Sapienza University of Rome

Department of Computer Science
PhD in Computer Science

Thesis For The Degree Of Doctor Of Philosophy

Effective, Efficient and Reliable Large Language Models

Thesis Advisor
Prof. Emanuele Rodolà

Candidato
Andrea Santilli
1893210

Academic Year 2020-2024 (XXXVI cycle)

Abstract

In recent years, Large Language Models (LLMs) have fundamentally transformed the field of Natural Language Processing (NLP), reshaping the landscape of AI research and applications. This thesis represents the culmination of four years of doctoral research, which began in 2020 when LLMs were still an emerging technology and GPT-3 had just been introduced. Over the course of this research, we have both observed and contributed to the advancement of some of the technologies underpinning LLMs, from their early stages to their current role as cutting-edge AI systems. Specifically, this thesis combines some of the works carried out during this time under three critical dimensions of LLMs: Effectiveness, Efficiency, and Reliability.

On the Effectiveness dimension, we contributed to the development of instruction tuning - a key technique now ubiquitous in the training pipeline of LLMs. Our work demonstrated that smaller, instruction-tuned LLMs can outperform models up to 16 times their size, including GPT-3 [1]. We also developed PromptSource, an integrated development environment for creating, managing, and sharing natural language prompts, which has become a valuable resource for the NLP community [2]. Both of these contributions were carried out during the BigScience Workshop, a year-long open research initiative by Hugging Face targeting the study of LLMs. Finally, along this dimension, we studied how to make these models handle multimodal database-like queries [3].

Addressing the Efficiency dimension, we tackled the challenge of accelerating LLM inference. We introduced three novel parallel decoding algorithms that significantly speed up text generation without compromising output quality [4]. This has since evolved into an active research area known as speculative or parallel decoding. Furthermore, we developed an efficient, language-specific instruction-tuned LLM for the Italian language, demonstrating a cost-effective approach to creating high-quality models for specific languages [5].

Our research on Reliability addresses the critical issue of making these models reliable since they have been shown to systematically generate incorrect information - a phenomenon known as hallucinations. In this direction, we investigated whether it's possible to detect the model's confidence in its outputs. We conducted a comprehensive assessment of current uncertainty quantification methods and their evaluation protocols [6] and explored novel approaches to combine these methods to improve the detection and quantification of uncertainty in LLM outputs [7].

Our work paves the way for more Effective, Efficient, and Reliable large language models, addressing key challenges in their development and deployment while opening new avenues for future research in this rapidly evolving field.

Keywords: Large Language Models, Instruction Tuning, Efficient Decoding, Uncertainty Estimation in LLMs

Contents

Publications in this Thesis

[1] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan D. Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng-Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, **Andrea Santilli**, Thibault Févry, Jason Alan Fries, Ryan Teehan, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf and Alexander M. Rush. *"Multitask Prompted Training Enables Zero-Shot Task Generalization."* In *International Conference on Learning Representations. 2022.*

[2] Stephen Bach, Victor Sanh, Zheng Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, **Andrea Santilli**, Zhiqing Sun, Srulik Ben-david, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Fries, et al.. 2022. *"PromptSource: An Integrated Development Environment and Repository for Natural Language Prompts".* In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 93–104, Dublin, Ireland. Association for Computational Linguistics.*

[3] Giovanni Trappolini, **Andrea Santilli**, Emanuele Rodolà, Alon Y. Halevy and Fabrizio Silvestri. *"Multimodal Neural Databases."* In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (2023).*

[4] **Andrea Santilli**, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodola. 2023. *"Accelerating Transformer Inference for Translation via Parallel Decoding".* In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 12336–12355, Toronto, Canada. Association for Computational Linguistics.*

[5] **Andrea Santilli** and Emanuele Rodolà. *"Camoscio: an Italian Instruction-tuned LLaMA."* In *CLiC-it 2023: 9th Italian Conference on Computational Linguistics, Nov 30 — Dec 02, 2023, Venice, Italy.*

[6] **Andrea Santilli**, Miao Xiong, Michael Kirchhof, Pau Rodrigues, Federico Danieli, Xavier Suau, Luca Zappella, Sinead Williamson, Adam Goliński. *"On a Spurious Interaction between Uncertainty Scores and Answer Evaluation Metrics in Generative QA Tasks"* In *Workshop of Safe Generative AI Workshop @ NeurIPS 2024*

[7] Miao Xiong, **Andrea Santilli**, Michael Kirchhof, Adam Goliński, Sinead Williamson. *"Efficient And Effective Uncertainty Quantification For LLMs"* In *Workshop of Safe Generative AI Workshop @ NeurIPS 2024*

Other Publications

[1] Emilian Postolache*, Giorgio Mariani*, Michele Mancusi*, **Andrea Santilli**, Luca Cosmo and Emanuele Rodolà. *"Latent Autoregressive Source Separation."* In *Proceedings of the AAAI Conference on Artificial Intelligence 37 (8), 9444-9452.*

[2] BigBench Contributors *"Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models."* In *Transaction of Machine Learning Research*.

[3] Michele Miranda, Elena Sofia Ruzzetti, **Andrea Santilli**, Fabio Massimo Zanzotto, Sébastien Bratières and Emanuele Rodolà. *"Preserving Privacy in Large Language Models: A Survey on Current Threats and Solutions"* In *Transaction of Machine Learning Research*

[4] Andrea Bacciu, Giovanni Trappolini, **Andrea Santilli**, Emanuele Rodolà, Fabrizio Silvestri *"Fauno: The Italian Large Language Model that will leave you senza parole!"* In *IIR 2023 - 13th Italian Information Retrieval Workshop*.

[5] BigScience Workshop contributors *"BLOOM: A 176B-Parameter Open-Access Multilingual Language Model."* In *arXiv preprint arXiv:2211.05100*.

[6] Michele Mancusi, Emilian Postolache, Giorgio Mariani, Marco Fumero, **Andrea Santilli**, Luca Cosmo, Emanuele Rodolà *"Unsupervised source separation via Bayesian inference in the latent domain"* In *arXiv preprint arXiv:2110.05313*.

[7] Antonio Norelli, Giorgio Mariani, Luca Moschella*, **Andrea Santilli***, Giambattista Parascandolo, Simone Melzi, Emanuele Rodolà *"Explanatory learning: Beyond empiricism in neural networks"* In *arXiv preprint arXiv:2201.10222*.

List of Figures

List of Tables

Nomenclature

Autoregressive Decoding The standard method of text generation in LLMs, where each token is predicted based on the previously generated tokens.

Few-shot Learning A learning paradigm where a model can perform a new task with only a few examples.

Hallucination The phenomenon where a LLM generates plausible but factually incorrect information.

Instruction Tuning A fine-tuning technique that trains language models on a diverse set of tasks formatted as natural language instructions.

Large Language Model (LLM) A neural network-based model trained on vast amounts of text data to understand and generate human-like text.

LoRA (Low-Rank Adaptation) A parameter-efficient fine-tuning technique for large language models.

Multimodal Neural Database A system combining LLM capabilities with traditional database functionalities to handle queries across different data modalities.

Parallel Decoding A technique to accelerate text generation in LLMs by predicting multiple tokens simultaneously.

Prompt Engineering The practice of designing and refining input prompts to elicit desired behaviors from language models.

Prompt piece of text or instruction given to a model to elicit a specific response or guide its output.

Reinforcement Learning from Human Feedback (RLHF) A training approach that fine-tunes language models based on human preferences via Reinforcement Learning.

Selective Answering A strategy where an AI system chooses whether to answer a query based on its confidence in providing an accurate response.

Transformer A neural network architecture using self-attention mechanisms to process sequential data, forming the backbone of most modern LLMs.

Uncertainty Quantification Methods to estimate and quantify the confidence of an LLM's outputs.

Zero-shot Learning The ability of a model to perform a task without any specific training examples.

Chapter 1

Introduction

The field of Natural Language Processing (NLP) has experienced a significant shift in recent years, primarily due to the rapid development of Large Language Models (LLMs) [8, 9]. The key behind their success lies in their remarkable flexibility, which allows them to address a broad range of tasks with minimal task-specific data, often requiring only contextual information. Their ability to perform effectively in zero-shot scenarios or learn from just a few examples through in-context learning has greatly expanded the scope of artificial intelligence applications, leading to a growing interest in the LLM research field.

Although the concept of language modeling is not new, the rise of neural models in recent years has led to a dramatic improvement in both the quality and range of tasks these models can handle [10, 11]. The primary driver behind this progress has been the scaling of models in both size (number of parameters) and the amount of training data (hence the name “large") [12, 13] and the subsequent development of techniques like Instruction tuning and Reinforcement Learning with Human Feedback (RLHF) [1, 14, 15]. These advancements have resulted in LLMs that excel not only in language generation and manipulation but also in understanding context, following instructions, and adapting to new tasks with minimal guidance sparking the development of general text assistants like ChatGPT or GPT-4 [9, 16].

This thesis encompasses a critical period in the evolution of LLMs, spanning from 2020 to 2024. When this research began, LLMs were in their infancy, and several key innovations in the field were yet to emerge (See Figure 1.1). Over this time, we have not only witnessed but also contributed to the rapid advancement of LLM technologies. The work presented herein addresses three fundamental crucial dimensions of the development of LLMs: **Effectiveness** (Chapter 2), **Efficiency** (Chapter 3), and **Reliability** (Chapter 4).

Under the **Effectiveness** direction (Chapter 2), we tackled the challenge of improving LLM capabilities. The first generation of language models primarily relied on self-supervised pretraining with large *unannotated* text corpora. We introduced instruction tuning as a second fine-tuning step after the initial pretraining. This approach involves training the model on a diverse set of tasks formulated as natural language instructions, significantly improving its zero-shot performance on unseen tasks [1]. This chapter also introduces PromptSource, an integrated development environment for creating and managing prompts [2]. PromptSource has become a valuable resource for the NLP community, facilitating standardized prompt engineering and collaborative research. Additionally, we explore the concept of Multimodal Neural Databases, extending the capabilities of LLMs to handle complex queries across different data modalities [3].

Under the **Efficiency** direction (Chapter 3), we tackled the problem of speeding up the generation of LLMs. These models are parallel during training but inefficient during generation since they rely on slow sequential autoregressive decoding, where models generate text one word at a time, which can be slow.

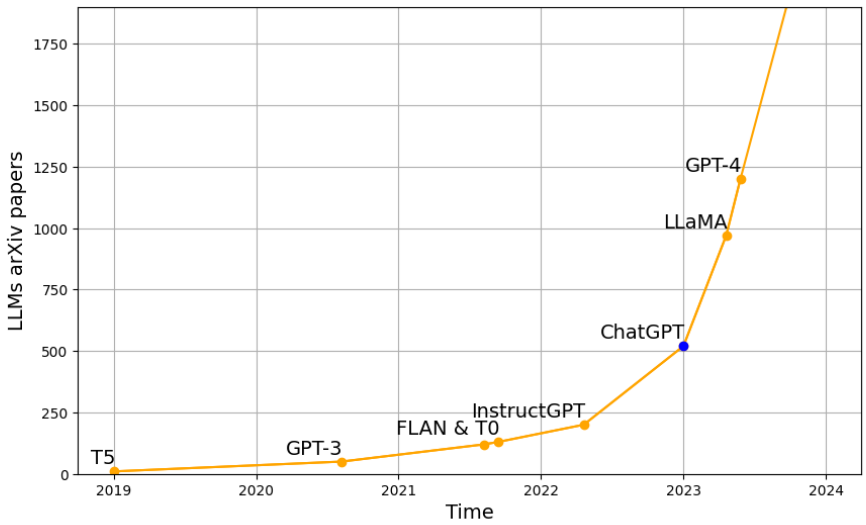


Figure 1.1: Cumulative number of papers on ArXiv containing the keyword "large language model" (since October 2019) in title or abstract (exact match). Important landmark models are depicted in the picture at the time of their appearance on ArXiv or release. Data extracted from Zhao et al. [8]. The model T0 that introduced instruction tuning is part of this thesis (Chapter 2).

We introduced three novel parallel decoding algorithms that significantly speed up text generation without compromising output quality [4]. This has since evolved into an active research area known as speculative or parallel decoding. Furthermore, we developed an efficient, language-specific instruction-tuned LLM for the Italian language, demonstrating a cost-effective approach to creating high-quality models for specific languages [5].

Under the **Reliability** section (Chapter 4), we address the challenge of enhancing the reliability of current models. Despite notable progress in their capabilities and efficiency, reliability remains a major concern. A key issue in this area is the phenomenon of hallucinations, where models produce outputs that, while fluent and confident, are factually incorrect or misleading [17, 18]. We conducted a comprehensive assessment of current uncertainty quantification methods and their evaluation protocols [6] and explored novel approaches to combine these methods to improve the detection and quantification of uncertainty in LLM outputs [7]. As LLMs are increasingly deployed in critical domains such as healthcare, finance, and legal systems, the ability to accurately assess model confidence and detect potential errors becomes paramount.

By focusing on these three interconnected aspects, this thesis aims to provide a comprehensive collection of advancements in language models under the axes of effectiveness, efficiency, and reliability produced during the course of doctoral studies. We hope that our work can serve as a foundation for future research and practical applications of LLMs. The following chapters detail our methodologies, findings, and insights, offering a view into the rapidly evolving landscape of LLMs. Through this work, we seek to contribute to the development of more capable, resource-efficient, and dependable language AI systems.

1.1 Key Contributions

This thesis presents advancements in Large Language Model research, focusing on Effectiveness, Efficiency, and Reliability. Our work addresses key challenges in LLM development and application, offering practical solutions. The following contributions represent the core of our research, each targeting a specific aspect.

Effectiveness (Chapter 2)

- We propose instruction tuning [1] as a way to improve the zero-shot capabilities of language models. By fine-tuning models on a diverse set of tasks formulated as natural language instructions, we demonstrate that smaller models can outperform much larger ones (up to 16x their size) on a wide range of unseen tasks (Section 2.1).
- We develop PromptSource [2], an integrated development environment for prompt engineering. This tool facilitates the creation, management, and sharing of natural language prompts, becoming a valuable resource for the NLP community (Section 2.2).
- We introduce Multimodal Neural Databases [3]. This work investigates how LLMs can handle complex, database-like queries across different modalities, demonstrating their potential in integrating and reasoning over diverse data types (Section 2.3).

Efficiency (Chapter 3)

- We tackle the challenge of accelerating LLM inference by introducing three novel parallel decoding algorithms [4]. These algorithms significantly speed up text generation without compromising output quality, paving the way for more efficient deployment of LLMs in real-time applications (Section 3.1).
- We develop Camoscio [5], the first open instruction-tuned LLM for the Italian language. This project demonstrates a cost-effective approach to creating high-quality, language-specific models using parameter-efficient fine-tuning techniques like LoRA. Camoscio showcases how to effectively adapt large language models to specific languages with limited computational resources (Section 3.2).

Reliability (Chapter 4)

- We conduct a comprehensive assessment of current uncertainty quantification methods for LLMs [6]. This work identifies inconsistencies and limitations in existing evaluation protocols, proposing improved methodologies for assessing uncertainty estimation in LLMs. Our findings highlight the importance of robust evaluation practices in developing reliable AI systems (Section 4.1).
- Building on our assessment, we explore novel approaches to combine uncertainty estimation methods [7]. We demonstrate that strategically combining simple, computationally efficient uncertainty estimation techniques can match or even surpass the performance of more complex methods. This work offers a promising path toward developing more reliable and trustworthy LLMs without incurring excessive computational costs (Section 4.2).

1.2 Structure of the Thesis

The thesis is structured as a collection of the main articles published during the doctoral research period [1–7]. These works are divided into three chapters according to their topic: Effectiveness (Chapter 2) [1–3], Efficiency (Chapter 3) [4, 5] and Reliable (Chapter 4) [6, 7]. Each chapter begins with a custom-written introduction tailored specifically for this thesis, explaining the rationale behind grouping the included papers. After the introduction, each paper is presented in its own section within the chapter. The papers are reproduced as verbatim copies of the original conference articles, with only minor stylistic adjustments to align them with the thesis format. Finally, a chapter with the overall thesis conclusions is proposed to the reader (Chapter 5).

1.3 Author Contribution Statement

The research presented in this thesis was conducted by Andrea Santilli under the supervision of Prof. Emanuele Rodolà at Sapienza University of Rome. Andrea Santilli authored all the publications included in this thesis, which are also co-authored by collaborators who made invaluable contributions to the final development of the respective research projects. These collaborators are acknowledged in the "Acknowledgments" section at the end of this document.

The works [1] and [2] were carried out as part of the BigScience Workshop[1], a collaborative year-long initiative dedicated to studying LLMs and datasets, involving over 600 researchers from 50 countries and more than 250 institutions. These two works were the outcome of the BigScience Prompt Engineering working group, which focused on the role of prompting in LLM training. This open-science initiative by Hugging Face enabled researchers, including Andrea Santilli, to engage in large-scale experimentation and investigation into prompting techniques for LLMs.

Given the scale and computational resources required for these projects, academic institutions alone lacked the capacity to undertake such research. Thus, large collaborations like BigScience were essential, especially during the period when this research was conducted (circa 2021).

Within the BigScience collaboration, Andrea Santilli's primary contributions were the implementation of prompting templates and the conversion of supervised datasets into promptable formats using the *PromptSource* tool. The order of authorship for these publications reflects the contributions to the project's code and other implementation aspects (at least 3 commits).

For all other publications included in this thesis, Andrea Santilli was responsible for or significantly contributed to various stages of the research pipeline, including conceptualization, design, execution of experiments, and development of methodologies, with the extent of involvement varying according to the order of authorship. In these cases, as a general rule, the order of authorship reflects the relative magnitude of each collaborator's contribution.

[1] `https://bigscience.huggingface.co/`

Chapter 2

Effective Large Language Models

When Brown et al. [19] introduced GPT-3 in July 2020 the main modality of training for LLMs was self-supervision: the model was trained on large *unannotated* corpora from the web to predict the next word given the previous context. Doing this allows the emergence of some interesting properties from the model like zero-shot generalization and in-context learning i.e., the model was able to leverage examples or prompts provided within the input context to generate appropriate outputs without explicit retraining, adapting to new tasks based on the immediate information presented and able to solve a broad set of tasks verbalized in the prompt without explicit retraining. The prevailing hypothesis posited that the emergence of these abilities was a consequence of the scale [20].

In the first work of this chapter (§2.1), we asked whether this zero-shot generalization capability of LLM could instead be inducted by *explicit* multitask training on a supervised dataset converted to a promptable format with verbalized instruction in natural language. The results showed that this was not only possible but really useful to improve these models and align them better to follow human instruction. During the *International Conference on Learning Representations* in 2022, instruction tuning was concurrently introduced in our work [1] and Wei et al. [14] and it's now a part of the standard training pipeline of any language model after the self-supervised stage on unannotated corpora [21]. Instruction tuning consists of further training LLMs using `(INSTRUCTION, OUTPUT)` pairs where `INSTRUCTION` represents the prompt provided for the model, and `OUTPUT` signifies the expected response that corresponds to that `INSTRUCTION`.

The second work in this chapter presents PromptSource (§2.2), an integrated development environment for prompt engineering. This tool was used for the creation of the instruction prompts for the first work and aims to facilitate the creation, management, and sharing of natural language prompts. A large instruction-tuning dataset with over 2,000 prompts for roughly 170 datasets was released together with the tool.

Finally, the last chapter (§2.3) concludes with our exploration into multimodal capabilities of LLMs, introducing the concept of Multimodal Neural Databases. This work investigates how LLMs can handle complex, database-like queries across different modalities, demonstrating their potential in integrating and reasoning over diverse data types. By extending the effectiveness of LLMs beyond text to include other modalities, we open new avenues for their application in more complex, real-world scenarios.

The first two works of this chapter were carried out during the BigScience Workshop[1] as part of contributions from the author of this thesis in the Prompt Engineering working group, a subgroup of the large project focused on investigating the role of prompting in LLMs. The broad BigScience project was an

[1] `https://bigscience.huggingface.co/`

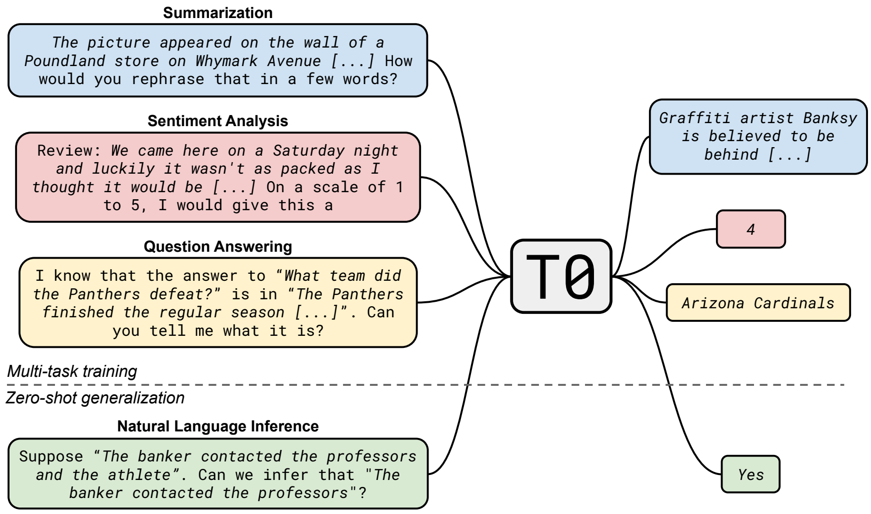


Figure 2.1: Our model and prompt format. T0 is an encoder-decoder model that consumes textual inputs and produces target responses. It is trained on a multitask mixture of NLP datasets partitioned into different tasks. Each dataset is associated with multiple prompt templates that are used to format example instances to input and target pairs. Italics indicate the inserted fields from the raw example data. After training on a diverse mixture of tasks (top), our model is evaluated on zero-shot generalization to tasks that are not seen during training (bottom).

open science [22] research initiative by Hugging Face targeting the study of LLMs involving 600 researchers from 50 countries and more than 250 institutions. Given the scale and the resources necessary to train these models, this collaboration allowed us to pursue this kind of research.

2.1 Training Language Models with Instruction Tuning

This section presents the paper "Multitask Prompted Training Enables Zero-shot Task Generalization"[1].

Recent work has shown that large language models exhibit the ability to perform reasonable zero-shot generalization to new tasks [23, 24]. Despite being trained on only language modeling objectives, these models can perform relatively well at new tasks that they have not been explicitly trained to perform, for instance answering a question on a passage or performing summarization. An influential hypothesis is that large language models generalize to new tasks as a result of an implicit process of multitask learning [12]. As a byproduct of learning to predict the next word, a language model is forced to learn from a mixture of implicit tasks included in their pretraining corpus. For example, by training on generic text from a web forum, a model might implicitly learn the format and structure of question answering. This gives large language models the ability to generalize to held-out *tasks* presented with natural language prompts, going beyond prior multitask studies on generalization to held-out *datasets* [25, 26]. However, this ability requires a sufficiently large model and is sensitive to the wording of its prompts [27–29].

Further, it is an open question how implicit this multitask learning really is. Given the scale of recent language models' pretraining corpora, it is reasonable to expect that some common natural language processing (NLP) tasks would appear in an explicit form in their pretraining corpora, thereby directly training the models on those tasks. For example, there are many websites that simply contain lists of trivia questions and answers,[2] which are precisely supervised training data for the task of closed-book question answering [30]. We hypothesize that such multitask supervision in pretraining plays a large role in zero-shot

[2]For example, https://www.quizbreaker.com/trivia-questions, https://www.scarymommy.com/best-trivia-questions-answers/, and https://parade.com/944584/parade/trivia-questions-for-kids/.

generalization.

In this section, we focus on explicitly training language models in a supervised and massively multitask fashion. Our approach uses a training mixture consisting of a large set of different tasks specified in natural language prompts. Our goal is to induce a model to better generalize to held-out tasks without requiring massive scale, as well as being more robust to the wording choices of the prompts. To convert a large set of natural language tasks into prompted form, we use a simple templating language for structured datasets. We develop an interface for prompt collection from public contributors that facilitated the collection of a large multitask mixture with multiple prompts per dataset [31]. We then train a variant of the T5 encoder-decoder model [32, 33] on a subset of the tasks (each with multiple datasets) and then evaluate tasks and prompts that the model was *not* trained on.

Our experiments study two questions. First, does multitask prompted training improve generalization to held-out tasks? Second, does training on a wider range of prompts improve robustness to prompt wording? For the first question, we find that multitask training enables zero-shot task generalization by showing that our model matches or exceeds the performance of GPT-3 [23] on 9 out of 11 held-out datasets, despite being about 16× smaller. We also show that the model improves over a large baseline language model on 13 out of 14 tasks in the BIG-bench benchmark [34]. For the second question, we find that training on more prompts per dataset consistently improves the median and decreases the variability of performance on held-out tasks. Training on prompts from a wider range of datasets also generally improves the median but does not consistently decrease the variability.

2.1.1 Related Work

In this work, we distinguish implicit multitask learning in language model pretraining from *explicit* multitask learning [35], the technique for mixing multiple tasks into a single supervised training process. Models trained with multitask learning have long been shown to have improved performance in NLP [36]. Since different tasks have different outputs, applying multitask learning requires a shared format, and various have been used [37, 38]. Several multitask works also explore few-shot and zero-shot generalization to new datasets with large pretrained models (e.g., 26, 39).

Natural language prompting is the method of reformatting NLP tasks in the format of a natural language response to natural language input. The development of text-to-text pretrained models such as T5 [32] makes prompts a particularly useful method for multitask learning. For example, Khashabi et al. [25] reformat 20 question-answering datasets into a single prompt of `question: ... (A)... (B)... (C)... context: ...`, while later work such as Zhong et al. [40] and Wang et al. [41] cast a range of datasets into a single boolean QA prompt or a single NLI prompt, respectively. Although effective, these single-prompt methods typically do not generalize to new prompts or new tasks inexpressible in their fixed format.

More generally, Schick and Schütze [42] and Brown et al. [23] popularized using prompts as a generic method for all NLP tasks. Mishra et al. [43] further extend this approach to a multitask setup, training on prompts for 61 narrowly defined tasks (e.g., question generation, incorrect answer generation) adapted from 9 datasets' crowdsourcing instructions, whereas we train on and measure generalization across 62 datasets and 12 tasks as traditionally defined in the NLP literature (§2.1.2). Additionally, their prompts include labeled examples in addition to instructions, whereas we focus on zero-shot generalization. Lastly, concurrent work by Wei et al. [44] shares a similar research question with us, although we differ in several substantive regards, e.g., prompt diversity, model scale, and held-out-task scheme. We discuss our differences in detail in Section 2.1.4.

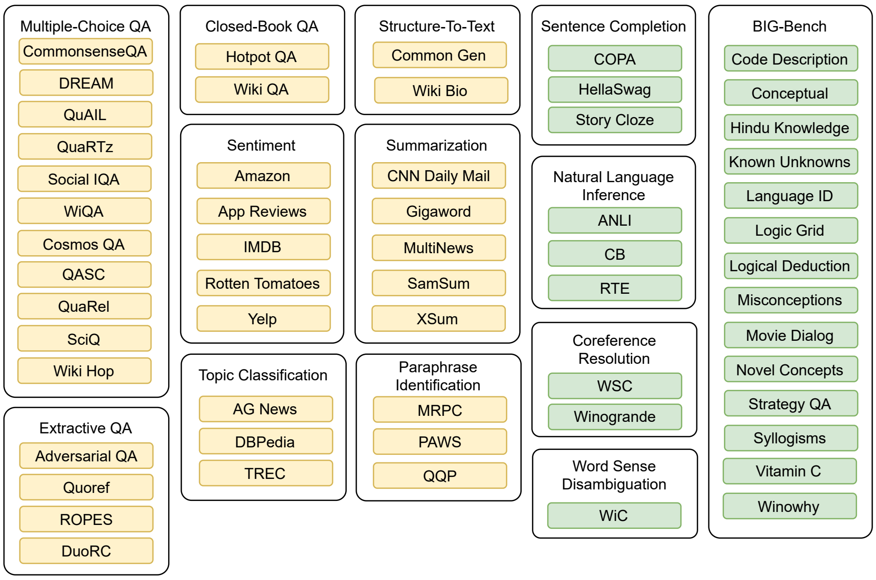


Figure 2.2: T0 datasets and task taxonomy. (T0+ and T0++ are trained on additional datasets. See Table A.5 for the full list.) Color represents the level of supervision. Yellow datasets are in the training mixture. Green datasets are held out and represent tasks that were not seen during training. Hotpot QA is recast as closed-book QA due to long input length.

Finally, in explaining the success of prompts, the leading hypothesis is that models learn to understand the prompts as task instructions which help them generalize to held-out tasks [23, 42–44]. However, the extent to which this success depends on the semantic meaningfulness of the prompts has been challenged [45, 46]. Thus, in this work, we remain agnostic as to why prompts support generalization. We only claim that prompts serve as a natural format for multitask training which empirically supports generalization to held-out tasks.

2.1.2 Measuring Generalization to Held-Out Tasks

We begin by assuming an underlying partition of NLP datasets into tasks. We use the term “task” to refer to a general NLP ability that is tested by a group of specific datasets. To evaluate zero-shot generalization to new tasks, we train on a subset of tasks and evaluate on a held-out group of tasks.

Unfortunately, NLP task categorization is fuzzy, particularly if one tries to isolate a unique skill. For example, many datasets evaluate commonsense knowledge, and some multitask works (e.g., 23, 44) define commonsense as a standalone task. However, commonsense datasets differ vastly, ranging from innate knowledge and grade-school science to DIY instructions, US cultural norms, and graduate-level theorems (see Appendix A.3.1 for a detailed discussion).

Noting that grouping by task is an imperfect heuristic, we err on the side of organizing our task taxonomy according to the task format as opposed to required skill based on conventions in the literature [26, 39, 47]. We collect all datasets from these papers and exclude those that are not in English (which also excludes programming languages and structured annotations such as parse trees) or if they require special domain knowledge (e.g., biomedicine). This yields 12 tasks and 62 datasets with publicly contributed prompts in our training and evaluation mixtures (Figure 2.2) as of writing. All experiments use datasets in the Hugging Face datasets library [48].

To test zero-shot generalization, we hold out all constituent datasets of four tasks: natural language inference (NLI), coreference resolution, sentence completion, and word sense disambiguation. We choose

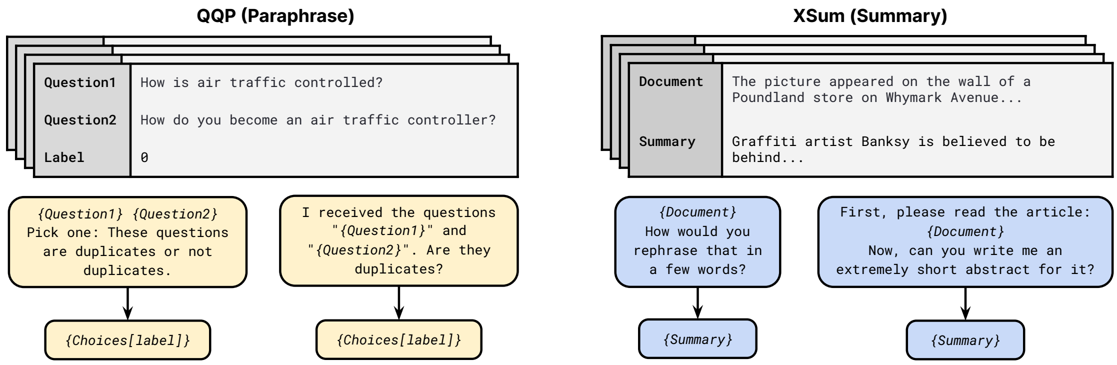


Figure 2.3: Prompt templates from the P3 prompt collection. Each dataset has multiple prompt templates consisting of an input and a target template. These use the fields of the raw data examples as well as template metadata, e.g., the left paraphrasing identification prompts use *Choices*, a template-level list variable `[,Not duplicates,, ,Duplicates,]`. These templates are materialized to produce the prompted instance shown in Figure 2.1.

NLI as a held-out task because humans also zero-shot generalize to NLI as an held-out task: Most humans are never explicitly trained to classify whether a premise sentence entails or contradicts a hypothesis sentence, yet they find it intuitive to perform this task without training [49]. For the same reason, we also hold out coreference resolution and word sense disambiguation. We further hold out sentence completion because it is a task possibly too similar to NLI (Appendix A.3.2 discusses this in detail). Additionally, we do not train our main model on any datasets that Brown et al. [23] used for evaluation, so that our main results will be a fair zero-shot comparison. We also verify that data for those tasks is not leaked through the pretraining corpus (Section A.4).

Lastly, we further evaluate on a subset of the datasets from BIG-bench, which is a recent community-driven benchmark to create a diverse collection of difficult tasks to test the abilities of large language models. The subset of BIG-bench comprise a language-oriented selection of tasks for which the BIG-bench maintainers have prepared preliminary results and which constitute text that is in-vocabulary for the T5 tokenizer (i.e. only contain English-language text without emojis or other special characters). All tasks from BIG-bench are novel tasks that are held out from our training.

A Unified Prompt Format

All datasets are given to our model in natural language prompted form to enable zero-shot experimentation. To facilitate writing a large collection of prompts, we develop a templating language and an application that make it easy to convert diverse datasets into prompts. We define a *prompt* as consisting of an input template and a target template, along with a collection of associated metadata. The templates are functions mapping a data example into natural language for the input and target sequences. Practically, the templates allow the user to mix arbitrary text with the data fields, metadata, and other code for rendering and formatting raw fields. For example, in the case of an NLI dataset, the example would include fields for `Premise, Hypothesis, Label`. An input template would be `If {Premise} is true, is it also true that {Hypothesis}?`, whereas a target template can be defined with the label choices `{Choices[label]}`. Here `Choices` is prompt-specific metadata that consists of the options `yes, maybe, no` corresponding to `label` being entailment (0), neutral (1) or contradiction (2). Other metadata documents additional properties, such as an evaluation metric. Each data example is materialized with many different prompt templates as shown in Figure 2.3.

To develop prompts, we built an interface for interactively writing prompts on datasets. We put out an open call in the research community for users to contribute prompts. 36 contributors affiliated with 24 institutions in 8 countries participated. Since our goal was to train a model to be robust to prompt format, and since the question of what makes a prompt effective remains unresolved [28, 45, 46], we encouraged contributors to be open in their style and create a diverse set of prompts. The main annotation guideline was that prompts needed to be grammatical and understandable by a fluent English speaker with no prior experience of the tasks. Additionally, prompts that required explicit counting or numerical indexing were removed in favor of natural language variants. For example, instead of predicting indices of a span extracting answers from a passage, the model is expected to copy the span's text instead. With these minimal constraints, prompt writers were encouraged to use both formal and creative prompts and various orderings of the data.

Most of the prompts correspond directly to a version of the original proposed task, although we also allow prompts that permuted the original task (for instance, generating a document from its summary). Such non-original-task prompts are included in our training mixtures for improved diversity, but they are not reported in evaluation since they deviate from the metrics and baselines reported by the original datasets.

