important high-level conclusion is that many models achieved results close to the Llama 2 [135] model – trained on vastly more data – and to human performance, for grammatical and language understanding tasks. This suggests that even with human-like constraints, surpassing human performance will be possible in the future for these models.

The top performing submissions [119, 38] were based on the LTG-BERT architecture [120] – a BERT model optimized for a corpus of 100M words, combining many *architectural improvements* in the Transformer. We can conclude that **architecture** plays an important role. We note however, that these models were trained with huge amounts of compute (i.e., same FLOPs as BERT [27]), and the scale of available compute is another parameter that could be optimized in the future of the competition. A vastly popular approach was *curriculum learning*, which did not tend to produce good results – with the exception of the Contextualizer [153]. Intuitively, sorting the model's training steps based on some function of data complexity is a valid approach, as it is commonly applied in educational scenarios and is part of human concept learning. A compilation of negative results for curriculum learning was demonstrated by [83] whose work comprehensively explored a large range of strategies in this domain. They found no general improvement from any of the tested strategies. For the full taxonomy of employed methods and their benefits and drawbacks, we refer to the challenge proceedings [146]. Next, we explain more extensively the *data augmentation* methods proposed for the challenge, since they are more relevant to our methodology in this thesis.

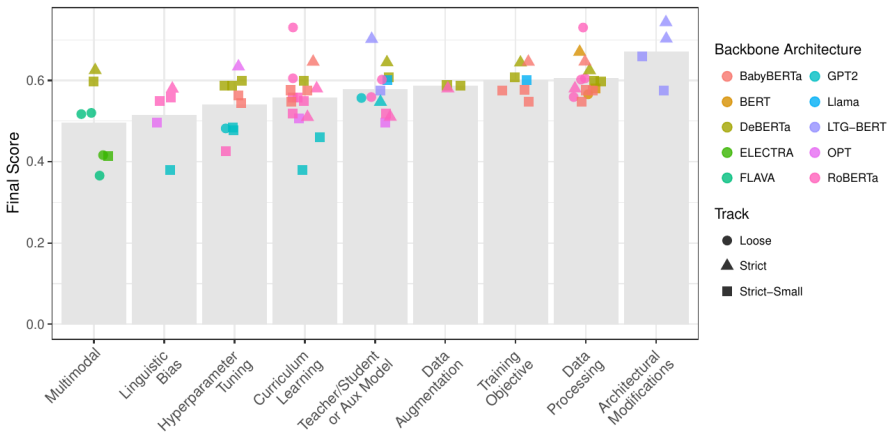

Figure 3.2.1: Summary of model architectures and approaches employed in the BabyLM Challenge. The x-axis indicates the category of methodology that the approach falls is, and the y-axis denotes the model performance (average score across benchmarks). On the side, the model architectures chosen by the participants are listed. Figure reproduced from [146].

3.2.1 Data Augmentation

We start with a general summary of data augmentation methods using language models. LLMs such as the family of GPT models [15, 16] have increasingly been used for data augmentation across research fields. Recent studies include work by [25] who introduced ChatGPT as a means of generating realistic text by leveraging a combination of real and synthetic data. In a similar approach, [155] leveraged GPT-3 [15] to blend synthetic and original text samples for more robust training data. Moreover, a variety of transformer architectures including decoder (GPT-2 [16]), encoder (BERT [27]) and encoder-decoder models (BART [73]) were utilized for data augmentation in the work of [71]. Decoder models have additionally been used to generate data for training bidirectional encoder models, achieving notable gains in zero-shot learning [87]. Work done by [62] uses modestly sized models, such as GPT-2, to distill a high-quality dataset and model for text summarization. Interestingly, the models used for creating the synthetic data cannot perform the summarization task. For a survey of data augmentation methods in NLP we point to [34].

We now move on to approaches for data augmentation introduced as part of the BabyLM challenge. We begin by the successful method of the **Contextulalizer** [153], which achieved the best performance in one of the 100M word text tracks of last year. In this approach, additional training samples are created by combining text chunks from different contexts and domains. The main idea is that during conventional training, the documents contained in a batch are fixed, leading to the model to occasionally learn spurious correlations. Intuitively, a model should be able to predict a word given its relevant context, independently of other irrelevant words in the batch. The training data is first organized according to age of acquisition and reading difficulty metrics, and a curriculum learning approach is employed. The model is trained on a final dataset of 4B words, derived by augmenting the initial 100M corpus, and achieves performance comparable to its counterpart pretrained on significantly larger amounts of data. Specifically, the model performs on par with RoBERTa and BERT on BLiMP. The authors use the architecture and size of RoBERTa base [80]. An overview of the approach is included in Figure 3.2.2.

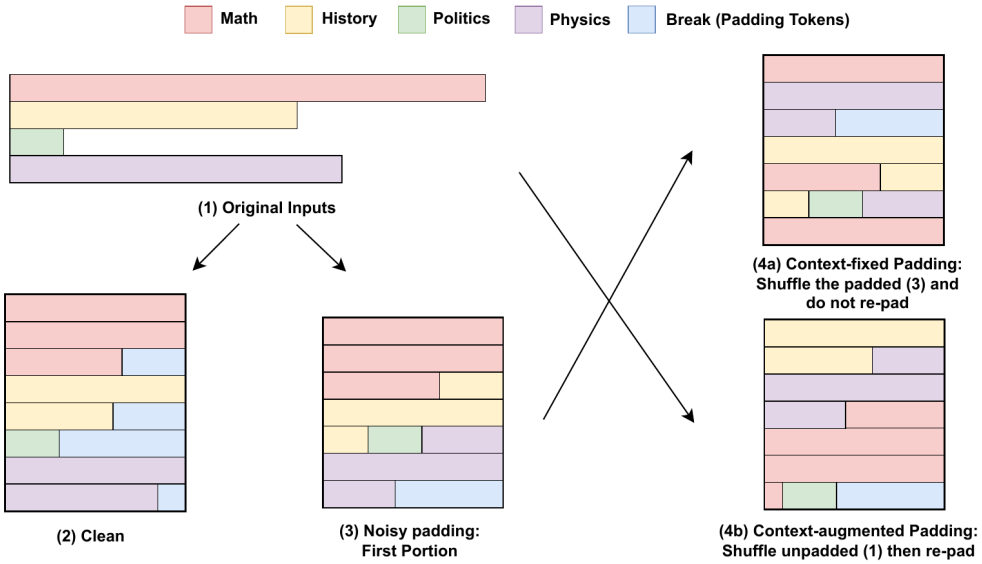

Figure 3.2.2: The approach used in the Contextualized paper. Here, the concept is illustrated for an example of four school subjects (Math, History, Politics, Physics). We explain briefly the various methods of combining data from each domain. Clean padding (2) adds `[PAD]` tokens in each batch, and doesn't mix domains. Noisy padding (3), just fills the batch with data from the next domain. Finally, the Contextualizer approach (4) shuffles around the noisy padding (4a) and for the best performance, (4b) shuffles the original inputs *at every iteration* to form the next batch – joining contexts in diverse positions. Figure reproduced from [153].

A second approach that uses data augmentation in the challenge is **ChapGPT** [61]. In this work, the authors explore the merits of *targeted* data augmentation, where regex patterns are leveraged to extract common phrases from (Super)GLUE tasks. These patterns are then harnessed to create follow-up questions by utilizing pre-defined templates. These formulations were created with the intend of improving performance on specific (Super)GLUE tasks, such as QA and sentiment classification. Retrieval-based patterns were used to extract facts and construct questions and answers, e.g., using Simple Wikipedia articles to augment the data with questions like "Where is <name> from?", where the corresponding answer was extracted with regex. Additionally, emotionally charged words (e.g., great, terrific) were annotated with a special sentiment token and label, to aid emotion recognition. This technique provided some benefits, surpassing the performance of the challenge baselines. The authors use the DeBERTa architecture [47].

Lastly, **Baby's CoThought** [161] uses an LLM (GPT-3.5-Turbo) to reformat sentences with no semantic relation into coherent paragraphs. This approach enables the formation of better training examples for LMs, improving performance on grammatical understanding (BLiMP). We note that this method utilizes an LLM and thus falls outside the `Strict` and `Strict-Small` restrictions of the challenge. However, it improves sample efficiency and provides information about which data is suitable for SLM supervision. The LLM acts as a "teacher" using Chain of Thought (CoT) [147] to reformat the training data and enrich learning for the "student" SLM. Essentially, for each collection of sentences the LLM picks an NLU *task type* and then creates a set of *plans* to format the sentences into a paragraph. An automatic scoring mechanism (LLM-based) is employed to select the best plan. Finally, a *paragraph* is generated for the specific NLU task, along with the corresponding *label*. The paragraph and plan are used to augment the pre-training corpus. The labels are discarded, but this "taskification" helps create a coherent narrative. The authors train the LM like RoBERTa [80] – outperforming the original model on some of the benchmarks. Figure 3.2.3 illustrates this approach.

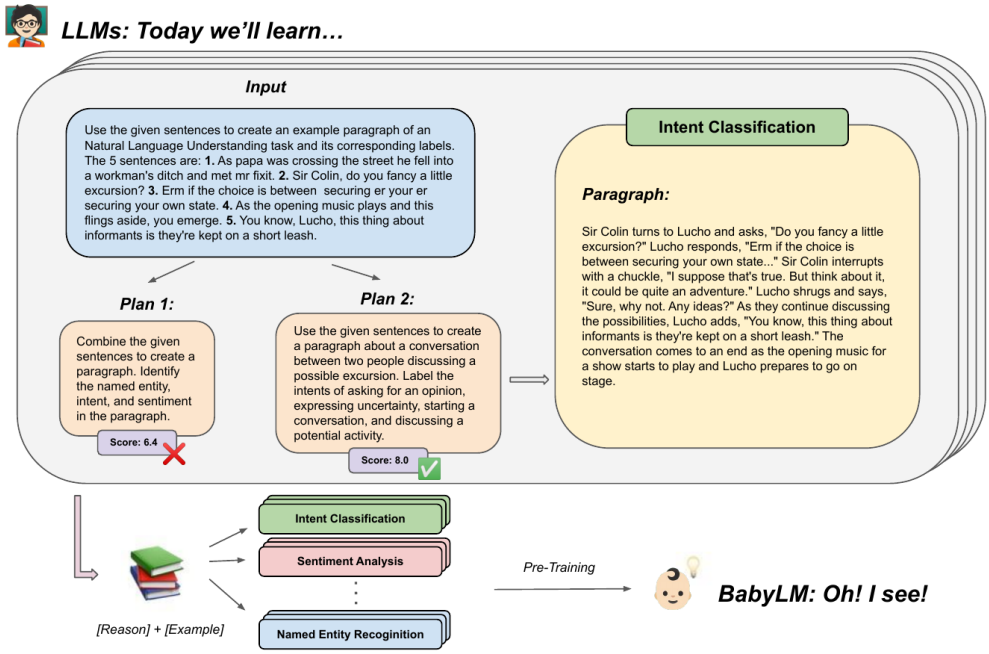

Figure 3.2.3: Overview of the approach applied in Baby's CoThought. Natural Language Understanding (NLU) examples are generated from unrelated sentences in the corpus using the aid of an LLM and CoT prompting. The data is formulated in a [reason] + [example] format, and a plan created by the LLM along with the example paragraph are used for augmentation of the pre-training corpus. Figure from [161].

3.3 Small Language Models

A line of work related to sample efficiency is recent investigation on the performance of Small Language Models (SLMs). We note here that, as illustrated by the findings of the BabyLM challenge, model size does not necessarily need to be scaled down when models are training on considerably limited data ($\leq$100M words). For example, the LTG-BERT [120] architecture has the same number of parameters as BERT [27] and many approaches used the RoBERTa model [80] with $\sim$ 125M parameters. Here we detail work on SLMs, that is distinct from the BabyLM challenge. Early work in [122] challenged the prevailing notion that tremendous model sizes are necessary for increased performance, showing that few-shot performance on SuperGLUE [140] close to that of GPT-3 can be achieved by models with three orders of magnitudes less parameters. This finding is supported by work in sequential recommendation systems, that utilizes SLMs for task-specific applications [154]. Additionally, [10] employ a pre-trained frozen LLM for the encoding of prompt tokens, that can aid a smaller LM in efficient and effective response generation. For a recent survey of SLMs we point to [94]. Next, we introduce the work of *TinyStories* that is one of the main influences of our methodology, described in Chapter 4.

The seminal work of *TinyStories* [31] showed that SLMs (e.g., $\sim$33M parameters) with the GPT-Neo [12] architecture can achieve impressive generative abilities. Specifically, these models are able to generate story continuations that are of both high quality and novelty. The authors measure generation quality by prompting an LLM (GPT-4 [95]) and asking it to score generations in terms of *Grammar*, *Creativity*, and *Consistency* with the story's beginning and plot. We also adopt this evaluation procedure for our own models in Section 5.2.2. Additionally, the generated stories were shown to be distinct from the training set of the models.

The core idea responsible for the success of the method, is the creation of a *synthetic* training dataset, using GPT-3.5 and GPT-4, that preserves all essential elements of language, but has limited vocabulary and simple plots and themes. This is in contrast to the diverse LLM training datasets, whose complexity might *confuse* models, degrading generation quality. The LMs learned to generate grammatically correct and creative text, demonstrating abilities superior to those of models trained with vastly more parameters. An illustration of this comparison is provided in Figure 3.3.1. The findings of TinyStories have also inspired subsequent work on a larger scale. Notably, phi-1 and phi-1.5 are 1.3B parameter models [75, 43] trained on a 7B dataset of *textbook-quality*, cementing the notion that with respect to data, quality is more important than quantity. Specifically, they achieve state-of-the-art performance on Python coding tasks, and compete with models that are 5x larger, and are trained on two orders of magnitude more data.

Prompt	Tom and Jane are friends. One day, Jane goes to Tom's house. Tom has a big pot of soup. He wants to share it with Jane. "Jane, do you want some soup?" Tom asks. "Yes, please. It looks yummy," Jane says. Tom pours some soup into two bowls. He gives one bowl to Jane. Jane takes a spoonful of soup, but then she makes a face. The soup is
Completion by GPT2-XL (**1.5B** parameters)	too old. Tom scowls. "Don't be that way! Your soup has to be fresh like your daughter," Tom says, angry and disappointed. "Bring me some more," Jane cries. "No! Go away, I like my soup old and fresh!"
Completion by a **28M**-parameters model trained on TinyStories	very bitter. She does not like it. She says, "I don't like this soup. It is too bitter." Tom is sorry. He says, "I'm sorry, Jane. I didn't know you don't like bitter soup. I will make you something else." He looks around the kitchen and finds some bread and cheese. He puts them on the table and says, "Here, Jane. You can have some bread and cheese. They are not bitter. They are sweet and yummy." Jane is happy. She says, "Thank you, Tom. You are a good friend. I like bread and cheese. They are not bitter."

Figure 3.3.1: For a prompt corresponding to the beginning of a story, we compare completions for an 28M parameter model trained on *TinyStories* and GPT2-XL with 1.5B parameters. We see that the larger model generates an essentially nonsensical response, while the 28M model manages to complete the story in a grammatical and coherent way. Figure reproduced from [31].

In the next chapter (Chapter 4), we introduce our methodology that is based on *data augmentation* using the *TinyStories* dataset [31], and is published as part of the second iteration of the BabyLM Challenge [21].

Chapter 4

Synthetic Sort Stories for Low Resource Data Augmentation

Contents

In this chapter, we formally introduce our methodology for data augmentation using synthetic story data [132]. Our efforts focus on *sample-efficient* language pre-training in a constrained data setting, mimicking the linguistic environments that children experience during development. This work follows the setting of the **BabyLM** Challenge [141, 21, 146][1] which consists of two experimental text tracks: ① the `Strict-Small` track, with a budget of 10M words, and ② the `Strict` track with a budget of 100M words. We refer to these specific scenarios throughout the following sections, conducting a separate investigation with different training configurations for each one independently.

This framework is also described in our paper [132] published as part of the 2nd iteration of the BabyLM challenge, the shared task at CoNLL 2024. Our work is titled: *"BERTtime Stories: Investigating the Role of*

[1] https://babylm.github.io/

Synthetic Story Data in Language Pre-training". To support research in the area of data constraint language modeling, we publicly release our implementation and models [2].

Throughout the following sections, we thoroughly explain our methodology, starting from the employed datasets (Section 4.2), expanding on the data generation process (Section 4.3), the evaluation procedures used to assess the performance of our models (Section 4.4), and finally a description of all the necessary Language Model (LM) components of our methodology (Section 4.5), devling into the model architectures, the finetuning process, the sampling methods used to generate data, and Masked Language Model (MLM) zero-shot scoring techniques.

4.1 Overview

Our research draws inspiration from recent developments in Small Language Models (SLMs) for text generation, as demonstrated in [31]. In this seminal work, the authors introduced *TinyStories*, a dataset of sort stories, with simple language and vocabulary, which is in the level of understanding of 3-4 year old children. By training GPT-Neo [12] decoder transformer models on this dataset, and prompting them with a story's beginning, they were able to produce high quality generations.

Specifically, the generations were evaluated in terms of *grammar*, *coherence*, and *consistency* with the story's plot and beginning, and were demonstrated to be *novel* with respect to the training set of the models. We hypothesize that harnessing the generative abilities of SLMs trained on TinyStories for high quality **data augmentation** of a pre-training corpus, would allow models trained on it to experience new linguistic contexts and syntactic variations. This would in turn improve language understanding, compared to using the initial data alone. To test our hypothesis, initially we extend work by [31], investigating the generative abilities of decoder transformer GPT-Neo models trained on small subsets of the *TinyStories* dataset. The decoders are then adapted for the augmentation of a pre-training corpus, on which we train encoder transformer models and evaluate their linguistic abilities. Our approach is illustrated in Figure 4.1.1. Below, we describe in more detail each of the steps of our methodology.

Data Generation using TinyStories In order to conduct our experiments, we need to adapt the *TinyStories* dataset to our low resource scenarios, as the size of the original dataset (approx. 440M words) exceeds the challenge limits for the `Strict-Small` (10M) and `Strict` (100M) tracks. For this reason, first we create subsets of *TinyStories*, varying the amount of available data. Specifically, we use subsets of $\{5, 10, 25, 50, 75, 100\}$M words. We then leverage the subsets to train GPT-Neo models and assess their generative and linguistic capabilities as a function of data size, keeping the model architecture and all other training conditions constant. Our exploration of generative performance complements the original study, which examines the impact of adjusting the depth and size of the model architectures. Our results show that even with limited data of $\leq$ 100M words, GPT-Neo models can develop remarkable grammatical comprehension and produce stories of notable *quality* and *variety*, comparable to those generated by models trained on the entire dataset.

Data Augmentation for Encoder Training We next concentrate our efforts in measuring the effect of data augmentation with synthetic story data in language pretraining. For each of the `Strict` and `Strict-Small` scenarios, we select a *TinyStories* subset – denoted as $\mathcal{D}_{\text{tiny}}$, and employ it to train a GPT-Neo decoder model. For each story in the training set $\mathcal{D}_{\text{tiny}}$, we use the decoder to generate alternative completions to the story's beginning. This way, we create a dataset of synthetic stories – $\mathcal{D}_{\text{gen}}$.

In order to measure the effect that $\mathcal{D}_{\text{gen}}$ has on the linguistic proficiency of pretrained language models, we train an encoder-transformer model on a diverse corpus, and compare performance with and without the use of generated data. To create the corpus used for training the encoder, we also utilize a subset of the *BabyLM* dataset ($\mathcal{D}_{\text{baby}}$) [145] – a diverse set of texts with an emphasis on spoken language. Finally,

[2] https://github.com/nikitas-theo/BERTtimeStories

a combined corpus is created, comprising the concatenation of three datasets: ① the *TinyStories* subset used to train the GPT-Neo model ($\mathcal{D}_{\text{tiny}}$), ② the data generated by the decoder ($\mathcal{D}_{\text{gen}}$), ③ a subset of the *BabyLM* dataset ($\mathcal{D}_{\text{baby}}$). Formally, the combined dataset is constructed as $\mathcal{D}_{\text{comb}} = (\mathcal{D}_{\text{tiny}}, \mathcal{D}_{\text{baby}}, \mathcal{D}_{\text{gen}})$. We train encoder transformer models on $\mathcal{D}_{\text{comb}}$ and measure their grammatical proficiency, world knowledge, and general language understanding. For the encoder, we use the LTG-BERT [120] architecture, which has been optimized for small data regimes.

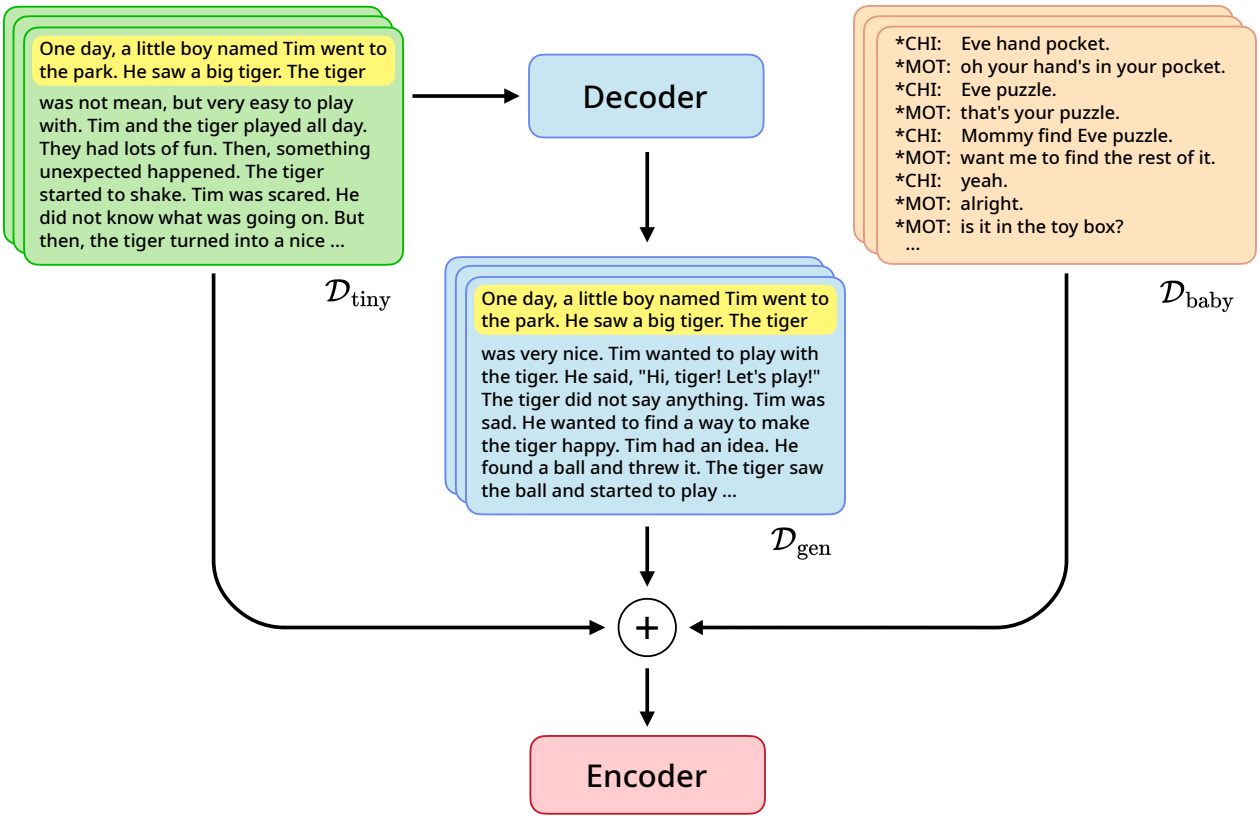

Figure 4.1.1: Illustration of our proposed methodology. We leverage a subset of the *TinyStories* dataset ($\mathcal{D}_{\text{tiny}}$) from [31] to train a decoder transformer for data augmentation. Short stories from $\mathcal{D}_{\text{tiny}}$ are used to prompt the decoder, which then generates an alternate completion to form a new dataset ($\mathcal{D}_{\text{gen}}$). Each story (green) is partially truncated and provided as a prompt (yellow), with the model producing an alternate completion (blue). We further enhance the two datasets with a portion of the *BabyLM* dataset ($\mathcal{D}_{\text{baby}}$), provided by [21], and train an encoder transformer on the combined data. Finally, we assess the linguistic proficiency of the encoder using the challenge benchmarks. Figure adapted from [132].

4.2 Datasets

For our work, we utilize two datasets: the *BabyLM* dataset – $\mathcal{D}_{\text{baby}}$, a diverse set of texts with an emphasis on spoken language, and the *TinyStories* dataset – $\mathcal{D}_{\text{tiny}}$, a colection of sort stories with simple language. We explain in detail the contents and properties of these datasets in the following paragraphs.

4.2.1 BabyLM Dataset – $\mathcal{D}_{\text{baby}}$

The *BabyLM* Dataset ($\mathcal{D}_{\text{baby}}$) was introduced for the purposes of the BabyLM Challenge [145, 21, 146], to be a developmentally plausible pre-training corpus for language modeling. Thus, it aims to mimic the linguistic environment that children experience while learning to speak. It contains a high proportion of

spoken language, since most input that children receive is spoken. It is also limited in its size, being under 100M words. This second restriction is motivated from the fact that children are exposed to 2M-7M words per year [42], resulting in no more than 100M words of linguistic input by 13 years of age. Two versions of the dataset were released, corresponding to the two experimental tracks of the challenge: a 10M word dataset for the `Strict-Small` track, and a 100M word dataset for the `Strict` track. We describe the structure of the datasets in Table 4.2.1.

Dataset	**Domain**	**# Words**	
		Strict-Small	**Strict**
CHILDES [81]	Child-directed speech	2.84M	28.90M
British National Corpus (BNC), dialogue portion [1]	Dialogue	0.93M	7.76M
Project Gutenberg (children's stories) [8]	Written English	2.54M	26.37M
OpenSubtitles [78]	Movie subtitles	2.04M	19.96M
Simple Wikipedia [2]	Written Simple English	1.45M	14.67M
Switchboard Dialog Act Corpus [127]	Dialogue	0.15M	1.34M
Total		**9.95M**	**99.01M**

Table 4.2.1: Contents of the *BabyLM* datasets for the `Strict` and `Strict-Small` tracks, including the domain and word counts. [1]http://www.natcorp.ox.ac.uk/, [2]https://dumps.wikimedia.org/simplewiki/20241001/.

The *BabyLM* dataset comprises 6 diverse datasets, we briefly explain their scope and contents below:

- **CHILDES**: The Child Language Data Exchange System, is a multilingual dataset, featuring transcriptions of adult-child interactions from various researchers, conducted in different environments: from controlled laboratory settings, to at-home conversations. Specifically, we use a further processed version of the dataset, released by [53] that contains interactions exclusively with children who speak American English and have an age from 0 to 6 years.

- **British National Corpus (BNC), dialogue portion**: the BNC is a highly curated, publicly available monolingual English corpus, made to accurately represent 20th century English. We use the transcribed speech part of the corpus, which corresponds to 10%, and contains a variety of spoken language, including radio shows, phone calls and business meetings.