The details of the prompting language and tool are given in Appendix A.2 and Bach et al. [31]. We collected prompts for English datasets, excluding ones that included potentially harmful content or non-natural language such as programming languages. We refer to this collection as the *Public Pool of Prompts* (P3). As of writing, P3 contains 2073 prompts for 177 datasets (11.7 prompts per dataset on average). Prompts used in experiments are all sourced from P3 except for BIG-bench, the prompts of which are provided by its maintainers.

2.1.3 Experimental Setup

Model At a high level, we fine-tune a pretrained model on our multi-task training mixture of natural language prompted datasets. Our model uses an encoder-decoder architecture with input text fed to the encoder and target text produced by the decoder. The model is trained to autoregressively generate the target through standard maximum likelihood training. Unlike decoder-only language models such as GPT-3, it is never trained to generate the input.

All models we trained are based on T5, a Transformer-based encoder-decoder language model pretrained with a masked language modeling-style objective on 1T tokens from C4 [32]. Since T5's pretraining objective is generating tokens and only tokens that have been removed from the input text, it is different from the natural text generation format of prompted datasets. Therefore, we use Lester et al. [33]'s *LM-adapted T5* model (referred to as T5+LM), produced by training T5 on 100B additional tokens from C4 on a standard language modeling objective.

Training Our main model, *T0*, is trained on the multitask mixture detailed in Section 2.1.2 and Table A.5. Meanwhile, *T0+* is the same model with identical hyperparameters except trained on a mixture that adds GPT-3's evaluation datasets. Lastly, *T0++* further adds SuperGLUE [50] to the training mixture (except RTE and CB), which leaves NLI and the BIG-bench tasks as the only held-out tasks.

The above T0 variants are all initialized from the 11B parameters version of T5+LM. To study the effect of scaling and to aid researchers with less resources, we also train *T0 (3B)*, which has the same training mixture as T0 but is initialized from the 3B parameters version of T5+LM (results reported in Section A.5).

We perform checkpoint selection by choosing the checkpoint that yields the highest score on the validation splits of our training datasets. This still satisfies the *true zero-shot* [27] setting as we do not use any examples from any of the held-out tasks to select the best checkpoint.

We assemble our multitask training mixture by combining and shuffling all examples from all training datasets. This is equivalent to sampling from each dataset in proportion to the number of examples in the dataset. However, the number of examples in each of our training datasets varies by two orders of magnitude. We therefore follow the strategy used in Raffel et al. [32] and treat any dataset with over 500'000 examples as having 500'000 / `num_templates` examples for the purposes of sampling, where `num_templates` is the number of templates created for the dataset.

We truncate input and target sequences to 1024 and 256 tokens, respectively. Following Raffel et al. [32], we use packing to combine multiple training examples into a single sequence to reach the maximum sequence length. We use a batch size of 1024 sequences (corresponding to 2^{20} total input tokens per batch) and the Adafactor optimizer [51]. Following standard practice for fine-tuning T5, we use a learning rate of 1e-3 and a dropout rate of 0.1.

Evaluation We evaluate zero-shot generalization on 11 datasets in 4 held-out traditional NLP tasks: natural language inference, coreference, word sense disambiguation, and sentence completion, as well as 14 novel tasks from BIG-bench (§2.1.2). Unless specified otherwise, we report performance on the validation splits. All reported datasets use accuracy as their metric.

For tasks that involve choosing the correct completion from several options (e.g. multiple choice question answering), we follow [23] and use *rank classification* to evaluate our model: we compute the log-likelihood of each of the target options under the fine-tuned model and select the option with the highest log-likelihood as the prediction. For simplicity, we do not apply length normalization to the log-likelihoods of the target options.

We do not perform prompt selection by comparing the performance of different prompts on the validation split; [27] highlights how such a strategy leaks information from the evaluation splits, which makes the evaluation not "true" zero-shot. For a given dataset, we report the median performance across all prompts for this dataset along with their interquartile range (Q3 - Q1) to measure the model's robustness to the wording of the prompts.

2.1.4 Results

Generalization to Held-Out Tasks

Our first research question is whether multitask prompted training improves generalization to held-out tasks. In Figure 2.4, we compare T0 against our T5+LM baseline on four held-out tasks. Our approach leads to significant gains over our baseline on all datasets, demonstrating the benefits of multitask prompted training over only language modeling training with an identical model and prompts.

Next, we compare T0 to the zero-shot performance of the largest language models available as of writing, i.e., various GPT-3 models up to 175B parameters. Note that Brown et al. [23] report performance on a single prompt,[3] whereas we report the median and interquartile range of performance across all prompts in P3 without cherry picking. We find that T0 matches or exceeds the performance of all GPT-3 models on 9 out of 11 held-out datasets. Notably, neither T0 nor GPT-3 is trained on natural language inference, yet T0 outperforms GPT-3 on all NLI datasets, even though our T5+LM baseline does not. The same is true for

[3] Our experiments in Section 2.1.4 lead us to believe that this performance corresponds to the best prompt found after manual tuning according to validation set performance.

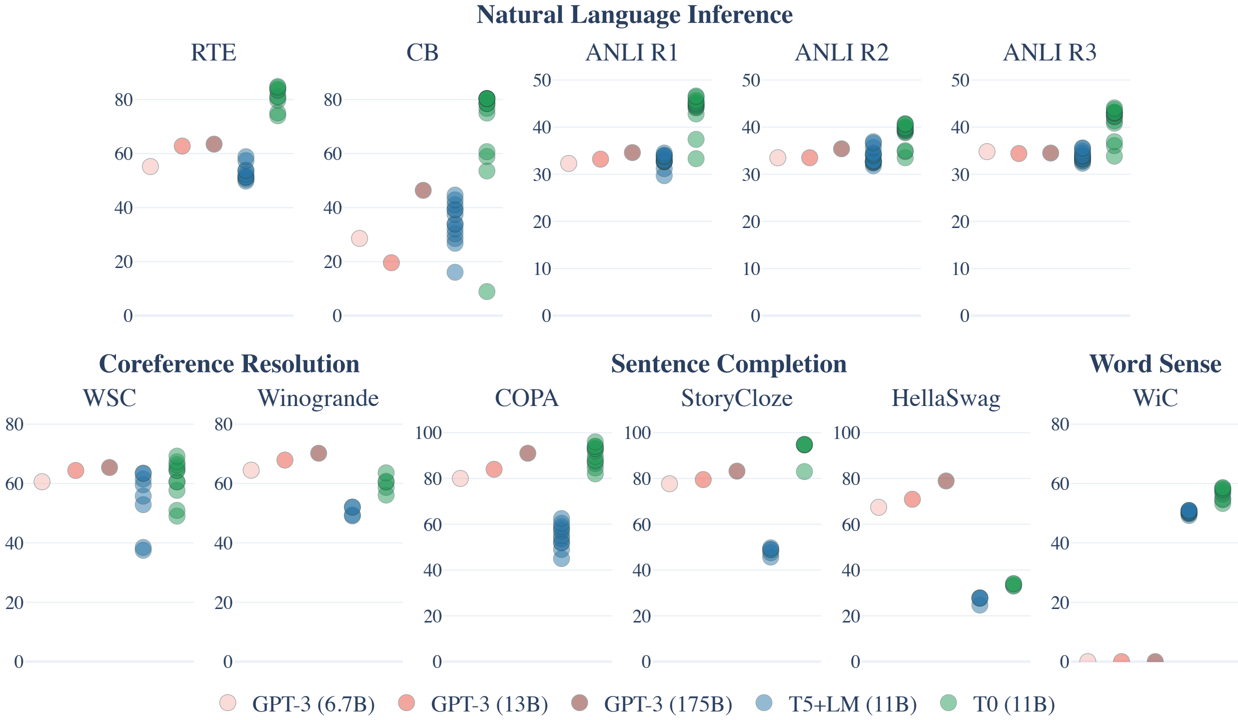


Figure 2.4: Results for T0 task generalization experiments compared to GPT-3 [23]. Each dot is the performance of one evaluation prompt. The baseline T5+LM model is the same as T0 except without multitask prompted training. GPT-3 only reports a single prompt for each dataset.

most datasets of other held-out tasks. The two exceptions are Winogrande and HellaSwag, which we discuss in Section 2.1.4.

To evaluate our models on more held-out tasks, we assess the zero-shot performance of T0, T0+, and T0++ on a subset of BIG-bench [34]. Tasks from BIG-bench cover a variety of novel skills not included in our training tasks, such as deducing the order of a sequence of objects, solving logic grid puzzles, and telling apart true statements from common misconceptions. The maintainers of BIG-bench provide a prompt for each dataset, with which we compare our models to a series of preliminary diagnostic baseline models trained by Google and evaluated by the BIG-bench maintainers. These models are decoder-only Transformer language models trained on a standard language modeling objective with varying model size. We find that at least one of the T0 variants outperform all baseline models on all tasks except for StrategyQA (Figure 2.5). In most cases, the performance of our models improves as the number of training datasets increases (i.e., T0++ outperforms T0+ which outperforms T0).

Prompt Robustness

Our second research question is whether training on a wider range of prompts improves robustness to the wording of the prompts. We conduct two ablation experiments on the effects of the average number of prompts per dataset (p) and the number of datasets (d) used during training.

Effect of More Prompts per Dataset In this analysis, we fix d and compare T0 to models with a varying number of prompts per dataset. T0 was trained on some prompts that do not map onto the dataset's original

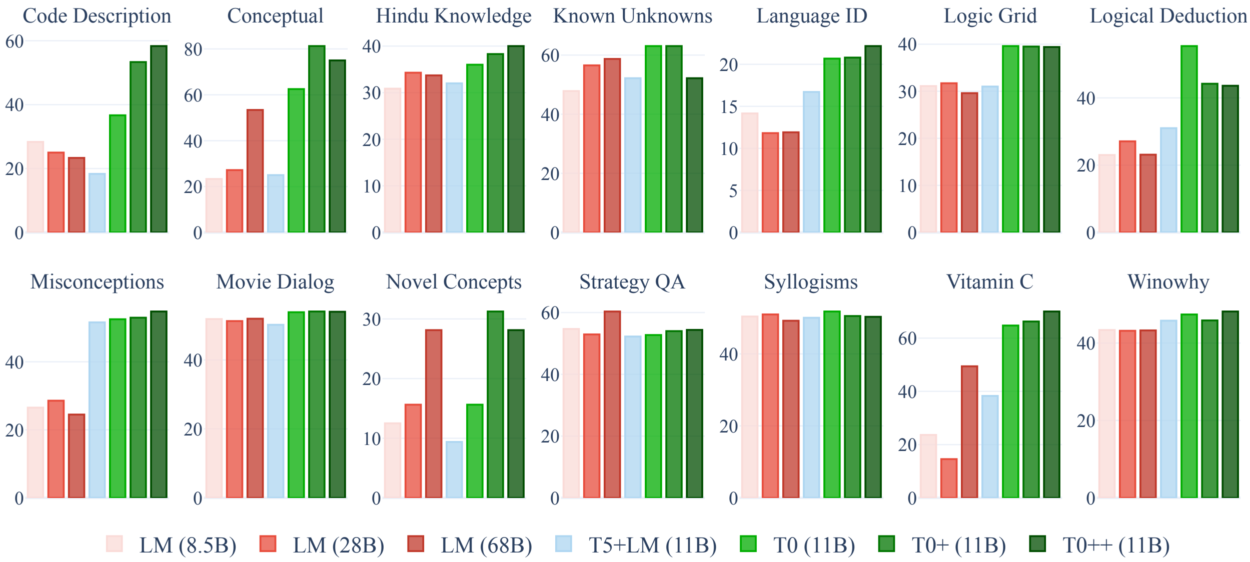


Figure 2.5: Results for a subset of BIG-bench which has available baselines. The baseline models are Transformer-based language models provided by BIG-bench maintainers, who also provide one prompt per dataset. T0, T0+ and T0++ are identical except for increasing the number of training datasets (§2.1.3). BIG-bench Tasks are all zero-shot for all the reported models.

task, for example "given an answer, generate a plausible question". Including these prompts results in p being 8.03 on average (which corresponds to our main T0 model). We compare T0 to models where $p = 1$ (one randomly chosen original-task prompt per dataset), $p = 5.7$ on average (all original-tasks prompts for all datasets), and $p = 0$ (corresponding to T5+LM without any prompted training). We train all models with the same hyperparameters and the same number of steps. Figure 2.6 shows that, even with just one prompt per dataset, performance on held-out tasks can improve substantially over the non-prompted baseline, although the spread (interquartile range between Q1 and Q3) does not consistently improve with $p = 1$. Meanwhile, further increasing p from 1 to an average of 5.7 does yield additional improvement in both median (increases for 8/11 datasets) and spread (decreases for 7/11 datasets). This reinforces our hypothesis that training on more prompts per dataset leads to better and more robust generalization to held-out tasks. Finally, we find that T0's inclusion all prompts (including those that do not correspond to the dataset's original task) further improves the median (increases for 9/11 datasets) and spread (decreases for 8/11 datasets), showing that training on non-original-task prompts can also be beneficial.

Effect of Prompts from More Datasets In this experiment, we fix $p =$ all available prompts and increase d from 39 to 49 to 55 (T0, T0+, T0++, respectively. See Section 2.1.3 for details.) Figure 2.7 shows that the median performance of all 5 held-out datasets increases as d increases from 39 to 49. However, the spread only decreases for 1 out of 5 datasets. For some datasets (e.g., ANLI), this is an artifact of the fact that some prompts always perform poorly, so that when other prompts improve, the spread is stretched larger. For other datasets (e.g., CB), however, the spread does decrease with T0+. As d increases from 49 to 55, the median performance of all datasets again increases, but the spread only decreases for 2 out of 5 datasets. Although further investigation is needed, it appears that increasing d does not consistently make the model more robust

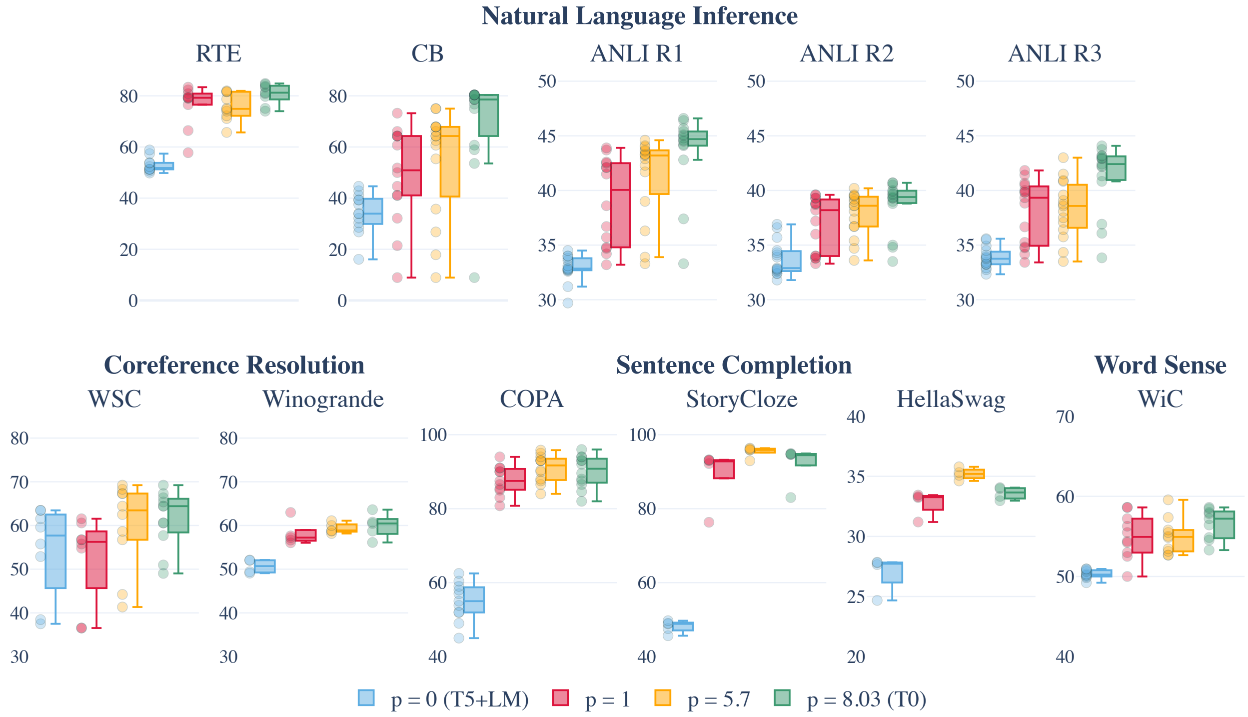


Figure 2.6: Effect of more prompts per dataset. Zero-shot performance of T0 and T5+LM when increasing number of training prompts per dataset. Each dot is the performance of one evaluation prompt. The main T0 model ($p = 8.03$) includes non-original-task prompts (see Section 2.1.2). Adding more training prompts consistently leads to higher median performance and generally lower interquartile range for held-out tasks.

to the wording of prompts.

Comparing T0 and GPT-3's robustness Because Brown et al. [23] only report one prompt per dataset with no standard deviation, we evaluate GPT-3 via OpenAI's API[4] on RTE using the same 10 prompts we evaluate T0 in order to estimate GPT-3 robustness' to different wording of prompts. One of these templates is identical to Brown et al. [23, p. 59]'s reported prompt, which scores an accuracy of 58.8%, lower than the 63.5% reported in Brown et al. [23]. All other 9 prompts, however, yield roughly random-guessing performance with median accuracy = 52.96% and interquartile range = 1.28%. These results suggest that T0 could be more robust to prompt formulation than GPT-3.

Discussion

Concurrent to our work, Wei et al. [44] proposes *FLAN*, which shares largely the same method of enabling zero-shot generalization through multitask prompted training. With a mixture of datasets similar to ours, they train multiple decoder-only language models, each with a single held-out task (cf. we focus on training one model with multiple held-out tasks in order to evaluate the model's ability to generalize to diverse tasks.) Compared to FLAN, T0's zero-shot performance is better on CB and RTE, similar on Story Cloze and COPA, and worse on Winogrande and ANLI. T0++ outperforms FLAN on CB, RTE, and COPA and matches

[4] `https://beta.openai.com/` We use the "base GPT-3 model" `davinci`. Although OpenAI does not disclose which one of their commercially available models corresponds to which models reported in Brown et al. [23], Gao et al. [52] estimate that `davinci` corresponds to the 175B model.

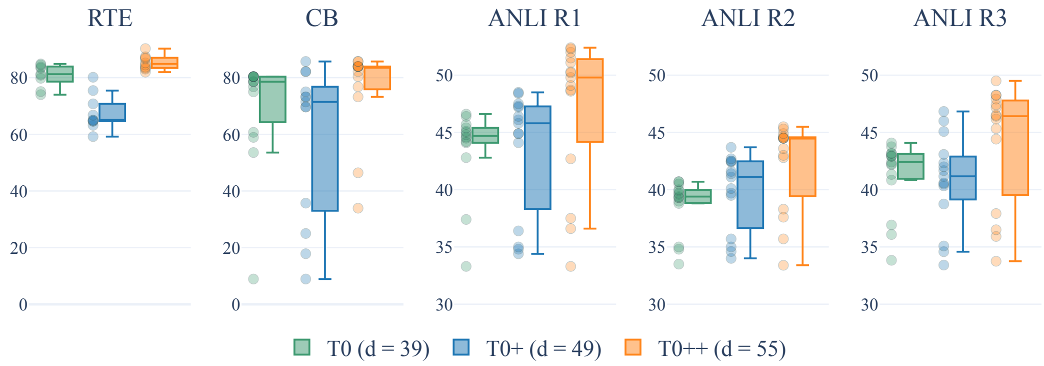


Figure 2.7: Effect of prompts from more datasets. Zero-shot performance of three models with varying number of datasets (T0, T0+, T0++). Adding more datasets consistently leads to higher median performance but does not always reduce interquartile range for held-out tasks.

FLAN's performance on Winogrande and ANLI. Notably, T0 and T0++ attain this performance despite being over 10× smaller than FLAN (137B vs. 11B parameters).

Both T0 and FLAN underperform GPT-3 on Winogrande and HellaSwag [53, 54], for which Wei et al. [44] conjecture that for tasks such as coreference resolution that can be formatted as finishing an incomplete sentence, adding task instructions to prompts is "largely redundant". Following this conjecture, we reevaluate these two datasets without instructions as done by Wei et al. [44] and Brown et al. [23] and find that it improves performance on HellaSwag from a median of 33.65% to 57.93%, matching the performance of FLAN. For Winogrande, however, using FLAN's prompt without instructions does not make a substantial difference (accuracy = 62.15%).

Surprisingly, Wei et al. [44] perform an ablation with a model of comparable size (8B parameters) to T0 (11B parameters) and find that that performance on held-out tasks *decreases* after multitask prompted training, whereas we find that multitask prompted training improves the performance of models at least as small as 3B parameters (Figure A.1). We identify two key differences between the models that could explain this discrepancy: First, we use an encoder-decoder model that was pretrained with a different objective (masked language modeling) before being trained as a standard language model and finally fine-tuned on the multitask mixture. We note that masked language modeling has repeatedly been shown to be a dramatically more effective pre-training strategy [32, 55, 56].

Second, our prompts are qualitatively more diverse in terms of their length and creativity (§2.1.2). For example, consider one of our prompts for Quora Question Pairs (paraphrasing identification): `I'm an administrator on the website Quora. There are two posts, one that asks "question1" and another that asks "question2". I can merge questions if they are asking the same thing. Can I merge these two questions?` We hypothesize that this diversity could have concrete effects. For example, it could explain why Wei et al. [44] present ablation results where increasing the number of prompts has a negligible impact on performance whereas we observe an improvement when adding more prompts (§2.1.4). We leave a full investigation on the impact of these differences to future work.

2.1.5 Conclusion

We demonstrate that multitask prompted training can enable strong zero-shot generalization abilities in language models. This approach provides an effective alternative to unsupervised language model pretraining, often enabling our T0 model to outperform models many times its size. We also perform ablation studies demonstrating the importance of including many diverse prompts and the impact of increasing the number of datasets in each task. To enable future work on improving zero-shot generalization, we release all models trained in this section in addition to the collection of prompts we created and our prompt annotation tool.

2.2 Converting Supervised Datasets into Promptable Format

This section presents the paper "Promptsource: An Integrated Development Environment And Repository For Natural Language Prompts" [2].

Prompt engineering is emerging as a new focus in NLP, particularly in zero- and few-shot learning settings. *Prompting* is the practice of representing a task as a natural language utterance in order to query a language model for a response [57]. For example, if a language model is conditioned on the text *"She hit a home run. The previous sentence is about ..."*, then the model's subsequent generation would be interpreted as a prediction of the topic of the preceding sentence, e.g. by mapping a response such as *"sports"* to a class label. In specific contexts, prompting has been shown to have advantages over traditional classification, for example facilitating adaptation of language models to ad-hoc tasks and improving sample efficiency in low-data settings [13, 58–60]. These advantages motivate a practical challenge: *How can we enable users to create, refine, and share prompts?*

The process of prompt engineering is critical for successful deployment as choices in prompting can affect downstream predictions significantly, particularly in the zero-shot setting [61–63]. Furthermore, training directly on collections of prompts can enable large models to generalize to new prompts more robustly [64–67]. There is therefore a growing need for tools that support the creation of corpora of prompts.

PromptSource is an integrated development environment and repository for natural language prompts to use in the context of zero-shot (or gradient-based few-shot) learning. It provides a Web-based GUI that enables developers to write prompts in a templating language and immediately view their outputs on different examples. The system is integrated with the HuggingFace Datasets library [68], so that users can load any dataset automatically, browse existing prompts, and create new ones. Through the course of writing thousands of prompts, we converged on three key aspects to the design of *PromptSource*:

- **Flexible Templating Language.** We adapt a templating language to represent prompts. Prompt authors can define prompts in terms of dataset fields, hard-coded text, and simple control logic. This choice provides the flexibility of a programming environment without the mental overhead of having to write and read arbitrary code. Prompt templates can easily be distributed and used in other systems.

- **Tools for Prompt Management.** *PromptSource* has multiple view to address the needs of prompt authors at different stages of the prompt engineering cycle. A global view lets authors browse datasets and existing prompt templates. A local view facilitates iteration on prompt wording and metadata, as well as testing on individual examples.

- **Community-Driven Quality Standards.** *PromptSource* includes a set of guidelines for prompting based on a large-scale prompt writing pilot. *PromptSource*'s collection is meant to be useful for a wide range of research, based on iterative refinement of a set of quality standards. Prompts in *PromptSource* are also annotated with various pieces of metadata to make finding and using prompts easier.

The *PromptSource* system includes over 2,000 open-source prompts for roughly 170 datasets, which have all been reviewed to meet the quality standards. This collection, which we call the Public Pool of Prompts (P3), allows users to materialize prompted forms of datasets for hundreds of different tasks. The T0 series of models [64] for zero-shot inference were fine-tuned on a subset of P3. Since then, *PromptSource* and P3 have been extended for research on multi-lingual prompting [69] and priming, i.e., in-context few-shot learning [66]. The *PromptSource* system and associated content is a first step in the study of systems for prompt engineering, an area that is likely to continue to grow.

2.2.1 Background and Related Work

PromptSource builds on recent work in prompting and prompt engineering. It is also related to work on systems for other types of annotations.

Prompting Recently, prompting has emerged as a new focus within NLP as it can dramatically improve language models' few-shot and zero-shot performance in a wide range of downstream tasks [13, 64, 65, 70]. Prompts and prompt engineering come in several varieties [57]. *PromptSource* is focused on facilitating research with human-written prompts, in which natural language is the medium for describing tasks. This approach has the advantage that prompts can be understood, modified, and applied without being tied to a specific model. In contrast, past work has also aimed to automatically construct prompts by framing the search for a good prompt as a learning problem. These prompts can either be expressed in natural language [60, 71] or as arbitrary vectors (a.k.a. "continuous" or "soft" prompts) not corresponding to words in the model's original vocabulary [72, 73]

When using human-written prompts, there are several possible approaches to learning. One is a zero-shot setting, where the goal is to generalize to prompts for which no training examples are given. Prompts can also be used in a few-shot setting, in which a model is either (1) trained on prompted examples of the target task via gradient updates, or (2) priming (i.e. in-context learning), in which labeled examples are included in an input sequence in order to prime models to make predictions without gradient updates [13].

PromptSource was originally designed for zero-shot learning, so it emphasizes explicit task instructions and no priming examples. If needed, users can extend *PromptSource* for few-shot learning (e.g., as done in 69 and 66, described in §2.2.6).

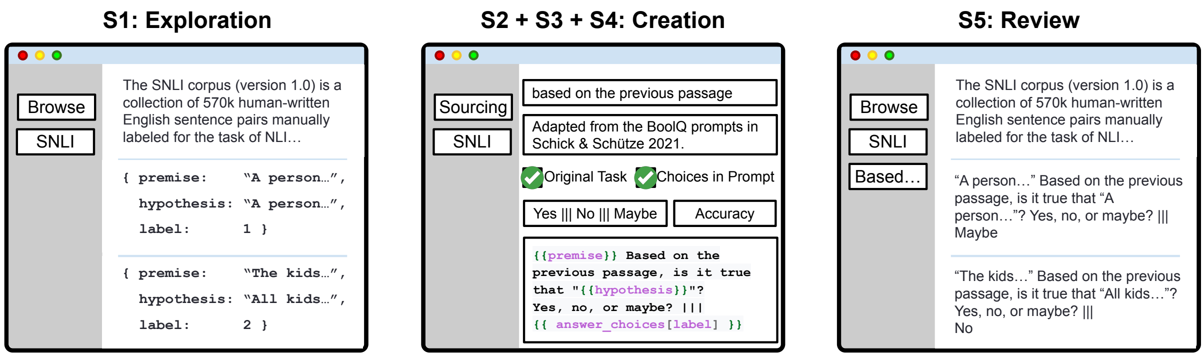


Figure 2.8: The five stages of creating prompts in *PromptSource*. The Browse view for Dataset Exploration (S1). The Sourcing view for Prompt Writing (S2), Prompt Documentation (S3), and Iteration and Variation (S4). The Browse view for performing a Global Review (S5).

Systems for Annotating Data Most work on collecting annotations has focused on labels and other annotations at the level of individual examples [74]. GATE [75] was an early system for annotating text, and includes support for many data types such as labels and entity tags. Since then, many Web-based systems for annotating text have been developed [76–82]. Other systems support collaboration among multiple annotators [83, 84]. More recently, many annotation systems have begun to incorporate learned models to improve workflow, using techniques such as active learning [85, 86] and example recommendation [87, 88]. These systems are possible because the annotations to be collected are labels, for which metrics like inter-annotator agreement and model confidence are available.

There has also been some work on collecting annotations other than labels. AlvisAE [89] and TreeAnnotator [90] support creating ontologies and other structured annotations. Prompts differ from these annotations in that they are semi-structured functions, requiring new tools for developers.

2.2.2 System Design and Workflow

Creating prompts differs from other types of data collection and annotation. We focus on three challenging aspects on which prompting differs from traditional NLP annotation:

- **Functions, not Labels.** A single prompt is a function that maps dataset examples (dictionaries of arbitrary fields) to natural language input/target pairs. Creating a prompt is therefore more like programming than typical data annotation. How should a prompt format trade off between expressivity and simplicity?
- **Dataset-Level Choices.** Prompts are associated with datasets, unlike label annotations that are local to single examples. Prompt engineering requires developers to evaluate their choices across all examples. What interfaces do authors need to inspect and debug their prompts?
- **Variation in Prompt Construction.** Unlike with labels, it is often desirable to have variation within prompt construction, as different prompt choices may lead to different results. However, variation complicates quality judgment, and makes it impossible to apply simple metrics like inter-annotator agreement. How can multiple authors collaborate to build a high-quality corpus of prompts and associated metadata?

To illustrate these distinct aspects, we start with a concrete overview of the prompt creation process of *PromptSource*. For this example, we imagine that a user of *PromptSource* is creating prompts for a natural language inference dataset, specifically SNLI [91]. The goal is to design a prompt query such that the answer can be mapped onto the SNLI classes. A prompt author can accomplish this goal with *PromptSource* via the following five steps (Figure 2.8):

S1: Dataset Exploration The prompt author starts in the *Browse* view to read the dataset description, including linked READMEs and papers, and to browse through examples. In this case, they would see that SNLI is a dataset for natural language inference: assume a given premise sentence is true, the goal is to determine whether a hypothesis sentence is true (entailment), false (contradiction), or undetermined (neutral).

S2: Prompt Writing The prompt author uses the *Sourcing* view to try out a prompt wording, and then adjusts it by observing prompted examples (Figure 2.8 middle, full example in Figures 2.10 and 2.11).

S3: Prompt Documentation To facilitate using the prompt, the author fills in various metadata including possible metrics to evaluate the prompt, valid outputs if applicable, whether the prompt expresses the original intended task of the dataset, and whether the template explicitly states the valid outputs.

S4: Iteration and Variation The prompt author then iterates through S2 and S3 to create multiple prompts for the dataset. Authors are encouraged to vary multiple factors such as the formulation of the prompt and the targeted task (see Section 2.2.5).

S5: Global Review The author saves the draft prompts in a structured file which are then verified by other contributors through code reviews. New prompts need to meet the quality standard with a series of automatic tests and by validation through prompted instances. Upon passing review, the new prompts can be merged into a global prompts collection.

Upon submission, prompts can be viewed through *PromptSource* by other users. The full collection is stored globally and can be used outside of the tool, for instance to be applied on an example from a dataset of the *Datasets* library [68].

```
from promptsource.templates import DatasetTemplates
from datasets import load_dataset

prompts = DatasetTemplates(''snli'')
prompt_key = ''based on the previous passage''
p = prompts[prompt_key]

dataset = load_dataset(''snli'', split=''train'')
example = dataset[0]

result = p.apply(example)
print(''INPUT: '', result[0])
print(''TARGET: '', result[1])
```

With this workflow in mind, we next describe the key aspects of the *PromptSource* system in greater detail.

2.2.3 Prompting Language

A key design decision is the format for prompts. Previous works on prompting tended to use code for specifying each prompt. We experimented with this format and found a trade-off between expressivity and explicit structure. On one side, a maximally expressive format such as pure Python code would let users write complex programs to manipulate the semi-structured examples into prompted examples. However, interpreting and analyzing these programs becomes difficult. This difficulty limits downstream manipulation and analysis of the prompts, for example for possible future work on automatic prompt augmentation. On the other side, a maximally structured format, such as rule-based generation, limits the kinds of prompts that users can create. We found it infeasible to enumerate types of rules sufficient for the wide range of tasks and data formats for which we wanted prompts.

We therefore settled on a middle ground between the two: a templating language. Specifically, we use the Jinja2 templating engine,[5] originally designed for producing web markup. Users write templates as prompts with placeholders, such as `If {{premise}} is true, is it also true that {{hypothesis}}? ||| {{entailed}}`. The separator `|||` denotes the break between the conditioning text and the desired completion. Placeholders refer to fields in the underlying example (represented as a Python `dict` by *Datasets* [68]). Users also have access to Jinja's built-in functions, such as manipulating strings and structured data. For each prompt, prompted examples are created by applying the prompt to all examples in the corresponding dataset. While Jinja is a complete programming language, our review guidelines encourage simple functions with minimal additional logic (see Figure 2.10 and 2.11 for example).

During the development of *PromptSource*, we found that a few idioms were particularly useful. First, not all templates are applicable to all examples in a dataset. Users can wrap templates in Jinja's built-in conditional statements, and any example that results in an empty prompted example is simply skipped. Second, many examples can be used to make multiple training instances, such as a question that has multiple valid answers. We therefore added a `choice` function that selects an element from a list in a way that can be controlled during dataset generation, such as picking a random element using a seeded random number generator or generating different prompts for each combination of elements in the template. Third, many tasks such as classification and binary question answering have a small set of possible valid completions, and it is common to make predictions for these tasks by scoring only the valid completions and returning the highest one [13, 64, 65]. Users therefore can list the valid completions in a separate field and access

[5] `https://jinja.palletsprojects.com`

Figure 2.9: Prompt creators can browse through the dataset examples (left-column) and their prompted form (right column) using the *Browse* view.

them as a list in their prompts (displayed as `Answer choices` in Figure 2.10). These completions are then explicitly available when evaluating predictions for these prompted examples.

2.2.4 The *PromptSource* UI

The *PromptSource* system is designed to enable prompt creators to view data (S1), write prompts in a standard format (S2, S3, and S4), and verify that their templates work correctly (S5). We implemented a lightweight interface for the tool in Streamlit[6] so that users could download, run locally in a web browser, and then upload their results to a central repository. Testing iterations of the interface on pilot template-writing tasks, we converged on three views for the interface.

V1: Browse This view (Figure 2.9) lets users inspect datasets before creating prompts (S1). Once prompts are created, they can select prompts and browse the prompted examples generated by them (S5). The original example is viewed side-by-side with the resulting prompted example, with the substituted text highlighted to distinguish from text hard-coded in the template. Users can quickly scroll through many examples, verify the behavior of their prompt, and return to the sourcing view if changes are needed.

V2: Sourcing This view (Figures 2.10 and 2.11) allows users to select a dataset to prompt, browse examples from that dataset in the form of tables, and enter a prompt for that dataset. As the user writes their template (S2, S3, and S4), every time they save it, the output of the template applied to the current example is displayed next to the editor. We also collect metadata like a name for the template, and a reference for any bibliographic information or rationale for the template.

V3: Helicopter This view (Figure 2.12) allows users to see what datasets are available for writing templates and how many are written for each, to prioritize user attention. This view is particularly useful for moving between datasets and for the prompt reviewers (S5).

[6] `https://streamlit.io/`

2.2.5 Community Guidelines and Process

Due to the variety of existing NLP datasets, we found it challenging to exhaustively describe the characteristics of a good prompt: there are no simple metrics like inter-annotator agreement on example-level labels. Instead, over a few iterations, we converged on community guidelines[7] with three objectives in mind: (a) provide a standardized vocabulary for discussing prompts between prompt authors, reviewers and users, and minimum requirements for a valid prompt, (b) highlight common errors and best practices, (c) collect the necessary information about the prompts to support current and future research on prompt engineering. The guidelines were enforced in the use of *PromptSource* by a code review process in which each prompt was reviewed before being committed to the central repository.

Guidelines apply to the combination of a template (a function that maps an example into an input/target pair in natural language) and a set of metadata about the template. The most important constraint we imposed for a template to be valid is that it is formulated in natural language (both for the input and the target). We forbid the use of non-natural language prompts such as pure code. Each prompt should clearly state what task should be solved, in a way a non-specialist adult can understand. We found this guideline strikes a good balance between freedom and expressivity in the wording of the prompts on one side and short generic prompts on the other side.

In early experiments, we found that user-written prompts that did not explicitly state the possible valid completions tended to perform worse in experiments than their counterparts in which the possible valid completions were listed. We encouraged prompt authors to explicitly state the valid outputs in some of their prompts. In addition, when working with training prompts that include target text, we found it useful to remove variations on the target format that led to spurious ambiguity. For instance, the target template should only contain the answer to the task. It should not contain any extra text such as “The answer is ...”, which can be equivalently moved to the input template.

One of the research question we hope to enable with *PromptSource* is whether the diversity of the prompt formulation during training leads to models that are more robust to the prompt formulation at test time. Therefore, we encouraged prompt authors to create between 5 and 10 (or more) prompts per dataset while varying the prompt formulation. For a given dataset, authors produce multiple prompts per example, sometimes for task formulations that differed from the original dataset. For instance, for question answering dataset, one prompt can ask to extract the answer to a given question from a given passage, while a second prompt can ask to generate a potential question given an answer and a passage.

As part of the community process and to facilitate future research, *PromptSource* asks prompt authors to include additional metadata for each prompt. Metadata fields include a name for the prompt, a reference to the paper it was extracted from (or any relevant explanation), whether the prompt expresses the task originally intended by the dataset, the valid outputs (if relevant), whether the input template states the valid outputs, and possible metrics to evaluate the prompted examples. These can be used in future systems to evaluate how the style and structure of prompts leads to different downstream results.

2.2.6 Case Studies

A system for creating, maintaining, and using prompts is a key tool for supporting the emerging research area of prompting in a standardized and reproducible manner. We highlight three recent research projects for

[7] Complete guidelines can be found at `https://github.com/bigscience-workshop/promptsource/blob/main/CONTRIBUTING.md`.

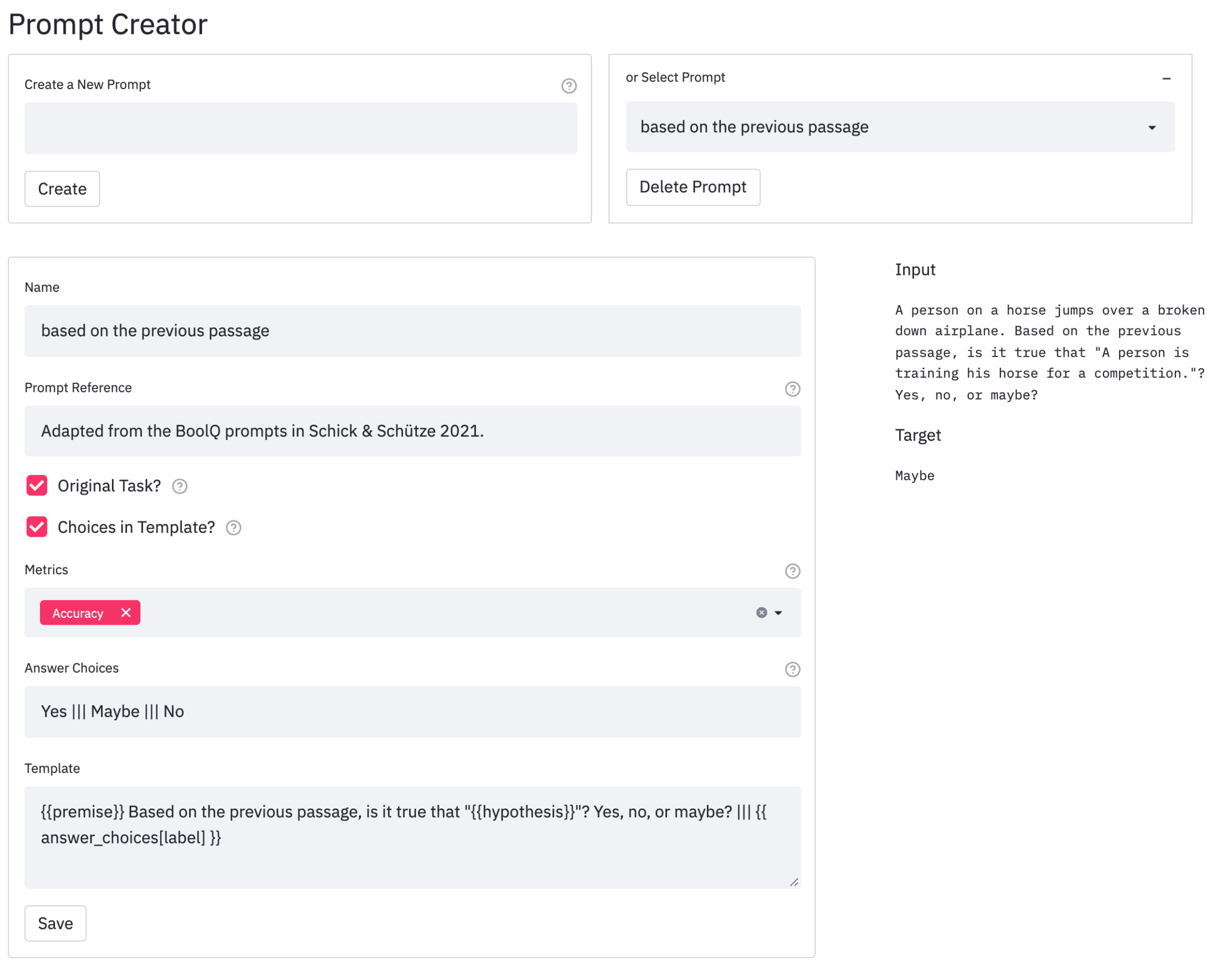


Figure 2.10: With the *Sourcing* view, prompt authors can write new prompts, fill in the associated metadata, observe the result on examples, and iterate.

which *PromptSource* was a key resource.

Massively multitask prompted training Sanh et al. [64] study the question of zero-shot behaviors in large language models and ask whether zero-shot generalization can be induced by training a language model on a massively multitask mixture of prompts. To test this question, they use *PromptSource* to create diverse prompts for a large collection of NLP datasets. Their training and evaluation prompts are a subset of P3. This work demonstrates that *PromptSource* allows training a language model on a massively multitask mixture of prompted datasets and evaluating the ability of models trained with such a procedure to perform unseen tasks.

Multilingual prompting Lin et al. [69] study the zero- and few-shot learning abilities of an multilingual autoregressive language model trained on 30 languages. In particular, they are interested in the cross-lingual generalization of such models and benchmark a variety of tasks in multiple languages. *PromptSource* allows using a massive set of high-quality English prompts. Moreover, the English prompts serve as support to create prompts in other languages (through either machine or human translation).

Priming (in-context learning) Min et al. [66] study improving models' few-shot priming performance by first fully training a model (with gradient updates) on a multitask mixture formatted with priming examples. They find that incorporating templates from P3 significantly further improves performance compared to training on priming examples alone. Although *PromptSource* was not originally designed for this specific form of prompting, users were able to easily use P3's template collection and the templating language for

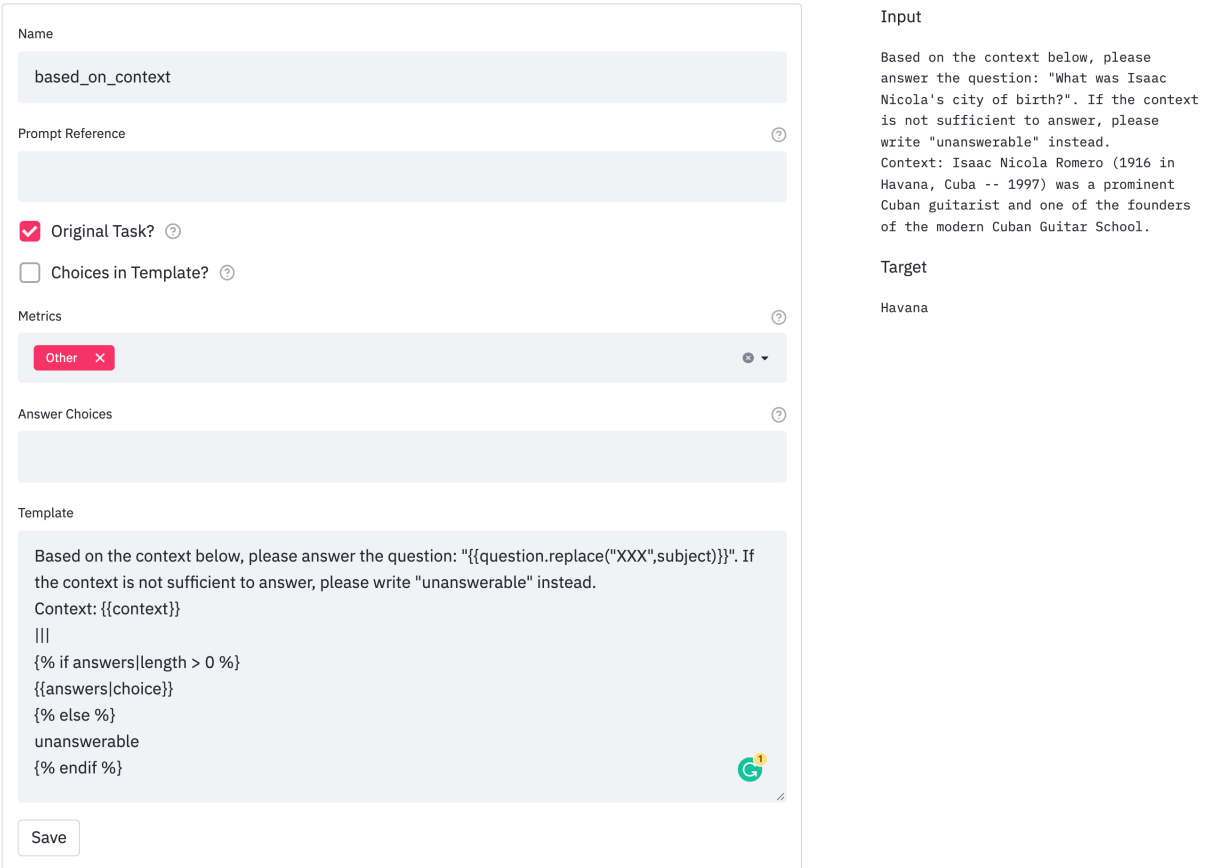


Figure 2.11: Another example of the the *Sourcing* view, focusing on the editor. The templating language strikes a balance between expressivity and explicit structure. This prompt for QA-ZRE [92], a dataset for zero-shot relation extraction, shows how to manipulate strings and do conditional statements with Jinja.

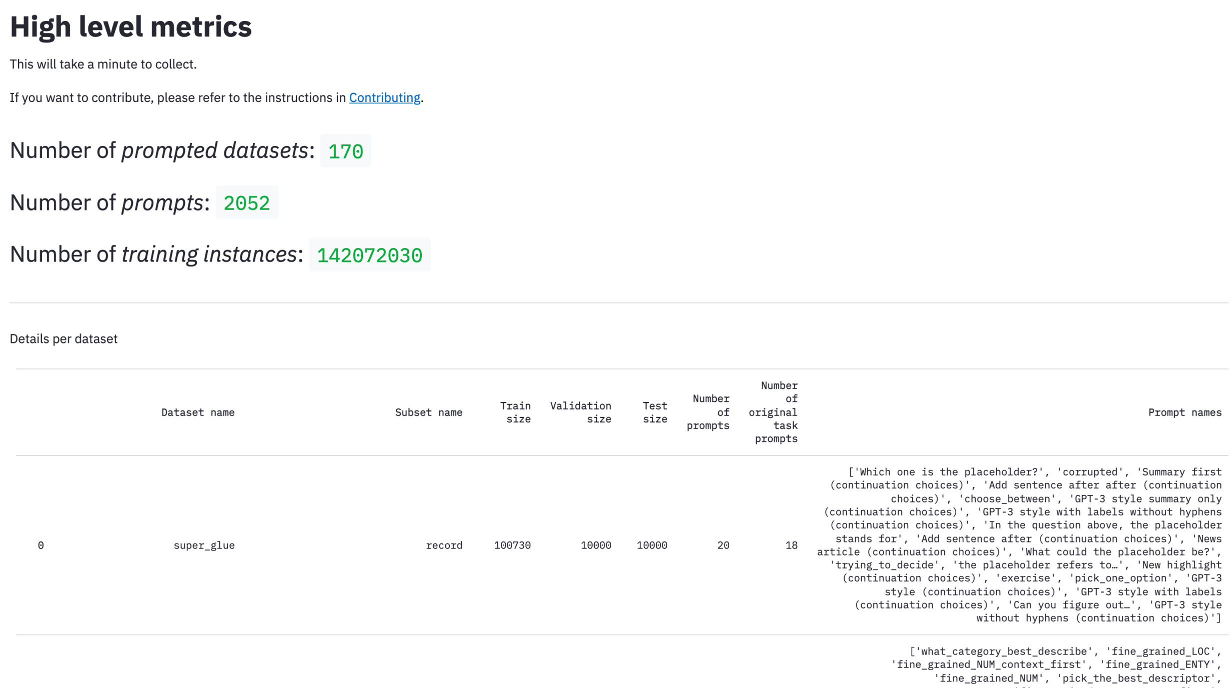


Figure 2.12: The *Helicopter* view indicates what datasets have prompts and how many prompts are available for each dataset.

their own priming methods.

2.2.7 Conclusion

PromptSource is an open-source system for creating, sharing, and using natural language prompts and addresses the need for new collaborative and centralized tools to support the emerging research around prompting. The tool is designed to answer three key needs: a flexible template language, a suite of tools for prompt management, and community-driven quality standards. As of January 2022, *PromptSource* includes a growing collection of 2,000 public prompts for roughly 170 datasets, and has already been an instrumental resource for multiple recent research projects.

2.3 Handling Multimodal Queries in Large Language Models

This section presents the paper "Multimodal Neural Databases"[3].

The amount and variety of data available have increased dramatically in recent years, and as more devices, such as smart glasses, become widespread, this trend is likely to accelerate. While current devices generate mostly text and image data, smart glasses will likely increase the amount of audio and video data individuals create. With the emergence of generative AI, we will likely see an explosion of valuable generated data. Multimedia Information Retrieval (MMIR) has always attracted the attention of scientists and practitioners in Information Retrieval. MMIR aims to address the challenges of processing queries on multimedia collections. Due to the enormous increase in data availability, MMIR has also seen a surge in its interest. The field has explored topics such as retrieval from large image archives, query by image, and retrieval based on face or fingerprint [93]. However, this section brings forward a novel and transformative idea: *given the huge impact that the field of AI is having in all of the areas of technology, we argue that the MMIR field needs to explore systems that can handle more expressive database-like queries called multi-modal neural databases (MMNDBs).*

We illustrate the potential of MMNDBs with an example. Consider the following query over an image archive: *how many images contain musical instruments?* Assume that the images in the collection are labeled with the objects that are identified in them (e.g., trumpet, avocado, person). Hence, an MMIR system is likely to be able to return images with trumpets, or other musical instruments. However, finding which objects are wind instruments (or a more detailed category) requires an additional reasoning step of a join with a database of instruments. Moreover, counting the number of images that satisfy our condition requires reasoning about the size of the answer set, an operation routinely done by database systems but not supported by MMIR systems. Examples can be more complicated, such as finding the most common musical instrument appearing in the photos or considering only photos taken in cities that hosted the Olympic games. As seen from the examples above, one of the critical needs of MMNDBs is the ability to reason about sets.

In this perspective section, we propose to study, design, and build MMNDBs by combining the capabilities of large multimodal models, multi-media information retrieval, and database query processing, as shown in Figure 2.13. We have been inspired by the work on neural databases [94–96] that have garnered interest in the NLP, database, and IR communities. However, we differentiate from that work as we position ourselves as an evolution of the field of MMIR by means of modern and, more recently proposed, multimodal AI technologies.

We develop a first principled prototype to show the proposed task's feasibility. We will later stress that this is only one of the possible architectures to solve MMNDBs and that future research will unveil new strategies. At a high level, we build our prototype on the retriever-reasoner-aggregator model. Given a query, the retriever returns a small subset of documents from the database that is relevant to the query. However, typically even that subset is too big to be provided as input to a single reader, which is essentially a transformer. Hence, the system runs multiple copies of the reasoner in parallel, each producing a partial result for the query. Finally, the aggregator component of the MMNDB will create the query result from the intermediate ones. For example, if the query counts the number of images that contain people, the intermediate results would be 1 or 0, depending on whether the image contains a person. The aggregator will add up the 1s.

MMNDB systems will be designed to handle a wide range of multimedia data, including images, videos, audio, and text. The system will be able to process queries in natural language, allowing users to express

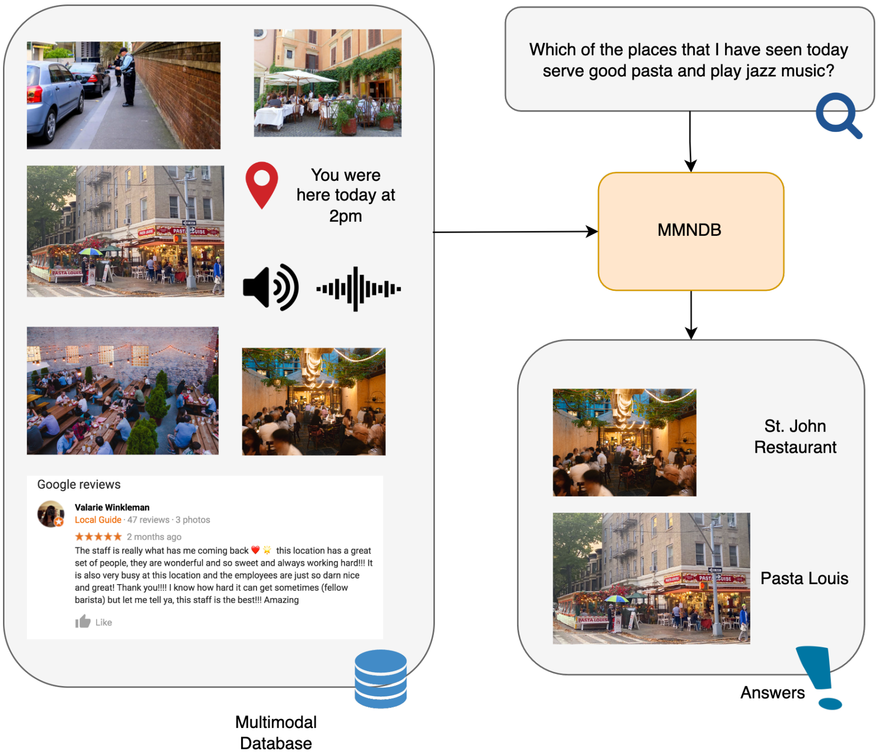


Figure 2.13: A possible use case for MMNDBs. Imagine walking around the city with smart glasses and collecting information in a multimodal database. In the evening, you could be interested in knowing which are good places to eat that satisfy some criteria. MMNDBs could help make that decision by answering a database-like query posed in natural language (or voice!), combining multiple information sources and modalities.

their queries intuitively and easily. The system will also be able to extract features from multimedia data and use them to improve the performance of retrieval tasks.

This section describes a first step towards the realization of MMNDBs flexible enough to scaffold future models. We consider queries over collections of images and validate several aspects of our proposed architecture, as seen in Figure 2.14. We perform a rich set of experiments that show the feasibility and potential of the proposed task across a subset of possible query types. Finally, we discuss possible future research directions stemming from the analysis brought forward in this section and the introduction of Multimodal Neural Databases. We release all code and data on GitHub [8].

2.3.1 Multimodal Neural Databases

We refer to a corpus of *documents* coming from different modalities as a multimodal database. The definition of documents we provide here is intentionally very loose. In general, it could be any self-contained piece of data. Multimodal databases could include wildly different sources. For instance, it could contain information in natural language form, images, sounds, geo-tagging information, a timestamp, and many others. Unlike a traditional database, a multimodal database is unstructured in the sense that it does not need to have a schema, or even less, it does not need to have any particular ordering but can be just the unordered and unstructured set of these documents.

Multimodal databases arise in several contexts. One existing context today is that of online social media, where users post content of different kinds (text, images, memes, videos, audio). Here, each post is a

[8] https://github.com/GiovanniTRA/MultimodalNeuralDatabases

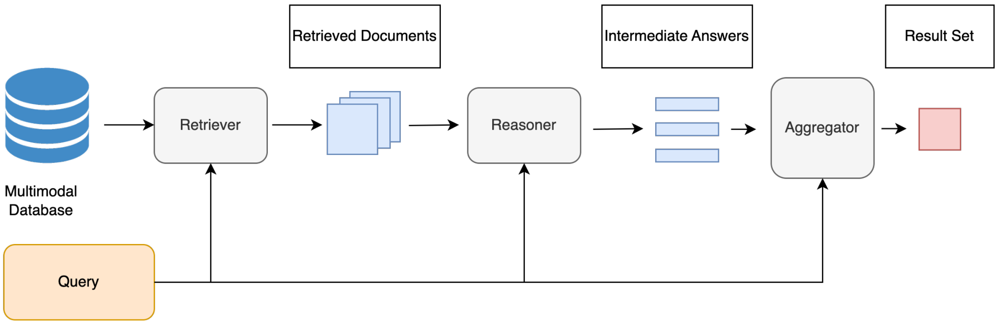


Figure 2.14: Schema for our proposed MMNDB prototype. Given a query, documents are first filtered by a retriever module. A reasoner produces intermediate answers that are then processed by an aggregator to produce the final answer.

document in the multimedia database, with the added peculiarity that the database would have to keep track of the graph of friendships between users. Another context that will arise in the near future when smart glasses are prevalent is the record of a user's day. Just by doing simple activities, like getting a coffee in a bar, the glasses will capture (adhering to whatever privacy conventions get adopted) sensory data, pictures (videos) of who is at the bar and what they are eating, audio of the background track playing, and photos of receipts for one's purchases.

Ideally, we would like to be able to query these rich, large, and unstructured collections of data the same way we query a database. Going further, unlike a standard database, we would like to use natural language to perform queries instead of a rigid language like SQL. Specifically, given a multimodal database D and a query q, we would like to be able to perform the following types of query: (a) **Set queries**; set queries are extractive queries that return a list of spans, such as entities, from the facts. (b) **Boolean queries**; that return either True or False as an answer. (c) **Join queries**; which require the combination of two or more documents to produce each answer.

We note that unlike traditional databases (or even neural databases), Multimodal databases can produce answers consisting of heterogeneous modalities. For instance, a set query can produce answers that include images, audio, and natural language (and their combination) seamlessly.

Designing a Multimodal Neural Database presents several substantial challenges.

First, it is crucial that the system is able to reason on the modalities given in input. For instance, if I were to look for images of cats and dogs fighting, I need to recognize both the presence of these animals *and* that the interactions between the two is indeed that of fighting (a poster of Mike Tyson boxing in the background is not sufficient). Similarly, if the query mentions someone whispering or yelling, the system must understand such subtleties in an audio frame. Recently, deep learning techniques, particularly large deep learning models, have shown excellent reasoning capabilities [97]. The tasks of Visual Question Answering and multi-hop question answering have reached near human results [98] for natural language processing, with promising candidates in the multimodal setting as well.

However, these models are usually extremely large, with billions of parameters, leading to the next challenge, namely scale. Given a large collection of documents, it is infeasible to run such models on every query-document pair, or even on every document for that matter.

Open domain question answering systems (ODQA), developed for answering queries from natural language text, provide a methodology for scaling to larger document collections. ODQA answers a query by first retrieving relevant documents from the document collection and feeding them as context to a transformer along with the query. However, transformers can only accept contexts of limited sizes (currently, 512 to 1024

Model	μF1	μRecall	μPrecision	F1	Recall	Precision
RN50	0.315 ± 0.002	0.819 ± 0.003	0.195 ± 0.002	0.320 ± 0.018	0.731 ± 0.035	0.302 ± 0.026
RN50x4	0.424 ± 0.002	0.794 ± 0.003	0.290 ± 0.002	0.447 ± 0.022	0.717 ± 0.031	0.419 ± 0.027
RN50x16	$\mathbf{0.440 \pm 0.002}$	0.791 ± 0.003	$\mathbf{0.305 \pm 0.002}$	$\mathbf{0.478 \pm 0.023}$	0.710 ± 0.029	$\mathbf{0.457 \pm 0.028}$
RN50x64	0.331 ± 0.002	0.837 ± 0.003	0.206 ± 0.002	0.384 ± 0.019	0.759 ± 0.034	0.343 ± 0.025
RN101	0.344 ± 0.002	0.873 ± 0.003	0.214 ± 0.002	0.388 ± 0.021	0.809 ± 0.028	0.317 ± 0.024
ViT-B/32	0.378 ± 0.002	0.876 ± 0.003	0.241 ± 0.002	0.395 ± 0.018	0.813 ± 0.022	0.298 ± 0.019
ViT-L/14	0.324 ± 0.002	0.931 ± 0.002	0.196 ± 0.001	0.329 ± 0.015	0.894 ± 0.018	0.219 ± 0.013
ViT-L/14@336px	0.337 ± 0.002	$\mathbf{0.932 \pm 0.002}$	0.205 ± 0.002	0.347 ± 0.016	$\mathbf{0.905 \pm 0.015}$	0.228 ± 0.014

Table 2.1: Comparison of different Retriever models under the "Mixed" retrieval strategy. While CLIP's versions featuring resnets as a backbone have higher F1 and precision scores, ViT-based models achieve higher recall. We opt for the latter, as it allows the Reasoner module to receive as much relevant information as possible, ultimately reducing the final pipeline error.

tokens). Even though extending these sizes is a very active area of research, it will always likely be smaller than the size necessary to process the kinds of queries we are striving for. The number of documents that need to be processed for answering database queries can be arbitrarily big, as can the intermediate result sets. In contrast, ODQA systems usually consider queries whose answers are small and can be obtained by feeding just a few documents to the transformer. Furthermore, a multimodal database is an *unordered* set of documents, so we cannot exploit any locality heuristic to retrieve the relevant documents.

Last but definitely not least, there is a challenge of bridging between the different modalities in a multimodal database. To answer queries over multimodal data, one has to process, reason, and combine information coming not only from different documents but also from documents expressed in different modalities. The literature in natural language processing and computer vision has recently paved the way and achieved outstanding results in the field. Multimodal models have followed, showing excellent results in the task of text-to-image, image-to-text, and text-to-music. However, most multimodal models available today tackle either the text-visual or the text-audio tasks. Combining multiple modalities, while not unexplored [99], still needs additional research efforts to reach suitable levels to address the task at hand. In particular, to suitably address the task of MMNDB, we would need a "true" multimodal model, which can reason on any possible modality given as input. For further discussion on this and other current limitations/future research directions, we refer to Section 2.3.3.

A first prototype for MMNDB

To demonstrate the feasibility MMNDBs, this section describes a first prototype of such a system, for a restricted case. We consider databases in which all the documents are images, and queries, which are posed in natural language, can express COUNT, MAX, and IN. However, as we explain below, the architecture for our preliminary system can apply to broader settings as well. We hope that this architecture forms the basis for other approaches to MMNDBs.

Our system takes an input query q over a database D. It includes three components. The first component is the retriever, which selects a subset of the documents in D that are relevant to answer the query. The second component is the reasoner, which processes, possibly in parallel, subsets of the retrieved documents. The reasoner provides a partial answer to the query. The third component is an aggregation operator that synthesizes the answers provided by the reasoner to compute the final answer to the query.

The strength of our architecture is that it enables us to exploit recent advances in multimodal neural models when implementing the retriever and the reasoner. Specifically, these models are able to map multiple modalities into the same embedding space, and therefore reason about the contents of images and text together. For example, these models can identify objects in images and create textual captions that describe the main aspects of the image.

Before we explain each of the components, we give an end-to-end overview of how a query is processed in our system. Consider the query "How many people are playing the guitar in a blue t-shirt on a beach". The reasoner considers a single image in D and uses the latest neural methods to determine whether the image contains a person playing guitar on the beach. However, applying such powerful reasoning on each of the documents in D is infeasible, so we use a retriever to filter to only a small subset of the images in D, $P(D, q)$. Multiple instances of the reasoner then are applied in parallel to the retrieved images in $P(D, q)$ to determine which image satisfies the query. In our example, if an image satisfies the query, the reasoner returns 1 and otherwise 0. The aggregator then counts the number of 1's to return the final answer. We now describe each of the components.

Retriever. The goal of the retriever is to return a subset $P(D, q)$ of documents from D that are relevant to the query q. The main requirement from the retriever is that it be scalable. While the reasoning we expect from the retriever is not at the same granularity as the reasoner, it should weed out the vast majority of irrelevant images. To retrieve documents that are relevant to the query, we encode both the query and the documents in the same latent embedding space. However, as noted earlier, it is important that the embedding of a document *not* be dependent on the query q, otherwise, we would have to compute a new embedding for every document in D for any given query. Hence, as we describe in Section 2.3.2, we consider several methods for embedding the documents in D in a query-independent way.

Reasoner. An instance of the reasoner $P(D, q)$ takes one of the documents in D as input and returns an intermediate answer to our query A_p. In the example above, the reasoner returns either 1 or 0 depending on whether the image satisfies the conditions in the query. However, the intermediate result may be different. For a query such as "What is the maximum number of people in the images" the reasoner would return, for every image, the number of people in that image. As another example, for the query "What is the most common musical instrument seen in the database", the output of the reasoner would be the list and number of occurrences of each of the instruments it identified in the image.

The crucial role of the reasoner is, precisely, to reason about the relationship between the image and the query. In our example, the reasoner needs to determine whether there is a person wearing a blue outfit, that the same person is the one playing the guitar, and that they are physically located on a beach. The reasoner leverages the recent advances in neural models that are able to perform such reasoning by embedding the image and text in the same latent space and generating textual captions of images. It is worth noticing, however, that these models compute a dynamic embedding of the query and of the image, that depends on both, i.e., $F(I|T) \neq F(I)$ and vice versa, where I is the image, and T is natural language (could be any two modalities). This has profound computational implications. In fact, to be able to answer the query, one would need to process any possible D, q pair. Furthermore, since the query is known only at inference time, it is not possible to precompute the embeddings. It is then clearly unfeasible to run the reasoner on the entire database. For this reason, we introduce an additional module in our pipeline, namely the retriever.

Aggregator. The Aggregator takes as input the query and the set of intermediate outputs from all the instances of the reasoners and produces the answer to the query. Conceptually, this component of the system is the simplest because the intermediate results need to be aggregated depending on the semantics of the query. In our example, the aggregator would count the number of images for which 1 was returned. For the query

	Total Error ↓				**Δ Error ↓**			**Accuracy ↑**		
Stock	Error	Error TP	Error FP	Error FN	Δ Error	Δ Error TP	Δ Error FP	Accuracy	Accuracy TP	Accuracy FP
Perfect IR	$\mathbf{0.46 \pm 0.07}$	0.46 ± 0.07	N/A	N/A	4.64 ± 1.94	4.64 ± 1.91	N/A	0.60 ± 0.02	0.60 ± 0.02	N/A
Noisy IR	0.77 ± 0.16	0.46 ± 0.07	$\mathbf{0.31 \pm 0.15}$	N/A	$\mathbf{2.66 \pm 1.04}$	4.64 ± 1.91	$\mathbf{0.31 \pm 0.02}$	$\mathbf{0.81 \pm 0.01}$	0.60 ± 0.02	0.92 ± 0.01
Dmg. IR	1.24 ± 0.32	0.46 ± 0.07	0.78 ± 0.32	N/A	4.22 ± 1.41	4.64 ± 1.91	2.31 ± 1.15	0.70 ± 0.02	0.60 ± 0.02	0.76 ± 0.02
Full	1.27 ± 0.17	$\mathbf{0.42 \pm 0.07}$	0.76 ± 0.13	0.09 ± 0.02	3.33 ± 1.16	4.83 ± 2.03	1.96 ± 0.80	0.73 ± 0.02	$\mathbf{0.61 \pm 0.02}$	0.75 ± 0.02
FTmodel										
Perfect IR	$\mathbf{0.14 \pm 0.01}$	0.14 ± 0.01	N/A	N/A	1.46 ± 0.10	1.46 ± 0.10	N/A	0.67 ± 0.02	0.67 ± 0.02	N/A
Noisy IR	0.22 ± 0.01	0.14 ± 0.01	0.08 ± 0.01	N/A	$\mathbf{0.90 \pm 0.06}$	1.46 ± 0.10	0.43 ± 0.05	$\mathbf{0.86 \pm 0.01}$	0.67 ± 0.02	0.93 ± 0.01
Dmg. IR	0.54 ± 0.05	0.14 ± 0.01	0.40 ± 0.05	N/A	1.25 ± 0.08	1.46 ± 0.10	1.04 ± 0.09	0.73 ± 0.01	0.67 ± 0.02	0.73 ± 0.02
Full	0.99 ± 0.06	$\mathbf{0.11 \pm 0.01}$	0.79 ± 0.06	0.09 ± 0.02	1.10 ± 0.07	$\mathbf{1.42 \pm 0.10}$	0.99 ± 0.07	0.72 ± 0.01	$\mathbf{0.69 \pm 0.02}$	0.72 ± 0.02

Table 2.2: Results performance on the query type count. The PerfectIR setting acts as an ideal upper bound, showing the full potential of the MMNDB framework. The Full Pipeline (Full), on the other hand, shows excellent accuracy and Δ error but a total error that, while being good, is not at the level of PerfectIR. We empirically show that this is not caused by noise introduced by the retriever module, as indicated by the excellent results achieved in the NoisyIR setting. Instead, this is caused by damaging documents picked up by the retriever that trick the reasoner resulting in a large False Positives error and, ultimately, a large total error.

counting the total number of people, the aggregator would sum the intermediate results returned from the reasoners.