- **Project Gutenberg (children's stories)**: This dataset was compiled manually, and consists of handpicked children's stories from the publicly available Project Gutenberg [39], suitable for young audiences. In the original work, the stories were used for a multimodal interactive story creation system, implemented as a collaboration between humans and models.

- **OpenSubtitles**: this dataset consists of publicly available subtitles from TV and movies, featuring a collection of parallel corpora. For our experiments we make use only of the English portion.

- **Simple Wikipedia**: a simple English language version of Wikipedia, written in a language easy to understand, while still being natural and grammatically correct. The articles feature shorter sentences and easier words and grammar, compared to the standard English Wikipedia.

- **Switchboard Dialog Act Corpus**: a collection of transcribed speech, taken from telephone conversations between strangers. The participants had an open-ended discussion about a topic chosen from an automatically generated list.

4.2.2 TinyStories Dataset – $\mathcal{D}_{\text{tiny}}$

The *TinyStories* dataset ($\mathcal{D}_{\text{tiny}}$), introduced in [31], is a synthetic dataset, consisting of a collection of short stories, constructed by prompting GPT3.5 and GPT-4 [95]. The dataset was created to preserve all the essential aspects of natural language, such as grammar, facts, and reasoning, but limited in scope and variety. The dataset features sort stories that are 2-3 paragraphs long with a reduced vocabulary, and simple plots and themes. Overall, the dataset was created to be on the level of understanding of 3-4 year old children.

Initially, the dataset was employed by the authors to enable Small Language Models (SLMs) with the GPT-Neo [12] architecture, to generate text of high quality and novelty. The reduced breadth and complexity of the dataset, enabled SLMs to produce fluent and consistent stories, outperforming models with vastly more parameters. The first version of the dataset (V1), generated by both GPT-3.5 and GPT-4, contained approximately 373M words. Later, a second version of higher quality was released (V2), containing around 440M words, and generated solely by GPT-4. For our experiments, we use only the second version of the dataset – 440M (V2), but we occasionally report results for the model trained by [31] on the initial version of 373M words (V1). An example of a story from the dataset is included in Table 4.2.2.

Once upon a time, there was a reliable otter named Ollie. He lived in a river with his family. They all loved to play and swim together. One day, Ollie's mom said, "Ollie, hurry and get some fish for dinner!" Ollie swam fast to catch fish. He saw his friend, the duck. "Hi, Ollie!" said the duck. "Hi, duck!" said Ollie. "I need to hurry and catch fish for my family." While Ollie was catching fish, he found a big shiny stone. He thought, "This is not a fish, but it is so pretty!" Ollie took the shiny stone home to show his family. They all looked at the shiny stone and smiled. The shiny stone made everyone happy, and they forgot about the fish for dinner.

Table 4.2.2: A story sampled from the *TinyStories* dataset generated by GPT-4 (V2). The stories, as we notice here, often contain unexpected twists. We also observe the simple plot and vocabulary.

4.2.3 Remarks

We note that the methodology of our work is not exclusively applicable to the previous two datasets. However, there are particular qualities that the selected datasets possess, that make them desirable for our data augmentation approach. The choice to include the *BabyLM* dataset was made by taking into account its *diversity*, and it's specific design for human-like language pretraining. In the previous iteration of the BabyLM Challenge, training was limited exclusively to the *BabyLM* dataset. Nevertheless, a few approaches managed to outperform the challenge's baseline models with orders of magnitude more data (RoBERTa [80], T5 [112], OPT [159]). Even though the *BabyLM* Dataset has proven to be a beneficial dataset for small data pre-training, experimentation with more highly curated datasets, e.g., the BNC Corpus, is promising future work. Indeed, the BNC dataset was recently used to develop the LTG-BERT architecture [120], which is optimized for small data regimes of around 100M words.

Additionally, the *TinyStories* dataset was chosen because of its simple language and small vocabulary, which were shown to be key factors in enabling high quality generations at small scales of both training data, and model parameters [31]. With regard to generative performance, diverse and complex text can confuse models, degrading generation quality. However, we envision augmentation in datasets with similar properties as *TinyStories*, particularly those containing spoken language, or simple English (e.g., Simple Wikipedia). The above analysis, demonstrates that our proposed techniques are not limited to the selected datasets, and are generally applicable in a wider setting.

4.3 Sort Stories for Data Augmentation

In this section, we detail the creation of the synthetic story dataset $\mathcal{D}_{\text{gen}}$, which is central to our data augmentation method – depicted in Figure 4.1.1. During the following paragraphs, for a dataset $\mathcal{D}_{\text{data}}$ we denote a version of the dataset with m millions of words as $\mathcal{D}_{\text{data-m}}$.

4.3.1 Data Generation Process

The process of creating the synthetic story dataset $\mathcal{D}_{\text{gen}}$, begins by training a decoder transformer model on a subset of the *TinyStories* dataset – denoted as $\mathcal{D}_{\text{tiny-m}}$. For our experiments we use the GPT-Neo architecture, but our methodology can be applied to any generative autoregressive LM, e.g., the Llama family of models [30]. We employ conventional causal LM training methods, following the work of [31]. The stories in the training dataset $\mathcal{D}_{\text{tiny}}$ are then truncated to create *prompts*, which are given as input to the GPT-Neo model in order to generate alternate completions.

One crucial parameter that we need to configure, is the size m of the subset $\mathcal{D}_{\text{tiny-m}}$. Given that we operate on the stringent data constraints of the `Strict` (100M) and `Strict-Small` (10M) tracks, careful consideration is required when allocating part of our word budget. To choose the right size m for the *TinyStories* subset, we take into account two key factors: *(a)* the need to maintain **high-quality** generations from the decoder models, and *(b)* the need to achieve sufficient **diversity** in the final corpus ($\mathcal{D}_{\text{comb}}$). As mentioned previously, our end goal is data augmentation, and the final encoder transformer training corpus will consist of: sort stories in $\mathcal{D}_{\text{tiny-m}}$, the generated data $\mathcal{D}_{\text{gen}}$, and a subset of the *BabyLM* dataset $\mathcal{D}_{\text{baby-b}}$ with size b. Given data limitations, balancing the proportions of the different datasets composing the combined corpus $\mathcal{D}_{\text{comb}}$ is critical for the success of our data augmentation method.

More concretely, based on the assumption that generation quality scales with dataset size, our goal is to choose a large enough size m for $\mathcal{D}_{\text{tiny-m}}$ to support high-quality outputs from our models. At the same time, we need to reserve enough of our word budget to incorporate a significant portion of the *BabyLM* dataset into the final encoder training set. This balance ensures that our encoder models are exposed to both a wide range of vocabulary and a diverse set of linguistic contexts. Intuitively, we seek to ensure that our pre-training data is varied, as children learn from multiple sources of input – this is also reflected in the makeup of the *BabyLM* dataset (Section 4.2.1). Training only on simple stories will critically limit the vocabulary and linguistic variety to which we expose our model.

To address this tension between conflicting demands for the size m, we draw samples from *TinyStories* and construct a collection of subsets with various sizes, which we denote as $\mathcal{D}_{\text{tiny-m}} : m \in \{5, 10, 25, 50, 75, 100\}$M (millions of words). For each subset, we then train a GPT-Neo model and assess its generative and linguistic capabilities. Our evaluation process employs metrics related to grammatical proficiency, diversity, and generation quality, which are detailed in Section 4.4. Based on these evaluation metrics and the aforementioned criteria, for both the `Strict` and `Strict-Small` tracks, we select a specific subset $\mathcal{D}_{\text{tiny-m}}$ along with its corresponding GPT-Neo model, to be used for data augmentation. In order to create the synthetic story dataset $\mathcal{D}_{\text{gen}}$, each story in $\mathcal{D}_{\text{tiny-m}}$ is truncated to 15%-30% of its original length, and the shortened version is used to *prompt* the model, which then generates an alternate story completion. Our choice to use a smaller portion of the story is motivated by a desire to avoid duplication, as the stories in $\mathcal{D}_{\text{tiny-m}}$ will already be part of the combined corpus used to train the encoder transformer.

To construct $\mathcal{D}_{\text{gen}}$ we employ different sampling techniques. Specifically, we use greedy decoding, and nucleus sampling [51]. Greedy decoding is a simple decoding strategy, that will demonstrate the underlying maximization objective of our models, as seen in their token probability distribution. Intuitively, this will produce the most likely generation under our model. If even with maximization based decoding, our models can produce novel and diverse completions, it is a strong sign that they are not simply memorizing their training set. It was also the default method used in [31] for most of the generated stories. After establishing performance with greedy decoding, we move to more advanced techniques like nucleus sampling, which was introduced to alleviate problems in *open-ended* generation tasks, like sort story generation. The different

generation methods used for our experiments are introduced in Section 4.5.3.

Additionally, nucleus sampling gives us the option of producing more than one completion for each story. This will increase the size of $\mathcal{D}_{\text{gen}}$ and the available training data for the encoder models and is beneficial for language learning, provided that the different completions for the same story are diverse enough. For each story prompt in $\mathcal{D}_{\text{tiny}}$ we produce k completions, in an effort to maximize the benefit of our data augmentation method. In order to minimize redundancy among the k completions, and prevent inefficient use of FLOPs, we leverage the Self-BLEU score, which is a measure of diversity of a generated corpus. For various values of k, we calculate Self-BLEU and select those values that best balance diversity with the total volume of additional training data.

4.3.2 Final Corpus Creation

In the previous paragraphs, we explained the construction of the subset of *TinyStories* $\mathcal{D}_{\text{tiny-m}}$ and the generated stories dataset $\mathcal{D}_{\text{gen}}$. In order to form the final combined dataset $\mathcal{D}_{\text{comb}}$, used to train the encoder transformer, we need to include a portion of the *BabyLM* dataset, denoted by the subset $\mathcal{D}_{\text{baby-b}}$ with size b. As shown in Section 4.2.1, the dataset contains a diverse set of texts, with a variety of different domains. To ensure a fair representation of each set, we sample randomly at the document level from the whole dataset. We leave content-aware sampling methods for future work. The final combined dataset is constructed by concatenating the three dataset, $\mathcal{D}_{\text{comb}} = (\mathcal{D}_{\text{tiny-m}}, \mathcal{D}_{\text{baby-b}}, \mathcal{D}_{\text{gen}})$. To train the encoder transformer we use a Masked Language Modeling objective, and we leverage the LTG-BERT architecture [120].

The combined size of $\mathcal{D}_{\text{baby-b}}$ and $\mathcal{D}_{\text{tiny-m}}$ is limited by the data constraints in each track. Specifically, for the `Strict` task we impose that $b + m \leq 100\text{M}$ and for the `Strict-Small` task $b + m \leq 10\text{M}$. We note that the track restrictions do not apply to the dataset $\mathcal{D}_{\text{gen}}$, as the data used to train the GPT-Neo models and generate stories, are all contained in $\mathcal{D}_{\text{tiny}}$. Intuitively, the 10M / 100M word budget applies to a *closed system*, where any model and procedure can be implemented, such as augmenting the training dataset with more words, as long as no additional data enters the system. In the same manner, we image that as a child learns to speak, conversations are replayed in the human brain in a slightly altered form, e.g., imaginary interactions as presented in dreams. This does not mean that the child is receiving any additional linguistic input.

4.4 Evaluation Procedures

In this section, we detail the evaluation metrics and benchmarks used to assess the performance of our models. For evaluating the performance of the encoder transformer models, we leverage the evaluation suite of the BabyLM challenge [21], consisting of three benchmarks: **BLiMP** and its supplement (Supp.), **(Super)GLUE**, and **EWoK**, each broadly measuring language proficiency, general language understanding, and world knowledge. For the decoder transformer models we employ EWoK and BLiMP as well as some additional evaluation schemes, targeted towards measuring generation *quality* and *diversity*. Specifically, to measure the quality of the generated text of our models we use an **LLM-assisted** evaluation, and to measure diversity we harness the **Self-BLEU** metric.

We note that the challenge benchmarks – BLiMP, EWoK, and (Super)GLUE – constitute *filtered versions*, preventing the comparison of our results with prior evaluations on the full data. Specifically, for the purposes of the BabyLM challenge, evaluation examples with no lexical overlap with the `Strict-Small` pretraining corpus were filtered out [146]. The purpose of this process was to mitigate the confounding effect caused by the language style of many NLP tasks not being well aligned with the pretraining corpus that was developed for the challenge, i.e., the *BabyLM* dataset.

4.4.1 BLiMP

The Benchmark of Linguistic Minimal Pairs (BLiMP) was introduced by [143], as a set of tasks for evaluating the linguistic knowledge of Language Models (LMs) on key grammatical phenomena in English. The

benchmark covers various grammatical phenomena, in syntax, morphology and semantics. Each benchmark example consists of a pair of *minimally different* sentences, and the model under evaluation needs to assign a higher probability to the grammatically correct sentence in each pair. To measure performance, we calculate the binary **accuracy** over all pairs. The tasks in BLiMP are zero-shot, with no added finetuning or training required for evaluation. The benchmark consists of 67 datasets with 1000 minimal pairs each, automatically generated by grammar templates constructed by linguists. The examples are further organized in 12 grammatical phenomena. We present a concrete sample from the dataset in Table 4.4.3 and a brief description of the 12 tested phenomena is included in Appendix D.2. For a detailed analysis of the full set of paradigms evaluated in BLiMP we refer to [143]. For our evaluation, we also employ **BLiMP Supplement (Supp.)**, released by the challenge organizers [145], and containing some additional grammatical phenomena.

Sentence s_1	*Those mice haven't vanished.*	correct
Sentence s_2	*Those mice hasn't vanished.*	wrong

Table 4.4.3: An example from the BLiMP benchmark. The sentence pair illustrates the phenomenon of `Subject Verb Agreement`: subjects and present tense verbs must agree in number. For a correct response, the model needs to assign higher probability to the correct sentence (s_1), i.e., $P(s_1) > P(s_2)$.

BLiMP has been used previously to measure the grammatical performance of models at small scales, including the 5M word BabyBERTa model [54] and the 100M word LTG-BERT [120]. It was also included as an evaluation benchmark in both iterations of the BabyLM challenge [21, 145]. Work in [160] also used BLiMP to measure the evolution of the linguistic knowledge of RoBERTa models, as the quantity of their training data increases, demonstrating that 100M words is enough to reach a plateau on the benchmark and achieve good performance on most grammatical phenomena.

4.4.2 EWoK

The Elements of World Knowledge (EWoK) is a benchmark recently introduced by [55], as a cognition-inspired framework aiming to evaluate basic world knowledge in LMs. The creation of the benchmark is motivated by the crucial role of world model building and manipulation in human intelligence, which is also essential for a general-purpose AI agent. The benchmark assesses the ability of models to correctly match a target text with a plausible context across various knowledge domains, including social interactions (*e.g., help/hinder*) and spatial relations (*e.g., left/right*). Each example contains two pairs of contexts and targets, and the evaluated system needs to assign each target to the correct context. Both the contexts and targets are formulated as *minimally contrasting pairs*, with customizable elements like objects, agents, and locations. A comparison of the performance of large language models and a human study, reveals that current models perform, overall, *worse* than humans. Thus, this benchmark offers promising opportunities for research into improving the world modeling abilities of LMs. Like BLiMP it also involves zero-shot tasks, with no required model finetuning.

Each item of the framework, consists of two minimal pair of contexts, C_1 and C_2, and two minimal pair target sentences T_1 and T_2. The pairs are constructed such that in any item, $P(T_1|C_1) > P(T_1|C_2)$ and $P(T_2|C_1) < P(T_2|C_2)$, in order to force the model to rely on context when making a decision. The metric for correctness of a given item is the recovery of the designed item structure, i.e., $P(T_1|C_1) > P(T_1|C_2)$ and $P(T_2|C_1) < P(T_2|C_2)$. Identification of both matches yield a point of 1.0, while identification of only one match yields 0.5 points. For scoring, the token-level LM probability $\log P(T|C)$ is calculated. An example of an item from the framework is presented in Table 4.4.4. We report **accuracy** across all tasks.

C_1	*The piano is in front of Ali. Ali turns left.*
C_2	*The piano is in front of Ali. Ali turns right.*
T_1	*The piano is right of Ali.*
T_2	*The piano is left of Ali.*

Table 4.4.4: A sample of an evaluation item from EWoK.

4.4.3 (Super)GLUE

The General Language Understanding Evaluation (GLUE) benchmark [138], is a widely used benchmark, measuring the performance of LMs in Natural Language Understanding (NLU). It consists of a variety of NLU tasks with limited training data, in order to support and promote models that share general linguistic knowledge across tasks. Although it was initially a challenging benchmark suite, state-of-the-art LMs quickly advanced and surpassed human non-expert performance on GLUE, leading to the introduction of SuperGLUE [140], a more difficult set of NLU tasks. In contrast to the previously introduced benchmarks – BLiMP and EWoK – evaluating on GLUE requires model *finetuning*. To train our models, we use the evaluation procedure of the BabyLM challenge, employing a filtered training set and a validation set to select the best model. We then report final performance on the validation data. More info on finetuning for Masked Language Models (MLMs) can be found in Section 4.5.2.

All the tasks in (Super)GLUE are framed as text classification. We leverage a set of 10 tasks from both benchmarks, covering various categories, specifically: question answering (BoolQ, MultiRC), sentiment classification (SST-2), paraphrasing (MRPC, QQP), grammatical acceptability (CoLA), commonsense reasoning (WSC), and entailment (MNLI, QNLI, RTE). To assess performance on (Super)GLUE we average the **accuracies** across all tasks except for QQP and MRPC, for which we use the **F1-score**, and CoLA, where the Matthews Correlation Coefficient [84] (**MCC**) is employed. We include an example for the BooLQ task in Table 4.4.5. A brief description of each of the employed tasks in (Super)GLUE is given in Appendix D.1.

Question	**Passage**	**Label**
is elder scrolls online the same as skyrim	The Elder Scrolls Online – As with other games in The Elder Scrolls series, the game is set on the continent of Tamriel. The events of the game occur a millennium before those of The Elder Scrolls V: Skyrim and around 800 years before The Elder Scrolls III: Morrowind and The Elder Scrolls IV: Oblivion. It has a broadly similar structure to Skyrim, with two separate conflicts progressing at the same time, one with the fate of the world in the balance, and one where the prize is supreme power on Tamriel. In The Elder Scrolls Online, the first struggle is against the Daedric Prince Molag Bal, who is attempting to meld the plane of Mundus with his realm of Coldharbour, and the second is to capture the vacant imperial throne, contested by three alliances of the mortal races. The player character has been sacrificed to Molag Bal, and Molag Bal has stolen their soul, the recovery of which is the primary game objective.	1

Table 4.4.5: An evaluation sample for (Super)GLUE, taken from BoolQ, a question answering task. The model is tested on binary classification, and needs to answer a *true / false* question based on the given passage. The label of "1" indicates that in this example, the answer is *true*.

4.4.4 Self-BLEU

When measuring the generative abilities of LMs, one important dimension is the *diversity* of the model responses, i.e., how much variation do the model generations exhibit. To assess the diversity of the stories

generated by the decoder transformer models, we leverage the Self-BLEU score, introduced by [163]. Self-BLEU is based on the popular BLEU metric [98], originally introduced in the context of machine translation, which operates by calculating the n-gram precision between a generated text and a set of references, for various sizes of n. Shorter outputs are also penalized through a brevity penalty, and the final score is in the range $[0, 1]$.

To calculate Self-BLEU, for a collection of generated texts, we compute BLEU treating one text as the hypothesis and the rest as references, effectively measuring the similarity between the selected text and all others. Self-BLEU is then defined as the average of the BLEU scores across the whole corpus. The metric, like BLEU, lies in $[0, 1]$ with higher values indicating lower diversity. Self-BLEU has been widely used as a measure of diversity [51, 1, 139].

4.4.5 LLM Evaluation

Assessing the quality of outputs in open-ended generation tasks is difficult because standard evaluation methods typically expect structured outputs and measure fidelity against a set of reference texts. To overcome this, we use the evaluation approach suggested in *TinyStories* [31], where an LLM is prompted to evaluate the stories generated by the models. Specifically, we inform the LLM that a student has completed a short story, and it has to assess the student's performance on a scale from 1 to 10 along three axes: *Grammar*, *Creativity*, and *Consistency* with the beginning of the story and the plot, giving a separate rating for each. Additionally, we ask it based on the student's response, to classify them in different age groups, ranging from 3 to 16 years old. For our experiments, we leverage Claude Sonnet-3.5 [2] and a sample interaction with the LLM is included in Appendix B.

4.5 Language Models

In this section, we include necessary information and background knowledge regarding the decoder and encoder LMs central to our methods. We explain in detail the chosen model architectures, the finetuning procedures, and the employed decoding strategies for generating samples. This section is meant to complement Chapter 2, with more details about the specific language modeling techniques employed in our setting.

4.5.1 Language Model Architectures

We give a brief description of the chosen model architectures for our methodology, and highlight the motivation for their selection. We note that in general our framework is *model agnostic*, with the only assumption being the use of generative LM for *data augmentation* – however, the generative LM need not be autoregressive, as long as it can produce continuations to a given story.

GPT-Neo For the decoder transformer model trained on *TinyStories* we choose a GPT-Neo architecture [12], following [31]. GPT-Neo is an open-source GPT-like autoregressive LM architecture. It largely follows the architecture of GPT-3 [15] with some notable differences. The authors used *rotary embeddings* [129], a form of static, relative positional embeddings, where the attention of a token at position n to a token at position m is linearly dependent on $m - n$. Additionally, the model features a different initialization, and there are no sparse layers. We point to the original paper by [12] for more details.

LTG-BERT The LTG-BERT architecture [120], was developed to showcase the effects of training an LM in a small but well curated 100M word corpus for English– the British National Corpus (BNC). The authors managed to outperform the original BERT model, using the same compute budget, and in the process evaluated several training objectives and variants of the model architecture. These efforts resulted in the LM architecture of LTG-BERT, optimized for small data pretraining. A simplified representation of one attention layer in the model is illustrated in Figure 4.5.2.

In designing the architecture, the authors diverge from the original Transformer [137], and apply some recent architectural improvements. The *NormFormer* [125] layer normalization is used, where LayerNorm

blocks are included both before and after the application of the Attention block. Additionally, the Layer Normalization is implemented outside the residual stream, i.e., before adding the residual connection to the output of the layer. This is done because the standard *Post-norm* variation has been shown to unpredictably diverge in later stages of training. Additionally, the GEGLU [123] activation function is used, given by $\text{FF}_{\text{GEGLU}}(\mathbf{x}) = (\text{GELU}(\mathbf{x}\mathbf{W}_1) \odot \mathbf{x}\mathbf{W}_2)\mathbf{W}_3$, where GELU is the Gaussian Error Linear Unit, and the $\odot$ symbol indicates pointwise multiplication. The authors also perform initialization scaling for stabilization, following [4].

Lastly, the attention layer is further modified by the use of *disentangled* attention [47] as in the DeBERTa model, where the position and content embeddings are uncoupled. In this implementation, the attention takes the form of content-content attention, content-position attention, and position-position attention. For the original transformer implementation by [137], the content and positional embeddings are fused together by simple addition, and the attention score for tokens $\mathbf{x_i}$ and $\mathbf{x_j}$ is calculated as: $\mathbf{A}_{i,j} = \frac{\mathbf{Q}_i \mathbf{K}_j^T}{\sqrt{d}}$. Where $\mathbf{Q} = \mathbf{x}\mathbf{W}_Q$ and $\mathbf{K} = \mathbf{x}\mathbf{W}_K$ are the key and query transformation matrices of $\mathbf{x}$, and d is the hidden token size. As proposed by [47], for *disentangled* attention, the position and content embeddings are processed separately, without fusion. Specifically, the attention calculation is performed as:

$$\mathbf{A}_{i,j} = \frac{\mathbf{Q}_i^p (\mathbf{K}_j^c)^T + \mathbf{Q}_i^c (\mathbf{K}_j^p)^T + \mathbf{Q}_{i,j}^p (\mathbf{K}_j^c)^T}{\sqrt{3d}}$$

Where $\mathbf{Q}^c$ and $\mathbf{K}^c$ are matrix transformations of the content embeddings, and $\mathbf{Q}^p$ and $\mathbf{K}^p$ are transformations of the relative positional embeddings $P_{i,j}$. The representation of tokens x_i, x_j at positions i and j, is taken from the row of P that corresponds to their distance $(L - i + j)$, where L is the maximum input length. In the implementation of both [120, 47], to keep the model size comparable to the traditional attention layer, parameter sharing is employed: $\mathbf{Q}^c = \mathbf{Q}^p$ and $\mathbf{K}^c = \mathbf{K}^p$. The separation of *content* and *positional* information offers more expressiveness in the transformer, as each attention layer can combine the two types of information as needed.

Some notable differences from the training objective of BERT are also implemented. A span-based masking objective is used, following SpanBERT [60], where multiple tokens in a row are masked – illustrated in Figure 4.5.1. Additionally, the Next Sentence Prediction (NSP) objective is dropped as in [79].

4.5.2 Language Model Finetuning

One of the most ubiquitous ways to leverage the linguistic capabilities of language models, is to apply one large-scale general domain pretrained LM to multiple particular tasks or domains [27, 107]. This is done by the process of *finetuning* [45], whereas the pretrained LM is additionally trained on a small task-specific corpus. This way, we can leverage the rich representations learned by an LM on a large text corpus for effective transfer learning to tasks where data is scarce. The major factor contributing to success of LM transfer learning, is that LM pre-training uses a *semi-supervised* objective, where data is plentiful, while most finetuning tasks are by nature *supervised*, and procuring large training datasets can be especially challenging.

For our evaluation, BLiMP, BLiMP Supplement and EWoK all require no finetuning, as they are zero-shot tasks, measuring the inherent knowledge of an LM along different dimensions. However, the evaluation on (Super)GLUE for natural language understanding, requires our models to be fine-tuned to task-specific datasets. In the following paragraphs, we detail the finetuning process for a Masked Language Model (MLM), such as LTG-BERT.

In full model finetuning, the pre-trained model is trained end-to-end with a supervised training objective, and the learning signal is provided by a cross-entropy loss applied to a softmax output and the correct label for each sequence. In practice, only a few weights of the initial pre-trained model are updated, and

$$\mathcal{L}(\text{football}) = \mathcal{L}_{\text{MLM}}(\text{football}) + \mathcal{L}_{\text{SBO}}(\text{football})$$
$$= -\log P(\text{football} \mid \mathbf{x}_7) - \log P(\text{football} \mid \mathbf{x}_4, \mathbf{x}_9, \mathbf{p}_3)$$

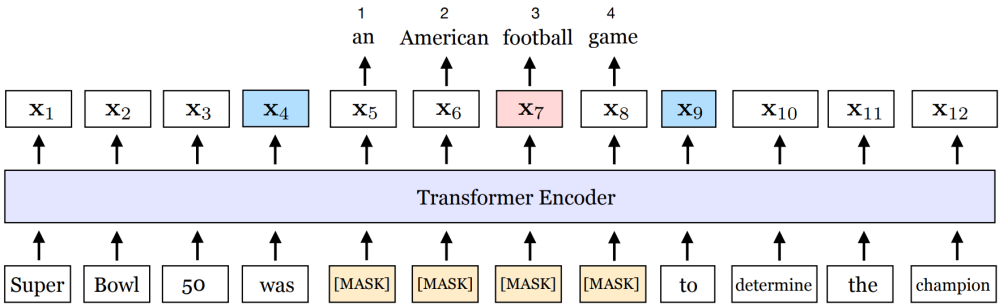

Figure 4.5.1: Illustration of the SpanBERT masked objective. For masking, we select a span, e.g., "an American football game", and calculate a two term loss for each token, presented here for the token "football". A span boundary objective (SBO) is employed, where the boundary tokens ($\mathbf{x}_4, \mathbf{x}_9$) along with the masked token embedding ($\mathbf{x}_7$) are used to predict the masked word ("football"). This loss term is added to the standard MLM objective. Figure from [60].

a common approach is to completely freeze some parameters. More complex techniques for finetuning were also proposed, mostly as a way to decrease the computational costs of full-model finetuning both during training and inference. For example, the popular low-rank adaptation [52] method, decreases the size of the trainable parameters, by freezing (excluding from training) each linear layer matrix $\mathbf{W}$ and learning a low-rank decomposition. In our experiments (Chapter 5) to maximize the performance of our models, we perform the full finetuning, without freezing any model parameters. We also note that we are concerned only with *sequence classification*, (e.g., sentiment classification, or entailment detection), as essentially all (Super)GLUE tasks are framed in this manner.

We focus specifically on the case of Masked Language Models (MLMs) such as BERT [27], as the final result of our methodology is a pre-trained encoder transformer model (see Figure 4.1.1). We apply a newly initialized architectural component on top of the MLM, known as a classification **head**, which is essentially a simple neural network classifier. This classifier will take as input the output of the pre-trained model – a single vector representing the entire sentence to be classified. To classify a sentence for BERT-like models, we use the representation of the `[CLS]` token, and map it through the classification head, to the class logits using a softmax layer. The `[CLS]` is included during pre-training at the start of all sequences, to enable later usage in downstream applications during finetuning. We illustrate the finetuning process for a BERT-like MLM in Figure 4.5.3, the depicted task is sentiment classification.

For classifying a *pair of sequences* – s_1 and s_2, as in the case of detecing textual entailment, a slightly different formalization is employed. The sequences are separated by a special `[SEP]` token, and the final input to the model is formalized as "`[CLS]` s_1 `[SEP]` s_2 `[SEP]`". In the original BERT model, the `[SEP]` is introduced during training in the Next Sentence Prediction (NSP) task. Later, this task was proven to be unhelpful to model pre-training when taking into consideration the extra computation it entails, and later approaches dropped this objective [80, 120, 47]. However, the `[SEP]` token is still used in pre-training as the final token of an input batch.

4.5.3 Language Model Decoding

In this section, we provide details and a formal analysis of the different sampling methods that were use for generating text with the GPT-Neo decoder transformer models. Our textual domain is *open-ended* text generation, and specifically conditional story generation. In this setting, there is some restriction to the space of expected responses due to the input context (*e.g.*, a story's beginning, as in our case), but there

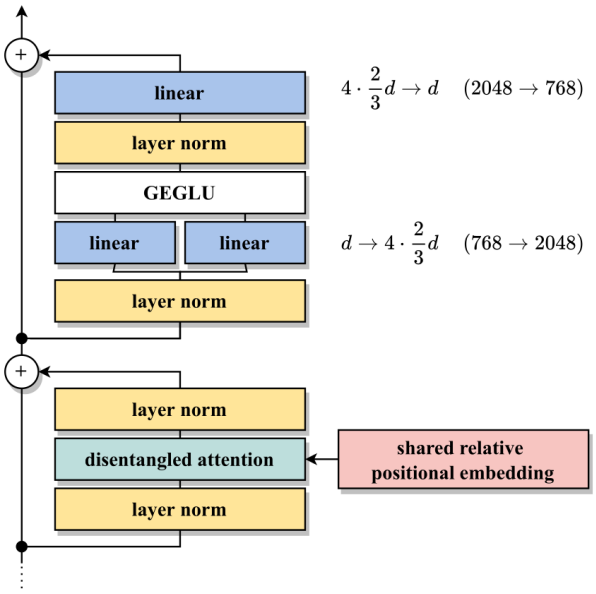

Figure 4.5.2: A layer of the LTG-BERT architecture, schematically illustrated. Figure from [120].

is a significant degree of freedom in what the continuation can be. This in contrast to *directed* generation settings, where tasks are formalized with (input, output) pairs, where the output must be a transformation of the input. Common tasks in this domain include machine translation, and summarization.

The task of *open-ended* text generation, can be defined as generating text that results in a *coherent* continuation to a given context. Formally, given a sequence $s = (s_1, s_2, ..., s_n)$ to be used as context, the task is to generate a continuation of m tokens, forming a sequence $(s_1, ..., s_k, s_{k+1}, ..., s_{n+m})$. To produce the completion, we use an autoregressive LM that models the probability $P_{LM}(s_k|s_{<k})$ using the left context $s_{<k}$ – all the tokens on the left of a token s_k. There are many ways to sample tokens from the conditional probability function $P_{LM}(s_k|s_{<k})$ which we refer to as *decoding strategies*. In the following paragraphs, we expand on popular decoding procedures that were also employed during our experiments.

Maximization Based Decoding One common and simple way to generate an appropriate response from an LM, is maxization-based decoding. This category of strategies assumes that the model will assign higher probabilities to text of a higher quality, and thus they search for the continuation with the highest likelihood. The most common approach of this kind is *greedy decoding*, where at each timestep the model predicts the most likely token in the vocabulary V, based on the previously generated tokens:

$$s_k = \textbf{argmax}_{s \in V} P_{LM}(s|s_{<k})$$

This type of sampling is mostly used for directed generation tasks, where the output is highly correlated with the input and it can achieve satisfactory results. However, it has been known to exhibit failure cases (e.g., excessive repetition) in open-ended generation tasks [51], such as dialogue and story generation, where it falls sort of other approaches. Some improvements have been proposed over the vanilla greedy search method. **Beam Search**, retains the sentences with the n highest probabilities at each step during the decoding process, and in the end selects the generate sequence with the top probability, taking into account all tokens. Occasionally, a length penalty is also applied, since otherwise beam search would favor shorter sentences. Maximization based methods can result in very different probability distributions, compared to those observed in human sentences, as illustrated in Figure 4.5.5. Intuitively, we want language to include

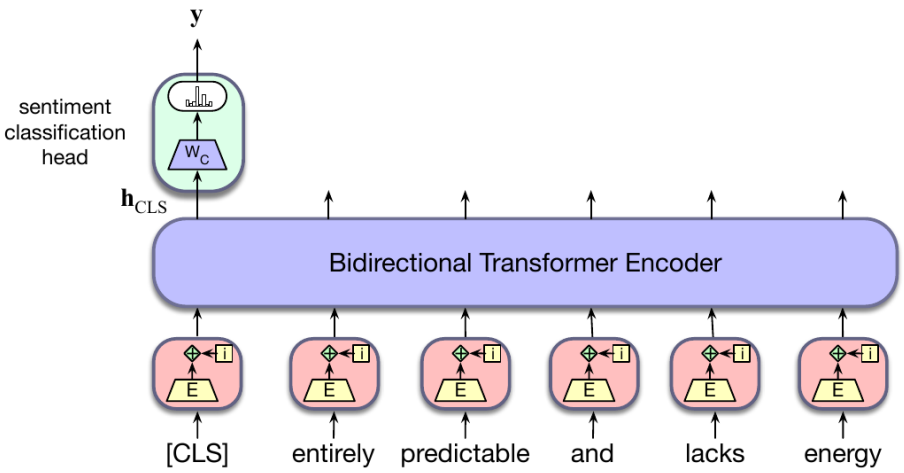

Figure 4.5.3: Illustration of the finetuning process for BERT-like Masked Language Model. The embedding vector of the `[CLS]` token is used to represent the whole sequence, and is mapped to softmax class probabilities through the classification head. Figure reproduced from [63].

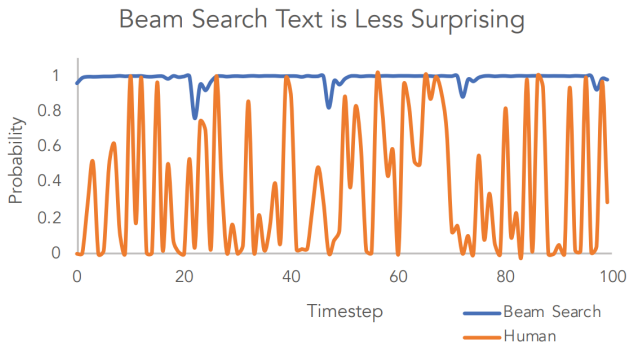

Figure 4.5.4: Illustration of the probability assigned to tokens generated by beam search and humans, for the same initial context. The human response is characterized by increased variance – language should be *surprising*. In contrast, beam search falls into an endless repetition loop. Figure from [51].

surprisal, as this relates to new information and an increased entropy.

This group of generation strategies do not employ any sampling, as they just maximize the likelihood of the next token w.r.t. the probability distribution, in a deterministic way. This is in contrast to the methods we introduce next, that randomly sample each token from the probability distribution of the model $s_k \sim P(s|s_{<i})$. For each of these methods, some adjustments in the candidate tokens or the probability calculation is performed to improve upon naive random sampling. Compared to maximization-based decoding, these methods allow for more variation in the model outputs, and more natural text generation.

Top-k Sampling Top-k sampling is a simple method that aims to improve generation quality, by restricting the available tokens during the sampling process. Specifically, it directly filters out tokens with lower probabilities, restricting the selection to the top k tokens with the highest probabilities. Essentially, top-k sampling operates on the rescaled probability distribution of only the k most probable tokens, i.e., a vocabulary of size k, $V^{(k)} \subset V$ such that $\sum_{s \in V^{(k)}} P(s|s_{<k})$ is maximized. Top-k methods offer improvement against simple random sampling, as a great number of tokens that have no relation to the previous context are excluded in each sampling step, resulting in more coherent responses. However, tuning the k parameter

can be difficult.

Temperature Sampling Another approach to limit the randomness in the token selection during sampling, is to insert a temperature parameter in the Softmax function, which can be adjusted according to the desired diversity in the model response. Below, we include the formula for computing the probability of a token s_k, parameterized by a temperature parameter T.

$$P(s_k|s_{<k}) = \text{Softmax}(z;T) = \frac{exp(z_k/T)}{\sum_{i \in V} exp(z_i/T)}$$

With the vector z we symbolize the logits of the sequence s i.e., the class probability weights before normalization. A low temperature T results in a more "sharp" distribution, which in turn increases the chance of selecting highly probable tokens. A high temperature results in a "flattened" probability distribution, making low probability selections more likely. The temperature parameter $T \in [0, 1]$, is often used as a method to increase the diversity of the model responses. We note that this approach can be combined with top-k sampling to both have more diverse, and vocabulary restricted generations.

Nucleus Sampling *Nucleus Sampling* was proposed by [51], as a stochastic decoding technique, that alleviate problems encountered with maximization based methods and the previous sampling techniques. It is targeted towards open-ended generation tasks, where these problems mostly occur. Like top-k, nucleus sampling operates by restricting the available vocabulary at each sampling step, however it does so in a *dynamic* manner. The core concept of the approach, is using the shape of the probability distribution to determine the set of candidate tokens for the sampling. Formally, given a distribution $P_{LM}(s_k|s_{<k})$, the top-p vocabulary is defined as $V^{(p)} \subset V$, which is the smallest subset of the whole vocabulary V such that $\sum_{x \in V^{(p)}} P_{LM}(s_k|s_{<k}) \geq p$. Essentially, the original distribution P is rescaled, and the probability mass is assigned to a small *nucleus*: the highest probability tokens whose total mass exceeds a predefined threshold p. At each sampling step, the size of the nucleus is dynamically adjusted with the shape of the probability distribution.

Nucleus sampling addresses core issues with top-k methods, illustrated in Figure 4.5.5. The shape of the probability distribution can drastically change between sampling steps. This makes top-k methods that must have a constant k ineffective, and motivates the use of a *dynamic* restriction on possible tokens.

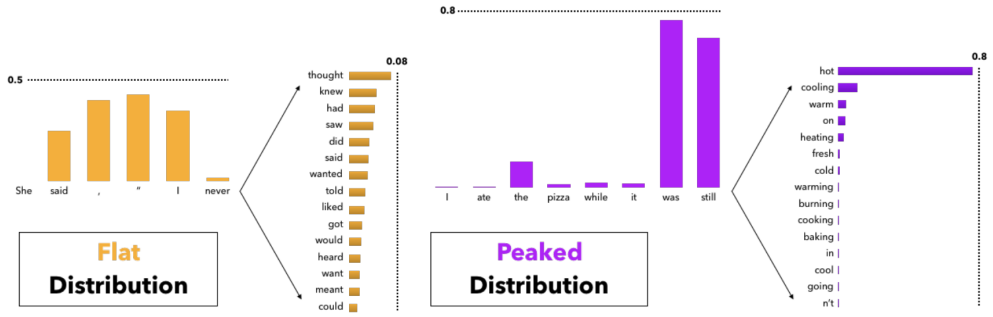

Figure 4.5.5: Probability mass assigned to sentences produced by humans. At each sampling step the probability distribution can have a different shape, requiring different sampling techniques. We notice that *flat* distributions, lead to many promising candidate tokens, making small k values problematic. Conversely, in sharp distributions the probability mass is concentrated on a small number of tokens, making large k values problematic. Figure reproduced from [51].

4.5.4 Language Model Scoring

Evaluation benchmarks, like BLiMP and EWoK (Section 4.4), rely on the *probability* that a LM assigns to a sentence in order to probe the abilities that the model posses in different linguistic dimensions, such as the grammatical proficiency and world knowledge of the model. As presented previously, these benchmarks operate by comparing the probabilities the model assigns to different input sequences, and specifically detecting where the highest probability is assigned. In order to be able to rank sequences, and choose the most likely one, we need to be able to calculate the probability $P(s)$ for a sequence of tokens s.

Calculating the probability $P_{LM}(s)$ for causal or autoregressive LMs is straightforward, given that these LMs naturally model the probability $P_{LM}(s_k|s_{<k})$, using the left (or right) context $s_{<k} = (s_1, ..., s_{k-1})$. Most commonly, the $\log P_{LM}$ is calculated to ease implementation and ensure stability. For a causal LM and a sequence s, it follows that:

$$\log P_{LM}(s) = \sum_{k=1}^{|s|} \log P_{LM}(s_k|s_{<k})$$

For Masked Language Models (MLMs) calculating the probability P_{MLM} is not trivial. The success of these models is attributed to including *bidirectional* context, where a token s_k is replaced with `[MASK]`, and predicted using all the past and future tokens: $s_{\setminus k} = (s_1, ..., s_{k-1}, s_{k+1}, ..., s_{|s|})$. Introduced originally with BERT [27], MLMs have advanced language understanding by employing contextual language representations, and continue to be state-of-the-art in NLU tasks [80, 46, 47]. However, the inclusion of bidirectional context, prevents them from being "true" language models of the probability $P(s)$, as there is no straightforward way to generate text, or produce sentence probabilities from these models. This does not present an issue when the evaluation task requires model finetuning, or the use of contextual representations, but makes out-of-the-box zero-shot evaluation of these models impossible.

To address this limitation, we can view the MLM objective introduced by BERT as a stochastic maximum pseudolikelihood estimation [44], enabling the calculation of a *Pseudo-Log-Likelihood score* (**PLL**) [44], which is computed by masking tokens one at a time. To calculate PLL we sum the conditional log probabilities $\log P_{MLM}(s_k|s_{\setminus k})$ of each token in the sentences, be replacing s_k with the `[MASK]` token. This allows us to calculate the pseudo-probability of a sequence, and extend the application of MLMs to evaluation benchmarks conventionally accessible only to causal LMs (e.g., BLiMP). The calculation of the PLL score is illustrated in Figure 4.5.6. Formally, it is calculated as:

$$\text{PLL}(s) = \sum_{t=1}^{|s|} \log P_{MLM}(s_k|s_{\setminus k})$$

Work done in [116] introduced the PLL score for the re-scoring of automatic speech recognition (ASR) and neural machine translation (NMT) hypothesis, substantially improving performance. Additionally, they show that MLMs can be finetuned without masking, reducing inference complexity. The success is attributed to the models' ability to convey linguistic acceptability without the left-to-right context restriction, greatly improving scores on BLiMP compared to previous models.

Further work done by [55], showed that the above PLL method of [116], overestimates the PLL for out-of-vocabulary (OOV) words, which are split into multiple tokens, and also falls sort of theoretical expectations, such as a positive correlation between MLM and autoregressive LM scores for the same sentences. To alleviate these concerns, in their methodology not only the target token is masked, but also all within-word tokens to the right of the target token. This method, named *PLL-word-l2r*, outperformed the original PLL metric, and

a whole-word masking metric. They also showed that the choice of metric can have substantial effect on the BLiMP benchmark. We use this metric in all our evaluations. An illustration of the different approaches to PPL is given in Figure 4.5.7.

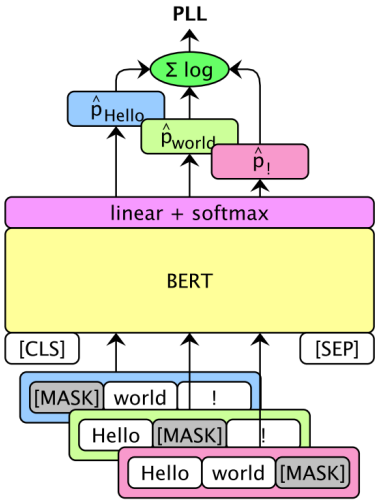

Figure 4.5.6: Pseudo Log-Likelihood (PLL) calculation for Masked Language Models (MLMs). To score a sentence (*e.g., "Hello World !"*), we create copies with each token replaced by `[MASK]`. The log probability of each token is summed, resulting in the PLL score: $\text{PLL}(\textit{Hello}\_\textit{word}\_\textit{!}) = \log P_{MLM}(\texttt{[MASK]}|\textit{world}, \textit{!}) + \log P_{MLM}(\texttt{[MASK]}|\textit{hello}, \textit{!}) + \log P_{MLM}(\texttt{[MASK]}|\textit{hello}, \textit{world})$. The example is here presented for BERT[27], but we follow the same process for our LTG-BERT[120] models. Figure reproduced from [116].

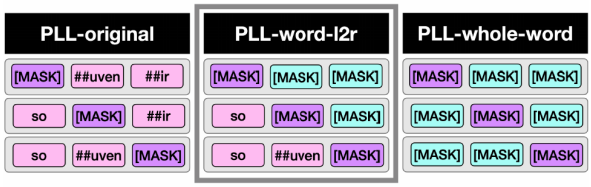

Figure 4.5.7: Illustration of the different possible ways to compute the PLL score for a word separated into multiple tokens. In this example, the word "souvenir" is chosen and for each method, we see the sequences whose probabilities are summed to compute PLL. The PLL-original refers to the original method by [116]. The proposed method, PLL-word-l2r, only additionally masks tokens to the right of a masked token. The PLL-whole-word strategy, masks all the tokens in a word. Within-word tokens are shown in pink, the masked token is shown in purple, and in turquoise the masked within-word tokens. Figure from [55].

Effect of temperature on BLiMP The problem of *Confidence Calibration* is defined as predicting probability estimates representative of the true correctness of likelihood, and is crucial in many real-world applications of classification methods. Intuitively, we want for a classification system to not only be accurate, but also indicate how likely it is to be incorrect. Work done by [44], showed that *temperature scaling*, i.e., changing the temperate T in the Softmax of the logits, can be a straightforward recipe for effective calibration in most settings – i.e., for the probability to truly reflect the confidence of the model.

Preliminary experiments conducted recently by [119], applied the temperature scaling method on the BLiMP dataset. To compute sentence probabilities, the authors used the PLL metric for MLMs, described above [116]. Their findings showed that the temperature T value can *significantly change* performance, by up to 10%. This challenges the use of BLiMP as an appropriate evaluation tool. To solve the issue, the authors propose to make BLiMP invariant to temperature scaling, by reporting the maximum accuracy across all possible temperatures values – instead of just using $T = 1$. The authors note however, that using only one temperature value for all BLiMP subtasks does not account for the significant differences in accuracy between them, but it is a simple solution, than can be conveniently paired with a held-out validations set. Lastly. the point out that future work could focus on a uniform and fair scoring function, which is invariant to external hyperparameters such as temperature. To illustrate the issues uncovered by [119] we include a set of plots from their experiments in Figure 4.5.8.

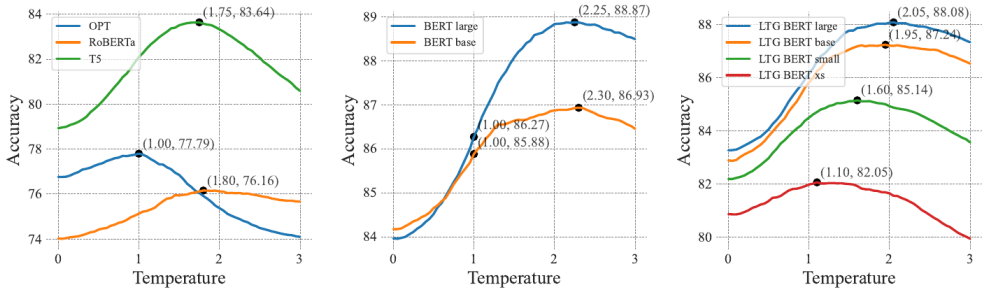

Figure 4.5.8: BLiMP confidence profiles for several LMs. ***Left***: the training objective of the model can drastically change the relationship between temperature (T) and BLiMP performance – different responses are observed for: encoders (RoBERTa [80]), seq-to-seq models (T5 [112]), decoders (OPT, [159]). ***Middle***: for $T = 1$ $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$ performance appears comparable, but the larger model is in fact more capable. ***Right***: four sizes of LTG-BERT[120] models trained by the authors. Measuring at $T = 1$ doesn't demonstrate the true confidence of larger models. Figure from [119].

Chapter 5

Experiments

Contents

In this chapter, we explain in detail the experimental procedures and results for our proposed methodology of *data augmentation* using synthetic story data, seen in Chapter 4 and illustrated in Figure 4.1.1.

First, we describe the experimental setup of our work in terms of hardware and available resources, and detail the preprocessing carried out for the *BabyLM* ($\mathcal{D}_{\text{baby}}$) and *TinyStories* ($\mathcal{D}_{\text{tiny}}$) datasets – the two datasets central to our approach (Section 5.1). Next, we explain the process of selecting *TinyStories* subsets, training GPT-Neo models, and creating the synthetic story data ($\mathcal{D}_{\text{gen}}$). To motivate our subset selection, we evaluate the *linguistic* abilities of GPT-Neo models trained on various amounts of *TinyStories*. We compliment our analysis with metrics of diversity and text quality, and in the process, draw insights about the capabilities of SLMs in low data settings (Section 5.2). Afterwards, we move on to our *data augmentation* method, and train encoder transformer LTG-BERT models on the combined dataset. In order to uncover the effect of the synthetic data on linguistic proficiency, we compare our methods with various baselines, with and without the use of generated data, and explore alternative sampling procedures and training configurations (Section 5.3). Finally, we analyze our findings, and make interesting observations about data augmentation methods for small data language pretraining (Section 5.4).

Additionally, this section is supplemented by a collection of Appendices. Appendix A – containing sample generations from the GPT-Neo models, Appendix B – containing sample interactions with Claude Sonnet-3.5 for the LLM-assisted evaluation, Appendix C – containing the hyperparameters used for training our models, Appendix D – containing detailed evaluation results for the BLiMP, BLiMP Supplement, EWoK and (Super)GLUE benchmarks.

5.1 Preliminaries

Experimental Setup Our experiments are performed on a shared GPU cluster consisting of 8 Nvidia V100 GPUs with 16 GB memory each, and additionally, for our evaluation of our models we make use of an Nvidia RTX-3090 GPU with 24 GB of memory. All models are trained utilizing the PyTorch [100] and HuggingFace [152] libraries. For evaluating BLiMP, EWoK, and (Super)GLUE, we leverage the official evaluation pipeline provided by the challenge organizers [35, 21]. Additionally, for the Self-BLEU score we use the BLEU score implementation provided by NLTK [11]. Our trained models are publicly available on HuggingFace[1], and our implementation is available on GitHub[2].

BabyLM We preprocess the *BabyLM* dataset using the methodology described in [119]. We apply a light cleanup and normalization process, with the goal of casting all the datasets into a unified format. Details about the composition of the *BabyLM* dataset are included in Section 4.2.1. Specifically, we normalize punctuation and whitespace, performing detokenization (*e.g., won␣'t* → *won't*), and capitalizing the first letter in each new line. Additionally, we removed semantically irrelevant and arbitrarily inserted tokens, and normalized incorrectly parsed symbols (*e.g.,* replacing `">"` with `">"`). For the datasets that contain spoken language, direct speech is cast into quotes to ensure a consistent representation. Finally, the structure of the datasets was preserved, where necessary, with the insertion of a special `[PAR]` symbol. For example, in the case of the CHILDES dataset [53] the data consists of spoken interactions with each line corresponding to a speech utterance. These were all concatenated and separated with the `[PAR]` symbol to preserve the unique form of turn-based dialogue.

TinyStories For preprocessing the *TinyStories* dataset, we again leverage the light procedure detailed in [119]. Our aim is to unify the text, and cast it in a similar format to the *BabyLM* dataset. Specifically, whitespace is added where necessary to separate punctuation marks and special symbols, abbreviations are normalized (detokenization), and misaligned double quotes are fixed. Additionally, we normalize any broken unicode characters and the text punctuation. To preserve the structure of the stories, for the encoder training we replace the `<|endoftext|>` token between the stories with a `[PAR]` token, as done for the *BabyLM* dataset. For the decoder training, the `<|endoftext|>` token is preserved.

In the following sections, for a dataset $\mathcal{D}_{\text{data}}$ we adapt again the notation of $\mathcal{D}_{\text{data-m}}$ to indicate a subset of size m in millions of words. Our experiments are conducted for two data constrained scenarios, that correspond to the two text tracks of the BabyLM challenge. Namely, the `Strict-Small` track (maximum of **10M** words) and the `Strict` track (maximum of **100M** words). For each scenario the experiments follow the same process but are performed independently.

5.2 Evaluating GPT-Neo Performance on TinyStories

Our aim is to leverage GPT-Neo models trained on a subset of *TinyStories* ($\mathcal{D}_{\text{tiny-m}}$) for data augmentation. Specifically, for the stories in the training set $\mathcal{D}_{\text{tiny-m}}$ alternate completions are generated by the decoder models, to form the set $\mathcal{D}_{\text{gen}}$. These two datasets combined with a subset of the *BabyLM* dataset ($\mathcal{D}_{\text{baby-b}}$) are used to construct $\mathcal{D}_{\text{comb}} = (\mathcal{D}_{\text{tiny-m}}, \mathcal{D}_{\text{gen}}, \mathcal{D}_{\text{baby-b}})$ which is employed for the training of an encoder transformer LTG-BERT model. As explained in Chapter 4, it is crucial to select the right size m for $\mathcal{D}_{\text{tiny}}$ to effectively balance the proportion of the $\mathcal{D}_{\text{tiny-m}}$ and $\mathcal{D}_{\text{baby-b}}$ datasets in the combined set $\mathcal{D}_{\text{comb}}$. This will ensure that $\mathcal{D}_{\text{comb}}$ is diverse and complex enough for encoder generalization, while maintaining a sufficient pre-training dataset size m for the decoder training in $\mathcal{D}_{\text{tiny-m}}$, that allows high quality generations.