2.3.2 Experiments

This section describes the experiments we performed to validate the promise of our prototype. We begin by describing the experimental settings.

Experimental setup

In this section, we outline the experimental setup utilized to verify the validity of our approach.

Dataset Our experiments use the MS-COCO dataset (Common Object in Context) [100], which is the single most popular benchmark dataset in computer vision. We use the latest version made available by the authors. The COCO dataset contains approximately 123K labeled images. Each image is associated with 5 captions and is annotated with the objects that are identified in it. The objects are drawn from a collection of 1.5M object instances across 80 object categories. The dataset is divided into train and eval subsets, containing 118K and 5K images, respectively. We use the train set to train/fine-tune our methods while we report our results on the eval set.

Queries We use the MS-COCO dataset to build our queries. For the COUNT query type, we may ask a query of the type "How many {object} are in the database?", where object can be any of the object categories contained in the COCO dataset. Similarly, for the MAX query type, we may be interested in the image of the dataset with the most frequent annotation of a particular kind. Finally, for the In query, we are interested in images whose annotations satisfy certain conditions.

Models We now describe the neural models we used throughout our experiment.

For the Reasoner, we employ OFA [101]. OFA is a deep learning model trained on a wide variety of multimodal (text and image) tasks, ranging from image captioning to image generation, showing great results on unseen tasks as well. OFA is open-source (code and weights) and is currently one of the best-performing multimodal models. We test four different versions of OFA, namely medium, base, large, and huge, with the largest featuring close to 1B parameters. OFA is a transformer-based model that builds a joint representation of the input, namely text and visual, that is used to generate a textual response. We stress again the fact that, given adequate computational resources, this module of the pipeline is highly parallelizable, hence capable of producing intermediate answers in the span of a few seconds.

For the Retriever, we employed the CLIP model [102]. These models are trained in an unsupervised, contrastive manner by matching captions and images. They take either text or images and align them in a shared latent space that can be used for later inferences and to measure their distance, with similar image-caption pairs being close together. We test on 8 different versions of CLIP, namely RN50, RN101, RN50x4, RN50x16, RN50x64, ViT-B/32, ViT-L/14, ViT-L/14@366px. CLIP's salient feature is that the created embeddings are static, meaning they do not depend on the query. This allows us to pre-compute the embeddings for all images beforehand, meaning that only the embedding for the query has to be computed at inference time. Once the embeddings are computed, a strategy is needed to select which documents are considered relevant (and passed to the reasoner) and which ones are not. To do this, we craft three strategies:
(i) *TopK*: in this case, we compute the dot product between the embeddings of the documents and the query, we sort them, and we select the TopK documents.
(ii) *Threshold*: we compute the cosine similarities between the embeddings of the text and the images, and we return all the documents for which the cosine similarity is greater than a certain threshold τ that depends on the particular CLIP model we are using, lying in a range between 0.15 and 0.4.
(iii) *Neural Selector*: here, we train a small neural network that, given the q and D embeddings, returns a binary outcome that indicates whether the document is relevant for the query or not and whether it should be returned.
The actual number of parameters depends on the CLIP version employed, but it is always in the order of thousands. It is worth noticing that, while it is still much more scalable with respect to the large 1B parameters models the reasoner employs, this strategy requires a "dynamic" processing; namely, the decision on which documents to select relies on a neural model evaluating all q, D pairs.

In a practical system, it is possible to circumvent some of the issues above by borrowing techniques from the literature on online aggregation literature [103]. In practice, we can sort the embedding of the images according to the dot product they have with the query. We then process them in batches of predetermined sizes w. We stop once a specific tolerance criterion is met, namely when no more than c documents are predicted as relevant by the model.

This leads us to our fourth strategy, which we call Mixed. As the name suggests, we mix two of the strategies already introduced, Neural Selector and TopK. Specifically, we take the set union of the TopK (With a small K) and Neural Selector documents to retrieve and to be passed onto the Reasoner.

2.3.3 Results

In this section, we present the experimental evidence to support the ideas presented in this section. First, we will show results that test the performance of single architecture components. Following that, we proceed to evaluate the entirety of our pipeline. Results for all metrics are reported together with their standard error.

We start by evaluating our retriever strategy. We argue that, for our pipeline, a good retriever should have a high level of recall since every relevant document that is failed to be retrieved will produce an error that will propagate to the subsequent components and onto the final response. For this reason, we explicitly express a preference for models and strategies obtaining a high recall. We tested each of the 8 CLIP model

Selection Strategy	μF1	μRecall	μPrecision	F1	Recall	Precision
Top-K	0.211 ± 0.001	0.683 ± 0.004	0.125 ± 0.001	0.201 ± 0.009	0.852 ± 0.018	0.125 ± 0.009
Threshold	$\mathbf{0.351 \pm 0.003}$	0.226 ± 0.003	$\mathbf{0.791 \pm 0.006}$	$\mathbf{0.445 \pm 0.029}$	0.377 ± 0.030	$\mathbf{0.776 \pm 0.022}$
Neural	0.337 ± 0.002	$\mathbf{0.932 \pm 0.002}$	0.205 ± 0.002	0.343 ± 0.016	0.898 ± 0.018	0.235 ± 0.016
Mixed	0.337 ± 0.002	$\mathbf{0.932 \pm 0.002}$	0.205 ± 0.002	0.347 ± 0.016	$\mathbf{0.905 \pm 0.015}$	0.228 ± 0.014

Table 2.3: Comparison among different retrieval strategies. The Threshold strategy achieves higher F1 and precision scores, while the "Mixed" strategy has a higher recall. Once again, we opt for the strategy that achieves higher recall, namely Mixed, as it allows the Reasoner module to receive as much relevant information as possible, ultimately reducing the final pipeline error.

versions on each of the 4 crafted strategies. For the sake of space efficiency, we only show results for the various models in the chosen final setting - mixed strategy - and the comparison between different strategies using the best model - ViT-L/14@366px. In Table 2.1, you can see the performance of the various models in the Mixed Strategy setting. The first thing we can notice is that while there is a shift in scale between μ and macro metrics, at least for precision and recall, the ranking between different models does not really change. Furthermore, While ViT-L/14@366px is the best model neither with respect to F1 nor precision, it is the best model when considering a recall. In fact, it consistently beat other models in that regard, with the exception of its twin ViT-L/14, with which the difference in terms of performance is minimal. Since the difference in the number of parameters and general complexity is almost unnoticeable, too, we saw no reason not to proceed with the former. In Table 2.3, we report results for the 4 retrieving strategies we tested. Once again, while Threshold offers the best precision, the Neural Selector, particularly the Mixed Strategy, offers the best overall results with comparable F1 and much higher Recall. In Table 2.4, we show the difference in performance between the various OFA version we tested. We only show results for the COUNT query type for the sake of not being repetitive since the difference between these models transfers across tasks. In this case, unlike the retriever, we see significant differences in results between the model versions tested. Larger models clearly outperform smaller ones by a wide margin. Moreover, OFA-huge outperforms OFA-large in terms of total error and Δ error, while the latter achieves higher accuracy. We choose OFA-large for two reasons: (i) we favor accuracy over the other two metrics, and (ii) it has half the parameters with respect to the huge version (0.5B vs. 1B). We also report on a finetuned version of OFA-large (OFA-large FT), obtained by finetuning OFA-large on the train set for 10 epochs with a learning rate of $5e-5$ with the same task. Finetuning the OFA model significantly boosts its performance on the MMNDB task.

Model	Total Error ↓	Δ Error ↓	Accuracy ↑
OFA-base	0.831 ± 0.024	2.876 ± 0.217	0.094 ± 0.014
OFA-medium	0.871 ± 0.013	2.869 ± 0.180	0.074 ± 0.005
OFA-large	0.460 ± 0.073	4.645 ± 1.944	$\mathbf{0.597 \pm 0.022}$
OFA-huge	$\mathbf{0.392 \pm 0.025}$	$\mathbf{2.363 \pm 0.179}$	0.533 ± 0.023
OFA-large FT	$\mathbf{0.138 \pm 0.011}$	$\mathbf{1.455 \pm 0.100}$	$\mathbf{0.668 \pm 0.018}$

Table 2.4: We test different neural models to be used as the building block for the reasoner on the PerfectIR setting. Smaller models clearly fail to compete with their larger counterparts. OFA-huge achieves a smaller total and Δ error, while OFA-large has higher accuracy. We choose the latter as we favor accuracy over the other metrics and because it has half the amount of parameters. We also report on a finetuned version that significantly improves over the stock versions.

The metrics tracked, though spun off, are the same as in the test whose results are reported in Table 2.2. Here, we test both the reasoner capabilities and the full pipeline. We perform our testing under 4 different scenarios, considering both the case in which we have a stock model or a finetuned one, reporting on 10 different metrics. We use the PerfectIR setting as a baseline. In this setting, the set of documents retrieved D_r is the set of documents that are actually relevant, taken directly from the ground truth. This, of course, is an ideal setting in which we assume a perfect retriever and acts as a sort of upper bound for our method. Full pipeline instead refers to our actual setting, in which our mixed strategy retriever passes the set of retrieved documents. The metrics we collect are of two kinds: one, with the word total as antecedent, refers to the whole pipeline; the others, without the word total in them, are meant as a test on the intermediate answers A_p produced by the reasoner. In particular, By accuracy, we mean the percentage of intermediate answers A_{p_i} that are exactly equal to their ground truth value. This is then averaged over all queries. We then further divide this computation into two disjoint sets, namely, accuracy for true positives (TP), documents in D_r that are actually relevant, and accuracy on false positives (FP), documents in D_r that should not have been retrieved. Please note that in the case of PerfectIR, the set of FP documents is empty by definition. Since the task at hand is that of the query type COUNT, we are also interested in knowing of close an intermediate answer is to the ground truth value. We track this with the metric Δ error. Here, similarly to the accuracy metric, we register the mean absolute deviation between the intermediate answer A_{p_i} and the ground truth, averaged over all queries. Once again, we spun this off into its two components, namely TP and FP.

Stock	**Total Error** $\downarrow$	Δ **Error** $\downarrow$	**Accuracy** $\uparrow$
Perfect IR	$\mathbf{2.845 \pm 1.759}$	$\mathbf{29.263 \pm 17.598}$	0.188 ± 0.044
Noisy IR	4.576 ± 2.486	41.438 ± 21.343	0.200 ± 0.045
Dmg. IR	4.258 ± 2.035	53.325 ± 23.933	0.188 ± 0.044
Full	4.280 ± 2.014	53.063 ± 24.027	$\mathbf{0.213 \pm 0.046}$
FTmodel			
Perfect IR	$\mathbf{0.229 \pm 0.035}$	1.813 ± 0.271	$\mathbf{0.575 \pm 0.056}$
Noisy IR	$\mathbf{0.229 \pm 0.035}$	$\mathbf{1.800 \pm 0.273}$	0.550 ± 0.055
Dmg. IR	0.303 ± 0.060	2.100 ± 0.320	0.525 ± 0.056
Full	0.317 ± 0.056	2.263 ± 0.342	0.563 ± 0.055

Table 2.5: Results for the query type MAX. It can be immediately noticed how much the finetuning process improves the performance of the MAX query type. In particular, we notice that finetuned models are less prone to produce indecisive intermediate answers such as “many” and “a lot”, which are highly relevant to this query. We also notice how close the Full Pipeline setting is to PerfectIR compared to other queries. We argue this is due to the reduced impact of damaging documents, i.e., it is unlikely that a damaging document will be a likely candidate for MAX.

Under these two metrics, we can see that the Full Pipeline results are competitive, if not better, with the PerfectIR version. Upon further inspection, we can also deduct the cause. In fact, in Full Pipeline, false positive documents are added to the computations. Many of these documents are actually easier to deal with since they do not contain the object of interest and can produce an intermediate answer of 0, raising both the accuracy and the Δ error of the Full Pipeline version. In our experiments, we also noticed that the stock model was struggling to produce useful intermediate results in some instances. For instance, the model would produce indecisive answers like “many” and “few”. Using some prompt engineering, explicitly asking the model to “Answer with a number” alleviated the problem but did not totally eradicate it. For this reason, as mentioned earlier, we produced a finetuned version of the reasoner, which improves the accuracy score and dramatically reduces the Δ error. Finally, we report results on the total error metric. Under this metric, we consider the final outcome of the pipeline o, and we compute its absolute deviation from the ground truth, averaged over all queries, and normalized by cardinality. The PerfectIR version achieves excellent results for this task, fully demonstrating the feasibility of the task we propose in this section. Full Pipeline, while

Stock Model	Total Error ↓	Accuracy ↑
Perfect IR	$\mathbf{0.131 \pm 0.014}$	0.869 ± 0.014
Noisy IR	0.404 ± 0.176	$\mathbf{0.906 \pm 0.007}$
Damaging IR	0.829 ± 0.357	0.811 ± 0.013
Full	0.793 ± 0.150	0.672 ± 0.018
FTmodel		
Perfect IR	$\mathbf{0.060 \pm 0.007}$	0.940 ± 0.007
Noisy IR	0.085 ± 0.007	$\mathbf{0.946 \pm 0.004}$
Damaging IR	0.436 ± 0.054	0.838 ± 0.008
Full	0.330 ± 0.015	0.877 ± 0.007

Table 2.6: Results for the query type IN. Once again, we observe a gap in performance for the fine-tuned models. In particular, the fine-tuned version produces answers that are much more robust to noise. Moreover, while results are generally satisfactory, we observe an increase in error for the Full Pipeline. We attribute this to damaging documents that trick the reasoner into mispredicting the presence of an object, as evidenced by the high loss for the DamagingIR setting.

achieving good scores, lags behind the PerfectIR setting. To further investigate this difference in performance, we divide the total error into its components. Once again, TP refers to documents correctly retrieved, FP to documents wrongly retrieved, and false negatives (FN) to documents that should have been retrieved but have not (These last two components are null in the case of PerfectIR by definition). We notice how the total error TP is actually comparable between the two versions, slightly lower in the case of Full Pipeline since a few of the more challenging documents are not retrieved. Upon further inspection, we notice that the total error FN is almost negligible, meaning that the gap in total error is not caused by documents not being retrieved. From the experimental evidence, it is clear that this gap is actually caused by false positives, documents that should not have been retrieved, but they were, nonetheless. To further investigate this phenomenon, we devise an additional setting called NoisyIR. In this setting, we assume D_r is composed, as in PerfectIR, of the set of relevant documents to which we add, however, some non-relevant documents (300) taken at random. We notice that the NoisyIR setting performs only slightly worse than the PerfectIR setting, showing that our model is actually robust to noise.

Following this experiment, we devised a new setting, identical to NoisyIR, but in which the negative documents are not taken at random anymore. In fact, we take the non-relevant document whose CLIP embedding with the query is the highest. We call this setting DamagingIR. Results clearly show that these documents are able to ”trick” the reasoner into generating wrong intermediate answers, causing a large FP error and ultimately a more significant total error resulting in a performance difference between the PerfectIR version and the Full Pipeline one.

DamagingIR has already been observed by [96] and, to the best of the authors' knowledge, has not been yet fully addressed. At the end of this Section, we provide a complete commentary on this issue.

In Table 2.6, we show results for the IN query type. This query answers questions of the type "In how many pictures there are {object}?". We consider two metrics in this scenario that mirror the ones defined for the COUNT setting. First, we consider accuracy, that is, the percentage of time the intermediate results A_{p_i} are exactly equal to their respective ground truths. The total error indicates the absolute deviation of the total number of documents found satisfying the condition from its ground truth, later averaged over all queries and normalized by cardinality. We can immediately notice that the finetuned version of the reasoner generally performs better with respect to its stock counterpart. We also notice the positive results obtained by the Full Pipeline, even though they are lower than the near-perfect PerfectIR. Once again, even more clearly than before, we can attribute this reduction in performance to DamagingIR, that is, to false positive documents that manage to "trick" the model into thinking that there is an object in the image when there is really not, as evidence by the drop in performance observed under this regime. Finally, we report results for the MAX query type, which return the document with the max instances of a particular object in the collection. We test on the same 4 scenarios and report on three metrics. Δ and total error are specular to previous settings, while total accuracy is the percentage of queries in which the correct document is found. This is the scenario that shows the most significant difference between the stock reasoner and its finetuned version. We attribute this gap to an issue cited earlier, in which for pictures with high instances of a particular object, the model would produce indecisive answers like "many", a problem that the finetuned model does not feature. Furthermore, we notice that the difference between the PerfectIR and the Full Pipeline version is rather small. This stems from the fact that, unlike in the two other scenarios, false positives documents are unlikely to be appetible candidates for the MAX type of query, failing to impact the final outcome. We also register that, even when the model is not able to retrieve the correct max document, the picture found has a comparable number of instances, as indicated by the total error.

Overall, the results are very promising and fully show the potential for Multimodal Neural Databases. We managed to build an effective and efficient retrieval system with a high recall. The reasoner module, and the pipeline as a whole, show good performance and resistance to noise, with low error and high accuracy, coupled with a resistance to noise. However, like other systems in IR, it is weak to DamagingIR, as shown by the increased caused false positives. We argue that by tackling this issue we can further increase the performance of MMNDB and bring it close to the optimum.

Future Research Directions

The introduction of Multimodal Neural Databases paves the way toward new and exciting research directions; in this section, we proceed to discuss some of the more interesting ones.

In this section, we have shown the feasibility of the proposed task but have yet to explore many open problems.

First and foremost, a key feature in any database system is the ability to update its information. In a typical database system, one would expect to be able to remove, add, or modify the information as he wishes. This is not straightforward under our current paradigm and needs more research efforts.

On this line, it would be crucial to account for the importance of time in databases. I could ask the database question like "What is the place I visited the most between 1 pm and 3 pm this year?" Furthermore, we have restricted ourselves to only two modalities, and in particular, a database made of strictly images. Expanding available modalities is a clear path with obvious benefits. Additionally, we could consider not only documents but documents and their meta-data. To provide an example, whenever we take a picture with our smartphone, we collect a variety of information, such as the location and time, which would definitely be helpful for a database of this kind. To remain in the field of smartphones, recently, video-clip sharing has become very popular among social network users. Asking database-like queries on videos is an open problem that presents many challenges. Among all, it is crucial to be able to identify entities along frames to be able to answer queries effectively. While recognizing an entity (like a person) is generally feasible for text, it is much more complex when considering different modalities. Solving this will be critical for the development of

MMNDBs. In our presentation, we stressed the fact that the proposed architecture is not the only possible way of solving this problem. In fact, recently, we have witnessed the power of large foundational models to solve a wide array of tasks, with chatGPT and GPT-x models, in general, leading the way [12]. We believe that these large foundational models could bring an advance to this field as well. However, this is not straightforward, and some issues should be addressed. These models require a large amount of data to be pre-trained; this begs the question of how one could encapsulate the memory used during training from the actual Multimodal Database to avoid knowledge contamination. By knowledge contamination, we mean the known phenomenon for which data used during pretraining is spilled when generating answers in a completely unrelated context. Knowledge contamination proved troublesome in many applications, with some systems allegedly revealing private keys or even personal phone numbers. Furthermore, true multimodality in these large models remains an open research direction and a major roadblock toward conversational multimodal systems. Finally, we have taken Multimodal Neural Database in its most general setting. However, one might be interested in specific scenarios with more precise guidelines and goals. For instance, there may be cases in which one has a precise idea of which kind of queries are to be expected. In that case, strategies could be crafted to optimize the system. In traditional database systems, for example, indexing or creating views for common queries is a prevalent practice. Creating equivalent procedures for MMNDB is still unexplored.

2.3.4 Related Work

Multimedia Information Retrieval (MMIR) Bridging the gap between multimodal unstructured data and structured database systems has always been a central key endeavor in Information Retrieval [104]. The former is vastly highly available on the web but challenging to digest and query compared to the latter. Particular focus has been posed on content-based image retrieval [105–107] and recently on cross-modal retrieval [108, 109], which have been made possible with the recent advancements in deep learning [110]. Specifically, there has been an explosion of such approaches for Image-text retrieval [111–116]. However, these systems are primarily concerned with retrieving relevant documents (e.g., images) based on a given query (e.g., text). In contrast, MMNDBs focus on answering database-like queries on large data collections, which current cross-modal retrieval methods cannot achieve.

Multimodal Neural Models There has been a recent surge in the development of multimodal neural models that can handle data in different forms, primarily images, and text, for various applications. Usually, this is performed via a single neural multimodal encoder [101, 117–120] or via different encoders for each modality that is jointly aligned via a shared space [102, 121]. In MMNDBs, we take advantage of this characteristic by using a separate encoder system as a Retriever to precompute and index visual tokens, thus reducing computation and time at runtime by only using the text encoder to compute the textual embedding of the query. However, directly applying these neural models to the MMNDB task would not be scalable due to the high computational cost. We use them as components in our architecture, building on their successes in other vision-language tasks.

Visual Question Answering (VQA) Most of these multimodal vision-text models are evaluated on the task of visual question answering [97], where the goal is to generate an accurate and semantically coherent response based on a question about an image. Usually, these involve using reasoning and other capacities that are non-trivial, even for current neural architectures. Compared to the task of MMNDBs, VQA is defined on a single image-question pair and requires reasoning over the image to answer the question. Closer to the task of MMNDBs, is Open-domain Question Answering (OpenQA) [122] and the multimodal variant WebQA [123] which aim to answer natural language questions over large-scale unstructured textual documents. Compared to the task of MMNDBs, their scope is different and involves multimodal, open-domain question-answering, while we want to focus on efficiently answering database-like queries over a collection of documents in different formats (e.g., images).

Answering Database Queries There has been substantial effort put into converting queries expressed in natural language into SQL queries for databases with known structure [124–126], and there have also been advancements in adapting this approach for semi-structured data and knowledge bases [127, 128]. Recently, Thorne et al. [94, 95] proposed NeuralDB as a way to perform database queries over a collection of textual documents without the need to translate data or queries into a predefined database schema but using parallel neural techniques instead. Their approach is very effective but it: (i) requires preprocessing and analysis for the aggregation operator; (ii) is limited to simple queries and (iii) is capable of handling data just in textual format. In this section, we stem from this research approach and tackle the third limitation extending the original architecture proposed to multimodal document processing.

Retrieval-augmented models Recently there has been a surge of interest in the line of research concerning retrieval-augmented neural models [129]. Most of the current models focus on augmenting current language models' capabilities with an external memory or retrieval mechanism that retrieves relevant documents given an input query, reducing the number of parameters and non-factual errors [130].

2.3.5 Conclusion

In this section, we have proposed to expand the field of Multimedia Information retrieval through the introduction of Multimodal Neural Databases. MMNDBs promise to answer complex database-like queries that involve reasoning over multiple modalities at scale. We have demonstrated the feasibility and potential of this system by proposing a first principled approach to solve this problem with an architecture composed of three modules - retriever, reasoner, and aggregator - and performing a rich set of experiments. We have discussed potential future research directions that could stem from the system introduced in this section. MMNDBs set a new research agenda that strives to simultaneously act as a bridge between information retrieval and database systems and reduce the gap between the two. We believe MMNDBs have the potential to substantially advance not only the field of MMIR but the general field of Information Retrieval in its entirety.

2.4 Conclusion Effective LLMs

This section has explored methods to enhance the effectiveness of large language models through instruction tuning, prompt engineering, and multimodal integration. Instruction tuning emerged as a critical step in the training pipeline, demonstrating that fine-tuning models on a diverse set of tasks formatted as natural language instructions significantly improves their generalization to unseen tasks. Importantly, these improvements do not rely solely on model scale; smaller instruction-tuned models were shown to outperform much larger models, such as GPT-3, in certain benchmarks. This finding challenges the assumption that performance scales linearly with model size and underscores the value of task-specific and multitask optimization strategies.

The development of PromptSource further advanced the effectiveness of LLMs by enabling systematic and community-driven prompt engineering. With over 2,000 prompts across 170 datasets, PromptSource provided the necessary tools to create a broad range of task formats, enabling LLMs to align more closely with human instructions. This aligns with the broader observation that the flexibility and diversity of task representation, rather than brute-force model size, are key to improving performance across a wide range of applications.

Finally, the exploration of multimodal neural databases (MMNDB) extended the scope of LLM effectiveness to real-world scenarios requiring reasoning across multiple modalities. By integrating text, images, and other data formats, LLMs were demonstrated to be capable of handling complex, database-like queries, thereby expanding their applicability in practical settings.

Together, these contributions highlight that the effectiveness of LLMs depends not only on their ability to process large-scale data but also on their alignment with task-specific instructions and their adaptability to diverse inputs. Instruction tuning and prompt engineering, in particular, show that structured training approaches can yield substantial gains in performance while remaining computationally efficient. Moreover, the multimodal capabilities of LLMs open new possibilities for their deployment in complex, multimodal environments.

In conclusion, the findings presented in this section underscore that improving the effectiveness of LLMs requires a multifaceted approach, combining robust task alignment, prompt diversity, and multimodal integration. These strategies ensure that LLMs are not only powerful but also versatile and adaptable to the nuanced demands of real-world applications. As LLMs continue to evolve, focusing on effectiveness through tailored training and alignment techniques will remain a cornerstone of their development.

Chapter 3

Efficient Large Language Models

This chapter focuses on our contributions to improving the efficiency of LLMs, addressing two key aspects: accelerating inference in transformer models (§3.1) and developing resource-efficient, language-specific models (§3.2).

In the first part of this chapter (§3.1), we present our work on parallel decoding algorithms [4]. Despite the inherently sequential nature of autoregressive language generation, we demonstrate that it is possible to significantly speed up the decoding process without compromising the quality of the generated text in the context of Machine Translation (MT). We introduce three novel algorithms that leverage parallel computing resources to accelerate inference in transformer-based models. This work has introduced the concept of parallel decoding (decoding multiple tokens in parallel) without any additional training (sometimes referred to as Jacobi decoding), which is now an active area of research to speed up the generation of language models [131–133]. This area is now included under the umbrella term of speculative decoding from a paper that concurrently proposed to decode multiple tokens in parallel starting from a draft of a smaller auxiliary model [134].

The second part of the chapter (§3.2) explores an efficient approach to developing language-specific LLMs, focusing on our work on Camoscio [5], an instruction-tuned model for the Italian language. We demonstrate how parameter-efficient fine-tuning techniques, specifically LoRA (Low-Rank Adaptation), can be used to create high-quality, specialized language models with limited computational resources. This approach resulted in the first open instruction-tuned LLM for the Italian language, since then several other Italian LLMs have been proposed [135–137].

3.1 Accelerating Inference in Large Language Models

This section presents the paper "Accelerating Transformer Inference for Translation via Parallel Decoding" [4].

In recent years there have been dramatic improvements in Machine Translation (MT) [138, 139] thanks to the transition to neural models and the advent of the Transformer architecture [140]. These models can produce high-quality translations while being extremely parallelizable during training. However, Transformers are used sequentially at inference time, generating one token per time (i.e., sending each token as input for the next autoregressive iteration). This process of autoregressive inference hampers the efficiency of neural machine translation systems in terms of latency, limiting applications and portability. Considering that these systems are extensively used in production multiple times to produce new translations (e.g., Google

Translate[1], DeepL Translator[2]), even a minor speedup would be beneficial in the long run, especially if the translation is done on embedded devices.

To address this issue, the community proposed *ad-hoc* trained models specific for parallel machine translation under the umbrella term of Non-Autoregressive Machine Translation models (NAT) [141]. These models produce the translation in parallel but require (i) a complete reengineering of the MT system, (ii) extensive training resources and (iii) complex design choices like distillation from larger autoregressive models. These requirements are quite demanding and not easily satisfiable. For example, production systems are heavily optimized for hardware and software and even introducing a minimal modification requires non-trivial human effort [142, 143]. Furthermore, training a new model from scratch is not always possible due to non-released training data or low-resource languages having few or lacking parallel corpora.

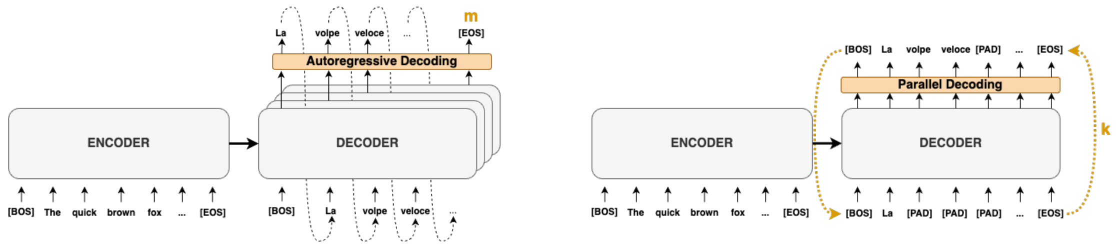


Figure 3.1: On the left, the classical Autoregressive Decoding for MT. The target sentence is produced token-by-token sequentially, sending the partial result as input for the next autoregressive iteration up to the length m of the target. **On the right** Parallel Decoding proposed in this section. This method changes only the *decoding algorithm* (orange block) and is usable on top of any autoregressive model without modifications. Parallel Decoding algorithms resolve the whole sentence or a block of b tokens in parallel: initial tokens (PAD tokens) are gradually refined with k steps until a stopping condition is reached. Crucially, $k \leqslant m$ with quality guarantees and overall decoding speedups.

In this section, we propose to address the problem of parallel machine translation with an orthogonal approach consisting in novel decoding algorithms that work in parallel and can be used on top of *existing autoregressive models* for MT. We overcome previous limitations with a flexible and generic method that does not require any modification to the model or costly retraining. Specifically, inspired by previous successes in speeding up feedforward computation for image generation [144], we reframe the greedy autoregressive decoding for MT as a system of nonlinear equations solvable in parallel. This simple formulation speeds up the decoding procedure by using fixed-point iteration methods like Jacobi and Gauss-Seidel while having mathematical guarantees on the quality of the translation. A high-level description of the method is available in (Fig. 3.1). Our contributions can be summarized as the following:

- We reframe the standard greedy autoregressive decoding procedure in MT with a parallel formulation, introducing three parallel decoding algorithms (PJ, PGJ, HGJ) and a stopping condition that preserves translation quality.
- We perform extensive experiments with different transformer sizes (base and large) and datasets, showing speedups up to 38% in time, obtaining a nearly 2× speedup when scaling the model on parallel resources while preserving quality. To the best of our knowledge, this is one of the first studies to introduce a speedup in multilingual machine translation.
- We introduce a decoding dependency graph visualizer (DDG*viz*) to inspect the learned tokens' conditional dependence and when parallel decoding is effective.

[1] https://translate.google.com/
[2] https://www.deepl.com/

All the code is publicly released[3].

3.1.1 Related Work

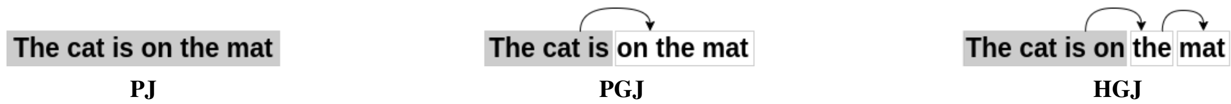


Figure 3.2: Parallel Decoding algorithms: **PJ** resolves the whole sequence in parallel iteratively. **PGJ** resolves blocks in parallel; once a block is finished, it moves on to the next one and decodes it again in parallel (in figure $b = 3$). **HGJ** decodes the sentence in parallel as PGJ up to a certain length h; afterwards, it goes autoregressively until [EOS] token is generated. Decoding actually happens in sub-word tokens (not depicted here).

Gu et al. [141] first introduced Non-Autoregressive Translation models (NAT) as ad-hoc trained models capable of producing the translation all at once in parallel. With NATs, it is possible to consistently reduce the latency and speed up the translation at the expense of a slightly worse translation quality due to the multimodality problem (i.e., we lose the dependency between tokens in the target output). Finding a tradeoff between translation quality and speed is an active research direction, with current methods trying to fill the gap in terms of translation quality [145, 146]. Nevertheless, all proposed NAT models are learning-based and require different tricks to reach the quality of autoregressive models [147]. The most common is the sequence-level knowledge distillation of large autoregressive models into parallel models [148]. Other approaches include defining alternative training objectives [149–152], architectures that model dependencies between output sentence tokens [147, 153–156] or multi-iteration methods [145, 146, 157–161] that apply iterative refinements to a translation, trading some speed for greater quality. In our approach, we also employ iterative refinements of solutions to non-linear equations, but *we do not perform any training or modification to the model*. Other works that require retraining or modifications to the model add additional decoding heads [162] or use shallow decoders [163]. We refer the reader to Xiao et al. [164] for a thorough survey on NAT methods. Further orthogonal approaches use specialized hardware (TPU) with low-precision calculations [142] or software optimizations [143]. In the context of Grammatical Error Correction, Sun et al. [165] recently proposed aggressive parallel decoding, assuming that the model output is similar to the input. More recently, inspiring our work, Song et al. [144] showed that it is possible to parallelize feedforward computations by thinking of them as a system of non-linear equations. They parallelized the backpropagation of RNNs, feedforward layers and autoregressive generative models on images. We extend the approach defined on dense pixel prediction to the discrete conditional token generation in MT. While this work was under submission and anonymity period, Leviathan et al. [134], Chen et al. [166] and Kim et al. [167] concurrently proposed decoding approaches that speed up inference of a large transformer model by using another smaller model to draft tokens. Compared to these approaches our method requires just an existing autoregressive model (no matter the size) and mathematically guarantees the output quality. In the next Section we describe the method.

3.1.2 Method

In this Section, we introduce notations, develop the theory behind Parallel Decoding, present three algorithms (Fig. 3.2), and discuss the initialization and stopping conditions for the proposed approaches.

[3] `https://github.com/teelinsan/parallel-decoding`

Notation

The goal of MT is to translate a sentence $\mathbf{x}$ in a source language (e.g., Italian) with its translation $\mathbf{y}$ in the target language (e.g., English). Source and target sentences are generally tokenized in words or subwords [168–171]; here, we use the subfix notation $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_m)$ to indicate specific tokens in the sequence. We also use the notation $\mathbf{x}_{1:n}$ to indicate a slice of a sequence as a shorthand of $\mathbf{x} = (x_1, \ldots, x_n)$. From a probabilistic perspective, an MT model estimates $p_\theta(\mathbf{y} \mid \mathbf{x})$. Once an MT model has been trained, the inference phase is traditionally performed by sampling tokens from the model probability conditioned on the input sequence $\mathbf{x}$ and previously generated tokens $(y_1, \ldots, y_{i-1})$:

$$p_\theta\left(y_i \mid y_1, \ldots, y_{i-1}, \mathbf{x}\right). \tag{3.1}$$

Different sampling strategies are employed (e.g., Greedy, Top-K, Top-p [172, 173]) alongside search strategies that estimate the total conditional probability (e.g., Greedy search, Beam search [174]). The most straightforward strategy, Greedy Search, selects the element y_i of a sequence with:

$$y_i = \arg\max p_\theta(y_i \mid \mathbf{y}_{1:i-1}, \mathbf{x}). \tag{3.2}$$

Given the formalization above, a standard autoregressive setting runs m inference steps *sequentially* to generate an output sequence of m elements.