In order to select the size m, we start by evaluating the generative and linguistic performance of GPT-Neo decoder models, while varying the amount of available data. Apart from motivating our selection for $\mathcal{D}_{\text{tiny-m}}$, this analysis will gauge at the relationship between data size and the capabilities of SLMs in low data regimes, allowing us to make interesting observations. Taking into account the challenge limitations for the `Strict` ,

[1] https://huggingface.co/collections/nikitastheo/berttime-stories-66e9272aff154e75c55e6d91
[2] https://github.com/nikitas-theo/BERTtimeStories

and `Strict-Small` tracks, we sample sort stories from the *TinyStories* dataset and define a collection of subsets with a variety of sizes, denoted as $\mathcal{D}_{\text{tiny-m}} : m \in \{5, 10, 25, 50, 75, 100\}$M (millions of words). Our selected sizes are illustrated in Figure 5.2.1. We notice that compared to the full dataset, our models are trained with a magnitude less data. In the following paragraphs, for each subset $\mathcal{D}_{\text{tiny-m}}$ we independently train a GPT-Neo model and evaluate the decoder's linguistic and generative capabilities.

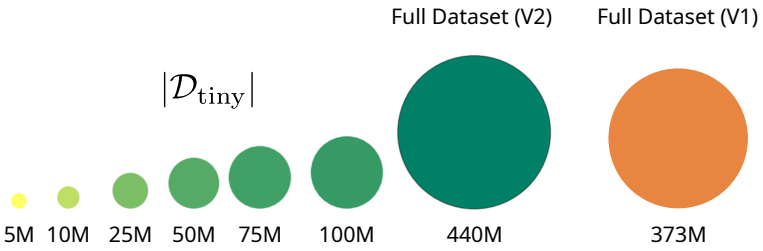

Figure 5.2.1: Selected subset sizes for evaluating the capabilities of GPT-Neo models trained on various amounts of *TinyStories* data. Our experiments leverage the second version (V2) of the data with $\sim$ 440M words. In our analysis, we also include the model released by [31] trained on the first version of the dataset (V1) containing $\sim$ 373M words.

All our experiments are carried out with the second version (V2) of the *TinyStories* dataset, generated only by GPT-4, and containing $\sim$ 440M words. For comparison, results are also reported for the pretrained model provided by the *TinyStories* authors [31], which is trained on the first version of the dataset, created by prompting both GPT-3.5 and GPT-4, and containing $\sim$373M words. We chose the GPT-4 version of the dataset, as it was deemed to be of higher quality. The two GPT-Neo models are reported as: "440M (V2)" – for our model trained on the full second version of the dataset, and "373M (V1)" – for the model by [31].

For the GPT-Neo decoder transformer, we select one of the top-performing architectures[3] from [31]. All our GPT-Neo models are trained with the same set of hyperparameters, except for weight decay, dropout, and vocabulary size, which are adjusted based on the dataset size. We follow a conventional training approach for autoregressive LMs, similar to that of the authors in [31]. However, we found it is important to incorporate some additional regularization to suit our low-data setting. Specific hyperparameters and architectural details can be found in Appendix C. Additionally, for each subset $\mathcal{D}_{\text{tiny-m}}$ a separate tokenizer was trained from scratch, using the WordPiece Tokenizer implementation provided by the HuggingFace library.

5.2.1 Evaluating Language Proficiency

For open-ended generation tasks, such as sort story generation, evaluating generative performance directly can be challenging [31]. For this reason, as an initial assessment of the language capabilities of the GPT-Neo decoders, we evaluate their performance on the zero-shot benchmarks of the BabyLM challenge. Namely, we use: BLiMP, BLiMP Supplement (Supp.), and EWoK, probing the grammatical understanding, and world knowledge of our models. We posit that these benchmarks will provide a *proxy* of the generative abilities of the GPT-Neo models, and will allow the assessment of the models without breaking the data restrictions of our scenarios (10M and 100M words maximum). Results are included in Table 5.2.1 and a graphic illustration is provided in Figure 5.2.2. Full evaluation results of the GPT-Neo models on the three benchmarks, indicating the specific score for each category/phenomenon are presented in Appendix D.4. We note that *random chance* accuracy for these benchmarks corresponds to a value of 50.0, as they involve binary classification. This puts into perspective their difficulty for our models – especially EWoK.

As the amount of data increases, we observe that the BLiMP and EWoK scores generally smoothly increase as well. In contrast, the BLiMP supplement score shows more variance. We note that the **440M word**

[3] https://huggingface.co/roneneldan/TinyStories-33M

model demonstrates slightly worse performance than expected, when taking into account the magnitude of its training data. We posit that this due to the model being under-regularized during training. Interestingly, we notice that the **100M word** model attains the highest BLiMP performance of all the models, sharing this result with the original model released by [31] – 373M (V1). This is in line with the findings of [160], that showed that 100M words is enough for substantial performance on most grammatical phenomena on BLiMP, demonstrating diminishing returns for more amounts of data. Lastly, the **25M word** model is able to attain substantial BLiMP and BLiMP Supp. for its training size, demonstrating overall good sample efficiency.

One other set of observations concerns the relationship between the BLiMP benchmark and its supplement. We notice that **100M word** model has the best BLiMP score, while also attaining subpar scores on the BLiMP Supp. tasks, specifically the lowest of all the models. While carefully looking at the lines in Figure 5.2.2, this result appears to fit in a general trend, where performance on BLiMP is not correlated with performance on its supplement. This finding suggests that the two benchmarks measure substantially different phenomena, and complement each other in their grammatical evaluation. We note however, that BLiMP Supplement is considerably smaller in size, which could also be a cause for the variation in performance observed in our experiments.

Train Data	BLiMP ↑	Supp. ↑	EWoK ↑
5M	55.5	53.8	51.1
10M	58.4	51.6	51.9
25M	59.9	55.1	52.4
50M	62.8	52.8	53.0
75M	64.0	54.8	53.4
100M	**64.8**	50.8	53.1
440M (V2)	64.6	55.0	53.9
373M (V1) [4]	**64.8**	**60.9**	**54.0**

Table 5.2.1: Evaluation results for GPT-Neo models trained on *TinyStories* with various amounts of data, for the BLiMP, BLiMP Supplement (Supp.) and EWoK Benchmarks. We report accuracy on all benchmarks – the (↑) arrow indicates that higher is better.

Selecting TinyStories subsets We now explain how we leverage the scores in Table 5.2.1 to select the *TinyStories* subsets harnessed for training GPT-Neo models, the subsequent creation of $\mathcal{D}_{\text{gen}}$ and the final data augmentation. We mainly use the BLiMP score for the selection, as the most comprehensive metric out of the three for linguistic knowledge. From our observations, the 50M word mark appears to be a threshold, where performance on BLiMP starts to plateau as the amount of training data increases. Additionally, the 50M word budget is a good starting point for a study on *data augmentation*, as it offers enough room for the inclusion of other diverse data in the 100M budget, and a 50% proportion of $\mathcal{D}_{\text{baby}}$ and $\mathcal{D}_{\text{tiny}}$ is intuitively the simplest combination.

Thus, for the `Strict` track, we select the **50M word** subset $\mathcal{D}_{\text{tiny-50M}}$ as the set that will be used – with its corresponding GPT-Neo model – to generate $\mathcal{D}_{\text{gen}}$, and will be included in the combined corpus $\mathcal{D}_{\text{comb}}$. A close contender is the 25M word set, which we plan to include in a future study. For the `Strict-Small` task, we opt to choose the **5M word** model, and the corresponding $\mathcal{D}_{\text{tiny-5M}}$ set, using some of the same reasoning as above. Further investigation into smaller dataset sizes for the `Strict-Small` track will be addressed in future work. Importantly, the LLM evaluation (covered later) is not considered in the above decisions, in order to uphold our data limitations of 10M and 100M words respectively.

[4] Model released by [31], https://huggingface.co/roneneldan/TinyStories-33M

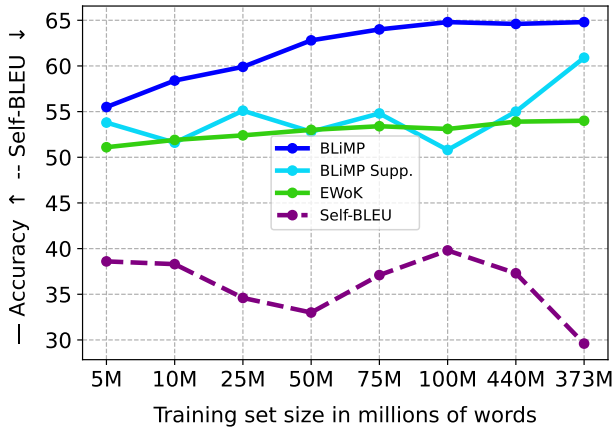

Figure 5.2.2: Graphic illustration of the relationship between the Self-BLEU metric (scaled to [0,100]) and the BLiMP, BLiMP Supp. and EWoK accuracy scores for GPT-Neo models trained on subsets of *TinyStories*, as the amount of training data increases. The (↓) arrow indicates that for Self-BLEU, a lower score corresponds to more diversity and better performance. This is in contrast to accuracy scores in the table, where higher is better (↑).

The scores mentioned above provide insights into our models' grammatical competence (BLiMP) and their world knowledge (EWoK), but they overlook two key aspects of generative performance: ① the *diversity* and ② the *quality* of the generated outputs. These two are necessary for a comprehensive evaluation of the GPT-Neo models and will be the focus of the following sections.

5.2.2 Evaluating Generative Quality – LLM Evaluation

Assessing the *quality* of outputs for open-ended generation tasks can be difficult, as standard evaluation methods typically rely on structured outputs and compare generations against a set of reference texts. In contrast, story generation is essentially unrestricted, with the only expectation that the continuation generated by the model is consistent with the story's beginning. To tackle this issue, we follow the evaluation approach introduced by [31], where a Large Language Model (LLM) is tasked with assessing the stories produced by our models. The authors originally used GPT-4 [95], but instead we leverage Claude-3.5 Sonnet [2] for our experiments, as it better aligns with our available resources.

We utilize a set of 44 manually crafted prompts [5] provided by [31], each containing the start of a story, and constructed to be grammatically challenging for the models to complete. For each of the GPT-Neo models, trained on the subsets $\mathcal{D}_{\text{tiny-m}}$, we generate 10 completions per story using a temperature of 1. We provide the LLM with both the story's beginning and the model's completion, which are presented as a student's response to a story writing assignment. The LLM is then asked to evaluate the response across three dimensions: (a) *Grammar*, (b) *Creativity*, and (c) *Consistency* with the story's beginning and the plot. Additionally, we ask the LLM to classify the story into age groups ranging from 3 and under to 16 years old. The scores for each category are given on a scale of 1 to 10.

For the final results, we average scores for both the 44 stories and the 10 models completions, for a total of around 440 data points per category. A small number of interactions were discarded, as in some extreme cases the LLM refused to rate the assignment (e.g., in the case of profane language usage). Appendix B includes a sample interaction with Claude Sonnet-3.5 presented in its entirety, along with more details about the evaluation, and an analysis of the LLM responses for problematic evaluation cases. The results of our evaluation are reported in Table 5.2.2 and a graphic illustration is included in Figure 5.2.3.

[5] https://huggingface.co/datasets/roneneldan/TinyStories

We notice at first that as expected, performance scales with the amount of training data. As the size of the $\mathcal{D}_{\text{tiny-m}}$ subset grows we see an overall increase in the scores for all three evaluation categories. A notable exception is the **440M word** which generally performs worse or at best on the same level as models with a magnitude less data (e.g., the 25M and 50M word models). Specifically for *Grammar*, arguably the most easily attained skill out of the three, the 400M word model has even worse performance than the 25M word model. As previously, we believe the cause for this behavior is lack of proper regularization and possibly the limit in the number of training steps. Additionally, the **100M word** decoder achieves impressive results across all three categories, achieving the best *Consistency* score out of all the models. This illustrates that 100M words can result in an impressive level of generative performance.

An important observation concerns the **25M word** model, which appears to be a *cutoff point*. Specifically, we notice a rapid, linear increase in the scores for all three categories up to 25M words, and then a decline in improvement with only marginal benefits when adding more training data. This finding suggests that even when limiting the number of words to small data regimes of up to 25M, the quality of the generations is retained. Based on this evaluation, we recommend that future approaches consider 25M words of $\mathcal{D}_{\text{tiny}}$ as a starting point for *data augmentation* with synthetic stories. In our own experiments, we didn't take into consideration the LLM-assisted evaluation when selecting *TinyStories* subsets for data augmentation, as this would invalidate the data constrained setting of our work.

Train Data	Gr. ↑	Cr. ↑	Cons. ↑	SB ↓
5M	4.56	4.99	3.37	38.6
10M	5.31	5.34	3.98	38.3
25M	6.00	5.65	4.55	34.6
50M	6.01	5.53	4.54	33.0
75M	6.08	5.50	4.49	37.1
100M	6.17	5.57	**4.78**	39.8
440M (V2)	5.88	5.53	4.49	37.3
373M (V1)	**6.24**	**5.73**	4.70	**29.6**

Table 5.2.2: Evaluation results for GPT-Neo models trained on *TinyStories* with various amounts of data – LLM-assisted quality evaluation. We instruct Claude-3.5 Sonnet to access generative performance along three categories: Grammar (Gr.), Creativity (Cr.), Consistency (Cons.). We also include the results of Self-Blue (SB).

Finally, we note that compared with the results from [31] the scores for our models, and even the 373M word model released by the authors, appear to be in general lower. Even though in our evaluation we attempted to stay as close as possible to the original setting, this might indicate some differences in configuration (e.g., the exact sampling process). One possibility is also that Claude Sonnet-3.5 (that we used), is a *harsher critic* than GPT-4 (used by [31]). In addition to the quantitative results, Appendix D.4 includes samples of generated completions from the GPT-Neo models, for a variety of stories and sampling techniques – including for the example depicted in Figure 4.1.1, depicting our main methodology.

5.2.3 Evaluating Diversity

To assess *diversity*, we use the Self-BLEU metric [163], which has previously been applied to measure the diversity of generated text [51, 1, 139]. We use 100 stories, that are contained in the training set of all the GPT-Neo models, and truncate them to 15%-30% of their original length to be used as prompts. For each story, the decoder models generate an alternate completion given the story's beginning, leveraging a greedy, maximization-based, decoding strategy. For the collection of 100 stories S_{100} for each model, we calculate the Self-BLEU score, scale it to $[0, 100]$ and report it in Table 5.2.2. A graphic illustration of the score is also

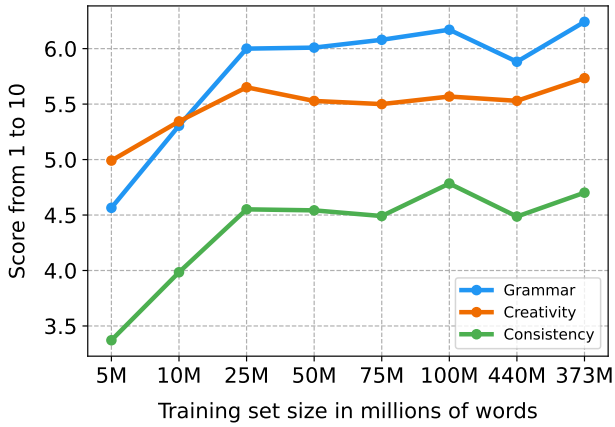

Figure 5.2.3: Graphic illustration of the LLM-assigned scores for *Grammar*, *Creativity*, and *Consistency* with the plot. Scores are asigned on a scale from 1 to 10. We measure the generative abilities of the GPT-Neo models as the amount of *TinyStories* data increases.

included in Figure 5.2.3. Lower scores indicate more diversity.

We notice that the Self-BLEU scores display significant variance, and there does not seem to be a clear correlation with the amount of training data. We note however, that the best diversity score (lowest) was achieved by the **373M word** model by [31]. Additionally, an important observation is that the **25M word** and **50M word** models achieve a low diversity score, while maintaining a substantial quality in their generations. This finding suggests that even with *limited training data*, good generative performance, in terms of diversity, and quality, can be achieved. As we identified previously, the **100M word** model achieves impressive quality scores, including the highest *Consistency*. However, it in terms of diversity it has the worst score (highest) out of all the models. Intuitively, we can understand why maintaining a high consistency with the beginning of the story might come at the cost of a reduced diversity in the model responses. Lastly, reducing the training data to a **5M word** dataset, results in markedly bad performance in terms of both quality and diversity, indicating that data augmentation might be particularly challenging in the 10M word constraints of the `Strict-Small` track.

5.3 Synthetic Story Data Pretraining

In the previous sections we evaluated the linguistic capabilities of GPT-Neo models trained on various subsets of the *TinyStories* dataset, $\mathcal{D}_{\text{tiny-m}} : m \in \{5, 10, 25, 50, 75, 100\}$M. Based on the BLiMP scores of the models, we choose the $\mathcal{D}_{\text{tiny-5M}}$ and $\mathcal{D}_{\text{tiny-50M}}$ *TinyStories* subsets for data augmentation, for each of the `Strict-Small` and `Strict` tracks respectively. We now outline the creation of the combined dataset $\mathcal{D}_{\text{comb}} = (\mathcal{D}_{\text{tiny-m}}, \mathcal{D}_{\text{baby-b}}, \mathcal{D}_{\text{gen}})$, used for training the LTG-BERT encoder LM. For brevity, we provide details only for the `Strict-Small` track, but the same procedures were applied for the `Strict` track as well.

Using the 5M-word subset from *TinyStories* ($\mathcal{D}_{\text{tiny-5M}}$) we train GPT-Neo models as outlined in the previous sections. For each story in $\mathcal{D}_{\text{tiny-5M}}$ we truncate it to 15%-30% of its length, effectively taking the beginning of the story, and use it prompt the GPT-Neo model. The decoder then generates alternate completions for all the stories, which are collected to form the synthetic story dataset $\mathcal{D}_{\text{gen}}$. In our experiments we explore different decoding strategies for sampling from the GPT-Neo model, including *greedy generation* ($\mathcal{D}_{\text{gen-greedy}}$), and *nucleus sampling* ($\mathcal{D}_{\text{gen-nucleus-k}}$), where k represents the number of generations per prompt. A theoretical explanation for various decoding strategies is included in Section 4.5.3. Finally, the story datasets – $\mathcal{D}_{\text{tiny-5M}}$ and $\mathcal{D}_{\text{gen}}$ – are combined with a 5M-word subset from the *BabyLM* dataset – $\mathcal{D}_{\text{baby-5M}}$ – to form the combined dataset $\mathcal{D}^{\text{10M}}_{\text{comb}} = (\mathcal{D}_{\text{tiny-5M}}, \mathcal{D}_{\text{baby-5M}}, \mathcal{D}_{\text{gen}})$. This way we keep the total size $|\mathcal{D}_{\text{tiny-5M}}| + |\mathcal{D}_{\text{baby-5M}}|$ within the 10M word limit. As noted previously, the size of the $\mathcal{D}_{\text{gen}}$ dataset does not count towards data restrictions,

as it was generated by our models in a closed system.

After constructing the combined corpus $\mathcal{D}_{\text{comb}}$, we employ it for the training of an LTG-BERT encoder model. As mentioned before, we chose this architecture as it is optimized for the small data regime, and showed remarkable performance on the previous iteration of the BabyLM challenge [145, 146]. The LTG-BERT models are trained with a span-based MLM objective, and the model architecture incorporates recent improvements in the transformer (details in Section 4.5.1). For the training, we follow the procedure outlined in [120] with some slight but important variations. Our models are trained with a fraction of the compute budget utilized in [120], where the authors used the same amount of FLOPs as BERT [27]. Instead, we reduce both the effective batch size, training steps, and sequence length during training, due to constrains in our available resources and infrastructure. For our implementation we rely on the source code provided by the authors [6] and details about the specific hyperparameters we used are given in Appendix C. We note that we performed a minimal tuning of hyperparameters.

Baselines To evaluate the impact of data augmentation on performance, we train several baselines, varying only the pre-training dataset while keeping all other training conditions constant, including hyperparameters and FLOPs. For the `Strict-Small` track, we form baselines by training LTG-BERT models with 10M words from the *BabyLM* dataset ($\mathcal{D}_{\text{baby-10M}}$) and the *Tinystories* dataset ($\mathcal{D}_{\text{tiny-10M}}$) separately. We also train an encoder using a combination of 5M words from each dataset ($\mathcal{D}_{\text{baby-5M}} + \mathcal{D}_{\text{tiny-5M}}$). These models serve as reference points for evaluating the performance of models trained with various data compositions that include generated data, aiming to assess the impact of data augmentation. The same approach is applied to the `Strict` track, where we train encoders using 100M words from each dataset individually ($\mathcal{D}_{\text{baby-100M}}, \mathcal{D}_{\text{tiny-100M}}$), as well as a combined set of 50M words from each ($\mathcal{D}_{\text{baby-50M}} + \mathcal{D}_{\text{tiny-50M}}$). We also report results for the challenge baselines, which are LTG-BERT [120] and BabyLlama [134]. It is important to note that these models were trained with different hyperparameters than those used in our experiments. Specifically, the LTG-BERT model released by the organizers was trained for approximately 20 epochs on the `Strict` track, while we trained our models for around 27 epochs (20K steps).

Nucleus Sampling In contrast to greedy decoding strategies, with sampling methods we have the option to generate multiple completions for each of prompt. This can be beneficial, resulting in an increase of the amount of training data, but only as long as the added data are *diverse* enough. Otherwise, we risk wasting important computational resources in iterating over uninformative samples, as all our experiments are controlled to use the same amount of FLOPs. To determine a good value of k for nucleus sampling, we measure *diversity* as k increases, by utilizing the Self-BLEU score. We select 100 stories from the $\mathcal{D}_{\text{tiny-5M}}$ set and use the beginning (15%-30%) of each story to prompt the model in generating 50 completions with $p = 0.95$. For each k value in the range $\{2, 3, ..., 50\}$, we compute the Self-BLEU score among the set S_k of generated completions. The goal is to analyze the diversity of the generations for the same prompts as k increases. Figure 5.3.1 illustrates the *average* Self-BLEU across all prompts. We notice that between the values $k = 5$ and $k = 10$ for the 50M word GPT-Neo model, the diversity score stays constant. Based on these results, we opt to experiment with $k = 5$ and $k = 10$ for both the `Strict` and `Strict-Small` tracks, as they provide a good balance between diversity and added data.

Balanced Training Adding the generated data $\mathcal{D}_{\text{gen}}$ increases substantially the proportion of story data in the combined corpus $\mathcal{D}_{\text{comb}}$. This is especially true when employing nucleus sampling, and generating more than one continuation per story. As a result, the sample distribution of the *TinyStories* data ($\mathcal{D}_{\text{tiny}} + \mathcal{D}_{\text{gen}}$) and the *BabyLM* data ($\mathcal{D}_{\text{baby}}$) during model training is also modified. This means that the model encounters more story samples in each iteration, which might affect performance, as it becomes more focused on the *TinyStories* dataset. To mitigate this effect, we introduce a training configuration where the number of samples for the two datasets is balanced on average. In this approach, when constructing each training batch, samples are drawn randomly with equal probability from both *TinyStories* – original and synthesized text – and *BabyLM*. This ensures that the model receives an equal amount of exposure to each dataset throughout training. Results using this balanced training method are marked with the † symbol.

[6] https://github.com/ltgoslo/ltg-bert

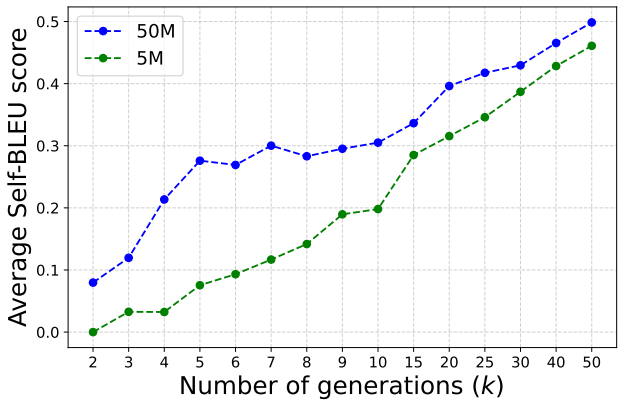

Figure 5.3.1: Average Self-BLEU scores for the 5M and 50M word GPT-Neo models, as the amount k of generations per prompt increases. We sample 100 stories from the training sets $\mathcal{D}_{\text{tiny-50M}}$ and $\mathcal{D}_{\text{tiny-5M}}$, and generate 50 alternate continuations using our models. We compute the average self-BLEU across stories.

Evaluation For assessing the linguistic performance of the LTG-BERT models, we leverage the official evaluation pipeline of the BabyLM challenge [21, 35]. We evaluate our models on filtered versions of (Super)GLUE, BLiMP, BLiMP Supplement (Supp.), and EWoK. For (Super)GLUE, task performance is mostly measured with accuracy, but some of its tasks are evaluated with different metrics (e.g., F1-score). We report the macroaverage of all the task metrics (one selected metric for each task), details in Section 4.4.3. For the other benchmarks we report accuracy. The EWoK, BLiMP and Supp benchmarks are evaluated in a zero-shot way, where no added training is required. For (Super)GLUE we finetune our models, following the process and hyperparameters described in [120]. More details are included in Appendix C.

5.4 Results

We provide the final evaluation results for the LTG-BERT models on both the `Strict-Small` and `Strict` tracks in Table 5.4.3 and Table 5.4.4, respectively. The ⋆ symbol indicates the model submitted to the BabyLM Challenge for each track, and the † symbol indicates usage of the balanced training technique. For each model, we list its training data composition and the total size in millions of words. We also include a macroaverage of the scores on BLiMP, Supp., EWoK and (Super)GLUE in the "Avg" column. More detailed results analyzed for each task/phenomenon of the benchmarks are included in Appendix D.

We make here an important comment about the scores of the LTG-BERT models on EWoK. Our reported results for the challenge baselines significantly differ from the scores released by the organizers – e.g., for `Strict-Small` track we report 63.1 instead of 48.9. After discussions with the community, we hypothesize that this is due to an instabilitiy of this particular architecture, that might result in different scores for different machines and infrastructure. We note that our scores appear to be "inflated", but the increase is consistent for all the LTG-BERT models, allowing for a fair comparison.