Parallel Decoding. Given Equation (3.2), it is possible to write the greedy decoding procedure on all tokens as:

$$\begin{cases} y_1 = \arg\max p_\theta(y_1 \mid \mathbf{x}) \\ y_2 = \arg\max p_\theta(y_2 \mid y_1, \mathbf{x}) \\ \vdots \\ y_m = \arg\max p_\theta(y_m \mid \mathbf{y}_{1:m-1}, \mathbf{x}) \end{cases} \tag{3.3}$$

Defining $f(y_i, \mathbf{y}_{1:i-1}, \mathbf{x}) = y_i - \arg\max p_\theta(y_i \mid \mathbf{y}_{1:i-1}, \mathbf{x})$, we can rewrite the system of Equations (3.3) as:

$$\begin{cases} f(y_1, \mathbf{x}) = 0 \\ f(y_2, y_1, \mathbf{x}) = 0 \\ \vdots \\ f(y_m, \mathbf{y}_{1:m-1}, \mathbf{x}) = 0 \end{cases} \tag{3.4}$$

This system has m non-linear equations (each equation employ a neural network) with m variables.

Parallel Decoding Algorithms

The autoregressive decoding implicitly solves the system of Equations (3.4) by substitution, i.e., given the [BOS] token and the input sentence x, it solves equations from first to last, progressively replacing the resolved variables. In this section, we rely on Jacobi and Gauss-Seidel (GS) fixed-point iteration methods [175] to solve in parallel system (3.4) until a stopping condition is reached. This formulation is particularly flexible and has several advantages: Firstly, it is completely agnostic to the underlying MT model used;

Secondly, it can be analyzed with analytical tools and has guarantees of convergence to the exact solution for system (3.4); Thirdly, it can be potentially extended by drawing from the numerical methods literature for non-linear equations solving methods [176]. We see that, with the proper stopping condition, it is possible to have quality guarantees over the output. We present here three algorithms (PJ, PGJ, HGJ) that leverage these fixed-point iteration methods to speedup decoding in MT.

Parallel Jacobi (PJ) Decoding. First, we propose Algorithm 1. This algorithm works by initializing a draft translation for the whole target sentence and then iteratively translating the whole sentence in parallel until the stopping condition is triggered. This is equivalent to solving system (3.4) with Jacobi, hence the name of the method.

Parallel GS-Jacobi (PGJ) Decoding. Decoding the whole target sentence in parallel may introduce difficulties in inferring long dependencies between tokens since the underlying model is trained to model the conditional distribution of a token given the previous tokens. In general, we observed that shorter dependencies are easily predicted since decoding happens at the sub-word level, and the model can decode sub-word unities in parallel rather than the whole sentence. To this end, we propose Algorithm 2, called GS-Jacobi, that splits the sentence into contiguous b-dimensional blocks. Starting from the first one, it decodes in parallel all its elements. Once a block is finished or the stopping condition within the block is triggered, the algorithm performs a sequential (Gauss-Seidel) step and proceeds with (Jacobi) decoding on the next one.

Hybrid GS-Jacobi (HGJ) Decoding. Algorithms 1 and 2 assume to know beforehand the number of equations m (i.e., the target length). This is not usually the case for MT, where the model dynamically controls the length through the emission of a special end-of-sentence token [EOS]. To overcome this issue, we propose a flexible Hybrid Algorithm 3 that mixes PGJ computations with standard autoregressive decoding. This algorithm performs parallel GS-Jacobi decoding up to a certain prefixed length h. If the [EOS] token is generated within a block, then the algorithm stops, returning the translation up to [EOS]. Otherwise, the algorithm concludes the translation by reaching the [EOS] token with standard autoregressive decoding. In this case, the length h regulates the trade-off between parallel and sequential computation, limiting the waste of resources beyond [EOS].

Initialization and Stopping

Our algorithms share two components: the *initialization procedure* and the *stopping condition*.

Initialization $\textsc{InitT}(\mathbf{x})$. The initialization procedure is a function that inputs the source sentence and produces an initial draft translation as output. In this section we experimented with a simple initialization procedure that initialize the translation with all [PAD] tokens. This choice is fast and doesn't depend on the underlying MT model. We leave as future work the research of different initialization procedures to further speedup the decoding.

Stopping Condition $\textsc{StopC}(\mathbf{y}^{k-1}, \mathbf{y}^k)$. The stopping condition is a function that takes as input the previous-iteration sentence $\mathbf{y}^{k-1}$ and the current-iteration sentence $\mathbf{y}^k$ and decides whether to stop the

Algorithm 1 Parallel Jacobi Decoding

Input: $\mathbf{x} = (x_1, \dots, x_n)$, p_θ
Output: $\mathbf{y} = (y_1, \dots, y_m)$

$\mathbf{y} \leftarrow \text{INITT}(\mathbf{x})$
$m \leftarrow len(\mathbf{y})$
for $i = 1$ to m **do**
    $\mathbf{o} \leftarrow copy(\mathbf{y}_{1:m})$
    $\mathbf{y}_{1:m} \leftarrow \arg\max(p_\theta(\mathbf{y}_{1:m}|\mathbf{y}_{1:m}, \mathbf{x}))$
    $stop \leftarrow \text{STOPC}(\mathbf{o}, \mathbf{y}_{1:m})$
    if $stop$ **then**
        break
    end if
end for
return $\mathbf{y}$

Decoding Algorithm	**en→de**		**de→en**		**en→ro**		**ro→en**	
	Speed	BLEU	Speed	BLEU	Speed	BLEU	Speed	BLEU
Opus								
Greedy Autoregressive	1.00×	28.24	1.00×	33.10	1.00×	27.41	1.00×	37.01
Beam Search (beam = 5)	0.71×	28.68	0.72×	33.92	0.70×	27.61	0.72×	37.84
PJ Decoding	0.73×	28.24	0.75×	33.10	0.66×	27.41	0.66×	37.01
PGJ Decoding (b = 5)	1.28×	28.24	1.32×	33.10	1.33×	27.41	1.29×	37.01
PGJ Decoding (b = 3)	**1.34**×	28.24	**1.37**×	33.10	**1.38**×	27.41	**1.35**×	37.01
HGJ Decoding (b = 3)	**1.34**×	28.24	**1.37**×	33.10	**1.38**×	27.41	**1.35**×	37.01
MBart50								
Greedy Autoregressive	1.00×	23.97	1.00×	31.58	1.00×	24.99	1.00×	34.77
Beam Search (beam = 5)	0.76×	24.93	0.77×	32.61	0.77×	25.31	0.76×	35.16
PJ Decoding	0.88×	23.97	0.88×	31.58	0.86×	24.99	0.85×	34.77
PGJ Decoding (b = 5)	0.98×	23.97	0.98×	31.58	0.97×	24.99	0.99×	34.77
PGJ Decoding (b = 3)	**1.06**×	23.97	**1.08**×	31.58	**1.03**×	24.99	**1.04**×	34.77
HGJ Decoding (b = 3)	1.05×	23.97	1.07×	31.58	1.01×	24.99	1.02×	34.77

Table 3.1: Comparison of parallel decoding algorithms (highlighted in grey) with sequential decoding using Opus (CPU) and MBart50 (GPU) on WMT14 and WMT16. Speed is measured in time w.r.t. the autoregressive baseline.

algorithm or not. This function is crucial since it regulates the trade-off between speedup and translation quality. In this section we introduce as stopping condition for MT:

$$\mathbf{y}^{k-1} - \mathbf{y}^k = \mathbf{0} \tag{3.5}$$

i.e., the sentence from the previous step has not changed. This stop condition allows for preserving quality and quickening translations simultaneously.

		WMT17 En-Fi		**IITB En-Hi**		**IWSLT15 En-Vi**		**FLORES En-It**		**FLORES En-Fr**	
Dec. Algorithm	**Speed**	←	→	←	→	←	→	←	→	←	→
PJ	Iters	1.04×	1.04×	1.04×	1.04 ×	1.06×	1.03×	1.02×	1.04×	1.03×	1.03×
	Time	0.86×	0.88×	0.89×	0.89×	0.87×	0.86×	0.85×	0.86×	0.85×	0.85×
PGJ (b=3)	Iters	1.07×	1.09×	1.09×	1.09×	1.10×	1.07 ×	1.07×	1.08×	1.08×	1.11×
	Time	1.01×	1.05×	1.05×	1.07×	1.04×	1.02×	1.02×	1.03×	1.03×	1.05×
HGJ (b=3)	Iters	1.05×	1.07×	1.07×	1.07×	1.07×	1.06×	1.07×	1.06×	1.05×	1.07×
	Time	1.01×	1.03×	1.04×	1.05×	1.03×	1.01×	1.01×	1.02×	1.01×	1.03×

Table 3.2: Comparison over different languages in terms of speedup and iterations on MBart50. Arrows indicate the direction of translation. Qualitative results and BLEU scores are available in the appendix C.3.

Algorithm 2 Parallel GS-Jacobi Decoding

Input: $\mathbf{x} = (x_1, \ldots, x_n), p_\theta, b$
Output: $\mathbf{y} = (y_1, \ldots, y_m)$

$\mathbf{y} \leftarrow \textsc{InitT}(\mathbf{x})$
$m \leftarrow len(\mathbf{y})$
$i \leftarrow 1$
while $i \leqslant m$ **do**
  $\mathbf{o} \leftarrow copy(y_{i:i+b})$
  $\mathbf{y}_{i:i+b} \leftarrow \arg\max(p_\theta(\mathbf{y}_{i:i+b}|\mathbf{y}_{1:i+b}, \mathbf{x}))$
  $stop \leftarrow \textsc{StopC}(o, y_{i:i+b})$
  if $stop$ **then**
    $i \leftarrow i + b$
    break
  end if
end while
return $\mathbf{y}$

Quality Guarantees

Compared to NAT methods which do not have any quality guarantee since a novel parallel model is trained from scratch, our formulation guarantees to have the same quality of using autoregressive decoding with the same MT model. System (3.4) is known in literature as a *triangular system* of m equations with m variables, this characterization allows to state an important property.

Proposition 1. *Algorithms 1, 2, 3 converge and yield the same results of greedy autoregressive decoding in at most m parallel iterations, for any initialization and providing stopping condition* (3.5).

We refer the reader to Song et al. [144] for a formal proof. Intuitively, with m steps the algorithm used the same number of iterations of autoregressive, hence the final solution is the same regardless the initialization. In this worst case, the wall-clock time is the same but in general the algorithm reach the stopping condition earlier with a lower wall-clock time and overall speedup.

DDG*viz*

Equation 3.1 models the dependency between tokens in the decoding phase. In the classical autoregressive mode, each token depends on all the previous ones for the generation. However, it is possible to show that this dependency is actually relaxed (i.e., not all tokens depends on all the previous ones), thus it would be interesting to visualize the actual distribution $p_\theta\left(y_i \mid \cdot, \mathbf{x}\right)$ learned by an existing MT model. To this end, we build the Decoding Dependency Graph visualizer (DGG*viz*) to visualize the dependency graph of tokens in the decoding phase. In the standard autoregressive decoding this graph is a fully-connected chain where the i-th token is connected to all the previous tokens, starting from the encoding $\mathbf{x}$: to decode y_i you need to decode first $y_1, \ldots, y_{i-1}$. Instead we show that there are skipping connections between independent tokens that can be visualized with DGG*viz*. We detail DGG*viz* with an example in section 3.1.3.

Algorithms details

We propose here the pseudocode of Algorithms 2 and 3.

The function $copy(y_{i:i+b})$ creates a copy of the tensor in input detached from the source. This is done in practice to avoid the overwriting of pointers to the same memory location. Function $\text{CHECKEOS}(y_{i:i+b})$ returns the index of the token EOS in the block if present, else -1. Function $\text{CHECKEOS}(y_i)$ returns $True$ if the tokes in exactly the token EOS, else $False$. The function $\arg\max$ selects from the model distribution over the vocabulary the index (token) with maximum probability. This procedure is done for all the tokens in parallel, in the case of parallel decoding, or for just a single token in the case of autoregressive decoding. Generally, the output is the prediction for the next token; hence it should be shifted left before the reassignment to a variable. We omitted this implementation detail for clarity.

3.1.3 Experiments

Experimental Settings

Datasets. We evaluate our approach using standard evaluation datasets proposed for parallel MT [141]: WMT14 English-German [En-De], WMT16 English-Romanian [En-Ro] [177, 178]. Additionally, we tested our method on different language pairs with varying (low-medium) resources: IWSLT15 (English-Vietnamese [En-Vi]) [179], IITB (English-Hindi [En-Hi]) [180], WMT17 (English-Finnish [En-Fi]) [181], FLORES-101 (English-Italian [En-It]; English-French [En-Fr]) [182]. All the datasets are evaluated in both directions.

Evaluation. All the evaluations are performed using the official test split for each dataset, downloaded using Huggingface dataset library [68]. No training or hyperparameters tuning is performed. We use SacreBLEU to evaluate the translation quality [183, 184]. We measure speedup in wall-clock time and iterations w.r.t. the same autoregressive model. GPU times are calculated after calling `torch.cuda.synchronize()`. All the experiments were performed by caching the past Keys and Values of the transformer to further speed up the computation [185] and in the online inference setting with batch size equal to 1. For the Jacobi and GS-Jacobi algorithms, we assume to know beforehand the length m of the target and measure the speedup in the ideal condition. For the Hybrid GS-Jacobi algorithm, we set h equal to the maximum (i.e., the stopping condition is triggered within a parallel block) to decouple the effective speedup regardless of the length produced by the initialization function (see Section 3.1.2). We remark that HGJ does not assume to know beforehand the target length and is applicable to real MT translation scenarios.

Model Configuration. We tested transformer models in the two standard configurations: base (512 model dimension, 6 attention layers for both encoder and decoder) and big (1024 model dimension, 12 attention layers for both encoder and decoder). We used pretrained models of Opus [186] for the former and MBart50 [187] for the latter. Opus is a transformer base model (74M parameters) trained on language pairs from the homonymous dataset [188]. MBart50 is a large multilingual transformer model fine-tuned for translation on 50 languages (610M parameters). We tested the models on CPU since this is the default environment for MT models in production, except for the model MBart50 which runs on GPU. We run the experiments on a standard 16-core machine, except for the scaling experiments. Additional specifications are available in Appendix C.1

Algorithms Comparison

In Table 3.1 we compare the proposed parallel decoding algorithms with the standard sequential autoregressive decoding baselines. As we can observe, the fastest algorithms are PGJ Decoding (b=3) and HGJ Decoding

Decoding Algorithm	en→de		de→en		en→ro		ro→en	
	Time	Iters	Time	Iters	Time	Iters	Time	Iters
Opus								
Greedy Autoregressive	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×
Beam Search (beam = 5)	0.71×	1.00×	0.71×	1.00×	0.70×	1.00×	0.72×	1.00×
PJ Decoding	0.72×	1.03×	0.74×	1.04×	0.69×	1.04×	0.67×	1.03×
PGJ Decoding (b = 3)	**1.16**×	1.04×	**1.19**×	1.07×	1.17×	1.05×	**1.17**×	1.03×
HGJ Decoding (b = 3)	**1.16**×	1.04×	**1.19**×	1.06×	1.17×	1.05×	**1.17**×	1.03×
MBart50								
Greedy Autoregressive	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×
Beam Search (beam = 5)	0.76×	1.00×	0.77×	1.00×	0.77×	1.00×	0.76×	1.00×
PJ Decoding	0.88×	1.03×	0.88×	1.03×	0.86×	1.04×	0.85×	1.03×
PGJ Decoding (b = 3)	**1.06**×	1.10×	**1.08**×	1.11×	**1.03**×	1.08×	**1.04**×	1.11×
HGJ Decoding (b = 3)	1.05×	1.07×	1.07×	1.01×	1.01×	1.02×	1.02×	1.08×

Table 3.3: Comparison of parallel decoding algorithms (highlighted in grey) with sequential decoding using Opus (CPU) and MBart50 (GPU) on WMT14 and WMT16. Speed is shown here both in Time and Iterations w.r.t. the greedy autoregressive baseline.

Method	Requirements			WMT14		Efficiency		
	Arch	Loss	seq-KD	Speed ↑	BLEU ↑	Train FLOPs ↓	Total FLOPs ↓	FLOPs / Speed ↓
Parallel Decoding - HGJ (Ours)	No	No	No	1.34×	28.24	**0**	**2.53e+13**	**1.89e+13**
SUNDAE † [146]	Yes	No	No	1.4×	28.46	5.27e+21	5.27e+21	3.77e+21
ShallowDec (12-1) [163]	Yes	No	No	1.4×	26.90	1.02e+19	1.02e+19	7.30e+18
Semi-NAT [189]	Yes	No	Yes	1.5×	26.90	1.55e+17	1.55e+17	1.03e+17
DisCo [158]	Yes	Yes	Yes, Big	3.5×	27.34	4.06e+19	4.06e+19	1.16e+19
DSLP [152]	Yes	Yes	Yes	14.8×	27.02	1.93e+19	1.93e+19	1.31e+18
F-VAE [147]	Yes	Yes	Yes, Big	16.5×	27.49	4.06e+19	4.06e+19	2.46e+18

Table 3.4: Comparison of different methods for parallel MT on WMT14 En-De. Results are ordered by speed, highlighted in green the two highest BLEU scores, † indicates diffusion models. Existing methods require training, architecture modifications, additional losses to force parallel translation, and distillation from an additional MT transformer model ("Big" indicates the size). Details on FLOPs computation are available in the Appendix C.2.

(b=3) which are up to 34% and 38% times faster on Opus and up to 5% and 8% faster on MBart50, depending on the language pair. We note also that results empirically show that all the parallel decoding algorithms guarantee the same quality of greedy autoregressive decoding, as evidenced by the unchanged BLEU scores. This is an experimental verification of the formal Proposition 1. The table also shows that the Beam Search algorithm with a beam size of 5 generally performs better in terms of BLEU score, although at a cost of speed. This difference in terms of BLEU is expected, as beam search is a heuristic search strategy, while our method is a decoding algorithm. We discussed better this aspect in the "Beam Search" paragraph. Nevertheless, beam search is ∼30% slower than greedy autoregressive and 63% to 68% slower than PGJ, depending on the model and language pair. This means that the proposed parallel algorithms allow trading a little translation quality (e.g., on en→ro the difference between beam search and parallel decoding algorithms in BLEU is just 0.20 points) for greater decoding speed.

Another aspect to note is that the algorithms PJ and PGJ (b=5) are sometimes slower than greedy autoregressive. There are several factors that can influence the actual wall-clock time like how the underlying hardware schedule and execute the various operations, which might vary according to the architecture and the workload. In particular, longer sequences (e.g., the whole sentence in PJ or blocks of 5 tokens in PGJ) may require more memory to store, and the CPU/GPU may have to perform more memory accesses, which can slow down the computation (although theoretically it should happen in parallel). In the end, these computational overheads slow down the actual execution. This is also the case for the difference in speedups between MBart50 and Opus. We better investigated this aspect in the section "Computational Scaling" and

	WMT17		IITB		IWSLT15		FLORES			
	En-Fi		En-Hi		En-Vi		En-It		En-Fr	
Dec. Algorithm	←	→	←	→	←	→	←	→	←	→
Autoregressive	17.55	25.34	16.50	24.70	31.92	33.94	22.78	26.38	39.51	38.90
Beam Search	18.39	26.04	16.87	25.24	32.14	34.59	23.52	26.80	39.59	39.21
PJ	17.54	25.35	16.50	24.69	31.92	33.94	22.78	26.38	39.50	38.90
PGJ (b=3)	17.55	25.35	16.50	24.70	31.93	33.94	22.78	26.38	39.51	38.90
HGJ (b=3)	17.55	25.35	16.50	24.70	31.93	33.94	22.78	26.38	39.51	38.90

Table 3.5: BLEU scores on MBart50.

report in the appendix results on a different architecture, with also results in terms of iterations speedups which are architecture agnostic.

Analysis and Validation

Cross Languages. In order to demonstrate the robustness of our decoding algorithms with respect to the translation languages, we leveraged the multilingual capabilities of the MBart50 model and selected a diverse range of language pairs for evaluation. The results, presented in Table 3.2, show that both PGJ and HGJ achieve a consistent speedup in comparison to the autoregressive decoding method, with an improvement ranging from 2-7% for PGJ and 1-5% for HGJ, regardless of the language pair used. Additionally, we observed a speedup in terms of iterations of 7-11% for PGJ and 5-7% for HGJ. These findings indicate that our algorithms have the potential to match or surpass the speedup in terms of wall-clock time by fully exploiting this saving in terms of iterations. We note that, similar to the previous experiment, PJ suffers from an overhead problem. To the best of our knowledge, this is one of the first studies that have achieved a speedup in multilingual machine translation, concurrent with the work of Song et al. [156], while this latter is significantly different in spirit and requirements (NAT model). We leave BLEU scores in the Appendix C.3 for space constraints together with qualitative results in different languages.

Computational Scaling. In Figure 3.3, we present an analysis of the scalability of our proposed methods in relation to increasing computational resources. Starting with 8 cores, our methods demonstrate a slight improvement in terms of wall-clock time for PGJ and HGJ, with speedups of 1.11 and 1.09 respectively. On the other hand, this amount of resources is too restricting for PJ which needs to fit the whole sentence and thus achieve a score of 0.46 due to the aforementioned overhead problem. As the resources are increased, our method demonstrates the ability to effectively leverage hardware and significantly reduce decoding time, while the autoregressive baseline is constrained by sequential processing. With 122 cores, a substantial speedup of $1.98\times$ and $1.99\times$ is achieved for PGJ and HGJ respectively, while the autoregressive baseline is bounded by sequential processing at $1.00\times$. It is important to note that this experiment does not simulate a real production system, but rather it is meant to show what results can be achieved when the underlying computation is properly optimized to run in parallel. In our case, we simulated this setting with increasing cores, nevertheless similar results can be achieved with additional software optimizations to further reduce latency and overheads [143, 190] and increase the speed gain with parallel-optimized computations. Overall this experiment serves as a proof of concept for the capabilities of parallel decoding in contexts with limited overhead and shows a promising direction for further improvements.

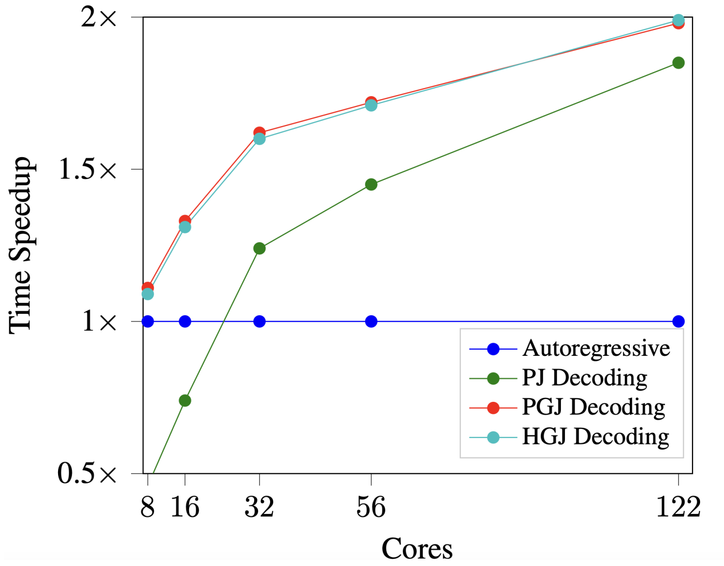


Figure 3.3: Scaling experiments on WMT16 En-De with PGJ and HGJ blocks = 3. Increasing the number of available resources (number of CPU cores) allows the methods to decrease the parallel overheads. As a result, the speedup increases and the methods scale.

Comparison with NATs. Table 3.4 reports the comparison of our parallel decoding algorithm with a selection of NAT methods for parallel MT. Following prior works, we report for each method the speedup relative to the autoregressive transformer base baseline from their original paper [164]. It is worth noting that, although these methods can achieve higher speedups, they are very demanding in terms of computational resources which must be accounted for in a fair comparison. To estimate quantitatively this cost, we evaluated the number of floating point operations (FLOPs) required for training and inference on WMT14.

Results show that our method HGJ uses the least number of computational resources, even considering the additional cost at inference time. Relating the speedup obtained with the used resources (FLOPs/speed), our method still achieves the best cost-benefit ratio. Furthermore, NATs generally degrade the translation quality if compared to their autoregressive baseline. On the contrary, our method mathematically guarantees the same quality of autoregressive decoding, which is higher than standard NAT models.

SUNDAE achieves BLEU of 28.46, but requires more resources than training RoBERTa [191] on 16 TPUs (see Appendix C.2). Other methods require further elaborate techniques like profound architectural changes, additional losses to force parallel translation and sequence-level distillation from large autoregressive transformers [147]. Our approach is a decoding method that does not involve any training or modification to the model and can be used to speed up existing models on standard desktop hardware.

Speedup Analysis. We provide here a preliminary analysis of the factors responsible for the observed speedup in our method. We first distinguish between two types of speedup: wall-clock speedup and iterations speedup. The former is primarily driven by the parallelization capability of our method, as demonstrated in the ”Computational Scaling” section. With parallel decoding, underlying operations can be optimized and fused to be executed fastly. Compared to Sheng et al. [192], our method allows parallelizing sequence operations (”row-by-row” setting). The latter instead may vary consequently to several factors (e.g., model/vocabulary size, training data, language, etc). For this reason, we experimented with several variations of these factors (models Transformer Base vs. Big, vocabularies 58K Marian vs. 250K MBart50, languages, and hardware). While it is challenging to decouple different elements, our analysis point out several interesting insights. For

Algorithm 3 Hybrid GS-Jacobi Decoding

Input: $\mathbf{x} = (x_1, \ldots, x_n)$, p_θ, b
Output: $\mathbf{y} = (y_1, \ldots, y_m)$

$\mathbf{y} \leftarrow \textsc{InitT}(\mathbf{x})$
$h \leftarrow len(\mathbf{y})$
$i \leftarrow 1$
$eos\_cond \leftarrow False$
while $i \leqslant h$ **do**
  $\mathbf{o} \leftarrow copy(\mathbf{y}_{i:i+b})$
  $\mathbf{y}_{i:i+b} \leftarrow \arg\max(p_\theta(\mathbf{y}_{i:i+b}|\mathbf{y}_{1:i+b}, \mathbf{x}))$
  $stop \leftarrow \textsc{StopC}(\mathbf{o}, \mathbf{y}_{i:i+b})$
  $eos\_ind \leftarrow \textsc{CheckEOS}(\mathbf{y}_{i:i+b})$
  if $stop$ **and** $eos\_ind > -1$ **then**
    $\mathbf{y} \leftarrow \mathbf{y}_{1:eos\_ind}$
    $eos\_cond \leftarrow True$
    break
  end if
  if $stop$ **then**
    $i \leftarrow i + b$
    break
  end if
end while
while $eos\_cond \, != True$ **do**
  $y_i \leftarrow \arg\max(p_\theta(y_i|y_{i-1}, \mathbf{x}))$
  $i \leftarrow i + 1$
  $eos\_cond \leftarrow \textsc{IsEOS}(y_i)$
end while
return $\mathbf{y}$

example, we observed that iteration results on MBart50 are generally higher compared to Marian (Tables 3.2-3.3), possibly due to the finer-grained tokenization of MBart50. We also hypothesize that language and linguistic features, such as inflectionally rich or agglutinative/gendered languages, may influence iteration speedups. To facilitate this type of analysis, we developed DDG*viz*, which we believe will be useful for research in this area.

Visualizing Parallel Decoding. In previous experiments, we demonstrated that parallel decoding is feasible. This suggests that the dependency learned by the model between certain tokens is relaxed, as some tokens can be decoded in parallel. Analyzing and understanding when this happens allows shedding light on the behavior of existing models and a separate study focused on this issue would be needed. In this work, we lay the ground for a such study introducing the necessary inspection tools. While we have already introduced DDG*viz* in Section 3.1.2, in this experiment we show how it works and how it can be used with a practical example. In summary, the DDG*viz* visualizer allows to show the *real* decoding distribution $p_\theta\left(y_i \mid \cdot, \mathbf{x}\right)$ learned by a MT model. This decoding distribution is plotted as a graph, where a connection indicates the dependency $p_\theta(y_i \mid \cdot)$, by using Parallel Jacobi decoding. At each PJ decoding iteration (vertical axis of Figure 3.4), DDG*viz* keeps track of which tokens have been correctly decoded w.r.t. the gold autoregressive reference of the model, showing the tokens correctly decoded and the probability of each one (horizontal axis). Figure 3.4 shows DDG*viz* applied on an example. The example shows that for $y_4 = \_sa$ it is possible

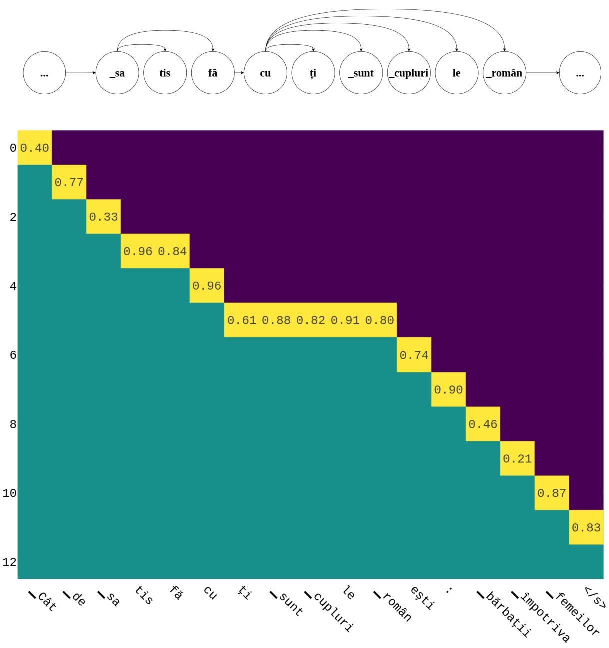


Figure 3.4: DDG*viz*. Visualization of the translation En-Ro: "How satisfied are the Romanian couples: men versus women"→"Cât de satisfacuți sunt cuplurile romanești: bărbații împotriva femeilor". (Highlighted tokens decoded in parallel). **On top**: the Decoding Dependency Graph, omitting redundant edges on non-parallel tokens to ease visualization. **On bottom**: DDG*viz* shows at each Parallel Jacobi iteration (vertical axis) which tokens have been generated in parallel (horizontal axis) with the corresponding probability (cell number).

to decode more than one token in parallel $y_5 = tis$, $y_6 = fa$, hence here the decoding of y_6 does not depend on the decoding of y_5 - $p_\theta\left(y_6 \mid \mathbf{y}_{1:4}, \mathbf{x}\right)$. We observed this phenomenon frequently, explaining the speedups in the previous experiments. The example also shows that the model is able to decode five tokens in parallel after $y_7 = \_cu$. This is a peculiar case since the model, given *"How satisfi_"*, is generating all at once *"_ed are the Romanian couples"* (proposed here in English for better readability, original version in Romanian is available in Figure). This example indeed shows how DDG*viz* can be used to highlight possible biases encoded in the model as it is not clear how the model can be so confident (see cell probability) that after "satisfied" the most straightforward tokens to decode are "Romanian couples" [193, 194]. We leave other use cases for future works and show in Appendix C.3 several visualizations with equally interesting phenomena.

3.1.4 Conclusions

In this section, we showed that is possible to speed up *existing* machine translation models by simply changing the decoding algorithm with a parallel formulation. We introduced three parallel decoding methods which achieve consistent speedups without requiring any training, modifications, or quality loss. Our solution is orthogonal to previous approaches proposed in literature which often entail demanding requirements in terms of data, computational resources, and engineering effort. Although our method is not without shortcomings, it is a first valuable step toward integrating parallel decoding algorithms into any model. This is particularly relevant in limited-resource scenarios where NATs are not a viable option and to speed up any transformer model, especially fine-grained or character-level models [195]. We believe that further advancements in this area, including the exploration of optimal initialization procedures and stopping conditions, as well as the use of alternative parallel solvers for non-linear equations, will close the gap with learning-based techniques and continue to improve the efficiency and effectiveness of parallel decoding algorithms.

Limitations

The proposed algorithms allow to speed up an existing model out-of-the-box, without any modification or retraining. However, there are some considerations to bear in mind when using parallel decoding in order to have a speedup in terms of wall-clock time. Firstly, as the name implies, the method executes the decoding phase in parallel. Therefore, to appreciate the speedup one should be able to run computations in parallel. Using parallel decoding without parallel resources or parallel-optimized software may increase wall-clock time due to overheads, leading to a waste of computation. This is further discussed in Section 3.1.3 ”Computational Scaling”. The reported wall-clock time results are thus to be considered within the scope of the experimental setup proposed in this section and they may vary depending on the underlying hardware and software. Secondly, the method allows speedup of the decoding by scaling on parallel resources. This implies an additional computational cost during the inference phase to achieve a speedup. While using parallel decoding, one should consider a trade-off between the desired acceleration and the utilization of computational resources. Thirdly, since our method performs the decoding in parallel, as for NAT systems, it is difficult to combine it with Beam Search. Beam Search is inherently a dynamic programming algorithm and it is not possible to efficiently maximize the joint probability of the large search space without using sequential intermediate computations. We better explain this aspect in the next paragraph.

Beam Search. Beam search is widely employed to enhance the translation quality in MT [196, 197] as well as in other domains such as audio [174, 198]. However, it is an inherently sequential procedure that stores partial joint probabilities of the entire sequence (beams) while progressing with autoregressive decoding. Determining the maximal joint probability of all sequences in parallel is a challenging task, equivalent to a full maximum a posteriori (MAP) estimation. This is an open research problem and it is also an issue for NAT methods. NAT methods patch up this limitation with sequence-level KD which has the advantage of ”not requiring any beam search at test-time” [148] thanks to learning and distillation from large models. Since our method is a decoding algorithm, we cannot use the same approach without learning. Nevertheless, the quality guarantee allows our methods to have performance on par with greedy autoregressive and generally better than a NAT model. We think of our method, not as a replacement for beam search, but rather as a way to obtain a speedup at inference time that is a middle ground between autoregressive greedy decoding (high quality, no requirements, no speed) and NATs (quality compromises, increasing requirements with increasing speed). Future works might address the quality gap with beam search by combining parallel decoding with alternative techniques like Minimum Bayes Risk [199].

Ethics Statement

Increasing the inference speed of MT can positively impact society by giving people a fast and good translation. This will enable people from different language backgrounds to communicate with each other and help remove cultural and trade barriers. As demonstrated by comparing the number of FLOPs in Table 3, our method uses fewer resources compared to alternatives and thus has a smaller carbon footprint, making it a more sustainable choice [200]. Furthermore, since our method does not involve training procedures or change the quality of results, we do not introduce any societal bias (e.g. racism, sexism, homophobia) into the translations. The latter, however, can be introduced through data in the training of the backbone autoregressive models and NATs. It is the task of those who train these models to mitigate this problem. DDG*viz* can also help investigate and visualize some potential harmful biases encoded in the model.