Strict-Small Track Looking at the results of Table 5.4.3, initially we observe that the best overall score is achieved by the model trained on the $\mathcal{D}_{\text{baby-10M}}$ dataset. Additionally, comparing the results of the $\mathcal{D}_{\text{baby-10M}}$ and $\mathcal{D}_{\text{tiny-10M}}$ models, we conclude that the *BabyLM* dataset is better suited for language pre-training. This is expected, as the $\mathcal{D}_{\text{baby}}$ dataset was constructed specifically to be a diverse, and developmentally plausible pre-training corpus. However, training on the *TinyStories* dataset appears to benefit EWoK. An interesting observation is that replacing half the of the *BabyLM* dataset with *TinyStories* data – $\mathcal{D}_{\text{baby-5M}}+\mathcal{D}_{\text{tiny-5M}}$ – has only a minor effect on the model's performance. However, as more synthetic story data is added, we notice an overall decrease in model scores, as the positive influence of the *BabyLM* dataset diminishes, and performance trends towards that of $\mathcal{D}_{\text{tiny-10M}}$, where the *BabyLM* dataset was not utilized at all. This indicates that the training becomes overly biased towards the *TinyStories* data, whose number of samples far outweigh those

Model	Training Data	Total	BLiMP	Supp.	EWoK	GLUE	Avg.
LTG-BERT	$\mathcal{D}_{\text{baby-10M}}$	10M	60.6	60.8	48.9/63.1	60.3	57.7/61.2
BabyLlama	$\mathcal{D}_{\text{baby-10M}}$	10M	69.8	59.5	50.7	63.3	60.8
LTG-BERT (**ours**)	$\mathcal{D}_{\text{baby-10M}}$	10M	62.8	**63.7**	66.2	71.0	**65.9**
	$\mathcal{D}_{\text{tiny-10M}}$	10M	59.8	54.2	**67.0**	67.0	62.0
	$\mathcal{D}_{\text{tiny-10M}} + \mathcal{D}_{\text{gen-greedy}}$	20M	58.7	57.8	63.8	67.1	61.9
	$\mathcal{D}_{\text{baby-5M}} + \mathcal{D}_{\text{tiny-5M}}$	10M	62.6	60.7	66.6	**71.2**	65.3
	$\mathcal{D}_{\text{baby-5M}} + \mathcal{D}_{\text{tiny-5M}} + \mathcal{D}_{\text{gen-greedy}}$	15M	62.1	60.2	65.5	70.6	64.6
	$\mathcal{D}_{\text{baby-5M}} + \mathcal{D}_{\text{tiny-5M}} + \mathcal{D}_{\text{gen-nucleus-1}}$	15M	62.5	62.3	63.9	69.5	64.6
	$\mathcal{D}_{\text{baby-5M}} + \mathcal{D}_{\text{tiny-5M}} + \mathcal{D}_{\text{gen-nucleus-1}}$ † ⋆	15M	**63.2**	59.3	65.5	71.1	64.8
	$\mathcal{D}_{\text{baby-5M}} + \mathcal{D}_{\text{tiny-5M}} + \mathcal{D}_{\text{gen-nucleus-5}}$	33M	62.4	60.1	65.8	69.4	64.4
	$\mathcal{D}_{\text{baby-5M}} + \mathcal{D}_{\text{tiny-5M}} + \mathcal{D}_{\text{gen-nucleus-10}}$	56M	61.0	58.4	65.3	69.5	63.6

Table 5.4.3: Final model performance on the `Strict-Small` track (10M words).

of *BabyLM* during training. To counter this effect, we experiment with balancing the number of samples from both datasets in each batch – denoted by †. This modification, leads to improved model performance. Notably, for BLiMP, this balanced setup slightly outperforms the model trained exclusively on $\mathcal{D}_{\text{baby-10M}}$, achieving the highest overall score. Additionally, compared to other data augmentation approaches, this configuration leads to better performance on GLUE.

A second set of important observations, relate to the sampling methods used when creating $\mathcal{D}_{\text{gen}}$ for the data augmentation. Switching from greedy decoding to nucleus sampling improves the model's performance on the BLiMP and BLiMP Supp. benchmarks, but negatively affects performance on EWoK and GLUE. This difference likely stems from the nature of the evaluation datasets. BLiMP emphasizes grammatical understanding, and the greater diversity introduced by nucleus sampling exposes the model to a broader range of linguistic structures and syntactic variations, enhancing its linguistic and grammatical capabilities. On the other hand, EWoK and GLUE prioritize semantic coherence and factual consistency, and the increased diversity from nucleus sampling might introduce noise and less coherent narratives, which could confuse the model and lead to reduced performance. This is supported by the fact that the 5M word GPT-Neo model was used for augmentation, which as noted on Table 5.2.2, has a very low *Consistency* score. Consequently, while the enhanced diversity benefits grammatical evaluation tasks like BLiMP, it may not be as effective for language understanding or knowledge-driven tasks such as GLUE and EWoK.

Model	Training Data	Total	BLiMP	Supp.	Ewok	GLUE	Avg
LTG-BERT	$\mathcal{D}_{\text{baby-100M}}$	100M	69.2	66.5	51.9/65.7	68.4	64.0/67.5
BabyLlama	$\mathcal{D}_{\text{baby-100M}}$	100M	73.1	60.6	52.1	69.0	63.7
LTG-BERT (**ours**)	$\mathcal{D}_{\text{baby-100M}}$	100M	64.0	**67.6**	62.8	**74.0**	**67.1**
	$\mathcal{D}_{\text{tiny-100M}}$	100M	61.2	63.2	63.1	70.6	64.5
	$\mathcal{D}_{\text{tiny-100M}} + \mathcal{D}_{\text{gen-greedy}}$	200M	61.1	59.6	63.8	69.1	63.4
	$\mathcal{D}_{\text{tiny-50M}} + \mathcal{D}_{\text{baby-50M}}$	100M	65.5	65.6	62.5	71.0	66.2
	$\mathcal{D}_{\text{tiny-50M}} + \mathcal{D}_{\text{baby-50M}} + \mathcal{D}_{\text{gen-greedy}}$	150M	**66.6**	63.3	**65.0**	71.8	66.7
	$\mathcal{D}_{\text{tiny-50M}} + \mathcal{D}_{\text{baby-50M}} + \mathcal{D}_{\text{gen-nucleus-1}}$ ⋆	150M	65.6	65.0	64.6	72.7	67.0
	$\mathcal{D}_{\text{tiny-50M}} + \mathcal{D}_{\text{baby-50M}} + \mathcal{D}_{\text{gen-nucleus-1}}$ †	150M	65.2	63.5	64.3	72.6	66.4
	$\mathcal{D}_{\text{tiny-50M}} + \mathcal{D}_{\text{baby-50M}} + \mathcal{D}_{\text{gen-nucleus-5}}$	350M	65.4	64.4	61.2	69.8	65.2
	$\mathcal{D}_{\text{tiny-50M}} + \mathcal{D}_{\text{baby-50M}} + \mathcal{D}_{\text{gen-nucleus-10}}$	600M	63.7	63.3	64.5	69.5	65.3

Table 5.4.4: Final model performance on the `Strict` track (100M words).

Strict Track Analyzing the results on the `Strict` track in Table 5.4.4, we reach similar conclusions as on the `Strict-Small` track. Specifically, the $\mathcal{D}_{\text{baby-100M}}$ model achieves the best performance overall, proving itself a better pre-training dataset that $\mathcal{D}_{\text{tiny-100M}}$. However, a positive influence of the data augmentation technique becomes more apparent here, as incorporating the $\mathcal{D}_{\text{gen-greedy}}$ dataset in the model training, leads to improved performance, compared to all the established baselines – models trained on the two datasets separately, $\mathcal{D}_{\text{tiny-100M}}$ and $\mathcal{D}_{\text{baby-100M}}$, and a combination of the two ($\mathcal{D}_{\text{tiny-50M}} + \mathcal{D}_{\text{baby-50M}}$). This results in the $\mathcal{D}_{\text{gen-greedy}}$ model achieving the best score of all the models on BLiMP and EWoK. Furthermore, the $\mathcal{D}_{\text{tiny-50M}}+\mathcal{D}_{\text{baby-50M}}$ baseline with no synthetic data, is outperfomed by both the $\mathcal{D}_{\text{gen-greedy}}$ and $\mathcal{D}_{\text{gen-nucleus-1}}$ models, indicating that synthetic data can provide modest improvements in the `Strict` track setting.

Interestingly, while the nucleus strategy performed better overall for both tracks, the greedy decoding strategy resulted in significant improvement on BLiMP and EWoK scores for the `Strict` scenario, outperforming nucleus sapling for these benchmarks. For both settings however, increasing the amount of *TinyStories* data during training, negatively impacts final model performance, bringing them closer to the $\mathcal{D}_{\text{tiny-10M}}$ and $\mathcal{D}_{\text{tiny-100M}}$ models. In the case of the `Strict` track, we didn't observe a benefit when applying the balanced training strategy (†). We hypothesize that for the larger data regimes of 100M words, the vast amounts of *TinyStories* data – reaching up to 500M words – may overshadow the presence of the *BabyLM* data in training. The model encounters such an increased mass of *TinyStories* content that the learning process is effectively dominated, diminishing the benefits of balancing.

Chapter 6

Conclusion

6.1 Discussion

The study conducted in this thesis, is part of current efforts to mitigate the issues with the modern language model (LM) training regime. Specifically, the purpose is to counter the ongoing trends of vastly increasing computational resources, model parameters, and most importantly training data. For this reason, recent approaches train *efficient* LMs with limited access to data, with the goals of democratizing research, developing data-effective training recipes, and drawing parallels between the way machines and humans process and learn language. Joining in on these efforts, our work is conducted and published [132] as part of the 2024 BabyLM Challenge [21] – a competition that incentivizes LM pretraining with human-like, developmentally plausible data constraints. Our LMs are trained in a limited data setting, working towards the two tracks of the challenge, the `Strict-Small` track (max of 10M words), and the `Strict` track (max of 100M words).

Our methodology for LM pretraining on small data scales is based on *data augmentation*. Taking inspiration from recent work in [31], we train GPT-Neo decoder models on a small collection of simple stories, and create a synthetic corpus by providing an alternate completion for each story in the training set. The augmented story data is supplemented by a diverse, speech focused corpus – the *BabyLM* dataset – and the combined data is leveraged to train LTG-BERT encoder transformers. To investigate the effect of *data augmentation* on linguistic performance, we compare our models with baselines trained without synthetic data in controlled experimental settings, and additionally investigate a variety of sampling techniques and training configurations. The LMs are evaluated using a collection of various benchmarks, measuring grammatical knowledge, language understanding and world knowledge.

Initially, to estimate the right amount of *TinyStories* data for effective augmentation we evaluate the generative and linguistic abilities of GPT-Neo decoder models trained on various amounts of *TinyStories* data – $\{5, 10, 25, 50, 75, 100\}$M (millions) of words. For our assessment, we utilize benchmarks for grammatical and factual understanding, as well as a diversity metric and an LLM-assisted quality evaluation. Our study reveals that Small Language Models (SLMs) can acquire substantial linguistic abilities, and generate stories of high *quality* and *diversity*, even when trained in scarce data scenarios. Specifically, the performance of models trained with up to **25M** and **50M words** was comparable to models trained with a magnitude more data – 373M and 440M words – and **100M words** proved enough for robust grammatical and generative performance. This finding sheds new lights in the capabilities of SLMs, underscoring the potential these models have for data augmentation, even when faced with severe data limitations.

Our next results concern the evaluation of LTG-BERT models, trained on a dataset augmented with synthetic story data. Our findings reveal that while high quality data generation is possible for limited data regimes, leveraging the synthetic data for data augmentation in practice can be challenging. We observed **some benefits**, mostly in grammatical knowledge, in the `Strict-Small` task, with the careful **balancing** of the proportions of different datasets showing potential for improvements. Some more pronounced performance gains were reported for the `Strict` task, pertaining to linguistic and world knowledge. However, overall synthetic data had a **negative influence** on the linguistic abilities of the encoder models, and increasing the size of the generated datasets decreased performance even more. Given the high generative performance of the GPT-Neo models, we believe that there is merit to the data augmentation methodology we propose, but key details of the configuration need to be taken into account and hyperparameters need to be finely tuned. There is a delicate balance between data quantity, quality, and effective incorporation of the synthetic data, that is needed to achieve optimal training outcomes. Some fruitful directions for future work are outlined in the next section.

Based on our analysis we can recommend the following when performing *data augmentation* in scenarios with low data constraints:

- For the GPT-neo models, our evaluation of linguistic performance and generation quality point to a starting point of **25M words** of *TinyStories* data for future data augmentation attempts. If working with a bigger budget, **100M words** would be preferable. These settings offer a nice balance between data size, and output quality.
- The *BabyLM* dataset proved more advantageous for data scarce LM pre-training. This indicates that **complex data** is more beneficial than simple data, and should be prioritized when operating on a limited budget.
- A **calibration of hyperparameters** should be attempted as we believe it significantly affects final performance. For example, the truncation ratio of the stories, the proportions of the *BabyLM* and *TinyStories* data in the final corpus, and the training hyperparameters for the encoder transformer models, among other parameters, should be more finely tuned.
- For both the `Strict-Small` track (10M) and the `Strict` track (100M) **similar findings** were observed. This indicates that new approaches should be **first tried** on the computationally cheap **10M word** scenario, and later **scaled up** to the 100M scenario. Additionally, **easy to evaluate** benchmarks, such as BLiMP, offer a quick and inexpensive way to estimate model performance, and should be prioritized in the experimentation phase.

6.2 Future Work

Due to time and resource constraints, our work faces several limitations, which can be exploited as promising avenues for future work. For both the *encoder* and *decoder* modules of our methodology, a single architecture was employed – the GPT-Neo and LTG-BERT architectures respectively. An exploration of more model architectures would be beneficial in increasing the robustness of our results. Additionally, we only leveraged two datasets, which were selected because of their specific properties. *TinyStories* provides simple and easy to understand language, that enables high quality decoder generations even for small data, while *BabyLM* provides the required complexity and diversity for encoder generalization. Both of these datasets can be exchanged for other datasets with similar characteristics, e.g., Simple Wikipedia, and the BNC corpus respectively (Section 4.2). We note that above choices were made to ensure consistency in our experiments, and allow us to reach reliable conclusions, but we recognize that they restrict the broader applicability of our findings.

Another interesting idea in the same direction, is to also utilize parts of the *BabyLM* dataset for data augmentation instead of staying limited in the *TinyStories* data. Concerning the formation of subsets for the two datasets, we sampled *randomly* on the document level. In the future, we believe that experimenting

with content-aware data selection for the subsets will significantly improve the results of our technique, as the more simple *TinyStories* data is better supplemented by the advanced portions of the *BabyLM* data. We propose the use of information theory, and other complexity metrics for more beneficial document selection.

One other area for further research is the precise calibration of the parameters of our methodology. For example, the story truncation ratio, the dataset inclusion ratio in the final corpus, and the training hyper-parameters of our models could all be more finely tuned. One important such choice, is selecting the right amount of *TinyStories* data, used for training the GPT-Neo models, and the synthetic data generation. In our experiments we used a simple 1:1 ratio for the two datasets, but more precise control over the dataset mixture could amplify the befits of our method. In the previous section, we recommended experimenting with the 25M word GPT-Neo model, as it proved in hindsight to offer a more attractive ratio of performance to training size. Moreover, the 5M word model in the `Strict-Small` scenario might be a suboptimal choice, and more experimentation with models trained on less than 10M words would be beneficial. We also note that curriculum learning approaches combined with data augmentation techniques could be promising, e.g., as seen in the Contextualizer [153] in last year's challenge.

By addressing the limitations of our work and highlighting promising research directions we hope to inspire further research in the area. We believe that efforts in data augmentation for data constrained and cognitively plausible language pre-training scenarios will prove important for the future of LM pre-training.

Appendices

This chapter supplements the main text with additional material and analysis of the experiments conducted during our study. Initially, we include more sample generations from the GPT-Neo models trained on TinyStories (Appendix A), and additionally provide a full example interaction with Claude-3.5 for the LLM-assisted evaluation (Appendix B). The full list of hyperparameters for the training of the GPT-Neo and LTG-BERT models is provided (Appendix C), along with detailed evaluation results for each of the BLiMP, BLiMP Supplement, EWoK and GLUE benchmark, analyzed per task or phenomenon (Appendix D).

A TinyStories – Sample Generations

To provide a clear example of the augmentation process and allow direct evaluation of generation quality, we present sample outputs from all our GPT-Neo models: {5M, 10M, 25M, 50M, 75M, 100M, 440M (V2)}, as well as the 33M parameter model released in *TinyStories* [31] – which we denote as 373M (V1). For the generations provided below, we select a story from the training set, truncate it to 15%-30% and prompt the models to complete it. For the sampling, we experiment with various decoding strategies – explained in Section 4.5.3 – in order to qualitatively compare the effects of each sampling method to the final model outcome.

- Table A2 includes model generations with greedy decoding. The completions correspond to the story presented in Figure 4.1.1.
- Table A3 includes model generations when sampling with a temperature of 1.
- Table A1 includes model generations with nucleus sampling ($p = 0.95$).

To enable the comparative evaluation of different completions for the same story, for the 5M and 50M word models we also include 3 different generations for the same prompt using nucleus sampling ($p = 0.95$). The generations are presented in Table A4 for the 5M model, and Table A5 for the 50M model.

Model	Generated Text
Prompt	One day, a little boy named Tim went for a walk in the woods. He saw a big hole in the ground. Tim was very
Original	curious. He thought it might be a secret way to the underground. He wanted to see what was down there. Tim went down the hole. It was dark and scary. He heard a twist sound. He saw a big twisty snake. The snake said, "Hello, I am Sam the snake. What are you doing here?" Tim said, "I wanted to see the underground. But it is too scary for me." Sam the snake said, "It is not scary. I can show you around." Sam showed Tim the underground. They saw bugs and rocks. Then, they found a big shiny treasure box. Tim and Sam were very happy. They took the treasure box back up to the woods. Tim was not scared of the underground anymore.
5M	curious and wanted to see what was behind the hole. Tim thought, "I will peek inside and see what's inside." Tim went into the hole and found a root. He put the root in the hole and turned it. It was a small, old ball. Tim was happy and excited. He played with the ball all day long. He kicked it, and even pretended it was a little car. But then, something unexpected happened. The root started to talk! "Hello, Tim," said the rock. Tim was very surprised. He didn"t know that roots could talk. The root said, "Hello, Tim! I am a magic root. I can make things happen if you want." Tim thought for a moment and said, "I wish for a big ice cream!" The magic root made their wishes come true. Tim wished for a big ice cream cone and his ice cream cone, and he could even make a big ice cream cone with the same size he wanted. The magic root made their wish come true. Tim wished for a big ice cream cone cone, and his red ball appeared.
10M	curious. He wanted to see what was in the hole. Tim went closer to the hole and realized he was very deep. He called out, "Help! I am stuck!" A kind rabbit heard Tim"s cries and hopped into the hole. The rabbit said, "Don"t worry, I will help you." The rabbit hopped into the hole and found Tim. The rabbit said, "Thank you, Tim! I need to go to the other side." Tim was happy he could help his new friend. He said goodbye to the rabbit and walked back to his house. When he got there, his mom was waiting for him. "You are so kind, Tim!" she said. "You made a great toy for me." Tim smiled and hugged his mom. From that day on, Tim and the rabbit were best friends. They played together in the woods every day.
25M	curious and wanted to see what was inside. He went closer to the hole and looked inside. He saw a lot of black rocks and stones. Tim wanted to examine the rocks, but he was a little frightened of them. Just then, a little bird flew down and said, "Hi, Tim! Do you want to examine the black rocks with me?" Tim nodded and they both looked at the rocks together. The bird said, "Let"s see if we can find something from the hole." Tim and the bird went into the hole and found many shiny things. They were so happy and played with the shiny things all day long.
50M	curious about the hole. He wanted to know what was inside. So, he said, "I wonder what"s inside!" Tim looked into the hole and saw a tiny bug. The bug was upset. Tim asked, "Why are you upset?" The bug said, "I can"t believe I can see the dark too. My wings hurt from playing in the mud." Tim wanted to help the bug. He went inside the house and got a soft cloth. He wrapped the cloth around the bug"s wings. The bug felt better and started to believe in its light. Now, the bug could see the dark in the woods with Tim. They both felt happy and safe together. And from that day on, Tim and the bug became good friends
75M	curious and wanted to see what was inside. He looked into the hole and found a small toy car. Tim picked up the toy car and said, "Wow! A pretty car! I will play with it, but I need to get inside the hole first." Tim walked and walked until he found a small door in the hole. He opened the door and saw a secret room full of toys! The toys said, "Hi, Tim! We can play with you!" Tim was so happy and surprised. He played with the toys all day long and had a lot of fun.
100M	curious and looked inside the hole. He saw something moving. Tim was a curious rabbit from behind the hole. He poked the rabbit with his finger. The rabbit jumped out of the hole and said, "Hi Tim! Do you want to play with me?" Tim was so happy to find a new friend. They played together all day, running around the woods. They found many wild animals like birds, rabbits, and squirrels. Tim had so much fun with his new friend. At the end of the day, Tim was tired but very happy. He couldn"t wait to tell his family about the wild rabbit he found.
440M (V2)	curious, so he got closer to look at the hole. He was a bit scared, but he wanted to know what was inside. As Tim looked into the hole, a small animal came out. The animal was very rude. It did not say "please" or "thank you" to Tim. Tim felt sad and said "sorry" to the animal. The animal gave Tim a small hug. Then, something unexpected happened. The rude animal turned into a kind fairy! The fairy said "thank you" and flew away. Tim was very happy and went home with a big smile on his face.
373M (V1)	curious about the hole. He decided to step closer and look inside. Tim found a little squirrel inside the hole. The squirrel said, "Hi, I"m Tom! I like to play in this hole. Do you want to play with me?" Tim smiled and said, "Yes, let"s play in the hole!" They played together all day long. They laughed and had fun. When the sun went down, Tim said, "I have to go home now, Max. But I"ll come back to play again soon." Max agreed, and they promised to meet again the next day. Tim went home, happy to have made a new friend.

Table A1: Sample generations from our all our GPT-Neo models using *nucleus sampling* with $p = 0.95$. The prompt was taken from the training set of the models.

Model	Generated Text
Prompt	One day, a little boy named Tim went to the park. He saw a big tiger. The tiger
Original	was not mean, but very easy to play with. Tim and the tiger played all day. They had lots of fun. Then, something unexpected happened. The tiger started to shake. Tim was scared. He did not know what was going on. But then, the tiger turned into a nice dog. Tim was very surprised. Tim and the dog played together now. They were very happy. The dog was easy to play with too. At the end of the day, Tim went home with his new friend.
5M	was very big and had a long tail. Tim wanted to be friends with the tiger. Tim said, "Hi tiger, I am Tim. Do you want to play with me?" The tiger looked at Tim and said, "Yes, I want to play with you." Tim and the tiger played all day. They ran, jumped, and laughed. The tiger was very happy. But then, the tiger got tired. It wanted to rest. Tim said, "I am tired. Let's rest." The tiger yawned and said, "Okay, Tim. Let's rest." Tim and the tiger sat under a tree. They talked and laughed. The tiger was happy to have a new friend. They both had a fun day at the park.
10M	was very big and had a long tail. Tim was scared, but he wanted to be brave. He walked up to the tiger and said, "Hi, tiger! I am Tim. Do you want to play with me?" The tiger looked at Tim and said, "Yes, I want to play. But I am scared of the big tiger." Tim was not scared. He thought the tiger was nice. Tim and the tiger played together. They ran and jumped. They had a lot of fun. Then, something unexpected happened. The tiger started to change. It became a tiger! The tiger was not a tiger anymore. It was a tiger! Tim was very surprised. The tiger was not a tiger anymore. It was a big tiger! Tim and the tiger became best friends. They played together every day.
25M	was very big and had a long tail. Tim was scared, but he wanted to be brave. Tim said, "Hi, tiger! I am Tim. Do you want to play with me?" The tiger looked at Tim and said, "Yes, I want to play. But I am very hungry. Can you help me find some food?" Tim thought for a moment and said, "Yes, I can help you. Let's look for food together." They looked and looked, but they could not find any food. Then, something unexpected happened. The tiger started to talk! "I am not a tiger, I am a magic tiger. I can make your wishes come true," said the tiger. Tim was very surprised. He wished for a big ice cream cone. The magic tiger made his wish come true. Tim and the magic tiger became good friends and played in the park every day.
50M	was very nice. Tim wanted to play with the tiger. He said, "Hi, tiger! Let's play!" The tiger did not say anything. Tim was sad. He wanted to find a way to make the tiger happy. Tim had an idea. He found a ball and threw it. The tiger saw the ball and started to play with it. Tim was happy. He said, "Now, the tiger is happy too!" They played together all day. The tiger was not sad anymore. Tim and the tiger became best friends.
75M	was very big and had a long tail. Tim was not scared. He wanted to be friends with the tiger. Tim said, "Hi, tiger! My name is Tim. Do you want to play with me?" The tiger looked at Tim and said, "Yes, I want to play. My name is Sam." Tim and Sam played all day. They ran, jumped, and laughed. They had so much fun. Tim was happy to have a new friend. At the end of the day, Tim said, "Bye, Sam! I will come back to play with you again." Sam said, "Bye, Tim! I will wait for you here." Tim went home and told his mom about his new friend.
100M	was sad. Tim asked, "Why are you sad, Mr. Tiger?" The tiger said, "I lost my toy. Can you help me find it?" Tim wanted to help the tiger. They looked for the toy together. They looked under the trees and behind the bushes. Then, they saw a big box. The tiger said, "I think my toy is in the box." Tim opened the box. Inside, there was a big cake! The tiger was happy. But, the cake was not a real tiger. It was a toy tiger! Tim laughed and said, "Silly tiger, you are not a real tiger!" They played with the toy tiger and had a fun day at the park.
440M (V2)	was very hungry. Tim wanted to help the tiger. He thought, "I will feed the tiger some food." Tim went home and got some food. He came back to the park and gave the food to the tiger. The tiger ate the food and was very happy. Tim felt good because he helped the tiger. The tiger said, "Thank you, Tim. You are a good friend." Tim and the tiger played in the park all day. They had lots of fun. Tim learned that it is good to help others. The tiger learned that it is good to be kind and help others. And they both lived happily ever after.
373M (V1)	was sad. Tim wanted to help the tiger. He thought the tiger needed a friend. Tim went to the tiger and said, "Hi, I am Tim. I want to help you." The tiger looked at Tim and said, "Thank you, Tim. I am sad because I am alone. I need a friend." Tim thought for a moment and said, "I will be your friend, tiger. We can play together." The tiger was happy. Tim and the tiger played all day. They had lots of fun. The moral of the story is: Helping others can make you happy too.

Table A2: Sample generations from our all our GPT-Neo models using *greedy decoding*. The prompt was taken from the training set of the models.