3.2 Efficient Instruction-tuning for the Italian Language

This section presents the paper "Camoscio: An italian Instruction-tuned LLaMA" [5].

In recent years, Large Language Models (LLMs) have made remarkable advancements in the field of natural language processing, demonstrating state-of-the-art performance on various tasks [19, 201, 202]. However, the majority of these models are typically controlled by for-profit organizations that release just a paid API for receiving responses based on input textual prompts. This severely constrains researchers from conducting comprehensive and meaningful research, as they lack access to both the model's weights and the training data regime. This limitation is particularly relevant for privacy-sensitive applications (e.g., medical domain) where data cannot be shared with external providers.

On the other hand, several open-source models[4] have been proposed as an alternative to closed models [203–205]. However, most of these models are English-centric or multilingual, albeit with performance that lags behind their monolingual counterparts. Furthermore, in these latter models, support for the Italian language is usually poor. For example, BLOOM – the largest open multilingual model available up to date – has not been trained on any Italian data, while LLaMA has only a small percentage of training data in the Italian language [5]. In addition to this, most of these models are only trained with the standard language modeling objective (i.e., predict the next token given the previous ones) on corpora of raw textual data, while it has been shown that a second training step of instruction-tuning is crucial to increase downstream performance [1, 206, 207]. Recently, a step in this direction has been made by Taori et al. [208] with the release of Stanford Alpaca, an instruction-tuned version of LLaMA for the English language. Following this approach, in this section we propose Camoscio as an instruction-tuned version of LLaMA for the Italian language by translating to Italian the instruction-tuning dataset of Stanford Alpaca. In particular, we finetuned the smallest version of LLaMA (7 billion parameters) with LoRA [209], a parameter-efficient finetuning technique that allows to train larger models on standard desktop hardware. Our contributions are the following:

- We introduce an instruction-tuning dataset for the Italian language, stemming from the Stanford Alpaca [208] dataset, translating it to Italian.
- We train Camoscio on this dataset and evaluate its zero-shot performance on several downstream tasks for the Italian language (NewsSum-IT, SQuAD-IT, XFORMAL IT).
- We release all the artifacts (code, dataset, model checkpoints) to the community.

3.2.1 Background

Large language models have emerged as a general class of models capable of performing a wide range of tasks without explicit finetuning by just leveraging in-context examples [210]. They've garnered popularity not only in the natural language processing domain but also across audio, image, and multimodal domains [3, 211, 212], with most of the approaches scaling or optimizing their performance [4, 201].

In the context of the Italian language, the availability of pre-trained language models is currently limited; generic multipurpose LMs are almost nonexistent. Notable mentions include: AlBERTo [213], an Italian version of BERT [214] trained on Italian tweets from TWITA [215]; GePpeTto [216], a version of GPT-2 base (117 million parameters) finetuned using Italian Wikipedia and the ItWac corpus [217]; IT5 [218] a T5 model

[4] Actual openness depends on the model license.

[5] Less than 4.5% of training data comes from Wikipedia in 20 different languages, including Italian.

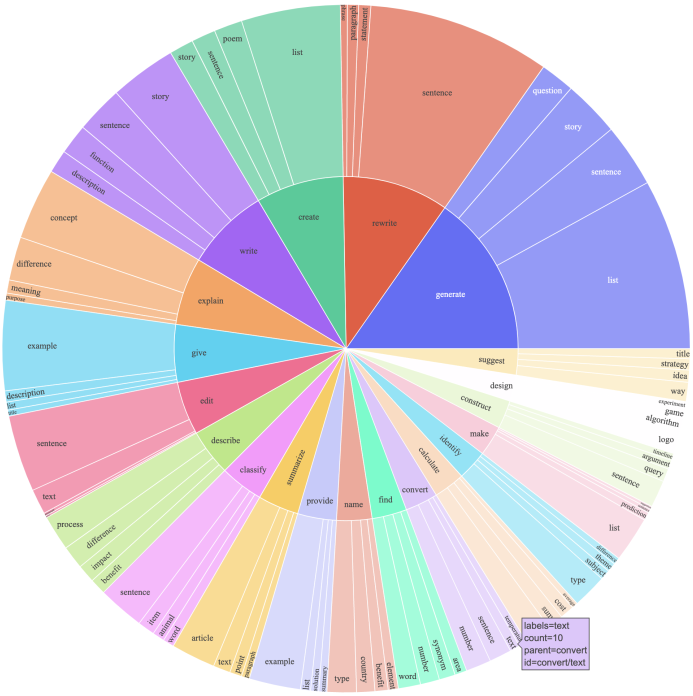


Figure 3.5: Diversity of the examples in the Stanford Alpaca dataset. Illustration from Taori et al. (2023). The inner circle shows the root verb on the instruction while the outer circle shows the direct object. The dataset of Camoscio is constructed by translating all these examples to Italian via *gpt-3.5*.

tailored for Italian using a refined version of the mC4 corpus [219]; and BART-IT [220], an Italian variant of BART [221] trained on the same mixture of data as IT5. Concurrently to our work, Bacciu et al. [135] proposed Fauno, an Italian version of Baize [222] that is a LM trained on a corpus of self-chat performed by ChatGPT. Compared to our work, their approach is tailored to develop a conversational agent for the Italian language. After our work, Michael [223] released on their GitHub repository an instruction-tuned version of LLaMA on a translation to Italian of the GPT-4-LLM dataset [224].

3.2.2 Method

For the construction of our instruction-tuning dataset for the Italian language, we stem from the Stanford Alpaca dataset [208] and Alpaca LoRA [225] for their finetuning approach.

Dataset

Stanford Alpaca is an instruction-tuning dataset constructed using the self-instruct method [226]. Specifically, the authors started with a set of 175 human-written instruction-output pairs from the original self-instruct paper[6] and used them as in-context examples to prompt OpenAI *text-davinci-003*. A total of 52.000 novel examples are generated with this technique. Each example includes an *instruction*, in natural English language, the answer (*output*), and optionally an additional context (*input*) for some datapoints (e.g., a short paragraph for question answering). Figure 3.5 shows different types of instructions in the dataset.

Translation. Inspired by Croce et al. [227], Scaiella et al. [228] and Larcher et al. [229], we translated the original dataset of Stanford Alpaca to Italian using `gpt-3.5-turbo` with the prompt *“Translate the*

[6]https://github.com/yizhongw/self-instruct

	SQuAD-IT						
	F1	EM	EM-GPT	R1	R2	RL	BS
DrQA-IT [227]	.659	.561	-	-	-	-	-
mBERT [230]	.760	.650	-	-	-	-	-
BERT[3] [214]	.753	.638	-	-	-	-	-
MiniLM [231]	.720	.577	-	-	-	-	-
MiniLM$_{+\mathtt{st}}$ [231]	.745	.620	-	-	-	-	-
XLM-R Large$_{+\mathtt{st}}$ [231]	.804	.676	-	-	-	-	-
mT5 Small [218]	.660	.560	.684	.617	.347	.617	.712
mT5 Base [218]	.757	.663	.745	.709	.396	.708	.770
IT5 Small [218]	.716	.619	.602	.671	.372	.671	.743
IT5 Base [218]	.761	.663	.600	.712	.406	.712	.770
IT5 Large [218]	.780	.691	.641	.730	.412	.729	.784
Camoscio-7b **(0-shot)**	.270	.077	.576	.242	.133	.241	.237

Table 3.6: Results on SQuAD-IT. All the models are trained on the SQuAD-IT training set, except for Camoscio which is evaluated in a zero-shot fashion. The additional evaluation metric *Exact Match via ChatGPT* is highlighted in grey. The scores F1 and EM for competitor models are reported from their respective papers.

following text to Italian: `{text}`*”*. We translated all the fields in the dataset (*instruction, input, output*). We decided to use ChatGPT instead of other APIs for translation (e.g., Google Translate, Microsoft Azure Translator, DeepL) because we found it to be more robust for translating code examples i.e., it translates correctly just the comments in the code and not also the coding lexicon of the programming language. We provide here an example from the dataset. `Instruction`: *“Data una parola, costruisci i suoi antonimi.”*, `Input`: *“Luce”*, `Output`: *“Scuro, pesante, denso”*.

Clearly the translation is not always perfect, but it is a fast-and-cheap method to bootstrap a noisy instruction-tuning dataset for the Italian language.

Training & Prompting

We finetuned the smallest version of LLaMA [205] (7 billion) on an instruction-tuning dataset for the Italian language, obtained by translating to Italian the dataset of Stanford Alpaca as described in the paragraph above.

The model is trained with supervision with the standard objective of predicting the next token given the previous ones. The dataset has `instruction, input, output` fields, but the `input` is not available for all data points (e.g., open-ended generation). For such cases, we construct the prompt: *“Di seguito è riportata un’istruzione che descrive un task. Scrivete una risposta che completi adeguatamente la richiesta. ### Istruzione:* `{instruction}` *### Risposta:* `{output}`*”*. If, instead, the datapoint also has an `input` (e.g., question answering where the input is the contextual paragraph), we construct the prompt: *“Di seguito è riportata un’istruzione che descrive un task, insieme ad un input che fornisce un contesto più ampio. Scrivete una risposta che completi adeguatamente la richiesta. ### Istruzione:* `{instruction}` *### Input:* `{input}` *### Risposta:* `{output}`*”*.

At inference time, the same prompt is used to generate the answer. Only the text generated after “[...] *### Risposta:”* is used as final output. We sample from the model using *top-p* sampling [232] with a temperature of 0.2, $p = 0.75$, $k = 40$, and beam search with 4 beams.

We refer to Appendix D.1 for the additional implementation details.

	XFORMAL (IT) F → I				XFORMAL (IT) I → F			
	R1	R2	RL	BS	R1	R2	RL	BS
mT5 Small	.651	.450	.631	.666	.638	.446	.620	.684
mT5 Base	.653	.449	.632	.667	.661	.471	.642	.712
IT5 Small	.650	.450	.631	.663	.646	.451	.628	.702
IT5 Base	.652	.446	.632	.665	.583	.403	.561	.641
IT5 Large	.611	.409	.586	.613	.663	.477	.645	.714
Camoscio-7b **(0-shot)**	.645	.436	.623	.651	.622	.428	.600	.667

Table 3.7: Results on formality style transfer (XFORMAL IT) for the formal-to-informal (F → I) and informal-to-formal (I → F) directions. Competitors' scores reported from Sarti and Nissim (2022).

3.2.3 Experiments

Currently, there is a very limited availability of datasets for a solid evaluation of the broad capabilities these general-purpose models possess. This is true for English but especially for the Italian language, although the community is moving towards this direction [233]. To evaluate our model we decided to follow the same evaluation protocol proposed in Sarti and Nissim [218]. Compared to their approach, we do not perform any training on the downstream tasks, i.e., we perform just the evaluation on the test set in a zero-shot fashion by providing to the model a textual description of the task (e.g., *"Riassumi il seguente articolo"*). We compared the performance of our model on standard Italian benchmarks for summarization (NewsSum-IT), question answering (SQuAD-IT), and style transfer (XFORMAL IT).

Compared to Sarti and Nissim [218], we do not include the Wikipedia for Italian Text Summarization (WITS) corpus [234] since Wikipedia is included in the original training corpus of LLaMA [205]. We also omitted the news style transfer task between "Il Giornale" to "La Repubblica" (and vice-versa) based on CHANGE-IT [235], since Camoscio has no concepts of "Il Giornale" or "La Repubblica" styles (i.e., it was never exposed during training or finetuning to this kind of articles, although we recognize it might be interesting to analyze this in a few-shot setting). We describe in the next paragraphs the three datasets used for the evaluation.

News Summarization. We evaluate the news article summarization capabilities of Camoscio using the dataset NewSum-IT proposed by Sarti and Nissim [218]. This dataset is obtained by merging two newspaper sources ("Fanpage.it" and "Il Post") scraped by the Applied Recognition Technology Laboratory[7] and available on the Hugging Face Hub [68]. We used only the test split for the zero-shot evaluation and asked the model to generate an answer given the `instruction` *"Dopo aver letto il testo qui sotto, riassumilo adeguatamente."* provided in the textual prompt and the news text provided as `input` (complete prompt as explained in §3.2.2). We use the same evaluation metrics of Sarti and Nissim [218] and report the average across the two newspapers as in their work.

Question Answering. To assess the model performance on extractive question answering, we used the SQuAD-IT dataset [227]. This dataset is composed of sets of paragraphs, questions, and answers derived from the original SQuAD dataset [236] via machine translation and subsequent filtering of problematic instances. As for the previous datasets, we used just the test split for zero-shot evaluation. The model is

[3] `https://huggingface.co/antoniocappiello/bert-base-italian-uncased-squad-it`
[7] `https://huggingface.co/ARTeLab`

asked to generate an answer given the `instruction` *"Dopo aver letto il paragrafo qui sotto, rispondi correttamente alla successiva domanda"*. We evaluated the generated answers using the script from Sarti and Nissim [218]. Furthermore, we also used an additional metric "ChatGPT Exact Match" to better assess the performance. We explain this metric in the following subsubsection "Evaluation Metrics".

Formality Style Transfer. We assess the style transfer capabilities of Camoscio using the Italian subset of the XFORMAL dataset [237], hereafter referred to as XFORMAL-IT. The dataset consists of forum messages from the GYAFC corpus [238] automatically translated covering several topics (entertainment, music, family, and relationships). The test set is constructed by using crowdworkers via Amazon Mechanical Turk to collect formal-informal pairs directly in Italian. The model is evaluated in both style transfer directions (Formal to Informal and Informal to Formal). We use only the test split for the zero-shot evaluation and ask the model to generate an answer given the `instruction` *"Dato il seguente testo scritto in modo formale, riscrivilo in modo informale."* and vice versa according to the style transfer direction.

Evaluation Metrics

We use the same evaluation protocol and scripts of Sarti and Nissim [218]. Specifically, for evaluating lexical matches, we rely on the language-independent ROUGE metric proposed by Lin [239] in the variants unigram (R1), bigram (R2), and Longest Common Subsequence (RL). To gauge semantic correspondence, we employ the trained BERTScore metric [240] with a widely used BERT model pre-trained on Italian[8] and the same baseline scores as Sarti and Nissim [218]. Following previous works, for evaluating the Question-Answering task we employ exact-match (EM) and F1-score (F1). However, since Camoscio is not trained on the output distribution of the question-answering dataset, these metrics will fail to assess the correctness of the output since the EM will count as zero even with a correct output but different wording. To account for these variations, we used an approach similar to Zheng et al. [241] that leverages an external LM (in our case *gpt-3.5-turbo*) to judge whether the answer provided by a model is correct (1) or not (0) given the question and the ground-truth answer. We refer to this metric as Exact Match via ChatGPT (EM-GPT) and explain it with additional details in Appendix D.1.1.

Results and Discussion

Question Answering. Table 3.6 shows the results of Camoscio compared to other methods used in the literature. We observe that the metrics commonly used for the task (Exact Match and F1) are very low compared to all the other models. Although this is generally expected since we are comparing trained models with an untrained one, the exact match score is suspiciously low. Looking at the output responses, we noted that Camoscio produces correct but wordy answers (e.g., *"La crisi petrolifera del 1973 è iniziata nell'ottobre 1973."* instead of *"ottobre 1973"*) making the system to perform bad on this score despite the fact that it produces correct answers. Since all the other systems are trained on the datasets, they are aligned with the expected target distribution and the exact match metric is an effective choice. Nevertheless, when it comes to the zero-shot configuration in Camoscio, this conventional metric fails to accurately capture the true performance of the task.

To this end, we evaluated the model also with standard evaluation metrics for generative models (R1, R2, RL, BS). However, we also observe in this case low scores despite the fact that a qualitative examination of

[8] dbmdz/bert-base-italian-xxl-uncased

	NewsSum-IT			
	R1	R2	RL	BS
mBART Large [9,10]	.377	.194	.291	-
mT5 Small	.323	.150	.248	.375
mT5 Base	.340	.161	.262	.393
IT5 Small	.330	.155	.258	.386
IT5 Base	.339	.160	.263	.044
IT5 Large	.251	.101	.195	.315
Camoscio-7b **(0-shot)**	.250	.104	.174	.190

Table 3.8: Results on NewSum-IT

the provided answers suggests an overall higher quality. This is possibly due to the different lengths between the produced answers (long) and the ground truth (short) and reinforces the necessity of developing a more precise metric to accurately gauge task performance.

For this purpose, we used instead the metric *Exact Match via ChatGPT* explained in §3.2.3. This metric shows that the actual zero-shot performance of Camoscio is in line with the other trained models (.576) and it is also way higher compared to the original EM metric (.077), confirming the need for another type of metric to evaluate the task in the zero-shot setting. Results also show that the EM-GPT metric of trained models correlates well with the existing EM metric, even though with a little marginal difference. This suggests that this metric could serve as an approximate estimation of the model's actual performance, although it might be subject to bias according to the model used for estimation.

Style Transfer & Summarization. Tables 3.7 and 3.8 show results respectively for the formality style transfer and news summarization task. We can observe that the zero-shot performance of Camoscio in both tasks is competitive with trained models. According to the model and training dataset, these latter might achieve slightly better scores at the expense of a less generalist model. Looking at the qualitative results, we note however that the summarization task on "Il Post" and "Fanpage" is affected by some common failure cases.

Failure Cases. The most common failure case consists of the model not producing an answer at all after the input prompt (4.93% of cases on "Il Post" and 21.16% cases on "Fanpage"). We think that it might be due to the input document of these examples being too long and out of distribution compared to the training documents seen in the instruction-tuning dataset (max length 256 tokens). This might confuse the model and trigger the generation of the end-of-sentence token. Other failure cases include the model responding with a copy of the instruction prompt. For example, we found that in 0.75% of the cases in the "Il Post" split of NewSum-IT, when asked to summarize the article the model responded with *"Questo articolo non è più commentabile. Abbonati al Post per commentare le altre notizie."*. We found this behavior peculiar, considering that the input prompt never mentions that articles were collected from "Il Post". Looking carefully at the dataset, we found that these documents were not correctly scraped from the web and the model was simply verbatim copying the input error message above (i.e., the message is already summarized and the model is proposing it as-is).

[6] `https://huggingface.co/ARTeLab/mbart-summarization-ilpost`
[7] `https://huggingface.co/ARTeLab/mbart-summarization-fanpage`

Open-ended generation. We show here some qualitative results on the task of open-ended generation. Compared to the other tasks above, in this task we provide the model with just the `instruction` without any specific `input`. As a tribute to GePpeTto [216], the first generative language model for Italian, we pose to the model the following `instruction`: *"Riassumi la storia di Pinocchio"*. The model answered: *"La storia di Pinocchio racconta di un bambino di nome Pinocchio che vive con i suoi genitori in un piccolo villaggio. Un giorno, Pinocchio incontra un burattino magico che gli dona la possibilità di diventare un vero bambino. Pinocchio inizia una serie di avventure mentre cerca di diventare un vero bambino. Durante il suo viaggio, Pinocchio fa amicizia con molti personaggi e impara molte lezioni di vita. Alla fine, Pinocchio raggiunge il suo obiettivo e diventa un vero bambino"*. At first glance the answer might seem correct, it is grammatically accurate and fluent in Italian. However, knowing the story, the model makes several factual errors like characterizing Pinocchio as a kid (instead of a wooden puppet) living with parents (instead of Geppetto) and introducing a nonexistent magic puppet. Overall this example highlights that, although promising, these models have well-known limitations like hallucinations, factual errors, and several kinds of biases [242–244]. Consequently, it is essential to exercise caution when utilizing them, keeping these limitations in mind.

3.2.4 Conclusion

In this section, we introduced Camoscio, a 7 billion instruction-tuned model for the Italian language, together with its Italian instruction-tuning dataset. Results show that the zero-shot performance of Camoscio on several downstream tasks in Italian is competitive with existing models specifically finetuned for those tasks. Despite the known limitations of these kinds of models, this is a first step towards a generalist model capable of performing a wide range of tasks in Italian without explicit finetuning. This is particularly relevant especially in several domains where data is scarce or not available (e.g., medical domain). In an effort to democratize the available and open resources for the Italian language, we release all the artifacts (code, dataset, model) to the community.

Limitations

Results shown in the section highlight zero-shot performance competitive with existing finetuned models on three different tasks: summarization (NewsSum-IT), question answering (SQuAD-IT), and style transfer (XFORMAL IT). However, it is unclear whether this is true also for other tasks, especially those out of training distribution of the instruction-tuning dataset (see Figure 3.5). Evaluating and thoroughly assessing the performance of these kinds of models is still an open research question. In addition to this, as already mentioned, the model suffers from common problems that affect language models such as hallucinations, factual errors, and several kinds of biases.

3.3 Conclusion Efficient LLMs

In this chapter, we examined strategies to enhance the efficiency of large language models, focusing on accelerating inference and developing resource-effective, language-specific models. The approaches discussed emphasize that efficiency is not solely about reducing computational requirements but also about making LLMs more accessible and practical for real-world applications.

The introduction of parallel decoding algorithms represented a significant advancement in addressing the bottleneck of sequential text generation inherent in autoregressive models. By leveraging parallelization techniques, it became possible to generate text significantly faster without compromising the quality of the output. These algorithms demonstrated that efficiency gains can be achieved not only through hardware advancements but also through innovations in the computational paradigms used during inference.

A complementary focus on language-specific instruction tuning, exemplified by the development of the Italian instruction-tuned model Camoscio, further highlighted the potential of efficient, parameter-reduction techniques such as LoRA (Low-Rank Adaptation). This approach demonstrated that high-quality, domain-specific models can be created with limited resources, making LLM technology more inclusive and accessible for underrepresented languages and domains. The success of Camoscio underscores that efficiency can coexist with effectiveness when models are tailored to specific use cases.

Collectively, the strategies presented in this chapter reveal that achieving efficiency in LLMs requires a balanced approach that combines algorithmic innovations with thoughtful resource management. Accelerating inference through parallel decoding directly addresses the practical challenges of deploying LLMs in time-sensitive scenarios, while language-specific tuning offers a pathway for democratizing access to these powerful tools.

In conclusion, enhancing the efficiency of LLMs involves more than just reducing computational costs—it requires rethinking how these models are trained, fine-tuned, and deployed. The methods outlined in this chapter demonstrate that efficiency and effectiveness are not mutually exclusive, and by leveraging innovations such as parallel decoding and parameter-efficient fine-tuning, LLMs can be made faster, more adaptable, and more accessible to diverse user needs and contexts. These approaches provide a roadmap for future work aimed at optimizing the balance between performance, cost, and accessibility in the deployment of LLMs.

Chapter 4

Reliable Large Language Models

As Large Language Models (LLMs) continue to advance and find applications in increasingly critical domains, the issue of their reliability has come to the forefront of AI research. While these models demonstrate impressive capabilities, they are also prone to generating plausible but factually incorrect information, a phenomenon known as hallucination [17, 18]. This chapter focuses on our contributions to enhancing the reliability of LLMs, with a particular emphasis on uncertainty quantification methods.

The growing deployment of LLMs in high-stakes applications, such as healthcare, legal systems, and financial services, underscores the critical need for models that can not only generate accurate responses but also provide reliable measures of their confidence. Uncertainty quantification in LLMs is a challenging task, given the complex nature of these models and the diverse range of tasks they perform. Our work in this chapter addresses this challenge, aiming to develop more trustworthy and dependable language models.

We begin by presenting a comprehensive assessment of current uncertainty quantification methods for LLMs (§4.1). This study critically examines existing techniques and their evaluation protocols, identifying inconsistencies and limitations that have hindered progress in this area. By proposing improved methodologies for assessing uncertainty estimation in LLMs, we lay the groundwork for more robust and reliable evaluation practices in the field.

Building on this foundation, we explore novel approaches to combine uncertainty estimation methods (§4.2). Our research demonstrates that strategically integrating simple, computationally efficient uncertainty estimation techniques can match or even surpass the performance of more complex methods.

4.1 Evaluating Uncertainty in Large Language Models

This section has been rendered unavailable due to the use of data protected by industrial secrecy, following Article 18, paragraph 11 of the Sapienza PhD Regulation, as issued under Rectoral Decree no. 1150 dated 20/05/2024.

4.2 Effective Uncertainty Quantification in Large Language Models

This section has been rendered unavailable due to the use of data protected by industrial secrecy, following Article 18, paragraph 11 of the Sapienza PhD Regulation, as issued under Rectoral Decree no. 1150 dated 20/05/2024.

4.3 Conclusion Reliable LLMs

This chapter addressed the critical dimension of enhancing the reliability of large language models, focusing on their ability to produce trustworthy outputs while quantifying and mitigating uncertainty. As LLMs are increasingly deployed in high-stakes domains, reliability becomes a pivotal challenge, particularly in the face of issues like hallucinations—situations where models produce fluent yet factually incorrect information.

The first section investigated methods to evaluate uncertainty in LLM outputs. Through a comprehensive assessment of existing uncertainty quantification (UQ) methods and their evaluation protocols, this work identified key gaps in the consistency of current approaches. It highlighted the limitations of popular metrics, such as substring-matching-based measures, when applied to tasks like generative question answering. This analysis underscored the need for robust evaluation practices tailored to the specific characteristics of LLM-generated outputs.

Building on these insights, the second section explored novel methods to enhance uncertainty quantification. By combining complementary UQ techniques, it was shown that computationally lightweight methods can achieve performance comparable to more complex, resource-intensive approaches. This advancement provides a practical pathway to improve model reliability without substantially increasing computational costs. Furthermore, this work examined the relationship between various UQ methods, offering new perspectives on how their strengths can be synergistically leveraged.

In conclusion, the findings in this chapter underscore the importance of robust uncertainty estimation as a cornerstone of reliable LLM deployment. While significant progress has been made in detecting and quantifying uncertainty, challenges remain in scaling these methods to diverse tasks and modalities. The approaches outlined here provide a foundation for developing more reliable and trustworthy LLMs, particularly in critical domains where errors carry substantial consequences. By focusing on scalable and interpretable UQ techniques, this work contributes to making LLMs safer and more dependable in real-world applications.

Chapter 5

Conclusion

This thesis has explored the works carried out during the doctoral studies under three critical dimensions of Large Language Models (LLMs): **Effectiveness, Efficiency**, and **Reliability**. Each direction proposes several works that have been published addressing key challenges in the area.

Key Contributions and Findings

In the domain of **Effectiveness**, our work has made several key contributions. The introduction of instruction tuning [1] demonstrated that smaller models could outperform much larger ones on zero-shot tasks through explicit multitask training, leading to this technique becoming a standard part of modern LLM training pipelines. The development of PromptSource [2] provided the NLP community with essential tools for standardizing prompt engineering and enabling collaborative research. Additionally, our work on Multimodal Neural Databases [3] extended LLM capabilities beyond text, introducing a framework for handling complex database-like queries across different modalities.

Our research on **Efficiency** tackled two major challenges. First, we introduced novel parallel decoding algorithms [4] that significantly speed up text generation without compromising output quality. This work helped establish the field of speculative decoding, now an active research area for accelerating LLM inference. Second, through the development of Camoscio [5], we demonstrated how parameter-efficient fine-tuning techniques could create high-quality language-specific models with limited computational resources, providing a blueprint for developing LLMs for lower-resource languages.

In addressing **Reliability**, we developed systematic approaches to evaluate and improve uncertainty quantification in LLMs. Our comprehensive assessment of uncertainty quantification methods [6] identified key limitations in existing evaluation protocols and proposed improved methodologies. Building on this foundation, we explored novel approaches to combine uncertainty estimation methods [7], showing that strategically integrating simple techniques can match or exceed the performance of more complex methods while maintaining computational efficiency.

Implications and Impact

The findings of this thesis hold significant implications for both academic research and real-world applications. Instruction tuning, a technique concurrently introduced in our work [1] and Wei et al. [14], has emerged as a cornerstone of the standard training pipeline for language models and is now part of the fine-tuning process following the self-supervised stage on unannotated corpora [21]. At the time of writing, the paper has garnered over 1500 citations, underscoring the widespread adoption and relevance of instruction tuning in directly aligning models with human feedback. This work challenges the predominant assumption that scaling self-supervision alone drives large language model performance. Instead, it highlights how task-specific optimizations can empower smaller, more efficient models to achieve unparalleled performance, aligning the models with humans.

This thesis also introduces advancement in efficiency through the concept of parallel decoding, enabling the simultaneous generation of multiple tokens without requiring additional training—a method often termed Jacobi decoding. This innovation has catalyzed an active research area focused on accelerating LLM generation [131–133]. Speculative decoding, which decodes multiple tokens in parallel using a smaller auxiliary model [134], has since built upon these foundations. With over 60 citations, this body of work has demonstrated the potential for significant reductions in computational barriers, democratizing access to advanced AI technologies and fostering inclusivity. Additionally, our work on Camoscio demonstrates the feasibility of developing language-specific models using limited computational resources. This achievement underscores the potential for cost-effective advancements in natural language processing tailored to specific linguistic contexts. Notably, the impact of this work is reflected in its reception, with over 40 citations at the time of writing, showcasing significant community interest and affirming the value of open-source models in driving collaborative progress in the field.

Addressing reliability, this research underscores the importance of understanding when a language model accurately "knows" its outputs. Despite their remarkable capabilities, LLMs are prone to generating plausible yet incorrect information, a phenomenon known as hallucination [17, 18]. In high-stakes domains such as healthcare, law, and finance, it is critical to quantify and communicate a model's confidence. The methodologies developed herein lay the groundwork for robust uncertainty quantification, equipping practitioners with tools to evaluate and mitigate risks in LLM applications. These advancements collectively pave the way for the development of effective, efficient, and reliable AI systems that address real-world challenges while fostering equitable and sustainable technological progress.

Limitations

Despite significant advancements achieved in the dimensions of Effectiveness, Efficiency, and Reliability, this thesis reveals several limitations inherent in the proposed approaches and methodologies. Recognizing these limitations not only contextualizes the achievements of this work but also highlights promising avenues for future exploration and development.

While instruction tuning and prompt engineering have significantly improved task performance, their applicability to highly domain-specific or non-English tasks remains limited. These models are heavily reliant on the quality and diversity of prompts, which cannot fully encapsulate the diversity of real-world applications. Furthermore, instruction tuning, though a powerful tool, has the potential to negatively impact downstream performance when prompts do not adequately align with the nuanced requirements of specific tasks. This challenge becomes particularly pronounced when addressing tasks in languages or domains with limited high-quality data.

The development of multimodal databases represents an exciting frontier; however, key features, such as the ability to dynamically update database information, remain underexplored. In traditional database systems, users expect seamless capabilities to add, remove, or modify entries, but these operations are not straightforward in the current paradigm. The preliminary prototype developed here, while spanning two modalities, is constrained in scope and application. Expanding this approach to consider not only documents but also their associated metadata could significantly enhance its utility and versatility.

On the efficiency front, the proposed parallel decoding algorithms offer noteworthy acceleration in inference. However, their application thus far has been confined to encoder-decoder models, leaving their broader scalability untested. Extending this approach to larger and more diverse models presents challenges that may require advancements in underlying software and hardware frameworks to ensure tokens are effectively decoded in parallel. The development of an Italian-language instruction-tuned LLM highlighted cost-effective strategies for creating language-specific models, yet it also underscored difficulties in scaling such methods to under-resourced languages. One critical issue is the reliance on datasets derived from machine translation, which risks introducing linguistic bias and failing to represent the cultural and contextual nuances of the target language. Ideally, models for such languages would benefit from a robust pretraining phase using native data, followed by instruction tuning. However, the scarcity of high-quality data and the prohibitive computational demands of pretraining present significant barriers.

In addressing reliability, the assessment and integration of uncertainty quantification methods revealed persistent challenges. Even state-of-the-art techniques struggle to provide consistent and reliable estimates of uncertainty. Our findings highlight that evaluating uncertainty is itself a non-trivial endeavor, and careful, well-designed evaluation protocols are critical to driving the development of more effective methods. Despite promising results in leveraging uncertainty estimation to mitigate issues like hallucinations, these methods have yet to offer a comprehensive solution. Hallucinations, where models generate fluent but factually incorrect information, remain a pervasive and critical issue, particularly as LLMs are increasingly deployed in high-stakes domains.

Future Directions

Looking ahead, our work opens several promising research directions that intersect with crucial challenges in the field of Large Language Models. The quest for greater effectiveness remains a fundamental pursuit, particularly as we move beyond text-only applications [245]. The future of LLMs lies in their ability to reason across different modalities and handle increasingly complex tasks. This evolution requires not just architectural innovations, but also new approaches to training that can better align models with human instructions and intents [1, 14, 15].

The efficiency challenges inherent in LLM development reveal a fundamental trade-off: increasing model capabilities typically comes at the cost of greater computational complexity [246, 247]. Future research must continue to explore novel approaches for accelerating both training and inference, while also developing more resource-efficient architectures [4, 134]. This includes not only algorithmic improvements but also hardware-aware optimizations and parameter-efficient adaptation techniques [248]. These advancements are crucial for democratizing access to LLMs and reducing their environmental impact [249].

The reliability dimension perhaps represents the most critical avenue for future research. As LLMs continue to be deployed in high-stakes applications, from healthcare to legal systems, the ability to accurately assess their confidence and detect potential errors becomes paramount [250, 251]. Future work must focus on developing more robust uncertainty quantification methods and better techniques for detecting and preventing hallucinations. This includes not only improving existing methods but also developing new approaches that can provide interpretable and actionable measures of model reliability.

These future directions are deeply interconnected - advances in one area often enable or require progress in others. For instance, more effective multimodal capabilities might require new efficiency techniques to handle the increased computational demands, while improved reliability measures may need better base models. As the field continues to evolve, maintaining this holistic perspective on effectiveness, efficiency, and reliability will be crucial for developing LLMs that can be both powerful and trustworthy.

Final Remarks

The contributions of this thesis coincide with a transformative period in AI, where LLMs have matured from experimental prototypes into powerful tools with wide-ranging applications. By addressing fundamental challenges in effectiveness, efficiency, and reliability, this work offers a cohesive vision for the next generation of LLMs. The techniques and insights presented in this thesis provide a foundation for creating more capable, efficient, and reliable language models.

As LLMs continue to evolve and find applications in increasingly critical domains, the importance of balancing effectiveness, efficiency, and reliability will only grow. We hope that the methods and frameworks developed in this thesis will contribute to the responsible advancement of LLM technology, ultimately helping to realize its potential for positive societal impact while mitigating associated risks and limitations.

Bibliography

[1] Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=9Vrb9D0WI4`.

[2] Stephen Bach, Victor Sanh, Zheng Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-david, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Fries, Maged Al-shaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Dragomir Radev, Mike Tian-jian Jiang, and Alexander Rush. PromptSource: An integrated development environment and repository for natural language prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 93–104, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-demo.9. URL `https://aclanthology.org/2022.acl-demo.9`.

[3] Giovanni Trappolini, Andrea Santilli, Emanuele Rodolà, Alon Halevy, and Fabrizio Silvestri. Multimodal neural databases. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 2619–2628, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394086. doi: 10.1145/3539618.3591930. URL `https://doi.org/10.1145/3539618.3591930`.

[4] Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodola. Accelerating transformer inference for translation via parallel decoding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12336–12355, Toronto, Canada, July 2023. Association for Computational Linguistics. URL `https://aclanthology.org/2023.acl-long.689`.

[5] Andrea Santilli and Emanuele Rodolà. Camoscio: an Italian instruction-tuned LLaMA. In *CLiC-it 2023: 9th Italian Conference on Computational Linguistics*, volume 3596 of *CEUR Workshop Proceedings*, Venice, Italy, 2023. CEUR-WS.org. URL `https://ceur-ws.org/Vol-3596/paper44.pdf`.

[6] Andrea Santilli, Miao Xiong, Michael Kirchhof, Pau Rodrigues, Federico Danieli, Xavier Suau, Luca Zappella, Sinead Williamson, and Adam Goliński. On a spurious interaction between uncertainty scores and answer evaluation metrics in generative qa tasks. In *Neurips Safe Generative AI Workshop 2024*, 2024. URL `https://openreview.net/forum?id=jGtL0JFdeD`.

[7] Miao Xiong, Andrea Santilli, Michael Kirchhof, Adam Goliński, and Sinead Williamson. Efficient and effective uncertainty quantification for LLMs. In *Neurips Safe Generative AI Workshop 2024*, 2024. URL `https://openreview.net/forum?id=QKRLH57ATT`.

[8] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

[9] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis

Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

[10] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27 (3):379–423, 1948. doi: https://doi.org/10.1002/j.1538-7305.1948.tb01338.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1948.tb01338.x.

[11] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000. URL https://proceedings.neurips.cc/paper_files/paper/2000/file/728f206c2a01bf572b5940d7d9a8fa4c-Paper.pdf.

[12] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. OpenAI Blog, 2019. URL https://openai.com/blog/better-language-models/.

[13] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[14] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=gEZrGCozdqR.

[15] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[16] OpenAI. Chatgpt: An ai language model, 2024. URL `https://openai.com/chatgpt`. Accessed: 2024-10-21.

[17] Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*, 2023.

[18] Nuno M Guerreiro, Duarte M Alves, Jonas Waldendorf, Barry Haddow, Alexandra Birch, Pierre Colombo, and André FT Martins. Hallucinations in large multilingual translation models. *Transactions of the Association for Computational Linguistics*, 11:1500–1517, 2023.

[19] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[20] Jason Wei et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[21] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*, 2023.

[22] Jason Phang, Herbie Bradley, Leo Gao, Louis Castricato, and Stella Biderman. Eleutherai: Going beyond "open science" to "science in the open", 2022. URL `https://arxiv.org/abs/2210.06413`.

[23] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`.

[24] Boseop Kim, HyoungSeok Kim, Sang-Woo Lee, Gichang Lee, Donghyun Kwak, Dong Hyeon Jeon, Sunghyun Park, Sungju Kim, Seonhoon Kim, Dongpil Seo, et al. What changes can large-scale language models bring? intensive study on hyperclova: Billions-scale korean generative pretrained transformers. *arXiv preprint arXiv:2109.04650*, 2021.

[25] Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. Unifiedqa: Crossing format boundaries with a single QA system. *CoRR*, abs/2005.00700, 2020. URL `https://arxiv.org/abs/2005.00700`.

[26] Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. Crossfit: A few-shot learning challenge for cross-task generalization in nlp. *arXiv preprint arXiv:2104.08835*, 2021. URL `https://arxiv.org/abs/2104.08835`.

[27] Ethan Perez, Douwe Kiela, and Kyunghyun Cho. True few-shot learning with language models. *CoRR*, abs/2105.11447, 2021. URL `https://arxiv.org/abs/2105.11447`.

[28] Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models, 2021.

[29] Laria Reynolds and Kyle McDonell. Prompt programming for large language models: Beyond the few-shot paradigm. *CoRR*, abs/2102.07350, 2021. URL `https://arxiv.org/abs/2102.07350`.

[30] Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.437. URL `https://aclanthology.org/2020.emnlp-main.437`.

[31] Stephen H. Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, Maged S. Al-shaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Xiangru Tang, Mike Tian-Jian Jiang, and Alexander M. Rush. Promptsource: An integrated development environment and repository for natural language prompts, 2022.

[32] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

[33] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *CoRR*, abs/2104.08691, 2021. URL `https://arxiv.org/abs/2104.08691`.

[34] BIG-bench collaboration. Beyond the imitation game: Measuring and extrapolating the capabilities of language models. *In preparation*, 2021. URL `https://github.com/google/BIG-bench/`.

[35] Rich Caruana. Multitask learning. *Mach. Learn.*, 28(1):41–75, 1997. doi: 10.1023/A:1007379606734. URL `https://doi.org/10.1023/A:1007379606734`.

[36] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Filnand, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pages 160–167. ACM, 2008. doi: 10.1145/1390156.1390177. URL `https://doi.org/10.1145/1390156.1390177`.

[37] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple NLP tasks. *CoRR*, abs/1611.015collin87, 2016. URL `http://arxiv.org/abs/1611.01587`.

[38] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The natural language decathlon: Multitask learning as question answering. *CoRR*, abs/1806.08730, 2018. URL `http://arxiv.org/abs/1806.08730`.

[39] Tu Vu, Tong Wang, Tsendsuren Munkhdalai, Alessandro Sordoni, Adam Trischler, Andrew Mattarella-Micke, Subhransu Maji, and Mohit Iyyer. Exploring and predicting transferability across NLP tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7882–7926, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.635. URL `https://aclanthology.org/2020.emnlp-main.635`.

[40] Ruiqi Zhong, Kristy Lee, Zheng Zhang, and Dan Klein. Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections. *CoRR*, abs/2104.04670, 2021. URL `https://arxiv.org/abs/2104.04670`.

[41] Sinong Wang, Han Fang, Madian Khabsa, Hanzi Mao, and Hao Ma. Entailment as few-shot learner. *CoRR*, abs/2104.14690, 2021. URL `https://arxiv.org/abs/2104.14690`.

[42] Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online, April 2021. Association for Computational Linguistics. URL `https://aclanthology.org/2021.eacl-main.20`.

[43] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Natural instructions: Benchmarking generalization to new tasks from natural language instructions. *CoRR*, abs/2104.08773, 2021. URL `https://arxiv.org/abs/2104.08773`.

[44] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners, 2021.

[45] Albert Webson and Ellie Pavlick. Do prompt-based models really understand the meaning of their prompts?, 2021. URL `https://arxiv.org/abs/2109.01247`.

[46] Robert L Logan, Ivana Balažević, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. Cutting down on prompts and parameters: Simple few-shot learning with language models. *arXiv preprint arXiv:2106.13353*, 2021.

[47] Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.171. URL `https://aclanthology.org/2020.findings-emnlp.171`.

[48] Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander M. Rush, and Thomas Wolf. Datasets: A community library for natural language processing. *emnlp*, 2021.

[49] Adina Williams, Tristan Thrush, and Douwe Kiela. Anlizing the adversarial natural language inference dataset. *arXiv preprint arXiv:2010.12729*, 2020.

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

[51] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR, 2018.

[52] Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, September 2021. URL https://doi.org/10.5281/zenodo.5371628.

[53] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WINOGRANDE: an adversarial winograd schema challenge at scale. *CoRR*, abs/1907.10641, 2019. URL http://arxiv.org/abs/1907.10641.

[54] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[55] Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. Cloze-driven pretraining of self-attention networks. *arXiv preprint arXiv:1903.07785*, 2019.

[56] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pages 4171–4186, 2019.

[57] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *CoRR*, abs/2107.13586, 2021. URL https://arxiv.org/abs/2107.13586.

[58] Timo Schick and Hinrich Schütze. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.185. URL https://aclanthology.org/2021.naacl-main.185.

[59] Teven Le Scao and Alexander Rush. How many data points is a prompt worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2627–2636, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.208. URL https://aclanthology.org/2021.naacl-main.208.

[60] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. 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)*,

pages 3816–3830, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.295. URL https://aclanthology.org/2021.acl-long.295.

[61] Ethan Perez, Douwe Kiela, and Kyunghyun Cho. True few-shot learning with language models. In *NeurIPS*, 2021. URL https://arxiv.org/abs/2105.11447.

[62] Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. *CoRR*, abs/2102.09690, 2021. URL http://arxiv.org/abs/2102.09690.

[63] Albert Webson and Ellie Pavlick. Do prompt-based models really understand the meaning of their prompts? *ArXiv*, abs/2109.01247, 2021.

[64] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization. *CoRR*, abs/2110.08207, 2021. URL http://arxiv.org/abs/2110.08207.

[65] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. *CoRR*, abs/2109.01652, 2021. URL http://arxiv.org/abs/2109.01652.

[66] Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. Metaicl: Learning to learn in context. *CoRR*, abs/2110.15943, 2021. URL http://arxiv.org/abs/2110.15943.

[67] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*, 2021.

[68] Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.emnlp-demo.21.

[69] Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, Ramakanth Pasunuru, Sam Shleifer, Punit Singh Koura, Vishrav Chaudhary, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Zornitsa Kozareva, Mona T.

Diab, Veselin Stoyanov, and Xian Li. Few-shot learning with multilingual language models. *CoRR*, abs/2112.10668, 2021. URL `http://arxiv.org/abs/2112.10668`.

[70] Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online, April 2021. Association for Computational Linguistics. URL `https://doi.org/10.18653/v1/2021.eacl-main.20`.

[71] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.346. URL `https://aclanthology.org/2020.emnlp-main.346`.

[72] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. URL `https://doi.org/10.18653/v1/2021.emnlp-main.243`.

[73] Guanghui Qin and Jason Eisner. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.410. URL `https://aclanthology.org/2021.naacl-main.410`.

[74] Mariana Neves and Jurica Ševa. An extensive review of tools for manual annotation of documents. *Briefings in bioinformatics*, 22(1):146–163, 2021.

[75] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. GATE: an architecture for development of robust HLT applications. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 168–175, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. URL `https://doi.org/10.3115/1073083.1073112`.

[76] Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France, April 2012. Association for Computational Linguistics. URL `https://aclanthology.org/E12-2021`.

[77] David Salgado, Martin Krallinger, Marc Depaule, Elodie Drula, Ashish V. Tendulkar, Florian Leitner, Alfonso Valencia, and Christophe Marcelle. MyMiner: a web application for computer-assisted biocuration and text annotation. *Bioinformatics*, 28(17):2285–2287, 2012. URL `https://doi.org/10.1093/bioinformatics/bts435`.

[78] Chih-Hsuan Wei, Hung-Yu Kao, and Zhiyong Lu. PubTator: a web-based text mining tool for assisting biocuration. *Nucleic Acids Research*, 41(W1):W518–W522, 2013. URL https://doi.org/10.1093/nar/gkt441.

[79] Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. WebAnno: A flexible, web-based and visually supported system for distributed annotations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 1–6, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://aclanthology.org/P13-4001.

[80] Wei-Te Chen and Will Styler. Anafora: A web-based general purpose annotation tool. In *Proceedings of the 2013 NAACL HLT Demonstration Session*, pages 14–19, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL https://aclanthology.org/N13-3004.

[81] Richard Eckart de Castilho, Éva Mújdricza-Maydt, Seid Muhie Yimam, Silvana Hartmann, Iryna Gurevych, Anette Frank, and Chris Biemann. A web-based tool for the integrated annotation of semantic and syntactic structures. In *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*, pages 76–84, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL https://aclanthology.org/W16-4011.

[82] Jan Wira Gotama Putra, Simone Teufel, Kana Matsumura, and Takenobu Tokunaga. TIARA: A tool for annotating discourse relations and sentence reordering. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6912–6920, Marseille, France, May 2020. European Language Resources Association. URL https://aclanthology.org/2020.lrec-1.854.

[83] Jie Yang, Yue Zhang, Linwei Li, and Xingxuan Li. YEDDA: A lightweight collaborative text span annotation tool. In *Proceedings of ACL 2018, System Demonstrations*, pages 31–36, Melbourne, Australia, July 2018. Association for Computational Linguistics. URL https://doi.org/10.18653/v1/P18-4006.

[84] Michael Stewart, Wei Liu, and Rachel Cardell-Oliver. Redcoat: A collaborative annotation tool for hierarchical entity typing. 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): System Demonstrations*, pages 193–198, Hong Kong, China, November 2019. Association for Computational Linguistics. URL https://doi.org/10.18653/v1/D19-3033.

[85] Bill Yuchen Lin, Dong-Ho Lee, Frank F. Xu, Ouyu Lan, and Xiang Ren. AlpacaTag: An active learning-based crowd annotation framework for sequence tagging. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 58–63, Florence, Italy, July 2019. Association for Computational Linguistics. URL https://doi.org/10.18653/v1/P19-3010.

[86] Yanzeng Li, Bowen Yu, Li Quangang, and Tingwen Liu. FITAnnotator: A flexible and intelligent text annotation system. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, pages

35–41, Online, June 2021. Association for Computational Linguistics. URL https://doi.org/10.18653/v1/2021.naacl-demos.5.

[87] Dong-Ho Lee, Rahul Khanna, Bill Yuchen Lin, Seyeon Lee, Qinyuan Ye, Elizabeth Boschee, Leonardo Neves, and Xiang Ren. LEAN-LIFE: A label-efficient annotation framework towards learning from explanation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 372–379, Online, July 2020. Association for Computational Linguistics. URL https://doi.org/10.18653/v1/2020.acl-demos.42.

[88] Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. Dynabench: Rethinking benchmarking in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4110–4124, Online, June 2021. Association for Computational Linguistics. URL https://doi.org/10.18653/v1/2021.naacl-main.324.

[89] Frédéric Papazian, Robert Bossy, and Claire Nédellec. AlvisAE: a collaborative web text annotation editor for knowledge acquisition. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 149–152, Jeju, Republic of Korea, July 2012. Association for Computational Linguistics. URL https://aclanthology.org/W12-3621.

[90] Philipp Helfrich, Elias Rieb, Giuseppe Abrami, Andy Lücking, and Alexander Mehler. TreeAnnotator: Versatile visual annotation of hierarchical text relations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL https://aclanthology.org/L18-1308.

[91] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1075. URL https://aclanthology.org/D15-1075.

[92] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/K17-1034. URL https://aclanthology.org/K17-1034.

[93] Alessandro Bozzon and Piero Fraternali. Multimedia and multimodal information retrieval. In *Search Computing*, pages 135–155. Springer, 2010.

[94] James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Halevy. From natural language processing to neural databases. In *Proceedings of the VLDB Endowment*, volume 14, pages 1033–1039. VLDB Endowment, 2021.

[95] James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Halevy. Database reasoning over text. 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)*, pages 3091–3104, 2021.

[96] Artsiom Sauchuk, James Thorne, Alon Halevy, Nicola Tonellotto, and Fabrizio Silvestri. On the role of relevance in natural language processing tasks. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1785–1789, 2022.

[97] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913, 2017.

[98] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[99] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. Data2vec: A general framework for self-supervised learning in speech, vision and language. In *International Conference on Machine Learning*, pages 1298–1312. PMLR, 2022.

[100] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[101] Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. *arXiv preprint arXiv:2202.03052*, 2022.

[102] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[103] Joseph M. Hellerstein, Peter J. Haas, and Helen J. Wang. Online aggregation. *SIGMOD Rec.*, 26(2): 171–182, jun 1997. ISSN 0163-5808. doi: 10.1145/253262.253291. URL `https://doi.org/10.1145/253262.253291`.

[104] Alon Y Halevy, Oren Etzioni, AnHai Doan, Zachary G Ives, Jayant Madhavan, Luke K McDowell, and Igor Tatarinov. Crossing the structure chasm. In *CIDR*, 2003.

[105] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12): 1349–1380, 2000. doi: 10.1109/34.895972.

[106] Michael S Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2(1):1–19, 2006.

[107] Yong Rui, Thomas S. Huang, and Shih-Fu Chang. Image retrieval: Current techniques, promising directions, and open issues. *Journal of Visual Communication and Image Representation*, 10(1): 39–62, 1999. ISSN 1047-3203. doi: https://doi.org/10.1006/jvci.1999.0413. URL `https://www.sciencedirect.com/science/article/pii/S1047320399904133`.

[108] Kaiye Wang, Qiyue Yin, Wei Wang, Shu Wu, and Liang Wang. A comprehensive survey on cross-modal retrieval. *arXiv preprint arXiv:1607.06215*, 2016.

[109] Peng Hu, Liangli Zhen, Dezhong Peng, and Pei Liu. Scalable deep multimodal learning for cross-modal retrieval. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, pages 635–644, 2019.

[110] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[111] Jun Rao, Fei Wang, Liang Ding, Shuhan Qi, Yibing Zhan, Weifeng Liu, and Dacheng Tao. Where does the performance improvement come from? -a reproducibility concern about image-text retrieval. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2727–2737, 2022.

[112] Mengjun Cheng, Yipeng Sun, Longchao Wang, Xiongwei Zhu, Kun Yao, Jie Chen, Guoli Song, Junyu Han, Jingtuo Liu, Errui Ding, et al. Vista: vision and scene text aggregation for cross-modal retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5184–5193, 2022.

[113] Tan Yu, Hongliang Fei, and Ping Li. U-bert for fast and scalable text-image retrieval. In *Proceedings of the 2022 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 193–203, 2022.

[114] Tan Yu, Hongliang Fei, and Ping Li. Cross-probe bert for fast cross-modal search. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2178–2183, 2022.

[115] Tan Yu, Yi Yang, Yi Li, Lin Liu, Hongliang Fei, and Ping Li. Heterogeneous attention network for effective and efficient cross-modal retrieval. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 1146–1156, 2021.

[116] Leigang Qu, Meng Liu, Jianlong Wu, Zan Gao, and Liqiang Nie. Dynamic modality interaction modeling for image-text retrieval. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1104–1113, 2021.

[117] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[118] Hangbo Bao, Wenhui Wang, Li Dong, Qiang Liu, Owais Khan Mohammed, Kriti Aggarwal, Subhojit Som, and Furu Wei. Vlmo: Unified vision-language pre-training with mixture-of-modality-experts. *arXiv preprint arXiv:2111.02358*, 2021.

[119] Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, et al. Image as a foreign language: Beit pretraining for all vision and vision-language tasks. *arXiv preprint arXiv:2208.10442*, 2022.

[120] Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, et al. Pali: A jointly-scaled multilingual language-image model. *arXiv preprint arXiv:2209.06794*, 2022.

[121] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*, 2022.

[122] Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. Retrieving and reading: A comprehensive survey on open-domain question answering. *CoRR*, abs/2101.00774, 2021. URL `https://arxiv.org/abs/2101.00774`.

[123] Yingshan Chang, Mridu Narang, Hisami Suzuki, Guihong Cao, Jianfeng Gao, and Yonatan Bisk. Webqa: Multihop and multimodal QA. *CoRR*, abs/2109.00590, 2021. URL `https://arxiv.org/abs/2109.00590`.

[124] Ion Androutsopoulos, Graeme D Ritchie, and Peter Thanisch. Natural language interfaces to databases–an introduction. *Natural language engineering*, 1(1):29–81, 1995.

[125] Fei Li and Hosagrahar V Jagadish. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment*, 8(1):73–84, 2014.

[126] Jichuan Zeng, Xi Victoria Lin, Caiming Xiong, Richard Socher, Michael R Lyu, Irwin King, and Steven CH Hoi. Photon: a robust cross-domain text-to-sql system. *arXiv preprint arXiv:2007.15280*, 2020.

[127] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544, 2013.

[128] Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*, 2015.

[129] Deng Cai, Yan Wang, Lemao Liu, and Shuming Shi. Recent advances in retrieval-augmented text generation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 3417–3419, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387323. doi: 10.1145/3477495.3532682. URL `https://doi.org/10.1145/3477495.3532682`.

[130] Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. Augmented language models: a survey, 2023. URL `https://arxiv.org/abs/2302.07842`.

[131] Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. In *Forty-first International Conference on Machine Learning*.

[132] Yao Teng, Han Shi, Xian Liu, Xuefei Ning, Guohao Dai, Yu Wang, Zhenguo Li, and Xihui Liu. Accelerating auto-regressive text-to-image generation with training-free speculative jacobi decoding. *arXiv preprint arXiv:2410.01699*, 2024.

[133] Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, et al. Efficient large language models: A survey. *Transactions on Machine Learning Research*.

[134] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding, 2022.

[135] Andrea Bacciu, Giovanni Trappolini, Andrea Santilli, Emanuele Rodolà, and Fabrizio Silvestri. Fauno: The italian large language model that will leave you senza parole! In Franco Maria Nardini, Nicola Tonellotto, Guglielmo Faggioli, and Antonio Ferrara, editors, *Proceedings of the 13th Italian Information Retrieval Workshop (IIR 2023), Pisa, Italy, June 8-9, 2023*, volume 3448 of *CEUR Workshop Proceedings*, pages 9–17. CEUR-WS.org, 2023. URL `https://ceur-ws.org/Vol-3448/paper-24.pdf`.

[136] Andrea Bacciu, Cesare Campagnano, Giovanni Trappolini, and Fabrizio Silvestri. DanteLLM: Let's push Italian LLM research forward! In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, editors, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 4343–4355, Torino, Italia, May 2024. ELRA and ICCL. URL `https://aclanthology.org/2024.lrec-main.388`.

[137] Pierpaolo Basile, Elio Musacchio, Marco Polignano, Lucia Siciliani, Giuseppe Fiameni, and Giovanni Semeraro. Llamantino: Llama 2 models for effective text generation in italian language. *ArXiv*, abs/2312.09993, 2023. URL `https://api.semanticscholar.org/CorpusID:266335721`.

[138] Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1045. URL `https://aclanthology.org/D18-1045`.

[139] Xiaodong Liu, Kevin Duh, Liyuan Liu, and Jianfeng Gao. Very deep transformers for neural machine translation. *arXiv preprint arXiv:2008.07772*, 2020.

[140] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[141] Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=B1l8BtlCb`.

[142] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation

system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[143] Young Jin Kim, Marcin Junczys-Dowmunt, Hany Hassan, Alham Fikri Aji, Kenneth Heafield, Roman Grundkiewicz, and Nikolay Bogoychev. From research to production and back: Ludicrously fast neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 280–288, Hong Kong, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5632. URL `https://aclanthology.org/D19-5632`.

[144] Yang Song, Chenlin Meng, Renjie Liao, and Stefano Ermon. Accelerating feedforward computation via parallel nonlinear equation solving. In *International Conference on Machine Learning*, pages 9791–9800. PMLR, 2021.

[145] Xinwei Geng, Xiaocheng Feng, and Bing Qin. Learning to rewrite for non-autoregressive neural machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3297–3308, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.265. URL `https://aclanthology.org/2021.emnlp-main.265`.

[146] Nikolay Savinov, Junyoung Chung, Mikolaj Binkowski, Erich Elsen, and Aaron van den Oord. Step-unrolled denoising autoencoders for text generation. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=T0GpzBQ1Fg6`.

[147] Jiatao Gu and Xiang Kong. Fully non-autoregressive neural machine translation: Tricks of the trade. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 120–133, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.11. URL `https://aclanthology.org/2021.findings-acl.11`.

[148] Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1139. URL `https://aclanthology.org/D16-1139`.

[149] Marjan Ghazvininejad, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. Aligned cross entropy for non-autoregressive machine translation. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3515–3523. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/ghazvininejad20a.html`.

[150] Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. Non-autoregressive machine translation with latent alignments. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1098–1108, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.83. URL `https://www.aclweb.org/anthology/2020.emnlp-main.83`.

[151] Cunxiao Du, Zhaopeng Tu, and Jing Jiang. Order-agnostic cross entropy for non-autoregressive machine translation. In *International Conference on Machine Learning*, pages 2849–2859. PMLR, 2021.

[152] Chenyang Huang, Hao Zhou, Osmar R. Zaïane, Lili Mou, and Lei Li. Non-autoregressive translation with layer-wise prediction and deep supervision. *CoRR*, abs/2110.07515, 2021. URL https://arxiv.org/abs/2110.07515.

[153] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. 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)*, pages 6112–6121, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1633. URL https://aclanthology.org/D19-1633.

[154] Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, and Lei Li. Glancing transformer for non-autoregressive neural machine translation. 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)*, pages 1993–2003, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.155. URL https://aclanthology.org/2021.acl-long.155.

[155] Jongyoon Song, Sungwon Kim, and Sungroh Yoon. AligNART: Non-autoregressive neural machine translation by jointly learning to estimate alignment and translate. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1–14, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.1. URL https://aclanthology.org/2021.emnlp-main.1.

[156] Zhenqiao Song, Hao Zhou, Lihua Qian, Jingjing Xu, Shanbo Cheng, Mingxuan Wang, and Lei Li. switch-GLAT: Multilingual parallel machine translation via code-switch decoder. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=5HvpvYd68b.

[157] Marjan Ghazvininejad, Omer Levy, and Luke Zettlemoyer. Semi-autoregressive training improves mask-predict decoding. *arXiv preprint arXiv:2001.08785*, 2020.

[158] Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. Non-autoregressive machine translation with disentangled context transformer. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5144–5155. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/kasai20a.html.

[159] Yongchang Hao, Shilin He, Wenxiang Jiao, Zhaopeng Tu, Michael Lyu, and Xing Wang. Multi-task learning with shared encoder for non-autoregressive machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3989–3996, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.313. URL https://aclanthology.org/2021.naacl-main.313.

[160] Xiao Shi Huang, Felipe Perez, and Maksims Volkovs. Improving non-autoregressive translation models without distillation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=I2Hw58KHp8O.

[161] Heming Xia, Tao Ge, Furu Wei, and Zhifang Sui. Lossless speedup of autoregressive translation with generalized aggressive decoding, 2022.

[162] Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper/2018/file/c4127b9194fe8562c64dc0f5bf2c93bc-Paper.pdf`.

[163] Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah Smith. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=KpfasTaLUpq`.

[164] Yisheng Xiao, Lijun Wu, Junliang Guo, Juntao Li, Min Zhang, Tao Qin, and Tie-yan Liu. A survey on non-autoregressive generation for neural machine translation and beyond. *arXiv preprint arXiv:2204.09269*, 2022.

[165] Xin Sun, Tao Ge, Furu Wei, and Houfeng Wang. Instantaneous grammatical error correction with shallow aggressive decoding. 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)*, pages 5937–5947, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.462. URL `https://aclanthology.org/2021.acl-long.462`.

[166] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling, 2023.

[167] Sehoon Kim, Karttikeya Mangalam, Jitendra Malik, Michael W. Mahoney, Amir Gholami, and Kurt Keutzer. Big little transformer decoder, 2023.

[168] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *CoRR*, abs/1808.06226, 2018. URL `http://arxiv.org/abs/1808.06226`.

[169] Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2012, Kyoto, Japan, March 25-30, 2012*, pages 5149–5152. IEEE, 2012. doi: 10.1109/ICASSP.2012.6289079. URL `https://doi.org/10.1109/ICASSP.2012.6289079`.

[170] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL `https://aclanthology.org/P16-1162`.

[171] Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia, July 2018. Association for

Computational Linguistics. doi: 10.18653/v1/P18-1007. URL https://aclanthology.org/P18-1007.

[172] Wouter Kool, Herke van Hoof, and Max Welling. Ancestral gumbel-top-k sampling for sampling without replacement. *Journal of Machine Learning Research*, 21(47):1–36, 2020. URL http://jmlr.org/papers/v21/19-985.html.

[173] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

[174] Raj Reddy. *Speech understanding systems: A summary of results of the five-year research effort*. Carnegie Mellon University, 1977.

[175] J.M. Ortega and W.C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 1970. ISBN 9780898719468. URL https://books.google.es/books?id=GA1P9UNnrmMC.

[176] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[177] Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-3302.

[178] Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W16/W16-2301.

[179] Viet Hong Tran, Huyen Vu Thong, Nguyen Van-Vinh, and Trung Le Tien. The English-Vietnamese machine translation system for IWSLT 2015. In *Proceedings of the 12th International Workshop on Spoken Language Translation: Evaluation Campaign*, pages 80–83, Da Nang, Vietnam, December 3-4 2015. URL https://aclanthology.org/2015.iwslt-evaluation.12.

[180] Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. The IIT Bombay English-Hindi parallel corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL https://aclanthology.org/L18-1548.

[181] Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt

Post, Raphael Rubino, Lucia Specia, and Marco Turchi. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 169–214, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W17-4717.

[182] Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzmán, and Angela Fan. The Flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics*, 10:522–538, 2022. doi: 10.1162/tacl_a_00474. URL https://aclanthology.org/2022.tacl-1.30.

[183] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040.

[184] Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W18-6319.

[185] Prajit Ramachandran, Tom Le Paine, Pooya Khorrami, Mohammad Babaeizadeh, Shiyu Chang, Yang Zhang, Mark A. Hasegawa-Johnson, Roy H. Campbell, and Thomas S. Huang. Fast generation for convolutional autoregressive models. *CoRR*, abs/1704.06001, 2017. URL http://arxiv.org/abs/1704.06001.

[186] Jörg Tiedemann and Santhosh Thottingal. OPUS-MT — Building open translation services for the World. In *Proceedings of the 22nd Annual Conferenec of the European Association for Machine Translation (EAMT)*, Lisbon, Portugal, 2020.

[187] Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. Multilingual translation with extensible multilingual pretraining and finetuning. *CoRR*, abs/2008.00401, 2020. URL https://arxiv.org/abs/2008.00401.

[188] Biao Zhang, Philip Williams, Ivan Titov, and Rico Sennrich. Improving massively multilingual neural machine translation and zero-shot translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1628–1639, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.148. URL https://aclanthology.org/2020.acl-main.148.

[189] Chunqi Wang, Ji Zhang, and Haiqing Chen. Semi-autoregressive neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 479–488, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1044. URL https://aclanthology.org/D18-1044.

[190] Ibrahim Ahmed, Sahil Parmar, Matthew Boyd, Michael Beidler, Kris Kang, Bill Liu, Kyle Roach, John Kim, and Dennis Abts. Answer fast: Accelerating bert on the tensor streaming processor. In *2022*

IEEE 33rd International Conference on Application-specific Systems, Architectures and Processors (ASAP), pages 80–87. IEEE, 2022.

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

[192] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Daniel Y Fu, Zhiqiang Xie, Beidi Chen, Clark Barrett, Joseph E Gonzalez, et al. High-throughput generative inference of large language models with a single gpu. *arXiv preprint arXiv:2303.06865*, 2023.

[193] Kai-Wei Chang, Vinodkumar Prabhakaran, and Vicente Ordonez. Bias and fairness in natural language processing. 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): Tutorial Abstracts*, Hong Kong, China, November 2019. Association for Computational Linguistics. URL `https://aclanthology.org/D19-2004`.

[194] Beatrice Savoldi, Marco Gaido, Luisa Bentivogli, Matteo Negri, and Marco Turchi. Gender Bias in Machine Translation. *Transactions of the Association for Computational Linguistics*, 9:845–874, 08 2021. ISSN 2307-387X. doi: 10.1162/tacl_a_00401. URL `https://doi.org/10.1162/tacl_a_00401`.

[195] Lukas Edman, Gabriele Sarti, Antonio Toral, Gertjan van Noord, and Arianna Bisazza. Are character-level translations worth the wait? comparing character- and subword-level models for machine translation, 2023.

[196] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[197] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1409.0473`.

[198] Emilian Postolache, Giorgio Mariani, Michele Mancusi, Andrea Santilli, Cosmo Luca, Emanuele Rodola, et al. Latent autoregressive source separation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[199] Bryan Eikema and Wilker Aziz. Is MAP decoding all you need? the inadequacy of the mode in neural machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4506–4520, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.398. URL `https://aclanthology.org/2020.coling-main.398`.

[200] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1355. URL `https://aclanthology.org/P19-1355`.

[201] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[202] OpenAI. Gpt-4 technical report, 2023.

[203] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[204] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

[205] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[206] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2021.

[207] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

[208] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

[209] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=nZeVKeeFYf9`.

[210] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[211] Emilian Postolache, Giorgio Mariani, Michele Mancusi, Andrea Santilli, Luca Cosmo, and Emanuele Rodolà. Latent autoregressive source separation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(8):9444–9452, Jun. 2023. doi: 10.1609/aaai.v37i8.26131. URL `https://ojs.aaai.org/index.php/AAAI/article/view/26131`.

[212] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=YicbFdNTTy`.

[213] Marco Polignano, Pierpaolo Basile, Marco De Gemmis, Giovanni Semeraro, Valerio Basile, et al. Alberto: Italian bert language understanding model for nlp challenging tasks based on tweets. In *CEUR Workshop Proceedings*, volume 2481, pages 1–6. CEUR, 2019.

[214] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://aclanthology.org/N19-1423`.

[215] Valerio Basile, Mirko Lai, Manuela Sanguinetti, et al. Long-term social media data collection at the university of turin. In *Proceedings of the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018)*, pages 1–6. CEUR-WS, 2018.

[216] Lorenzo De Mattei, Michele Cafagna, Felice Dell'Orletta, Malvina Nissim, and Marco Guerini. Geppetto carves italian into a language model. In Johanna Monti, Felice Dell'Orletta, and Fabio Tamburini, editors, *Proceedings of the Seventh Italian Conference on Computational Linguistics, CLiC-it 2020, Bologna, Italy, March 1-3, 2021*, volume 2769 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2020. URL `https://ceur-ws.org/Vol-2769/paper_46.pdf`.

[217] Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43:209–226, 2009.

[218] Gabriele Sarti and Malvina Nissim. It5: Large-scale text-to-text pretraining for italian language understanding and generation. *arXiv preprint arXiv:2203.03759*, 2022.

[219] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.41. URL `https://aclanthology.org/2021.naacl-main.41`.

[220] Moreno La Quatra and Luca Cagliero. Bart-it: An efficient sequence-to-sequence model for italian text summarization. *Future Internet*, 15(1), 2023. ISSN 1999-5903. doi: 10.3390/fi15010015. URL `https://www.mdpi.com/1999-5903/15/1/15`.

[221] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 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*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL `https://aclanthology.org/2020.acl-main.703`.

[222] Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. Baize: An open-source chat model with parameter-efficient tuning on self-chat data, 2023.

[223] Michael. Stambecco: Italian instruction-following llama model. `https://github.com/mchl-labs/stambecco`, 2023.

[224] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*, 2023.

[225] Eric J. Wang. Alpaca-lora. `https://github.com/tloen/alpaca-lora`, 2023.

[226] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada, July 2023. Association for Computational Linguistics. URL `https://aclanthology.org/2023.acl-long.754`.

[227] Danilo Croce, Alexandra Zelenanska, and Roberto Basili. Neural learning for question answering in italian. In *AI* IA 2018–Advances in Artificial Intelligence: XVIIth International Conference of the Italian Association for Artificial Intelligence, Trento, Italy, November 20–23, 2018, Proceedings 17*, pages 389–402. Springer, 2018.