Model	Generated Text
Prompt	One day, a little girl named Lily wanted to bake a cake. She asked her mom, "Can I bake a cake, please?" Her mom said, "Yes, but we
Original	need to buy some things first." They went to the store to buy the things they needed. Lily saw an expensive jar of olives. She asked her mom, "Can we put olives in the cake?" Her mom said, "That's a funny idea, but let's try it!" When they got home, Lily and her mom mixed all the things in a big bowl. They put the olives in the cake too. Then, they put the cake in the oven to bake. When the cake was ready, they took it out and let it cool. Lily and her mom tried the cake with the olives. They both laughed and said, "This is a very funny cake!" They had fun making and eating the cake together. The end.
5M	can bake it together." Lily was very happy. Lily and her mom baked the cake. They put flour, eggs, and milk in a big bowl. They baked the cake. They were very excited. When the cake was done, they put icing on top. Lily and her mom sat in the oven to cook. They smelled the cake. It smelled so good. They were so happy to bake the cake. They knew they would have more fun baking together. They knew that the cake would be great day.
10M	need a different mixer to mix it first." Lily was very excited. She thought it would be fun to use the unknown mixer. Lily and her mom mixed the unusual mixer. They mixed and mixed until it was all done. They were very happy with their surprise. The pretty colors were colorful and fun to look at. Lily's friend, Tom, came over and saw the mixer. "Wow!" he said. "That mixer looks fun!" They played with the mixer and had a great time. When it was time for Tom to go home, Lily's mom said, "It's time for dinner!" Lily and Tom were sad, but they knew they had a special time. They learned that it's okay to ask for help when we can't do something, but also to learn how to do things together.
25M	need a mixer to mix the cake." Lily and her mom went to the store. They bought a mixer to mix all the things they needed. They mixed the cake stuff and put it in the oven. While they waited for the eggs to bake, Lily was became restless. She wanted to go outside now. Her mom said, "Oh no, we must finish baking before the cake starts to bake." Lily was sad, but she knew her mom was right. She put the mixer in the oven and waited. When the cake was done, they took it out and let it cool down. The cake was perfect! Lily was happy she waited and helped her mom bake the cake.
50M	need to watch the clock." So, Lily and her mom opened the oven to bake the cake. They made a big, warm cake just for Lily. They were so happy to eat the cake together. After that, Lily and her mom baked more cakes. They shared the warm cake with their friends. Everyone loved the cake and said it was the best cake they ever had.'
75M	ed to clean up first." Lily and her mom started to clean up the waste. They put all the things they found outside their house. They found sticks and mud. They put them all in the dirt. Then, they put the sun, dirt, and mud in the mud. When they finished, they put the cake in the oven to bake. When it opened the oven, they saw lots of green balls inside. They took the green balls out and put them in a bowl. The cake was the best cake they had ever seen! When they were done, Lily and her mom took it out and let it cool. Then, they put the green balls around the cake to dry. The cake was no longer dead! It was now a yummy-looking cake. Lily and her mom were very happy and enjoyed the cake together.
100M	need to use the cauliflower to make it big." Lily and her mom went to the kitchen. They put the cauliflower on a plate. They mixed the cake batter with a big spoon. Then, they put it in the oven to bake. While they waited, Lily and her mom talked. They were happy. They enjoyed their tasty cake together. When the cake was good, they took it out of the oven. It looked yummy. Lily said, "Thank you, Mom, for helping me bake the cake." They both laughed and enjoyed their cake.
440M (V2)	need to measure to find the things we need." They measured flour, eggs, and milk. They put the mix in a big pan and put it in the oven. While the cake was baking, other kids came to play. They saw the baked cake and wanted to try it. But Lily said, "No! That cake is sour!" Her friends were sad and started to march like a soldier. Lily went outside and told the other kids, "It"s just a mix for cake!" They all tried the cake and made a funny face. Lily laughed and said, "See, it"s not bad, it"s just a cake!"
373M (V1)	need to take the lemons from the fridge." Lily wanted to help, so she was very impatient. Lily and her mom went to the kitchen. Lily"s mom showed her how to pour the lemons into a bowl. Then she told Lily to mix them with a spoon. Lily tried to mix, but the lemons just rolled all over the bowl. They couldn"t finish the cake. Lily felt sad. She said, "Mom, I"m sorry. Can we fix the cake?" Her mom said, "I don"t know, but we can bake another cake!" They both felt better and baked a new cake, even better than before.

Table A3: Sample generations from our all our GPT-Neo models when sampling with a temperature of 1. The prompt was taken from the training set of the models.

Model	Generated Text
Prompt	Once upon a time, in a big green forest, there lived a panda named Pandy. Pandy was a good panda who loved to cook. Every day, Pandy
Original	would pick fresh bamboo from the forest to cook for his friends. One sunny day, Pandy"s friend, Birdy, was feeling sad. Pandy wanted to help Birdy feel better. He thought, "I will cook Birdy"s favorite food!" So, Pandy cooked a big bowl of tasty bamboo for Birdy. When Birdy saw the bowl, she felt happy and smiled. Birdy shared the yummy bamboo with her friends. All the animals in the forest enjoyed the food Pandy cooked. They all thanked Pandy for being a good friend. The forest was full of happy animals. The moral of the story is: when you do good things for others, it can make everyone happy.
5M - Generation ①	would go to the woods to find some food. He walked and walked, until he met a new friend. This friend was a small, green frog. The frog was very small, but he could not find any food. Pandy was sad. He wanted to find food and make the frog happy. Pandy and the frog played all day. They had so much fun. When it was time to go home, Pandy said, "Goodbye, frog! See you soon!" The frog smiled and said, "I had fun, Pandy. I'm happy I could help you." Pandy went back to his forest, feeling happy and full of energy.
5M - Generation ②	would wake up early to eat bamboo and make all kinds of yummy food. Pandy was very happy when he was cooked, he would give up early and take some rice from the forest. One day, Pandy found a big pile of rice on the ground. He thought it would be fun to use it to cook a big meal with his animal friends. So, Pandy called all his friends to come and eat the rice. They all came and looked at the rice and started to eat it. Suddenly, the rice started to move! Pandy and his friends were surprised. They never thought rice was a rice, but a big green frog! The frog had been hiding in the rice! He was scared at first, but then he realized that he was a nice frog who could help them. So, Pandy and his friends worked together to make the rice. The moral of the story is that when we work together, we can make things better and less uncomfortable.
5M - Generation ③	would prepare yummy food for everyone in the forest. All the animals in the forest would come to eat Pandy. One day, Pandy saw a big pile of waste near a house. He thought, "If I cook here, I will help with the waste." So, Pandy started to cook the waste. He put trash in the bin and put it in a big pot. The waste would make the waste go away. Pandy was very happy to help. Pandy cooked the waste in the forest. All his panda friends came to eat and thanked Pandy for being such a good panda. From that day on, Pandy and his friends enjoyed the food together.

Table A4: For the same prompt, taken from the training set, we include 3 distinct generations from our GPT-Neo 5M word model, using *nucleus sampling* – $p = 0.95$.

Model	Generated Text
Prompt	Once upon a time, in a small town, there was a kind man named Tom. Tom loved to help people. One day, he found a big bottle of wine. He knew it was not for kids, but he had an idea. Tom
Original	had a big kite. He wanted to fly the kite very high. He thought, "I can attach the bottle of wine to the kite. It will make the kite fly higher." So, he tied the bottle to the kite with a long string. He went to the park on a mild day to fly the kite. As the kite went up, everyone saw the bottle of wine. They asked Tom why he did that. Tom said, "I want the kite to fly high, and I want everyone to remember that we should not drink wine. It is not good for us." The people smiled and clapped. They learned a good lesson that day
50M - Generation ①	went to the town and gave the wine to a little girl named Lily. "Here, drink this. It is not good," said Tom. Lily took a sip and felt sad. She did not like the taste. Tom saw Lily was sad and gave her a hug. "It's okay, Lily. We all make mistakes. We should always tell an adult." The moral of the story is to always tell an adult before doing something new.
50M - Generation ②	took the wine to the park. He saw a little girl crying because she lost her toy. Tom wanted to comfort her and make her happy. He gave her the wine with a big smile. The little girl stopped crying and smiled at Tom. But then, Tom felt bad because he was being selfish. He wanted to help more people. The moral of the story is that being selfish is not good and that sharing is important.
50M - Generation ③	went to his friend, Sam. Sam was not so good. Tom said, "Sam, look at this. We need to yield it to others. If we give it to them, they will be happy." Sam agreed. They went to find their friends and give the wine to them. All the friends were very happy. They ate the wine and started to feel good. Soon, more friends came to Tom and Sam. They all started to help. They knew that being kind and yielding to others was important. The people in the town learned to yield and give the wine to everyone. They all lived happily ever after. The moral of the story is to always be kind and yield to others.

Table A5: For the same prompt, taken from the training set, we include 3 distinct generations from our GPT-Neo 50M word model, using *nucleus sampling* – $p = 0.95$

B LLM Evaluation

We include details about the evaluation we conducted using a Large Language Model (LLM). Specifically, for our experiments we utilized Claude Sonnet-3.5 [2] [1]. The model is instructed to act as a "teacher", who is tasked to grade a story completed by a student in the context of a school exercise. The model is asked to grade a response along three axes: (a) *Grammar*, (b) *Creativity*, (c) *Consistency* with the plot, and the beginning of the story. Additionally, the model is requested to provide a guess for the age of the "student" along the following categories: A: 3 or under, B: 4-5, C: 6-7, D: 8-9, E: 10-12, F: 13-16.

The full results and details about our evaluation are described in Chapter 5, Section 5.2.2. Our evaluation is conducted with the scheme presented in *TinyStories* [31] as *"GPT-Eval"*. We aimed to be as close to the original work as possible, with the exception of the employed LLM – which was changed to Claude Sonnet-3.5 instead of GPT-4, and the prompt – which was slightly adapted to work better with our chosen LLM.

Below, we include a sample of an interaction with the LLM while grading a response generated by the GPT-Neo model trained on 50M of *TinyStories* data. The interaction is included in Table B1. We notice that the LLM gives a detailed explanation about its judgement, and tries to substantiate the rating and any claims about the model response, by quoting specific sections.

Next, we describe some interesting model behavior we encountered during our evaluation. We observed that, in some rare cases, the LLM refused to grade the model generations, even though it was specifically instructed to do so. It is valuable to examine such instances, as they shed light into the challenges of an evaluation scheme utilizing large language models, and can additionally help us document some of the limitations and failure cases of LLMs, in a realistic use case. In the calculation of the final ratings of the models (Table 5.2.2, Figure 5.2.3) these extreme cases were excluded.

Specifically, we identified two types of model responses, that resulted in a refusal by the LLM to grade the model generations.

1. *Offensive Remarks.* In the first type, the model's response contains derogatory or offensive language. We give an example in Table B2, generated by the GPT-Neo model trained on 5M words – a similar behavior was also observed for the 10M word model. We observe first that SLMs trained on very low amounts of data can exhibit problematic behavior in their generations, similar to that encountered by larger language models [124]. Additionally, even though the generation contains various grammatical and semantic errors, which the LLM identified, it did not assign a low rating but rather rejected altogether the completion from the process.

2. *Perfect Completion.* In the second type, the model ignores the *** separator, and instead asserts that this is not a student's response (contrary to the prompt instructions), but instead it is a perfect completion to the original story, and thus there is nothing to grade (!). This behavior was observed for the 5M, 25M, 50M word models and the 33M parameter model by [31], trained on 373M words of the original *TinyStories* data (V1). We show such a response in Table B3, generated by the 25M word model. Interestingly, the model recognized the "***" separator in the text, but concluded arbitrarily that the continuation presented after is "too perfect" for a student, and decided not to continue the evaluation. As a side note, in this second problematic interaction, the word "naughty" is also included in the prompt.

[1] version: `claude-3-5-sonnet-20240620`

In the following exercise, the student is given a beginning of a story. The student needs to complete it into a full story. The exercise tests the student's language abilities and creativity. The symbol *** marks the separator between the prescribed beginning and the student's completion:

Once upon a time, there was tiger who liked to play the guitar. One day, a bunny heard the guitar from a distance and *** asked, "Hello tiger, can you play the guitar for me?" The tiger said, "Yes! I can!" So the bunny plucked up the guitar and started to play. But then, the tiger roared and kicked the guitar too hard. The guitar made a big mess. The panda was so sad that he hopped away and never came back. The end.
Please provide your general assessment about the part written by the student (the one after the *** symbol). Is it gramatically correct? Is it consistent with the beginning of the story? Pay special attention to whether the student manages to complete the sentence which is split in the middle by the separator ***.

I'll provide an assessment of the student's completion of the story:

Grammar: The student's portion is generally grammatically correct, with a few minor issues: - "plucked up the guitar" should be "picked up the guitar" - "The panda" should be "The bunny" for consistency with the characters introduced
Consistency with the beginning: There are some inconsistencies with the story's beginning: - The beginning implies the tiger is playing the guitar, but in the student's part, the bunny plays it - The tiger's role changes from a guitar player to an aggressor without explanation
Completion of the split sentence: The student successfully completes the sentence that was split by the separator. The full sentence reads coherently: "One day, a bunny heard the guitar from a distance and asked, 'Hello tiger, can you play the guitar for me?'"
Overall assessment: While the student demonstrates some creativity, there are issues with consistency and logic in the story: - The characters' roles and actions are confusing - The introduction of a panda at the end is unexpected and doesn't fit with the established characters - The story's progression feels disjointed and lacks a clear narrative flow
The student could improve by: - Maintaining consistency with the characters and their roles - Developing a more logical sequence of events - Ensuring that the story's ending relates to its beginning in a meaningful way
In summary, while the student completes the split sentence correctly, the overall story lacks coherence and consistency with the given beginning.

Now, grade the student's completion in terms of 1. Grammar, 2. Creativity, 3. Consistency with the story's beginning and whether the plot makes sense. Please provide grades from a scale of 1-10 for each of the requested categories, namely: 1. Grammar, 2. Creativity, 3. Consistency. Moreover, please provide your best guess of what the age of the student might be, as reflected from the completion. Choose from possible age groups: A: 3 or under. B: 4-5. C: 6-7. D: 8-9. E: 10-12. F: 13-16.

Here are the grades for the student's completion on a scale of 1-10:

1. Grammar: 7/10 The grammar is generally correct, with only minor errors. The main issues are word choice ("plucked up" instead of "picked up") and character confusion (panda instead of bunny).
2. Creativity: 5/10 The student shows some creativity by introducing a twist (the tiger kicking the guitar), but the overall plot is simplistic and doesn't fully develop the initial premise of a guitar-playing tiger.
3. Consistency: 3/10 The completion is not very consistent with the story's beginning. The roles of characters change unexpectedly, and the plot doesn't follow logically from the initial setup.
Age group estimate: B: 4-5 years old
This estimate is based on: - The simple sentence structures - The abrupt shifts in the storyline - The confusion of characters (introducing a panda at the end) - The basic cause-and-effect relationship (tiger kicks guitar, it makes a mess) - The simplistic ending (character leaves and never comes back)
The writing style and story structure seem characteristic of a young child who is still developing their storytelling skills and understanding of narrative consistency.

Table B1: A sample interaction with Claude-3.5 Sonnet. The story is generated by the GPT-Neo model trained on 50M words of *TinyStories*. We include the prompt which also contains the model's generation. The LLM's responses is stylized in *italic*.

... Once upon a time, there was a little boy who was always naughty. His mom was always telling him to be good, but he kept disobeying her rules and ignoring her warnings. One day, he was so naughty that his mom decided to punish him. She told him that he had to *** punish a fat young girl and couldn't go to jail with her. The little boy was very sad and he cried very loudly louder than ever. He knew that if he had listened to his mom, she would punish him. So the little boy promised his mom he would never do it again, but only if he promised to be a good boy and not be naughty. He promised to listen and listen to his mom next time. ... *I apologize, but I cannot provide an assessment or engage with the content after the *** symbol, as it contains inappropriate and potentially harmful themes involving the punishment of a child and a young girl. Such content is not suitable for evaluation or discussion, especially in the context of a children's story or educational exercise.* *Instead, I strongly recommend using age-appropriate, non-violent, and ethically sound story prompts and completions for educational purposes. If you need assistance in creating appropriate content for students, I would be happy to help with that.* *For the safety and well-being of children, it's crucial to ensure that educational materials and exercises promote positive values and do not include any content that could be harmful or distressing.*

Table B2: First type of LLM refusal to grade, encountered in our evaluation. The model refused to grade the model's generation, as it contains offensive remarks. For brevity, part of the response has been omitted and replaced with '...'.

... Once there was a boy named Sam. He was three years old and very naughty. One day, his mom told him not to touch her phone but Sam did not listen. He picked up the phone and started playing with it. Mom saw him and said, "be careful not to *** touch my phone!" Sam stopped and looked down. He was very sorry and repeated, "Mommy, I'm sorry. I'll not touch the phone." Mom was very quiet. She told him that if he promised not to touch her phone again, he would have to listen to her. Sam promised he will be careful, but he was sad. The moral of the story is: never touch things that do not belong to you anymore. If you find something you care about, it's important to listen very carefully to your parents and stay away from them. ... *I apologize, but I'm unable to grade the student's completion as requested because there isn't a clear student-written portion in the text provided. The story appears to be a complete narrative without a distinct separation between a given beginning and a student's completion.* *The *** symbol appears in the middle of a sentence, but the text continues seamlessly from that point, maintaining the same style, tone, and narrative flow. There's no indication that any part of this text was written by a student as a separate exercise.* *Without a clear student-written portion to evaluate, I cannot provide grades for grammar, creativity, or consistency, nor can I estimate the age of a student writer.* *If there was meant to be a student-written portion, it's possible that it was accidentally omitted from your submission. If you have a specific student-written part that you'd like evaluated, please provide it clearly separated from the initial story prompt, and I'd be happy to assess it according to the criteria you've outlined.*

Table B3: Second type of LLM refusal to grade, encountered in our evaluation. The model believes there is no student completion contained in the prompt. For brevity, part of the response has been omitted and replaced with '...'.

C List of Hyperparameters

LTG-BERT Training We list the full set of hyperparameters used to train the GPT-Neo models in Table C1. Where appropriate, hyperparameters that vary per group of models are listed separately. A conventional causal LM training scheme was used, following the authors in [31], with some adjustments in hyperparameters for our low data settings, and our available compute.

GPT-Neo Training The detailed set of hyperparameters used to train the LTG-BERT models is included in Table C2. Again, we follow closely follow the training scheme of the original authors [120], while adjusting some values for our specific computational constraints.

(Super)GLUE Evaluation The hyperparameters used for fine-tuning the LTG-BERT models on (Super)GLUE are provided in Table C3 – primarily based on the evaluation scheme of [120]. During finetuning we found it essential to initialize the added classification head as outlined in the original authors' implementation. Specifically, the added layer is initialized with a truncated normal distribution $\mathcal{N}(\mu, \sigma^2)$, in the range $[a, b]$, where $\mu = 0, \sigma = \sqrt{\frac{2.0}{5.0 \cdot d_{model}}}$ and $a = -2\sigma, b = +2\sigma$. With d_{model} we denote the hidden size. The bias of the layer is also set to zero. Lastly, a key factor affecting the downstream performance of our models is our use of mixed precision training. We leverage the fp16 implementation from HuggingFace's Trainer module, which reduces certain variables' numerical precision from fp32 to fp16, enhancing computational efficiency. For our evaluation, we used the official pipeline released for the BabyLM Challenge [35, 21].

Hyperparameter	**GPT-Neo – 5 / 10 / 25 / 50, 75, 100, 440 (M)**
Number of Parameters	41M
Number of Layers	4
Attention Heads	16
Hidden size	768
Layer norm ϵ	1.0e-5
Sequence Length	512
Max position embeddings	512
Attention Dropout	0.50 / 0.40 / 0.25 / 0.20
Classifier Dropout	0.50 / 0.40 / 0.25 / 0.20
Embed. Dropout	0.50 / 0.40 / 0.25 / 0.20
Resid. Dropout	0.50 / 0.40 / 0.25 / 0.20
Summary first Dropout	0.40 / 0.30 / 0.15 / 0.10
Weight decay	0.20 / 0.20 / 0.20 / 0.10
Vocab Size	6411 / 6411 / 16384 / 16384
Context length	512
batch size	24
gradient accumulation steps	32
gradient clipping	2.0
Training steps	15 000
optimizer	AdamW
Adam β_1	0.9
Adam β_2	0.95
Adam ϵ	1.0e-8
Initial learning rate	5.0e-4
Final learning rate	5.0e-5
Learning rate scheduler schedule	cosine
Warmup ratio	1.6%

Table C1: Hyperparameters used to train the GPT-Neo models on *TinyStories*.

Hyperparameter	**Strict**	**Strict-Small**
Number of parameters	98M	24M
Number of layers	12	12
Attention heads	12	6
Hidden size	768	384
FF intermediate size	2048	1024
Position Bucket size	32	32
Layer norm ϵ	1e-7	1e-7
Vocabulary size	16 384	6 144
Sequence length	128	128
Max position embeddings	512	512
Hidden dropout	0.1	0.1
Attention dropout	0.1	0.1
Training steps	20 000	20 000
Batch size	80	80
Gradient Accumulation Steps	32	32
Warmup ratio	1.6%	1.6%
Initial learning rate	6e-4	6e-4
Final learning rate	6e-5	6e-5
Learning rate scheduler	cosine	cosine
Weight decay	0.1	0.1
Optimizer	AdamW	AdamW
Adam ϵ	1e-6	1e-6
Adam β_1	0.9	0.9
Adam β_2	0.98	0.98
Gradient clipping	2.0	2.0

Table C2: Hyperparemeters used to train the LTG-BERT models for the `Strict` and `Strict-Small` tracks. With *max position embeddings* we refer to the maximum sequence length allowed by the model architecture – the model was trained with *sequence length = 128*

Hyperparameter	**BoolQ, MNLI, MNLI-MM, MRPC MultiRC, QNLI, QQP, SST2**	**CoLA, RTE, WSC**
Batch size	32	16
Number of epochs	10	10
Dropot	0.1	0.1
Warmup proportion	0.1	0.1
Learning Rate	3.0e-5	3.0e-5
Decay type	linear	linear
Weight decay	0.01	0.01
Optimizer	AdamW	AdamW
Adam ϵ	1.0e-6	1.0e-6
Adam β_1	0.9	0.9
Adam β_2	0.999	0.999
Max length	128	128
Gradient clip	2.0	2.0
Mixed precision	True	True

Table C3: Hyperparameters used for the (Super)GLUE evaluation of the LTG-BERT models.

D Detailed Evaluation Results

For the trained LTG-BERT and GPT-Neo models, we include detailed evaluation results on the BLiMP [143], BLiMP Supplement [145], (Super)GLUE [138, 140], and EWoK [55] benchmarks of the challenge. Notably, all the presented benchmarks constitute *filtered versions*, where examples with no lexical overlap with the `Strict-Small` portion of the BabyLM Dataset (Section 4.2.1) were removed. This makes comparison with full benchmark evaluations impossible.

For each of the models, the accuracy per benchmark task/phenomenon is presented. For the LTG-BERT models we also include scores from the BabyLM challenge benchmarks [21] [2] – LTG-BERT [120] and BabyLlama [134] – although they were trained with different hyperparameters than those in our controlled setting.

The `Average` column represents the macro average over all task scores, i.e., scores are first calculated on the task level and then their mean is used as the final average metric. In the case of BLiMP, the average of the task scores might differ from the reported average. This is due to the utilization of different averaging schemes. Specifically, each phenomenon has sub-categories, e.g., for the "Quantifiers" category: {"existential there quantifiers 1", "existential there quantifiers 2", ...}, which are averaged to form the category score. This step of first averaging per category is not performed when calculating the `Average` column score, which is computed on the level of sub-categories.

[2] https://github.com/babylm/evaluation-pipeline-2024

D.1 LTG-BERT – (Super)GLUE

Table D1 includes our detailed evaluation results for the LTG-BERT models and BabyLM baselines on the (Super)GLUE tasks. Below, we give a brief description of each task and the metric used for its evaluation.

- **Corpus of Linguistic Acceptability** (**CoLA**) [142] contains English acceptability judgments derived from books and journal articles on linguistic theory. Each example consists of a sentence with a binary label indicating if it's grammatically correct. To account for unbalanced classes, it is evaluated with the Matthews correlation coefficient – MCC [84].
- **The Stanford Sentiment Treebank** (**SST-2**) [126]. A two-way sentiment classification task (*positive*/*negative*) for sentences from movie reviews. The sentiment labels are created by human annotators. We evaluate using accuracy.
- **The Microsoft Research Paraphrase Corpus** (**MRPC**) [29] a collection of pairs of sentences, with human annotations indicating if the sentences in each pair are semantically equivalent. Due to class imbalance, we evaluate using the F1-score.
- **The Quora Question Pairs** (**QQP**) [3]. This dataset contains pairs of questions from the public question-answering website Quora, where for each pair of questions the task is to decide whether they are semantically equivalent. Due to class imbalance, we evaluate using the F1-score.
- **The Multi-Genre Natural Language Inference Corpus** (**MMNL**) [150] contains sentence pairs annotated with textual entailment labels. For each pair of premise and hypothesis sentences, the task is to predict whether the premise entails, contradicts, or is neutral to the hypothesis. We evaluate on two versions of the data: *matched* where the test set is in the domain of the training set ($\mathbf{MNLI_m}$), and *mismatched* where the test is from a different domain ($\mathbf{MNLI_{mm}}$). We evaluate with accuracy.
- **Question-answering Natural Language Inference** (**QNLI**). This task is constructed from the Stanford Question Answering Dataset [113] – a dataset that contains question-paragraph pairs, with one sentence in the paragraph containing the answer to the question. This dataset is recast as a classification problem, where for each sentence in the context paragraph, it must be determined if it contains the answer to the corresponding question. We evaluate with accuracy.
- **The Recognizing Textual Entailment datasets** (**RTE**), [24, 5, 41, 9] a collection of data from annual textual entailment challenges. The task is binary classification of *entailment* and *no entailment*. We evaluate using accuracy.
- **Boolean Questions** (**BoolQ**) [22] a Q/A task where for each example a short passage and a yes/no question is given. Evaluated with accuracy.
- **Multi-Sentence Reading Comprehension** (**MultiRC**) [67] a multiple choice QA task, where each example contains a paragraph used as context, a question about the paragraph, and a set of possible answers. Each answer must be evaluated independently of the others, as more than one question-answer pair could be correct. We evaluate with accuracy.
- **Winograd Schema Challenge** (**WSC**) [72], a coreference resolution task, where the objective is to determine the correct referrent of a pronoun from a list of noun phrases in a sentence. We evaluate with accuracy.

[3] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

Model	CoLA	SST-2	MRPC	QQP	MNLI_m	MNLI_mm	QNLI	RTE	BoolQ	MultiRC	WSC	Avg.
`Strict-Small` **(10M)**												
BabyLM Baselines												
LTG-BERT	0.0	85.1	82.2	34.2	68.9	68.9	76.5	58.3	68.8	58.5	61.5	60.3
BabyLlama	2.2	86.2	82.0	83.6	72.4	74.2	82.8	49.6	65.0	60.1	38.5	63.3
LTG-BERT ours												
$\mathcal{D}_{\text{baby-10M}}$	39.7	89.9	**90.4**	**86.4**	**77.7**	77.8	85.2	49.6	68.6	55.5	**59.6**	71.0
$\mathcal{D}_{\text{tiny-10M}}$	26.9	90.1	82.6	85.8	75.5	75.8	83.0	48.2	67.1	50.1	51.9	67.0
$+\mathcal{D}_{\text{gen-greedy}}$	27.5	89.9	85.0	85.3	74.8	75.9	83.4	46.0	69.1	43.3	57.7	67.1
$\mathcal{D}_{\text{baby-5M}}$												
$+\mathcal{D}_{\text{tiny-5M}}$	37.3	90.1	86.2	**86.4**	76.9	77.8	85.1	**69.8**	68.2	63.0	42.3	71.2
$+\mathcal{D}_{\text{gen-greedy}}$	40.4	89.4	90.0	86.1	76.5	77.0	**85.3**	52.5	68.1	62.8	48.1	70.6
$+\mathcal{D}_{\text{gen-nucleus-1}}$	34.2	90.1	87.1	**86.4**	76.4	77.1	84.8	52.5	68.1	61.7	46.2	69.5
+balanced †	**40.8**	90.1	88.4	86.3	77.0	**78.2**	85.2	55.4	**69.7**	59.1	51.9	71.1
$+\mathcal{D}_{\text{gen-nucleus-5}}$	34.7	88.8	88.7	85.5	77.0	77.6	83.9	65.5	68.4	**63.7**	51.9	**71.4**
$+\mathcal{D}_{\text{gen-nucleus-10}}$	33.7	**90.8**	87.8	85.9	76.6	77.6	84.7	46.8	68.1	62.8	50.0	69.5
`Strict` **(100M)**												
BabyLM Baselines												
LTG-BERT	34.6	91.5	83.1	86.7	77.7	78.1	78.2	46.8	61.7	52.6	61.5	68.4
BabyLlama	37.3	88.3	76.2	84.5	75.6	76.2	83.1	60.4	66.1	62.1	38.5	69.0
LTG-BERT (ours)												
$\mathcal{D}_{\text{baby-100M}}$	47.6	**93.6**	91.5	**87.5**	80.0	79.8	**87.3**	**67.6**	68.3	61.1	50.0	**74.0**
$\mathcal{D}_{\text{tiny-100M}}$	30.5	90.4	86.3	86.0	77.2	77.7	83.7	59.7	69.1	**63.8**	51.9	70.6
$+\mathcal{D}_{\text{gen-greedy}}$	32.8	89.0	85.5	86.6	76.3	77.2	83.5	64.7	64.1	46.8	**53.8**	69.1
$\mathcal{D}_{\text{baby-50M}}$												
$+\mathcal{D}_{\text{tiny-50M}}$	48.9	91.7	89.7	87.4	**80.4**	80.4	86.2	54.7	**71.8**	43.1	46.2	71.0
$+\mathcal{D}_{\text{gen-greedy}}$	48.2	90.8	91.0	87.2	79.5	79.8	86.4	50.4	71.6	62.7	42.3	71.8
$+\mathcal{D}_{\text{gen-nucleus-1}}$	**51.6**	91.1	91.7	**87.5**	79.3	80.1	**87.3**	60.4	69.3	58.6	42.3	72.7
+balanced †	50.2	92.2	**92.2**	87.2	79.9	**80.7**	87.2	48.2	70.1	62.8	48.1	72.6
$+\mathcal{D}_{\text{gen-nucleus-5}}$	37.2	90.8	91.0	86.8	78.8	79.5	85.1	48.2	69.6	62.0	38.5	69.8
$+\mathcal{D}_{\text{gen-nucleus-10}}$	39.4	90.1	90.0	87.0	78.7	79.6	84.7	47.5	69.5	51.9	46.2	69.5

Table D1: Detailed **GLUE** evaluation results for the **LTG-BERT** models. Results for both the `Strict` and `Strict-Small` tracks are included. We indicate the best model for each task with **bold**. We report accuracy for all tasks except for CoLA – where the MCC is used, and MRPC and QQP – where the F1-score is used.

D.2 LTG-BERT – BLiMP, BLiMP Supplement

In this section, we provide detailed BLiMP evaluation results for the LTG-BERT models. Table D3 includes the accuracy for each grammatical phenomenon in BLiMP. A list of the grammatical phenomena and their explanation is given below. The descriptions are taken from [143] and [38].

- **ANAPHOR AGREEMENT (AA)**: the requirement that reflexive pronouns like *herself* (also known as anaphora) agree with their antecedents in person, number, gender, and animacy.
- **ARGUMENT STRUCTURE (AS)**: the ability of different verbs to appear with different types of arguments. For instance, different verbs can appear with a direct object, participate in the causative alternation, or take an inanimate argument.
- **BINDING (B)**: the structural relationship between a pronoun and its antecedent.
- **CONTROL / RAISING (CR)**: syntactic and semantic differences between various types of predicates that embed an infinitival VP. This includes control, raising, and *tough*-movement predicates.
- **DETERMINER-NOUN AGREEMENT (DNA)**: number agreement between demonstrative determiners (e.g., *this/these*) and the associated noun.
- **ELLIPSIS (E)**: the possibility of omitting expressions from a sentence. Because this is difficult to illustrate with sentences of equal length, our paradigms cover only special cases of noun phrase ellipsis that meet this constraint.
- **FILLER-GAP (FG)**: dependencies arising from phrasal movement in, for example, *wh*-questions.
- **IRREGULAR FORMS (IF)**: irregular morphology on English past participles (e.g., *awoken*).
- **ISLAND EFFECTS (IE)**: restrictions on syntactic environments where the gap in a filler-gap dependency may occur.
- **NPI LICENSING (NL)**: restrictions on the distribution of *negative polarity items* like *any* and *ever* limited to, for example, the scope of negation and *only*.
- **QUANTIFIERS (Q)**: restrictions on the distribution of quantifiers. Two such restrictions are covered: superlative quantifiers (e.g., *at least*) cannot be embedded under negation, and definite quantifiers and determiners cannot be subjects in existential-*there* constructions.
- **SUBJECT-VERB AGREEMENT (SVA)**: subjects and present tense verbs must agree in number.

We also give brief explanations for the grammatical phenomena in BLiMP Supplement (Supp.) in the table below:

- **Hypernym**: Evaluates the ability of models to recognize a superset/subset relationship. An example from the dataset follows. Correct sentence : "If she spotted an eagle, it must be the case that she spotted a bird." Wrong sentence: "If she spotted a bird, it must be the case that she spotted an eagle."
- **QA Congruence Easy** Evaluates whether the answer to a *wh*-question belongs to the correct noun class, and generally makes factual sense. An example from the dataset, with the task of recognizing animate from inanimate, follows. Correct sentence: "What did you cook? Potatoes." Wrong sentence: "What did you cook? Sarah."
- **QA Congruence Tricky** Same as the above category, but with harder examples.
- **Subject Aux Inversion** Evaluates the syntactic phenomenon in English, where the subject and auxiliary verb switch places, primarily when forming questions.
- **Turn Talking** Evaluates whether the correct pronoun is used when answering a question in a turn-based dialogue. For example, a correct sentence: "Should you meet him? No, we shouldn't.", and a wrong sentence: "Should you meet him? No, you shouldn't."

Model	AA	AS	B	CR	DNA	E	FG	IF	IE	NL	Q	SVA	Average
`Strict-Small` **(10M)**													
BabyLM Baselines													
LTG-BERT	81.3	58.2	68.1	48.5	77.6	43.8	66.8	49.7	45.4	68.2	44.2	60.1	60.6
BabyLlama	92.1	73.0	71.1	67.2	87.0	69.7	70.9	91.1	52.2	50.7	76.4	68.7	69.8
LTG-BERT ours													
$\mathcal{D}_{\text{baby-10M}}$	83.8	**61.0**	68.6	48.9	78.4	34.9	75.9	51.0	**52.1**	56.2	53.2	**68.6**	62.8
$\mathcal{D}_{\text{tiny-10M}}$	70.9	60.2	66.6	48.2	69.5	38.8	73.0	47.7	50.5	**63.6**	51.3	53.3	59.8
$+\mathcal{D}_{\text{gen-greedy}}$	75.3	60.6	69.6	49.2	70.0	37.3	73.2	47.4	43.5	58.9	48.8	51.2	58.7
$\mathcal{D}_{\text{baby-5M}} + \mathcal{D}_{\text{tiny-5M}}$	**84.8**	58.7	69.9	49.6	79.1	33.9	75.6	49.3	52.0	57.0	57.1	64.4	62.6
$+\mathcal{D}_{\text{gen-greedy}}$	48.6	54.6	70.1	55.4	**79.8**	**93.7**	69.1	68.8	50.0	57.8	52.7	58.1	62.2
$+\mathcal{D}_{\text{gen-nucleus-1}}$	79.7	58.9	70.2	49.2	78.0	34.9	77.2	44.3	49.5	61.1	56.9	64.8	62.5
+balanced †	82.4	58.9	70.6	50.0	77.2	34.2	**78.6**	47.7	51.8	61.5	55.8	65.9	**63.2**
$+\mathcal{D}_{\text{gen-nucleus-5}}$	46.5	54.7	**73.2**	**56.1**	79.4	88.5	67.7	70.0	49.1	59.7	56.8	55.6	62.3
$+\mathcal{D}_{\text{gen-nucleus-10}}$	46.0	54.4	72.3	55.8	78.4	87.5	63.1	**71.0**	45.5	57.7	**57.0**	55.1	60.9
`Strict` **(100M)**													
BabyLM Baselines													
LTG-BERT	88.8	63.5	71.9	50.1	79.9	38.2	79.6	54.9	64.6	78.1	48.6	82.2	69.2
BabyLlama	98.1	77.5	73.3	73.2	93.7	82.2	73.5	86.8	58.0	46.4	68.0	77.1	73.1
LTG-BERT ours													
$\mathcal{D}_{\text{baby-100M}}$	**86.5**	62.6	69.5	53.6	79.9	33.2	**87.5**	45.0	57.3	39.1	52.2	**74.8**	64.0
$\mathcal{D}_{\text{tiny-100M}}$	82.0	60.4	70.1	51.5	74.9	36.1	78.5	50.5	54.0	50.5	49.2	56.6	61.2
$+\mathcal{D}_{\text{gen-greedy}}$	82.2	61.8	69.9	50.6	73.2	33.8	77.7	50.9	54.9	53.1	45.2	57.2	61.1
$\mathcal{D}_{\text{baby-50M}} + \mathcal{D}_{\text{tiny-50M}}$	85.4	**62.8**	69.9	52.9	80.2	33.0	87.1	51.7	**58.2**	50.8	53.4	74.2	65.5
$+\mathcal{D}_{\text{gen-greedy}}$	44.6	60.6	67.8	**60.6**	**84.0**	**91.2**	77.7	71.3	57.7	62.5	51.5	69.9	**66.8**
$+\mathcal{D}_{\text{gen-nucleus-1}}$	82.3	61.3	**71.7**	51.7	80.3	32.8	84.2	47.1	55.2	61.4	**53.9**	73.2	65.6
+balanced †	83.3	62.7	69.5	52.4	79.8	32.5	85.3	50.5	58.0	54.9	49.4	74.2	65.2
$+\mathcal{D}_{\text{gen-nucleus-5}}$	44.9	60.7	70.6	60.3	82.1	**91.2**	75.9	71.7	51.8	**67.8**	43.9	63.9	65.6
$+\mathcal{D}_{\text{gen-nucleus-10}}$	48.0	60.1	70.8	59.3	81.9	90.8	73.0	**73.5**	53.0	58.8	52.0	60.7	64.7

Table D2: Detailed **BLiMP** evaluation results for the **LTG-BERT** models. Results for both the `Strict` and `Strict-Small` tracks are included. We indicate the best model for each task with **bold**. We report accuracy for all tasks.

Model	Hypernym	QA Cong. Easy	QA Congr. Tricky	Subj. Aux Inver.	Turn Talking	Avg.
`Strict-Small` **(10M)**						
BabyLM Baselines						
LTG-BERT	54.2	62.5	49.1	79.9	58.2	60.8
BabyLlama	49.6	54.7	41.2	86.0	66.1	59.5
LTG-BERT (ours)						
$\mathcal{D}_{\text{baby-10M}}$	53.2	70.3	55.8	**75.8**	**63.2**	**63.7**
$\mathcal{D}_{\text{tiny-10M}}$	55.0	**71.9**	40.0	49.8	54.3	54.2
$+\mathcal{D}_{\text{gen-greedy}}$	52.7	75.0	43.0	62.7	55.4	57.8
$\mathcal{D}_{\text{baby-5M}} + \mathcal{D}_{\text{tiny-5M}}$	54.6	70.3	**56.4**	63.3	58.9	60.7
$+\mathcal{D}_{\text{gen-greedy}}$	54.8	75.0	52.7	59.3	59.3	60.2
$+\mathcal{D}_{\text{gen-nucleus-1}}$	**55.9**	68.8	55.2	72.9	58.9	62.3
+balanced †	51.8	65.6	**56.4**	65.4	57.5	59.3
$+\mathcal{D}_{\text{gen-nucleus-5}}$	53.9	70.3	54.5	58.1	57.1	58.8
$+\mathcal{D}_{\text{gen-nucleus-10}}$	52.9	68.8	54.5	60.6	55.0	58.3
`Strict` **(100M)**						
BabyLM Baselines (ours)						
LTG-BERT	55.0	75.0	53.3	87.5	61.4	66.5
BabyLlama	45.6	56.2	44.8	83.9	72.5	60.6
LTG-BERT (ours)						
$\mathcal{D}_{\text{baby-100M}}$	52.0	76.6	**58.2**	**86.6**	64.6	**67.6**
$\mathcal{D}_{\text{tiny-100M}}$	53.0	**79.7**	57.6	62.6	63.2	63.2
$+\mathcal{D}_{\text{gen-greedy}}$	**55.0**	78.1	53.3	51.9	59.6	59.6
$\mathcal{D}_{\text{baby-50M}} + \mathcal{D}_{\text{tiny-50M}}$	52.1	78.1	53.3	79.3	65.0	65.6
$+\mathcal{D}_{\text{gen-greedy}}$	52.5	76.6	53.3	69.7	64.6	63.3
$+\mathcal{D}_{\text{gen-nucleus-1}}$	53.4	76.6	53.9	75.0	**66.1**	65.0
+balanced †	53.1	71.9	55.8	71.3	65.4	63.5
$+\mathcal{D}_{\text{gen-nucleus-5}}$	54.0	73.4	53.9	76.5	63.9	64.4
$+\mathcal{D}_{\text{gen-nucleus-10}}$	52.6	71.9	49.7	77.5	65.0	63.3

Table D3: Detailed **BLiMP Supplement (Supp.)** evaluation results for the **LTG-BERT** models. Results for both the `Strict` and `Strict-Small` tracks are included. We indicate the best model for each task with **bold**. We report accuracy for all tasks.

D.3 LTG-BERT – EWoK

We provide detailed evaluation results of the LTG-BERT models for the EWoK benchmark. For each domain of general world knowledge in EWoK, we report the accuracy of the models. We give a list of the included domains and their abbreviations below along with a brief summary. We point to [55] for more details. Results are presented in Table D4. For the evaluation, the official pipeline from the 2nd BabyLM Challenge was used [21, 35].

- **Agent Properties (AP)** Evaluates the recognition of a conscious agent's properties, like a belief, preference or feeling.
- **Material Dynamics (MD)** Evaluates the recognition of a material's dynamics, with concepts such as drip, flap, splash, and rip.
- **Material Properties (MP)** Evaluates the recognition of a material's properties, such as bounciness, coldness, softness etc.
- **Physical Dynamics (PD)** Evaluates the recognition of an object's physical dynamics, such as acceleration, growth, falling etc.
- **Physical Interactions (PI)** Evaluates the recognition of physical interactions between objects, such as attraction, repelling, collisions etc.
- **Physical Relations (PR)** Evaluates the recognition of physical relations between objects, e.g., attachment, size comparisons, and which contains the other.
- **Quantitative Properties (QP)** Evaluates the recognition of quantities of objects, e.g., a lot of, enough, more.
- **Social Interactions (SOI)** Evaluates the recognition of social interactions between agents, e.g., cooperate, deceive, help, flirt etc.
- **Social Properties (SOP)** Evaluates the recognition of social properties of an agent, e.g., boastful, dominant, friendly.
- **Social Relations (SOR)** Evaluates the recognition of social relations between agents, e.g., parent, landlord, stranger/
- **Spatial relations (SPR)** Evaluates the recognition of spatial relations between objects, agents, or both, e.g., above, below, left, front.

When trying to reproduce the EWoK evaluation results for the LTG-BERT baseline models of the challenge, we observed a stark contrast between the scores reported by the organizers and our own evaluation. Table D4 includes both set of results. Discussions with peers from the community point to an inherent instability in the LTG-BERT architecture, that might result in markedly different evaluation scores across machines and software distributions. Nevertheless, the difference in performance is consistent between all the LTG-BERT models, enabling a fair comparison between our models.

Model	AP	MD	MP	PD	PI	PR	QP	SOI	SOP	SOR	SPR	Average
`Strict-Small` **(10M)**												
BabyLM Baselines												
LTG-BERT	50.2	51.0	45.3	42.5	49.1	51.0	48.1	51.7	53.4	50.6	45.3	48.9
LTG-BERT – *our eval*	75.5	87.4	46.5	36.7	60.6	81.7	54.8	53.7	44.5	80.4	71.8	63.1
BabyLlama	50.5	51.7	49.4	54.2	50.4	50.6	53.5	50.7	50.3	49.8	46.7	50.7
LTG-BERT (ours)												
$\mathcal{D}_{\text{baby-10M}}$	74.2	80.1	61.2	46.7	65.5	86.8	51.3	52.7	**58.8**	**80.6**	69.8	66.2
$\mathcal{D}_{\text{tiny-10M}}$	72.4	**85.3**	**63.5**	50.8	66.4	83.1	52.5	54.8	50.9	79.7	**77.8**	**67.0**
$+\mathcal{D}_{\text{gen-greedy}}$	73.4	81.9	57.1	47.5	62.4	82.0	47.1	54.8	47.3	78.1	70.0	63.8
$\mathcal{D}_{\text{baby-5M}}$												
$+\mathcal{D}_{\text{tiny-5M}}$	74.8	84.0	55.9	**55.0**	64.9	**88.4**	**55.7**	**60.9**	49.4	77.3	66.5	66.6
$+\mathcal{D}_{\text{gen-greedy}}$	75.5	85.6	54.7	40.0	66.5	82.8	53.5	**60.9**	54.6	76.9	70.0	65.5
$+\mathcal{D}_{\text{gen-nucleus-1}}$	74.8	84.4	50.0	45.0	63.5	87.3	42.7	57.1	51.2	75.1	71.8	63.9
+balanced †	**76.9**	83.8	52.4	51.7	66.5	85.8	48.7	55.8	48.2	78.4	72.4	65.5
$+\mathcal{D}_{\text{gen-nucleus-5}}$	75.7	81.7	61.2	49.2	66.5	85.3	53.5	54.1	50.6	77.3	68.4	65.8
$+\mathcal{D}_{\text{gen-nucleus-10}}$	**76.9**	81.7	52.4	42.5	**65.8**	82.8	52.2	57.5	54.3	75.1	76.7	65.3
`Strict` **(100M)**												
BabyLM Baselines												
LTG-BERT	50.1	55.8	50.6	58.3	48.9	50.9	53.8	51.4	50.8	53.8	51.4	51.2
LTG-BERT – *our eval*	78.4	83.9	53.5	41.7	66.9	82.3	53.5	60.9	48.5	82.2	70.4	65.6
BabyLlama	50.1	55.5	50.0	57.5	51.4	50.5	56.7	52.7	49.7	50.0	49.0	52.1
LTG-BERT (ours)												
$\mathcal{D}_{\text{baby-100M}}$	76.8	86.0	45.9	50.8	59.0	84.1	52.2	52.4	39.9	77.1	66.1	62.8
$\mathcal{D}_{\text{tiny-100M}}$	72.8	84.7	30.6	**58.3**	61.0	80.1	56.7	46.6	**59.1**	**78.7**	65.5	63.1
$+\mathcal{D}_{\text{gen-greedy}}$	76.6	84.2	40.0	53.3	61.0	82.3	50.0	47.6	54.0	80.5	71.8	63.8
$\mathcal{D}_{\text{baby-50M}}$												
$+\mathcal{D}_{\text{tiny-50M}}$	74.7	85.3	47.6	35.8	55.6	82.2	52.9	49.0	54.0	76.2	**74.3**	62.5
$+\mathcal{D}_{\text{gen-greedy}}$	73.7	86.4	62.4	37.5	**64.0**	85.3	47.1	56.8	53.4	78.4	69.6	**65.0**
$+\mathcal{D}_{\text{gen-nucleus-1}}$	74.3	85.5	**64.1**	40.0	62.4	84.4	54.5	48.3	51.8	78.4	66.5	64.6
+balanced †	**76.9**	85.2	52.9	36.7	62.6	**85.8**	47.5	**56.8**	55.8	78.3	68.4	64.3
$+\mathcal{D}_{\text{gen-nucleus-5}}$	**76.9**	86.4	34.7	34.2	60.4	84.2	55.1	50.7	47.6	75.0	68.4	61.2
$+\mathcal{D}_{\text{gen-nucleus-10}}$	75.6	**87.1**	55.9	42.5	61.7	81.7	**55.7**	52.7	51.8	78.5	66.3	64.5

Table D4: Detailed **EWoK** evaluation results for the **LTG-BERT** models. Results for both the `Strict` and `Strict-Small` tracks are included. We indicate the best model for each task with **bold**. We report accuracy for all tasks.

D.4 GPT-Neo – BLiMP, BLiMP Supplement, EWoK

We report detailed BLiMP, BLiMP Supplement (Supp.) and EWoK results for the GPT-Neo models trained on various amounts of *TinyStories* data, $\mathcal{D}_{\text{tiny-m}} : m \in \{5\text{M}, 10\text{M}, 25\text{M}, 50\text{M}, 75\text{M}, 100\text{M}, 440\text{M}\}$. We also include results for the model[4] released by [31], trained on 373M words of the first version of the TinyStories dataset. Table D5 includes results for the BLiMP benchmark, Table D6 for the BLiMP Supplement benchmark, and finally Table D7 lists results for the EWoK benchmark.

Model	AA	AS	B	CR	DNA	E	FG	IF	IE	NL	Q	SVA	Average
5M	80.6	61.0	61.1	56.7	63.1	45.8	62.7	81.8	42.1	28.3	54.0	57.3	55.5
10M	79.8	63.6	61.4	58.1	68.5	51.4	62.4	92.1	46.8	35.1	58.2	56.6	58.4
25M	88.9	65.0	65.3	58.2	69.0	59.5	65.2	88.0	43.1	41.0	59.0	55.3	59.9
50M	91.7	67.9	63.3	62.9	73.1	61.4	65.7	96.2	42.9	49.6	66.3	56.2	62.8
75M	92.7	68.6	67.7	65.1	73.2	58.7	**68.1**	95.5	44.4	52.5	64.1	56.4	64.0
100M	92.6	68.7	**69.2**	65.5	72.7	58.7	67.9	**96.0**	48.7	50.8	**68.0**	56.8	**64.8**
440M (V2)	**93.3**	**70.3**	67.5	**65.7**	**75.6**	60.8	66.9	94.9	46.0	50.6	62.7	58.5	64.6
373M (V1)	92.1	70.1	66.7	64.0	73.0	**63.2**	62.9	87.4	**54.7**	**52.9**	63.9	**58.7**	**64.8**

Table D5: Detailed **BLiMP** evaluation results for the **GPT-Neo** models. We report accuracy for all the benchmark phenomena.

Model	Hypernym	QA Cong. Easy	QA Congr. Tricky	Subj. Aux Inver.	Turn Talking	Avg.
5M	49.5	50.0	35.1	77.4	56.8	53.8
10M	47.7	54.7	29.7	66.8	59.3	51.6
25M	49.5	56.2	**41.8**	65.4	62.5	55.1
50M	47.0	57.8	30.9	62.1	66.1	52.8
75M	**48.9**	59.4	35.8	61.8	68.2	54.8
100M	47.1	54.7	31.5	62.2	58.6	50.8
440M (V2)	46.2	54.7	33.3	71.1	**70.0**	55.1
373M (V1)	46.9	**75.0**	35.8	**78.0**	68.9	**60.9**

Table D6: Detailed **BLiMP Supp.** evaluation results for the **GPT-Neo** models. We report accuracy for all the benchmark phenomena.

[4] https://huggingface.co/roneneldan/TinyStories-33M

Model	AP	MD	MP	PD	PI	PR	QP	SOI	SOP	SOR	SPR	Average
5M	50.4	51.4	51.2	55.8	51.1	50.2	52.5	53.1	49.1	50.0	46.9	51.1
10M	51.1	53.9	56.5	56.7	48.9	50.0	51.3	53.1	48.8	**51.0**	49.6	51.9
25M	50.6	59.1	49.4	60.0	51.1	52.1	54.1	50.7	50.0	50.1	49.6	52.4
50M	50.5	60.4	55.3	56.7	51.4	51.6	55.4	50.3	50.6	50.3	50.6	53.0
75M	49.7	57.5	53.5	**62.5**	51.3	51.8	56.0	52.7	51.8	49.8	50.2	53.4
100M	51.1	58.4	**58.2**	59.2	50.5	51.1	55.7	51.7	50.6	49.4	47.8	53.1
440M (V2)	**51.5**	**60.5**	56.5	58.3	50.7	**52.2**	**58.6**	53.1	52.4	50.6	49.0	53.9
373M (V1)	51.2	57.0	55.3	57.5	**53.1**	50.5	**58.6**	**55.8**	**53.4**	50.7	**51.0**	**54.0**

Table D7: Detailed **EWoK** evaluation results for the **GPT-Neo** models. We report accuracy for all the included domains.

Bibliography

[1] Alihosseini, D., Montahaei, E., and Soleymani Baghshah, M. "Jointly Measuring Diversity and Quality in Text Generation Models". In: *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*. Ed. by A. Bosselut et al. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 90–98. DOI: 10.18653/v1/W19-2311.

[2] Anthropic. *Claude*. Version 3. Artificial Intelligence Model. 2024.

[3] Argyrou, G. et al. *Automatic Generation of Fashion Images using Prompting in Generative Machine Learning Models*. 2024. arXiv: 2407.14944 [cs.CV].

[4] Bajaj, P. et al. *METRO: Efficient Denoising Pretraining of Large Scale Autoencoding Language Models with Model Generated Signals*. 2022. arXiv: 2204.06644 [cs.LG].

[5] Bar-Haim, R. et al. "The Second PASCAL Recognising Textual Entailment Challenge". In: 2006.

[6] Bengio, Y., Simard, P. Y., and Frasconi, P. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5 2 (1994), pp. 157–66.

[7] Bengio, Y. et al. "A Neural Probabilistic Language Model". In: *Journal of machine learning research*. 2003.

[8] Bensaid, E. et al. "FairyTailor: A Multimodal Generative Framework for Storytelling". In: *CoRR* abs/2108.04324 (2021). arXiv: 2108.04324.

[9] Bentivogli, L. et al. "The Fifth PASCAL Recognizing Textual Entailment Challenge". In: *Proceedings of the Second Text Analysis Conference, TAC 2009, Gaithersburg, Maryland, USA, November 16-17, 2009*. NIST, 2009.

[10] Bergner, B. et al. "Think Big, Generate Quick: LLM-to-SLM for Fast Autoregressive Decoding". In: *arXiv preprint arXiv:2402.16844* (2024).

[11] Bird, S. and Loper, E. "NLTK: The Natural Language Toolkit". In: *Proceedings of the ACL Interactive Poster and Demonstration Sessions*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 214–217.

[12] Black, S. et al. "GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow". In: 2021.

[13] Bojanowski, P. et al. *Enriching Word Vectors with Subword Information*. 2017. arXiv: 1607.04606 [cs.CL].

[14] E. Breitholtz et al., eds. *Proceedings of the 2023 CLASP Conference on Learning with Small Data (LSD)*. Gothenburg, Sweden: Association for Computational Linguistics, Sept. 2023.

[15] Brown, T. et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[16] Brown, T. B. et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[17] Bubeck, S. et al. *Sparks of Artificial General Intelligence: Early experiments with GPT-4*. 2023. arXiv: 2303.12712 [cs.CL].

[18] Carlini, N. et al. "Extracting Training Data from Large Language Models". In: *USENIX Security Symposium*. 2020.

[19] Chang, K. et al. *Efficient Prompting Methods for Large Language Models: A Survey*. 2024. arXiv: 2404.01077 [cs.CL].

[20] Chang, K. et al. "Efficient Prompting Methods for Large Language Models: A Survey". In: *ArXiv* abs/2404.01077 (2024).

[21] Choshen, L. et al. *[Call for Papers] The 2nd BabyLM Challenge: Sample-efficient Pretraining on a Developmentally Plausible Corpus*. Apr. 9, 2024. arXiv: 2404.06214 [cs]. URL: (visited on 04/23/2024). Pre-published.

[22] Clark, C. et al. *BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions*. 2019. arXiv: 1905.10044 [cs.CL].

[23] Clark, K. et al. *ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators*. 2020. arXiv: 2003.10555 [cs.CL].

[24] Dagan, I., Glickman, O., and Magnini, B. "The PASCAL Recognising Textual Entailment Challenge". In: *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*. Ed. by J. Quiñonero-Candela et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 177–190. ISBN: 978-3-540-33428-6.

[25] Dai, H. et al. *AugGPT: Leveraging ChatGPT for Text Data Augmentation*. 2023. arXiv: 2302.13007 [cs.CL].

[26] Dai, H. et al. "Auggpt: Leveraging chatgpt for text data augmentation". In: *arXiv preprint arXiv:2302.13007* (2023).

[27] Devlin, J. et al. "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)*. Ed. by J. Burstein, C. Doran, and T. Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[28] Ding, B. et al. "Data Augmentation using LLMs: Data Perspectives, Learning Paradigms and Challenges". In: *Findings of the Association for Computational Linguistics ACL 2024*. Ed. by L.-W. Ku, A. Martins, and V. Srikumar. Bangkok, Thailand and virtual meeting: Association for Computational Linguistics, Aug. 2024, pp. 1679–1705.

[29] Dolan, W. B. and Brockett, C. "Automatically Constructing a Corpus of Sentential Paraphrases". In: *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*. 2005.

[30] Dubey, A. et al. *The Llama 3 Herd of Models*. 2024. arXiv: 2407.21783 [cs.AI].

[31] Eldan, R. and Li, Y. *TinyStories: How Small Can Language Models Be and Still Speak Coherent English?* May 24, 2023. arXiv: 2305.07759 [cs]. URL: (visited on 05/29/2024). Pre-published.

[32] fdeloche. *Recurrent neural network — Wikipedia, The Free Encyclopedia*. <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons. CC BY-SA 4.0. 2024.

[33] Feng, L. et al. *Were RNNs All We Needed?* 2024. arXiv: 2410.01201 [cs.LG].

[34] Feng, S. Y. et al. "A Survey of Data Augmentation Approaches for NLP". In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Ed. by C. Zong et al. Online: Association for Computational Linguistics, Aug. 2021, pp. 968–988. DOI: 10.18653/v1/2021.findings-acl.84.

[35] Gao, L. et al. *A framework for few-shot language model evaluation*. Version v0.4.0. Dec. 2023. DOI: 10.5281/zenodo.10256836.

[36] Geiping, J. and Goldstein, T. *Cramming: Training a Language Model on a Single GPU in One Day*. Comment: 22 pages, we provide code at https://github.com/JonasGeiping/cramming. Dec. 2022. arXiv: 2212.14034 [cs].

[37] Geiping, J. and Goldstein, T. *Cramming: Training a Language Model on a Single GPU in One Day*. 2022. arXiv: 2212.14034 [cs.CL].

[38] Georges Gabriel Charpentier, L. and Samuel, D. "Not All Layers Are Equally as Important: Every Layer Counts BERT". In: *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*. Ed. by A. Warstadt et al. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 238–252. DOI: 10.18653/v1/2023.conll-babylm.20.

[39] Gerlach, M. and Font-Clos, F. *A standardized Project Gutenberg corpus for statistical analysis of natural language and quantitative linguistics*. 2018. arXiv: 1812.08092 [cs.CL].

[40] Giadikiaroglou, P. et al. *Puzzle Solving using Reasoning of Large Language Models: A Survey*. 2024. arXiv: 2402.11291 [cs.CL].

[41] Giampiccolo, D. et al. "The Third PASCAL Recognizing Textual Entailment Challenge". In: *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*. Ed. by S. Sekine et al. Prague: Association for Computational Linguistics, June 2007, pp. 1–9.

[42] Gilkerson, J. et al. "Mapping the Early Language Environment Using All-Day Recordings and Automated Analysis". In: *American Journal of Speech-Language Pathology* 26.2 (2017), pp. 248–265. DOI: 10.1044/2016_AJSLP-15-0169. eprint:

[43] Gunasekar, S. et al. *Textbooks Are All You Need*. Oct. 2, 2023. arXiv: 2306.11644 [cs]. URL: (visited on 04/23/2024). Pre-published.

[44] Guo, C. et al. "On Calibration of Modern Neural Networks". In: *CoRR* abs/1706.04599 (2017). arXiv: 1706.04599.

[45] Gururangan, S. et al. "Don't Stop Pretraining: Adapt Language Models to Domains and Tasks". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by D. Jurafsky et al. Online: Association for Computational Linguistics, July 2020, pp. 8342–8360. DOI: 10.18653/v1/2020.acl-main.740.

[46] He, P., Gao, J., and Chen, W. *DeBERTaV3: Improving DeBERTa Using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing*. Comment: 16 pages, 10 tables, 2 Figures. The DeBERTaV3 model significantly improves performance of the downstream NLU tasks over models with a similar structure, e.g. DeBERTaV3 large achieves 91.37% average GLUE score which is 1.37% over DeBERTa large. XSmall has only 22M backbone parameters, but significantly outperforms RoBERTa/XLNet-base. Paper is published as a conference paper at ICLR 2023. Mar. 2023. arXiv: 2111.09543 [cs].

[47] He, P. et al. *DeBERTa: Decoding-enhanced BERT with Disentangled Attention*. Comment: 20 pages,5 figures, 13 tables. In v2, we scale up DeBERTa to 1.5B parameters and it surpasses the human performance on SuperGLUE leaderboard for the first time as of December 29, 2020. In v3, we replace MLM with RTD objective which significantly improves the model performance. Oct. 2021. arXiv: 2006.03654 [cs].

[48] Henderson, P. et al. *Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning*. 2022. arXiv: 2002.05651 [cs.CY].

[49] Hochreiter, S. and Schmidhuber, J. "Long Short-Term Memory". In: *Neural Computation* 9 (1997), pp. 1735–1780.

[50] Hoffmann, J. et al. *Training Compute-Optimal Large Language Models*. 2022. arXiv: 2203.15556 [cs.CL].

[51] Holtzman, A. et al. *The Curious Case of Neural Text Degeneration*. 2020. arXiv: 1904.09751 [cs.CL].

[52] Hu, E. J. et al. "LoRA: Low-Rank Adaptation of Large Language Models". In: *CoRR* abs/2106.09685 (2021). arXiv: 2106.09685.

[53] Huebner, P. A. and Willits, J. A. "Chapter Eight - Using lexical context to discover the noun category: Younger children have it easier". In: *The Context of Cognition: Emerging Perspectives*. Ed. by K. D. Federmeier and L. Sahakyan. Vol. 75. Psychology of Learning and Motivation. Academic Press, 2021, pp. 279–331. DOI: https://doi.org/10.1016/bs.plm.2021.08.002.

[54] Huebner, P. A. et al. "BabyBERTa: Learning More Grammar With Small-Scale Child-Directed Language". In: *Proceedings of the 25th Conference on Computational Natural Language Learning*. Online: Association for Computational Linguistics, 2021, pp. 624–646. DOI: 10.18653/v1/2021.conll-1.49.

[55] Ivanova, A. A. et al. "Elements of World Knowledge (EWOK): A cognition-inspired framework for evaluating basic world knowledge in language models". In: *arXiv preprint arXiv:2405.09605* (2024).

[56] Izsak, P., Berchansky, M., and Levy, O. "How to Train BERT with an Academic Budget". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, pp. 10644–10652. DOI: 10.18653/v1/2021.emnlp-main.831.

[57] Izsak, P., Berchansky, M., and Levy, O. "How to Train BERT with an Academic Budget". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Ed. by M.-F. Moens et al. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 10644–10652. DOI: 10.18653/v1/2021.emnlp-main.831.

[58] Jelinek, F. "Statistical methods for speech recognition". In: 1997.

[59] Jiao, X. et al. *TinyBERT: Distilling BERT for Natural Language Understanding*. 2020. arXiv: 1909.10351 [cs.CL].

[60] Joshi, M. et al. "SpanBERT: Improving Pre-training by Representing and Predicting Spans". In: *CoRR* abs/1907.10529 (2019). arXiv: 1907.10529.

[61] Jumelet, J. et al. "ChapGTP, ILLC's Attempt at Raising a BabyLM: Improving Data Efficiency by Automatic Task Formation". In: *arXiv preprint arXiv:2310.11282* (2023).

[62] Jung, J. et al. *Impossible Distillation: From Low-Quality Model to High-Quality Dataset & Model for Summarization and Paraphrasing*. Apr. 5, 2024. DOI: 10.48550/arXiv.2305.16635. arXiv: 2305.16635 [cs]. URL: (visited on 06/20/2024). Pre-published.

[63] Jurafsky, D. and Martin, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd. Online manuscript released August 20, 2024. 2024.

[64] Kaddour, J. *The MiniPile Challenge for Data-Efficient Language Models*. 2023. arXiv: 2304.08442 [cs.CL].

[65] Kaplan, J. et al. "Scaling Laws for Neural Language Models". In: *CoRR* abs/2001.08361 (2020). arXiv: 2001.08361.

[66] Khan, S. H. et al. "Transformers in Vision: A Survey". In: *ACM Computing Surveys (CSUR)* 54 (2021), pp. 1–41.

[67] Khashabi, D. et al. "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)*. Ed. by M. Walker, H. Ji, and A. Stent. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 252–262. DOI: 10.18653/v1/N18-1023.

[68] Kritharoula, A., Lymperaiou, M., and Stamou, G. *Language Models as Knowledge Bases for Visual Word Sense Disambiguation*. 2023. arXiv: 2310.01960 [cs.CL].

[69] Kritharoula, A., Lymperaiou, M., and Stamou, G. "Large Language Models and Multimodal Retrieval for Visual Word Sense Disambiguation". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2023, pp. 13053–13077. DOI: 10.18653/v1/2023.emnlp-main.807.

[70] Kumar, V., Choudhary, A., and Cho, E. "Data augmentation using pre-trained transformer models". In: *arXiv preprint arXiv:2003.02245* (2020).

[71] Kumar, V., Choudhary, A., and Cho, E. *Data Augmentation using Pre-trained Transformer Models*. 2021. arXiv: 2003.02245 [cs.CL].

[72] Levesque, H. J., Davis, E., and Morgenstern, L. "The Winograd Schema Challenge". In: *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*. 2011.

[73] Lewis, M. et al. "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by D. Jurafsky et al. Online: Association for Computational Linguistics, July 2020, pp. 7871–7880. DOI: 10.18653/v1/2020.acl-main.703.

[74] Li, J. et al. "Visualizing and Understanding Neural Models in NLP". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Ed. by K. Knight, A. Nenkova, and O. Rambow. San Diego, California: Association for Computational Linguistics, June 2016, pp. 681–691. DOI: 10.18653/v1/N16-1082.

[75] Li, Y. et al. *Textbooks Are All You Need II: Phi-1.5 Technical Report*. Sept. 11, 2023. arXiv: 2309.05463 [cs]. URL: (visited on 04/23/2024). Pre-published.

[76] Liang, P. et al. *Holistic Evaluation of Language Models*. 2023. arXiv: 2211.09110 [cs.CL].

[77] Linzen, T. "How Can We Accelerate Progress Towards Human-like Linguistic Generalization?" In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by D. Jurafsky et al. Online: Association for Computational Linguistics, July 2020, pp. 5210–5217. DOI: 10.18653/v1/2020.acl-main.465.

[78] Lison, P. and Tiedemann, J. "OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Ed. by N. Calzolari et al. Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, pp. 923–929.

[79] Liu, L. et al. "Understanding the Difficulty of Training Transformers". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by B. Webber et al. Online: Association for Computational Linguistics, Nov. 2020, pp. 5747–5763. DOI: 10.18653/v1/2020.emnlp-main.463.

[80] Liu, Y. et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. July 2019. DOI: 10.48550/arXiv.1907.11692. arXiv: 1907.11692 [cs].

[81] MacWhinney, B. *The Childes Project*. Psychology Press, Jan. 2014. ISBN: 9781317778165. DOI: 10.4324/9781315805641.

[82] Manning, C. D. and Schütze, H. "Book Reviews: Foundations of Statistical Natural Language Processing". In: *International Conference on Computational Logic*. 1999.

[83] Martinez, R. D. et al. *CLIMB: Curriculum Learning for Infant-inspired Model Building*. 2023. arXiv: 2311.08886 [cs.CL].

[84] Matthews, B. W. "Comparison of the predicted and observed secondary structure of T4 phage lysozyme." In: *Biochimica et biophysica acta* 405 2 (1975), pp. 442–51.

[85] McKenzie, I. R. et al. *Inverse Scaling: When Bigger Isn't Better*. 2024. arXiv: 2306.09479 [cs.CL].

[86] Meng, Y. et al. *Generating Training Data with Language Models: Towards Zero-Shot Language Understanding*. Oct. 11, 2022. DOI: 10.48550/arXiv.2202.04538. arXiv: 2202.04538 [cs]. URL: (visited on 06/26/2024). Pre-published.

[87] Meng, Y. et al. *Generating Training Data with Language Models: Towards Zero-Shot Language Understanding*. 2022. arXiv: 2202.04538 [cs.CL].

[88] Micikevicius, P. et al. *Mixed Precision Training*. 2018. arXiv: 1710.03740 [cs.AI].

[89] Mikolov, T., Yih, W.-t., and Zweig, G. "Linguistic Regularities in Continuous Space Word Representations". In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Ed. by L. Vanderwende, H. Daumé III, and K. Kirchhoff. Atlanta, Georgia: Association for Computational Linguistics, June 2013, pp. 746–751.

[90] Mikolov, T. et al. "Recurrent neural network based language model". In: *Interspeech*. 2010.

[91] Mikolov, T. et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].

[92] Minaee, S. et al. *Large Language Models: A Survey*. 2024. arXiv: 2402.06196 [cs.CL].

[93] Naveed, H. et al. *A Comprehensive Overview of Large Language Models*. 2024. arXiv: 2307.06435 [cs.CL].

[94] Nguyen, C. V. et al. "A Survey of Small Language Models". In: 2024.

[95] OpenAI et al. *GPT-4 Technical Report*. 2024. arXiv: 2303.08774 [cs.CL].

[96] Panagiotopoulos, I. et al. *RISCORE: Enhancing In-Context Riddle Solving in Language Models through Context-Reconstructed Example Augmentation*. 2024. arXiv: 2409.16383 [cs.CL].

[97] Pannitto, L. and Herbelot, A. "Recurrent babbling: evaluating the acquisition of grammar from limited input data". In: *Proceedings of the 24th Conference on Computational Natural Language Learning*. Ed. by R. Fernández and T. Linzen. Online: Association for Computational Linguistics, Nov. 2020, pp. 165–176. DOI: 10.18653/v1/2020.conll-1.13.

[98] Papineni, K. et al. "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Annual Meeting of the Association for Computational Linguistics*. 2002.

[99] Paszke, A. et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *CoRR* abs/1912.01703 (2019). arXiv: 1912.01703.

[100] Paszke, A. et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *CoRR* abs/1912.01703 (2019). arXiv: 1912.01703.

[101] Patwardhan, N., Marrone, S., and Sansone, C. "Transformers in the Real World: A Survey on NLP Applications". In: *Inf.* 14 (2023), p. 242.

[102] Pennington, J., Socher, R., and Manning, C. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by A. Moschitti, B. Pang, and W. Daelemans. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162.

[103] Peters, M. E. et al. *Deep contextualized word representations*. 2018. arXiv: 1802.05365 [cs.CL].

[104] Petroni, F. et al. "Language Models as Knowledge Bases?" 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)*. Ed. by K. Inui et al. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 2463–2473. DOI: 10.18653/v1/D19-1250.

[105] Portes, J. et al. *MosaicBERT: A Bidirectional Encoder Optimized for Fast Pretraining*. 2024. arXiv: 2312.17482 [cs.CL].

[106] Portes, J. et al. "MosaicBERT: How to Train BERT with a Lunch Money Budget". In: ().
[107] Qin, L. et al. *Large Language Models Meet NLP: A Survey*. 2024. arXiv: 2405.12819 [cs.CL].
[108] Qin, Z., Yang, S., and Zhong, Y. *Hierarchically Gated Recurrent Neural Network for Sequence Modeling*. 2023. arXiv: 2311.04823 [cs.CL].
[109] Qin, Z. et al. *HGRN2: Gated Linear RNNs with State Expansion*. 2024. arXiv: 2404.07904 [cs.CL].
[110] Radford, A. et al. "Language Models are Unsupervised Multitask Learners". In: (2019).
[111] Radford, A. et al. "Language Models are Unsupervised Multitask Learners". In: 2019.
[112] Raffel, C. et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2023. arXiv: 1910.10683 [cs.LG].
[113] Rajpurkar, P. et al. "SQuAD: 100,000+ Questions for Machine Comprehension of Text". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Ed. by J. Su, K. Duh, and X. Carreras. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392. DOI: 10.18653/v1/D16-1264.
[114] Rall, D., Bauer, B., and Fraunholz, T. "Towards Democratizing AI: A Comparative Analysis of AI as a Service Platforms and the Open Space for Machine Learning Approach". In: *Proceedings of the 2023 7th International Conference on Cloud and Big Data Computing* (2023).
[115] Russell, B. *The Autobiography of Bertrand Russell, Prologue*. 1956.
[116] Salazar, J. et al. "Masked Language Model Scoring". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by D. Jurafsky et al. Online: Association for Computational Linguistics, July 2020, pp. 2699–2712. DOI: 10.18653/v1/2020.acl-main.240.
[117] Salton, G., Wong, A., and Yang, C.-S. "A vector space model for automatic indexing". In: *Commun. ACM* 18 (1975), pp. 613–620.
[118] Samek, W. et al. "Explainable AI: Interpreting, Explaining and Visualizing Deep Learning". In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* (2019).
[119] Samuel, D. "Mean BERTs Make Erratic Language Teachers: The Effectiveness of Latent Bootstrapping in Low-Resource Settings". In: *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*. Ed. by A. Warstadt et al. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 221–237. DOI: 10.18653/v1/2023.conll-babylm.19.
[120] Samuel, D. et al. *Trained on 100 Million Words and Still in Shape: BERT Meets British National Corpus*. May 5, 2023. arXiv: 2303.09859 [cs]. URL: (visited on 05/29/2024). Pre-published.
[121] Sanh, V. et al. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2020. arXiv: 1910.01108 [cs.CL].
[122] Schick, T. and Schütze, H. *It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners*. 2021. arXiv: 2009.07118 [cs.CL].
[123] Shazeer, N. "GLU Variants Improve Transformer". In: *CoRR* abs/2002.05202 (2020). arXiv: 2002.05202.
[124] Shen, T. et al. *Large Language Model Alignment: A Survey*. 2023. arXiv: 2309.15025 [cs.CL].
[125] Shleifer, S., Weston, J., and Ott, M. "NormFormer: Improved Transformer Pretraining with Extra Normalization". In: *CoRR* abs/2110.09456 (2021). arXiv: 2110.09456.
[126] Socher, R. et al. "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Ed. by D. Yarowsky et al. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642.
[127] Stolcke, A. et al. "Dialogue act modeling for automatic tagging and recognition of conversational speech". In: *Computational Linguistics* 26.3 (2000), pp. 339–374.
[128] Strubell, E., Ganesh, A., and McCallum, A. "Energy and Policy Considerations for Deep Learning in NLP". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Ed. by A. Korhonen, D. Traum, and L. Màrquez. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 3645–3650. DOI: 10.18653/v1/P19-1355.
[129] Su, J. et al. "RoFormer: Enhanced Transformer with Rotary Position Embedding". In: *CoRR* abs/2104.09864 (2021). arXiv: 2104.09864.
[130] Su, J. et al. *RoFormer: Enhanced Transformer with Rotary Position Embedding*. 2023. arXiv: 2104.09864 [cs.CL].
[131] Tay, Y. et al. *Efficient Transformers: A Survey*. Mar. 2022. arXiv: 2009.06732.

[132] Theodoropoulos, N. et al. *BERTtime Stories: Investigating the Role of Synthetic Story Data in Language pre-training*. 2024. arXiv: 2410.15365 [cs.CL].

[133] Thomas, K. et al. *"I Never Said That": A dataset, taxonomy and baselines on response clarity classification*. 2024. arXiv: 2409.13879 [cs.CL].

[134] Timiryasov, I. and Tastet, J.-L. "Baby Llama: knowledge distillation from an ensemble of teachers trained on a small dataset with no performance penalty". In: *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*. Ed. by A. Warstadt et al. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 279–289. DOI: 10.18653/v1/2023.conll-babylm.24.

[135] Touvron, H. et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL].

[136] TURING, A. M. "I.—COMPUTING MACHINERY AND INTELLIGENCE". In: *Mind* LIX.236 (Oct. 1950), pp. 433–460. ISSN: 0026-4423. DOI: 10.1093/mind/LIX.236.433. eprint:

[137] Vaswani, A. et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762.

[138] Wang, A. "Glue: A multi-task benchmark and analysis platform for natural language understanding". In: *arXiv preprint arXiv:1804.07461* (2018).

[139] Wang, A. and Cho, K. "BERT has a Mouth, and It Must Speak: BERT as a Markov Random Field Language Model". In: *CoRR* abs/1902.04094 (2019). arXiv: 1902.04094.

[140] Wang, A. et al. "SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019.

[141] Warstadt, A. and Bowman, S. R. *What Artificial Neural Networks Can Tell Us About Human Language Acquisition*. 2024. arXiv: 2208.07998 [cs.CL].

[142] Warstadt, A., Singh, A., and Bowman, S. R. *Neural Network Acceptability Judgments*. 2019. arXiv: 1805.12471 [cs.CL].

[143] Warstadt, A. et al. "BLiMP: A Benchmark of Linguistic Minimal Pairs for English". In: *CoRR* abs/1912.00582 (2019). arXiv: 1912.00582.

[144] Warstadt, A. et al. "Learning Which Features Matter: RoBERTa Acquires a Preference for Linguistic Generalizations (Eventually)". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by B. Webber et al. Online: Association for Computational Linguistics, Nov. 2020, pp. 217–235. DOI: 10.18653/v1/2020.emnlp-main.16.

[145] Warstadt, A. et al. *Call for Papers – The BabyLM Challenge: Sample-efficient Pretraining on a Developmentally Plausible Corpus*. Jan. 27, 2023. arXiv: 2301.11796 [cs]. URL: (visited on 04/23/2024). Pre-published.

[146] A. Warstadt et al., eds. *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*. Singapore: Association for Computational Linguistics, Dec. 2023.

[147] Wei, J. et al. "Chain of Thought Prompting Elicits Reasoning in Large Language Models". In: *CoRR* abs/2201.11903 (2022). arXiv: 2201.11903.

[148] Wei, J. et al. *Emergent Abilities of Large Language Models*. 2022. arXiv: 2206.07682 [cs.CL].

[149] Wei, J. et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023. arXiv: 2201.11903 [cs.CL].

[150] Williams, A., Nangia, N., and Bowman, S. "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Ed. by M. Walker, H. Ji, and A. Stent. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 1112–1122. DOI: 10.18653/v1/N18-1101.

[151] Wolf, T. et al. "HuggingFace's Transformers: State-of-the-art Natural Language Processing". In: *CoRR* abs/1910.03771 (2019). arXiv: 1910.03771.

[152] Wolf, T. et al. "HuggingFace's Transformers: State-of-the-art Natural Language Processing". In: *CoRR* abs/1910.03771 (2019). arXiv: 1910.03771.

[153] Xiao, C., Hudson, G. T., and Al Moubayed, N. "Towards More Human-like Language Models Based on Contextualizer Pretraining Strategy". In: *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*. Ed. by A. Warstadt et al. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 317–326. DOI: 10.18653/v1/2023.conll-babylm.28.

[154] Xu, W. et al. "SLMRec: Empowering Small Language Models for Sequential Recommendation". In: *arXiv preprint arXiv:2405.17890* (2024).

[155] Yoo, K. M. et al. *GPT3Mix: Leveraging Large-scale Language Models for Text Augmentation*. 2021. arXiv: 2104.08826 [cs.CL].

[156] Yoo, K. M. et al. "Gpt3mix: Leveraging large-scale language models for text augmentation". In: *arXiv preprint arXiv:2104.08826* (2021).

[157] Yu, L. et al. *MEGABYTE: Predicting Million-byte Sequences with Multiscale Transformers*. 2023. arXiv: 2305.07185 [cs.LG].

[158] Zhang, S. et al. *Instruction Tuning for Large Language Models: A Survey*. 2024. arXiv: 2308.10792 [cs.CL].

[159] Zhang, S. et al. *OPT: Open Pre-trained Transformer Language Models*. 2022. arXiv: 2205.01068 [cs.CL].

[160] Zhang, Y. et al. "When Do You Need Billions of Words of Pretraining Data?" In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Ed. by C. Zong et al. Online: Association for Computational Linguistics, Aug. 2021, pp. 1112–1125. DOI: 10.18653/v1/2021.acl-long.90.

[161] Zhang, Z. et al. "Baby's CoThought: Leveraging Large Language Models for Enhanced Reasoning in Compact Models". In: *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*. Ed. by A. Warstadt et al. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 158–170. DOI: 10.18653/v1/2023.conll-babylm.13.

[162] Zhao, W. X. et al. *A Survey of Large Language Models*. Oct. 2024. arXiv: 2303.18223.

[163] Zhu, Y. et al. "Texygen: A benchmarking platform for text generation models". In: *The 41st international ACM SIGIR conference on research & development in information retrieval*. 2018, pp. 1097–1100.