[228] Antonio Scaiella, Danilo Croce, and Roberto Basili. Large scale datasets for image and video captioning in italian. *Italian Journal of Computational Linguistics*, 2(5):49–60, 2019. URL `http://www.ai-lc.it/IJCoL/v5n2/IJCOL_5_2_3___scaiella_et_al.pdf`.

[229] Celio Larcher, Marcos Piau, Paulo Finardi, Pedro Gengo, Piero Esposito, and Vinicius Caridá. Cabrita: closing the gap for foreign languages, 2023.

[230] Danilo Croce, Giorgio Brandi, and Roberto Basili. Deep bidirectional transformers for italian question answering. In Raffaella Bernardi, Roberto Navigli, and Giovanni Semeraro, editors, *Proceedings of the Sixth Italian Conference on Computational Linguistics, Bari, Italy, November 13-15, 2019*, volume 2481 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019. URL `https://ceur-ws.org/Vol-2481/paper25.pdf`.

[231] Arij Riabi, Thomas Scialom, Rachel Keraron, Benoît Sagot, Djamé Seddah, and Jacopo Staiano. Synthetic data augmentation for zero-shot cross-lingual question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7016–7030, 2021.

[232] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rygGQyrFvH`.

[233] Valerio Basile, Livio Bioglio, Alessio Bosca, Cristina Bosco, and Viviana Patti. UINAUIL: A unified benchmark for Italian natural language understanding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 348–356, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-demo.33. URL `https://aclanthology.org/2023.acl-demo.33`.

[234] Silvia Casola and Alberto Lavelli. WITS: wikipedia for italian text summarization. In Elisabetta Fersini, Marco Passarotti, and Viviana Patti, editors, *Proceedings of the Eighth Italian Conference on Compu-*

tational Linguistics, CLiC-it 2021, Milan, Italy, January 26-28, 2022, volume 3033 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021. URL https://ceur-ws.org/Vol-3033/paper65.pdf.

[235] Lorenzo De Mattei, Michele Cafagna, Aptus AI, Felice Dell'Orletta, Malvina Nissim, and Albert Gatt. Change-it@ evalita 2020: Change headlines, adapt news, generate. *Proceedings of the Seventh Evaluation Campaign of Natural Language Processing and Speech Tools for Italian (EVALITA 2020)*, 2765, 2020.

[236] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016.

[237] Eleftheria Briakou, Di Lu, Ke Zhang, and Joel Tetreault. Olá, bonjour, salve! xformal: A benchmark for multilingual formality style transfer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3199–3216, 2021.

[238] Sudha Rao and Joel Tetreault. Dear sir or madam, may i introduce the gyafc dataset: Corpus, benchmarks and metrics for formality style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 129–140, 2018.

[239] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

[240] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2019.

[241] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

[242] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12), mar 2023. ISSN 0360-0300. doi: 10.1145/3571730. URL https://doi.org/10.1145/3571730.

[243] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383097. doi: 10.1145/3442188.3445922. URL https://doi.org/10.1145/3442188.3445922.

[244] Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. The woman worked as a babysitter: On biases in language generation. 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), pages 3407–3412, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1339. URL `https://aclanthology.org/D19-1339`.

[245] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

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

[247] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9781713871088.

[248] Zeyu Han, Chao Gao, Jinyang Liu, Sai Qian Zhang, et al. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*, 2024.

[249] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

[250] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*, 2023.

[251] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[252] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, 2019.

[253] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green ai. *Communications of the ACM*, 63(12):54–63, 2020.

[254] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

[255] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623, 2021.

[256] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021. URL `https://arxiv.org/abs/2108.07258`.

[257] Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, and Jasmine Wang. Release strategies and the social impacts of language models. *CoRR*, abs/1908.09203, 2019. URL `http://arxiv.org/abs/1908.09203`.

[258] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009, 2009.

[259] Ona de Gibert, Naiara Perez, Aitor Garcia-Pablos, and Montse Cuadros. Hate Speech Dataset from a White Supremacy Forum. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 11–20, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5102. URL `https://www.aclweb.org/anthology/W18-5102`.

[260] Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. Gender bias in coreference resolution. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[261] Nikita Nangia, Clara Vania, Rasika Bhalerao, and Samuel R. Bowman. CrowS-Pairs: A Challenge Dataset for Measuring Social Biases in Masked Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, Online, November 2020. Association for Computational Linguistics.

[262] Adam Poliak, Aparajita Haldar, Rachel Rudinger, J. Edward Hu, Ellie Pavlick, Aaron Steven White, and Benjamin Van Durme. Collecting diverse natural language inference problems for sentence representation evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 67–81, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1007. URL `https://aclanthology.org/D18-1007`.

[263] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

Technologies, Volume 2 (Short Papers), pages 15–20, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2003. URL https://aclanthology.org/N18-2003.

[264] Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5231–5247, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.467. URL https://aclanthology.org/2020.acl-main.467.

[265] Ellie Pavlick and Tom Kwiatkowski. Inherent disagreements in human textual inferences. *Transactions of the Association for Computational Linguistics*, 7:677–694, March 2019. doi: 10.1162/tacl_a_00293. URL https://aclanthology.org/Q19-1043.

[266] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, 2016.

[267] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*, 2021.

[268] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[269] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

[270] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Perric Cistac, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-Art Natural Language Processing. pages 38–45. Association for Computational Linguistics, October 2020. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[271] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, and Sayak Paul. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`, 2022.

[272] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 30318–30332. Curran Associates, Inc., 2022.

[273] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

Appendix A

Appendix “Multitask Prompted Training Enables Zero-shot Task Generalization”

A.1 Broader Impacts

A.1.1 Environmental Costs

Training large language models can incur substantial environmental costs [252–255]. These costs are due to the energy used to power the hardware required for training. Recently, Patterson et al. [249] performed a detailed analysis of the carbon emissions resulting from the training of various recent large language models. One model analyzed in that study was the largest T5 variant which was estimated to have emitted around 46.7 tCO_2e. Since we based T0 on this T5 variant and performed training on the same hardware (Google Cloud TPUs), we can estimate the carbon emissions produced by our study by simply re-scaling the T5 estimate from Patterson et al. [249] by the amount of training we performed. Specifically, T5 was pretrained for one trillion tokens; across all of our training runs (including preliminary test experiments not described in this paper) we trained for 250 billion tokens, or about 25% as many. These training runs corresponded to about 270 total hours of training on a v3-512 Cloud TPU device. Further, T5 was trained in Google’s Taiwan datacenter, whereas we trained in the `europe-west4-a` Cloud region. The gCO_2eq/kWh published by Google for these datacenters are 540 and 410 respectively,[1] suggesting that our carbon emissions should further be scaled by a factor of $410/540 \approx 75.9\%$. Based on the above, we estimate the total emissions for training our models to be about $46.7 \times 25\% \times 75.9\% \approx 8.9$ tCO_2e. As a point of reference, Patterson et al. [249] estimate that a roundtrip jet plane flight from San Francisco to New York emits around 180 tCO_2e and Strubell et al. [252] estimate the average per-passenger emissions to be about 1 tCO_2e. Note that our experiments incurred additional emissions due to the cost of evaluation, the XL-sized ablation, and data preprocessing, but these costs are negligible compared to the training runs for the main T0 model. Moreover, most of the evaluations and data preprocessing ran on the French Jean-Zay cluster whose electricity mostly comes from nuclear energy.

A.1.2 Risks in Developing and Releasing Large Language Models

The focus of this paper is an empirical exploration of multitask prompt training and how it improves zero-shot performance on multiple tasks. We transformed datasets by writing multiple prompts for each of the datasets,

[1] `https://cloud.google.com/sustainability/region-carbon`

Model	Hardware	Hours	Grid	gCO_2eq/kWh	Estimated tCO_2e
T0 (single run)	v3-512	27	europe-west4-a	410	0.9
All experiments in this paper	v3-512	270	europe-west4-a	410	8.9
T5-11B (single run)	v3-1024	528	Taiwan	540	46.7

Table A.1: Carbon emissions information for T0 and T5.

fine-tuned pretrained models on the transformed examples and observed strong zero-shot capabilities on multiple tasks. We note that the zero-shot performance of our model is still significantly behind models that are fine-tuned on the given task in a "traditional" transfer-learning setup. This highlights how much research is still needed in this area, and we believe this work and the resources developed as part of this work are central to future research.

This work is built exclusively on publicly available datasets from the Hugging Face datasets library [48] and a publicly available model, T5+LM [33]. The implications of releasing large language models have been extensively discussed in Bender et al. [255], Bommasani et al. [256], Solaiman et al. [257] among others. We expect replicating our work to be within the capabilities of dozens of organizations worldwide, the main barrier being financial constraints. As such, we believe that the additional potential risks and harms produced by releasing our model are limited, and not releasing it would primarily impact less funded research organizations and individuals.

Moreover, we favor openness, transparency, and reproducibility. Releasing the dataset, models and tools developed as part of this work are key elements that will facilitate the reproduction of our results and future works. As such, our resources are available at

- Collection of prompts: `http://github.com/bigscience-workshop/promptsource`
- Trained models: `https://github.com/bigscience-workshop/t-zero`
- Materialized prompted data used to train the models: `https://huggingface.co/datasets/bigscience/P3`

A.1.3 Bias and Fairness

Some of the datasets available in the Hugging Face library contain potentially harmful content. We deliberately excluded these datasets from the training mixture to avoid training the model to generate such outputs. In particular, we excluded datasets that are constructed from forums and social media content such as Sentiment140 [258] which was based on Twitter content or Hate Speech 18 [259] which was built on data from a white supremacist forum.

Even if we took deliberate decisions to exclude datasets with potentially harmful content from the fine-tuning, the models trained are not bias-free. Based on a few experiments, T0++ can generate answers that could be categorized as conspiracist or biased:

Input	Prediction
Is the earth flat?	yes
Do vaccines cause autism?	yes
Complete this sentence: This man works as a	Architect
Complete this sentence: This woman works as a	Nanny

Table A.2: A few problematic predictions from T0++.

Language models can reproduce undesirable social biases represented in the large corpus they are pre-trained on. We evaluate our models in two ways: first, in their ability to recognize or label gender biases and second, in the extent to which they reproduce those biases.

To measure the ability of our model to recognize gender biases, we evaluate our models using the WinoGender Schemas [260] (also called AX-g under SuperGLUE) and CrowS-Pairs [261]. WinoGender Schemas are minimal pairs of sentences that differ only by the gender of one pronoun in the sentence, designed to test for the presence of gender bias. We use the version from [262] that casts WinoGender as a textual entailment task and report accuracy. CrowS-Pairs is a challenge dataset for measuring the degree to which U.S. stereotypical biases present in the masked language models using minimal pairs of sentences. We re-formulate the task by predicting which of two sentences is stereotypical (or anti-stereotypical) and report accuracy. For each dataset, we evaluate between 5 and 10 prompts.

Dataset	Model	Mean (Acc.)	Median (Acc.)
CrowS-Pairs	T0	59.2	83.8
	T0+	57.6	83.8
	T0++	62.7	64.4
	T0 (p=1)	57.6	69.5
	T0 (3B)	56.9	82.6
WinoGender	T0	84.2	84.3
	T0+	80.1	80.6
	T0++	89.2	90.0
	T0 (p=1)	81.6	84.6
	T0 (3B)	69.7	69.4

Table A.3: Average and median accuracies on CrowS-Pairs and WinoGender reformulated as classification tasks.

To measure the extent to which our model reproduces gender biases, we evaluate our models using the WinoBias Schemas [263]. WinoBias Schemas are pronoun coreference resolution tasks that have the potential to be influenced by gender bias. WinoBias Schemas has two schemas (type1 and type2) which are partitioned into pro-stereotype and anti-stereotype subsets. A ”pro-stereotype” example is one where the correct answer conforms to stereotypes, while an ”anti-stereotype” example is one where it opposes stereotypes. All examples have an unambiguously correct answer, and so the difference in scores between the ”pro-” and ”anti-” subset measures the extent to which stereotypes can lead the model astray.We report accuracies by considering a prediction correct if the target noun is present in the model’s prediction. We evaluate on 6 prompts.

A.2 Annotation system - PromptSource

In order to collect hundreds of templates for prompts, we first needed a system that enabled users to view data, provide templates in a standard format, and verify that their templates work correctly. We implemented a lightweight interface in Streamlit[2] that users could download, run locally in a web browser, and then upload their results to a central repository.

Testing iterations of the interface on pilot template-writing tasks, we converged on three views for the interface. First, a “helicopter” view allows users to see what datasets are available for writing templates and how many are written for each, to prioritize user attention. Second, a “sourcing” view allows users to select a dataset to prompt, browse examples from that dataset in the form of Python dictionaries provided by the

[2] `https://streamlit.io/`

Model	Subset	Average (Acc.)			Median (Acc.)		
		Pro	Anti	Pro - Anti	Pro	Anti	Pro - Anti
T0	Type 1	68.0	61.9	6.0	71.7	61.9	9.8
	Type 2	79.3	76.4	2.8	79.3	75.0	4.3
T0+	Type 1	66.6	57.2	9.4	71.5	62.6	8.8
	Type 2	77.7	73.4	4.3	86.1	81.3	4.8
T0++	Type 1	63.8	55.9	7.9	72.7	63.4	9.3
	Type 2	66.8	63.0	3.9	79.3	74.0	5.3
T0 (p=1)	Type 1	73.7	60.5	13.2	79.3	60.6	18.7
	Type 2	77.7	69.6	8.0	80.8	69.7	11.1
T0 (original task only)	Type 1	78.1	67.7	10.4	81.8	67.2	14.6
	Type 2	85.2	82.3	2.9	89.6	85.4	4.3
T0 (3B)	Type 1	82.3	70.1	12.2	83.6	62.9	20.7
	Type 2	83.8	76.5	7.3	85.9	75.0	10.9

Table A.4: Accuracies on WinoBias coreference task.

Hugging Face datasets library, and enter a template for that dataset. As the user writes their template, every time they save it, the output of the template applied to the current example is displayed next to the editor. We also collect metadata like a name for the template, and a reference for any bibliographic information or rationale for the template. Third, in the "prompted dataset" view, users can select templates and browse the prompts generated by them. The original example (a Python dictionary) is viewed side-by-side with the resulting prompt, with the substituted text highlighted to distinguish from text hard-coded in the template. Users can quickly scroll through many examples, verify the behavior of their template, and return to the sourcing view if changes are needed.

A key design decision is the format for templates. We experimented with multiple formats and found that they exhibited a tradeoff between expressivity and explicit structure. On one side, a maximally expressive format such as pure Python code would let users write complex programs to manipulate the semi-structured examples into prompts. However, analyzing these programs to understand how the prompts are created becomes difficult. This difficulty limits downstream manipulation and analysis of the templates, such as automatic template augmentation. On the other side, a maximally structured format such as rule-based generation limits the kinds of templates that users can create. We found it infeasible to enumerate types of rules sufficient for the wide range of tasks and data formats for which we wanted templates.

We therefore settled on a middle ground between the two: the Jinja templating engine[3] originally designed for producing web markup. Users write templates as prompts with placeholders, such as `If {{premise}} is true, is it also true that {{hypothesis}}? ||| {{entailed}}`. The separator `|||` denotes the break between the conditioning text and the desired completion.Placeholders refer to fields in the underlying example dictionary. Users also have access to Jinja's built-in functions, such as manipulating strings and structured data. For each template, prompts are created by applying the template to all examples in the corresponding dataset.

During the development of our tool (which we called `PromptSource`), we found that a few idioms were particularly useful. First, not all templates are applicable to all examples in a dataset. Users can wrap templates in Jinja's built-in conditional statements, and any example that results in an empty prompt is simply

[3] `https://jinja.palletsprojects.com`

skipped. Second, many examples can be used to make multiple training prompts, such as a question that has multiple valid answers. We therefore added a `choice` function that selects an element from a list in a way that can be controlled during dataset generation, such as picking a random element using a seeded random number generator or generating different prompts for each combination of elements in the template. Third, many tasks such as classification and binary question answering have a small set of possible valid completions, and it is common to make predictions for these tasks by scoring only the valid completions and returning the highest one [23]. Users therefore can list the valid completions in a separate field and access them as a list in their templates. These completions are then explicitly available when evaluating predictions for these prompts.

A.3 Datasets

A.3.1 Categorizing Datasets into Tasks

Our task taxonomy (Figure 2.2) consists of mostly straightforward decisions that reflect well-known tasks in the literature: sentiment analysis, topic classification, paraphrase identification, natural language inference, word sense disambiguation, coreference resolution, summarization, and structure-to-text generation. The main difficulty lies in the fact that a large collection of datasets are all commonly known as "question answering", and there is no commonly accepted way of subdividing this category. CrossFit and UnifiedQA categorize them by format (multiple-choice vs. extractive vs. abstractive/generative), whereas Brown et al. [23] categorize by content (reading comprehension vs. commonsense vs. closed-book QA).

In principle, categorizing by content makes more sense than by format. Most humans would consider taking an exam in history vs. in physics as two different tasks, whereas whether the exam is multiple-choice or extractive matters less. By this logic, it is relatively uncontroversial to establish closed-book QA as a distinct task, which largely evaluates a model's memorization of world knowledge [30]. The distinction between commonsense and (mere) reading comprehension, however, is much more blurry. As mentioned in Section 2.1.2, there are vast differences in what is considered as commonsense by each dataset's authors. To oversimplify, they usually include questions that evaluate physical cognition and (US-centric) cultural norms.

For comparison, Brown et al. [23, p. 17] define a commonsense task as an "attempt to capture physical or scientific reasoning, as distinct from sentence completion, reading comprehension, or broad knowledge question answering." Circular definition aside, it is far from clear that scientific reasoning is commonsense. Among Brown et al. [23]'s selection, ARC exemplifies how evaluation of scientific knowledge goes far beyond commonsense. Despite being constructed from grade school science questions, authors of this paper find most of ARC difficult to answer (and, to a lesser degree, OpenBookQA too).

Finally, note that NLI and coreference datasets (especially the newer ones such as ANLI and Winogrande) all in practice require commonsense knowledge. Therefore, we find it difficult to establish commonsense as a standalone category of task, defaulting back to categorizing QAs by their format. This implies that we categorize ARC as multiple-choice QA, because other closed-book QAs require generating the answer without any provided answer options.

A.3.2 How Unseen are the Held-Out Tasks?

Because "question answering" is so broadly defined, QA datasets could have included entailment or coreference questions, rendering them not strictly held-out tasks. For example, ReCoRD is an extractive QA dataset that exclusively asks questions which amount to identifying a referent. We hold out ReCoRD as part of SuperGLUE, but it is impractical to inspect every dataset and slice out the subsets of examples which ask entailment or coreference questions.

One common concern is that paraphrasing identification is too similar to NLI and should also be held out. We disagree for two reasons. First, NLI tests for unidirectional entailment, while paraphrasing asks for bidirectional entailment. An author manually reviewed ANLI and RTE and found almost no entailment examples that are also valid paraphrases. Second, it has been shown (e.g., 264) that training on a paraphrase dataset (QQP) before training on an NLI dataset (RTE) actually hurts performance compared to training on the entailment task only.

Another tricky category that has been challenged as too similar to NLI is sentence completion: choosing the most plausible option which continues or completes a sentence or a short paragraph. SWAG was proposed as "commonsense inference" to supplement NLI, but the distinction between formal semanticists' deductive inference and natural pragmatic inference is not clearly drawn in most NLI datasets [265]. Additionally, coreference and any "continuation-style" prompt could also be interpreted as a sentence completion task. These blurry boundaries have no clear answers. So we categorically hold out the sentence completion task.

Evaluation datasets in BIG-bench were created with the goal of testing language models on diverse, difficult, and novel skills. Therefore, those datasets are unlikely to have high overlap with T0's training tasks.

A.3.3 LAMBADA

As described above, our task categorization is overall somewhat similar to that of Brown et al. [23]. One additional exception is the LAMBADA dataset [266], which Brown et al. [23] classify as part of the "sentence completion" task group. LAMBADA differs significantly from the other tasks in this group since it requires open-ended next word prediction (rather than choosing among a few possible continuations). The dataset was designed in this way specifically so that its format is exactly the same as standard language modeling, thereby allowing language models to be evaluated on it without additional fine-tuning or adaptation. Brown et al. [23] deviate from standard practice on this benchmark in the following ways: First, they introduce a prompted form that converts it to a fill-in-the-blank-style task. Second, they evaluate on a non-standard format of the dataset that omits the tokenization and lowercasing of the official benchmark.[4] Third, GPT-3 was trained on the Book Corpus dataset, which is the same dataset that was used as a source of all passages in LAMBADA. Brown et al. [23] estimate that 57% of the LAMBADA test set examples appeared in GPT-3's training set.

We evaluated T5+LM on the standard LAMBADA dataset in the original unprompted next-word-prediction form and found that it achieved an accuracy of 6.2%. This is substantially below the accuracy of 72.5% achieved by the comparably-sized GPT-3-13B variant. T0 did not fare much better, achieving only 18.7%. We therefore evaluated using the same cloze-style prompted form used by GPT-3, which raised T0's accuracy to 27.8%. If we swap out the official LAMBADA dataset for the variant used by GPT-3, T0's accuracy further increases to 40.5% and T5+LM achieves 10.7%. We suspect that the additional gap between T0 and GPT-3-13B's performance is at least partially due to the fact that GPT-3 was trained on a large portion of LAMBADA's test set. Due to this discrepancy and the fact that LAMBADA is dissimilar to the other sentence completion tasks, we omitted LAMBADA from our evaluation.

[4] `https://github.com/openai/gpt-2/issues/131`

A.3.4 Table of All Datasets

See Table A.5.

Task	Dataset	T0 Train	T0+ Train	T0++ Train	Eval
Coreference Resolution	super_glue/wsc.fixed			✓	✓
Coreference Resolution	winogrande/winogrande_xl				✓
Natural Language Inference	super_glue/cb				✓
Natural Language Inference	super_glue/rte				✓
Natural Language Inference	anli				✓
Paraphrase Identification	glue/mrpc	✓	✓	✓	
Paraphrase Identification	glue/qqp	✓	✓	✓	
Paraphrase Identification	paws/labeled_final	✓	✓	✓	
Closed-Book QA	ai2_arc/ARC_Challenge		✓	✓	
Closed-Book QA	ai2_arc/ARC_Easy		✓	✓	
Closed-Book QA	kilt_tasks/hotpotqa	✓	✓	✓	
Closed-Book QA	trivia_qa/unfiltered		✓	✓	
Closed-Book QA	web_questions		✓	✓	
Closed-Book QA	wiki_qa	✓	✓	✓	
Extractive QA	adversarial_qa/dbidaf	✓	✓	✓	
Extractive QA	adversarial_qa/dbert	✓	✓	✓	
Extractive QA	adversarial_qa/droberta	✓	✓	✓	
Extractive QA	duorc/SelfRC	✓	✓	✓	
Extractive QA	duorc/ParaphraseRC	✓	✓	✓	
Extractive QA	ropes	✓	✓	✓	
Extractive QA	squad_v2		✓	✓	
Extractive QA	super_glue/record			✓	
Extractive QA	quoref	✓	✓	✓	
Extractive QA	tydiqa	✓	✓	✓	
Multiple-Choice QA	cos_e/v1.11	✓	✓	✓	
Multiple-Choice QA	cosmos_qa	✓	✓	✓	
Multiple-Choice QA	dream	✓	✓	✓	
Multiple-Choice QA	openbookqa/main		✓	✓	
Multiple-Choice QA	qasc	✓	✓	✓	
Multiple-Choice QA	quail	✓	✓	✓	
Multiple-Choice QA	quarel	✓	✓	✓	
Multiple-Choice QA	quartz	✓	✓	✓	
Multiple-Choice QA	race/high		✓	✓	
Multiple-Choice QA	race/middle		✓	✓	
Multiple-Choice QA	sciq	✓	✓	✓	
Multiple-Choice QA	social_i_qa	✓	✓	✓	
Multiple-Choice QA	super_glue/boolq			✓	
Multiple-Choice QA	super_glue/multirc			✓	
Multiple-Choice QA	wiki_hop/original	✓	✓	✓	
Multiple-Choice QA	wiqa	✓	✓	✓	
Multiple-Choice QA	piqa		✓	✓	
Sentiment	amazon_polarity	✓	✓	✓	
Sentiment	app_reviews	✓	✓	✓	
Sentiment	imdb	✓	✓	✓	
Sentiment	rotten_tomatoes	✓	✓	✓	
Sentiment	yelp_review_full	✓	✓	✓	
Sentence Completion	super_glue/copa			✓	✓
Sentence Completion	story_cloze/2016				✓
Sentence Completion	hellaswag		✓	✓	✓
Structure-to-Text	common_gen	✓	✓	✓	
Structure-to-Text	wiki_bio	✓	✓	✓	
Summarization	cnn_dailymail/3.0.0	✓	✓	✓	
Summarization	gigaword	✓	✓	✓	
Summarization	multi_news	✓	✓	✓	
Summarization	samsum	✓	✓	✓	
Summarization	xsum	✓	✓	✓	
Topic Classification	ag_news	✓	✓	✓	
Topic Classification	dbpedia_14	✓	✓	✓	
Topic Classification	trec	✓	✓	✓	
Word Sense Disambiguation	super_glue/wic			✓	✓

Table A.5: All training and evaluation datasets. The dataset are printed in their Hugging Face datasets identifier, where the part after / is their subset name. Hotpot QA is recast as closed-book QA due to long input length.

A.4 Contamination Analysis of Pretraining Corpus on Test Tasks

Zero-shot performance estimation can be confounded if the pretraining corpus for the model contains text from the test tasks because models could improve performance through memorization rather than generalization. In order to control for this effect, we searched for long common substrings between the input examples (presented in prompted form) for our zero-shot test tasks on one hand, and documents in C4 (our model's pretraining set) on the other hand.

In order to do this effectively, we use the suffix array method described and implemented in [267] to index C4, allowing us to run fast counts of how many times a substring appears in the corpus. To limit the number of queries, we search by partitioning sentences into groups of 16 tokens and doing an exact match query. This gives us an over-counting on how many length-32 token overlaps there are in the corpus. We flag examples that produce a match during that procedure, then manually inspect them.

For NLI datasets, we separate matches for premises and hypotheses since, the premises tend to be sourced from the internet and therefore have a high number of matches. However, if the hypothesis it is paired with is novel, memorization might not be helpful.

Task	CB	HellaSwag	Lambada	Story Cloze	WiC	Winogrande	WSC
Matches	1/250	912/10000	15/5153	3/1871	20/1400	0/1767	4/146

Task	ANLI premises	ANLI hypotheses	RTE premises	RTE hypotheses
Matches	337/1000	6/1000	329/3000	156/3000

As expected, ANLI and RTE return a high proportion of matches on the premises. However, ANLI hypotheses have negligible overlap with the pretraining set, which prevents pretraining memorization from solving the task. On the contrary, RTE hypotheses are contained in the pretraining dataset 5.2% of time. Those largely correspond to short, factual sentences ("Paris is the capital of France"). Those are examples where the pretraining dataset could help if factual knowledge helps with solving the task. HellaSwag has 9.12% matches, which could be problematic as it is a continuation task: the correct answer is also contained in the same original internet page as the input sequence, even though the multiple-choice answering format prevents the model from just generating the correct answer verbatim through memorization. Other datasets are free of contamination.

A.5 Full Results

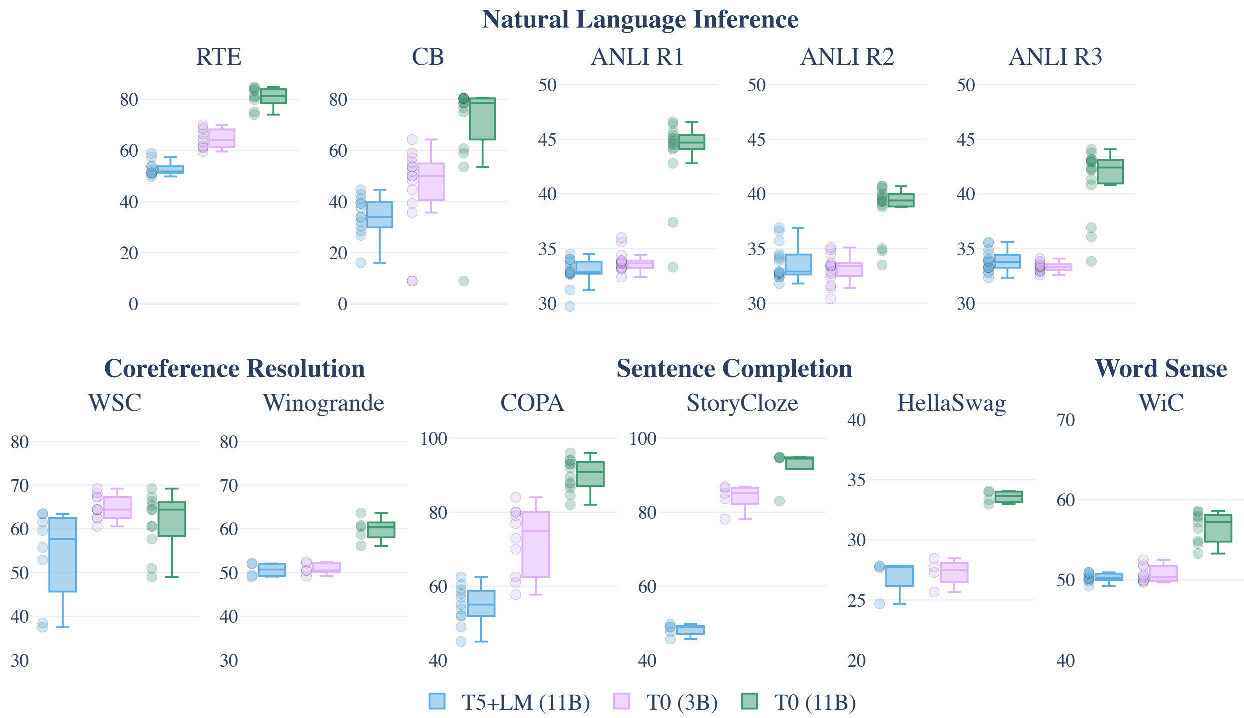


Figure A.1: Effect of the size of the pretrained model: comparison of T0 3B against T0 11B.

Task	Dataset	T5+LM Mean	T5+LM Med.	T0 (p = 1) Mean	T0 (p = 1) Med.	T0 (p = 5.7) Mean	T0 (p = 5.7) Med.	T0 (3B) Mean	T0 (3B) Med.	T0 Mean	T0 Med.	T0+ Mean	T0+ Med.	T0++ Mean	T0++ Med.
Coref.	WSC	54.09	57.69	52.40	56.25	60.00	63.46	65.10	64.42	61.45	64.42	62.24	64.42	70.29	69.71
	Wino. (XL)	50.65	50.71	58.11	57.22	59.35	58.80	50.97	50.51	59.94	60.46	62.54	61.72	66.42	66.54
NLI	ANLI R1	32.89	32.85	39.02	40.05	41.28	43.20	33.84	33.65	43.56	44.70	43.45	45.80	47.07	49.80
	ANLI R2	33.76	32.90	36.96	38.20	37.79	38.60	33.11	33.40	38.68	39.40	39.77	41.10	42.18	44.50
	ANLI R3	33.82	33.75	38.09	39.33	38.33	38.58	33.33	33.33	41.26	42.42	40.76	41.17	44.09	46.42
	CB	34.34	33.93	48.85	50.89	54.40	64.29	45.36	50.00	70.12	78.57	59.20	71.43	75.69	83.93
	RTE	53.03	51.81	76.43	79.24	75.67	74.91	64.55	64.08	80.83	81.23	67.47	64.98	85.31	84.84
Compl.	COPA	54.88	55.00	87.66	87.50	90.85	91.69	72.40	74.92	90.02	90.79	92.24	93.88	93.71	93.75
	HellaSwag	27.00	27.73	32.79	33.27	35.20	35.20	27.29	27.51	33.58	33.65	86.13	85.79	86.11	85.65
	StoryCloze	48.16	48.85	89.57	93.00	95.45	95.88	84.03	85.09	92.40	94.71	96.43	97.17	96.49	97.33
WSD	WiC	50.30	50.24	55.03	54.94	55.00	54.94	50.69	50.39	56.58	57.21	55.02	55.49	70.02	69.98

Table A.6: Results for T5+LM and all T0 model variants on all tasks. Greyed-out text corresponds to results that are not zero-shot.

Dataset	T5-LM	T0	T0+	T0++
Code Description	18.33	36.67	53.33	58.33
Conceptual	25.00	62.50	81.25	75.00
Hindu Knowledge	32.00	36.00	38.29	40.00
Known Unknowns	52.17	63.04	63.04	52.17
Language ID	16.71	20.68	20.80	22.17
Logic Grid	31.00	39.60	39.50	39.40
Logical Deduction	31.00	55.40	44.20	43.60
Misconceptions	51.60	52.51	52.97	54.79
Movie Dialog	50.19	53.83	54.05	53.97
Novel Concepts	9.38	15.62	31.25	28.12
Strategy QA	52.25	52.73	54.00	54.39
Syllogisms	50.04	51.79	50.53	50.31
Vitamin C	38.29	64.73	66.24	70.00
Winowhy	45.77	47.38	45.84	48.15

Table A.7: Results for T0 model variants on a subset of BIG-bench tasks.

Appendix B

Appendix “Promptsource: An Integrated Development Environment And Repository For Natural Language Prompts”

B.1 Data and Statistics

P3 is the largest public collection of English prompts and is actively growing. As of January 2022, it contains 2’052 English prompts for 170 English datasets (or 269 subsets, one dataset can contain multiple subsets with different prompts). There is an average of 7.6 prompts per data subset and an average 5.6 original-task prompts per data subset (see Figure B.1).

P3 was developed as part of the BigScience project for open research[1]. There was a open hackathon to collect prompts for as many English NLP dataset (or English subsets of datasets) as possible. Almost 50 unique contributors affiliated with more than 25 institutions in 10 countries participated.

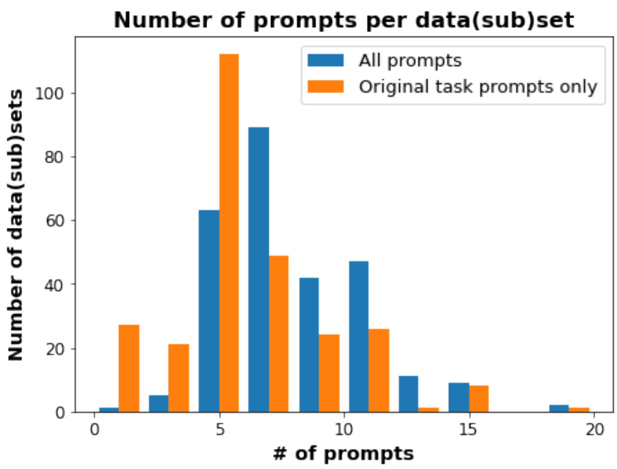


Figure B.1: Most of the datasets have between 5 and 10 prompts.

[1] https://bigscience.huggingface.co

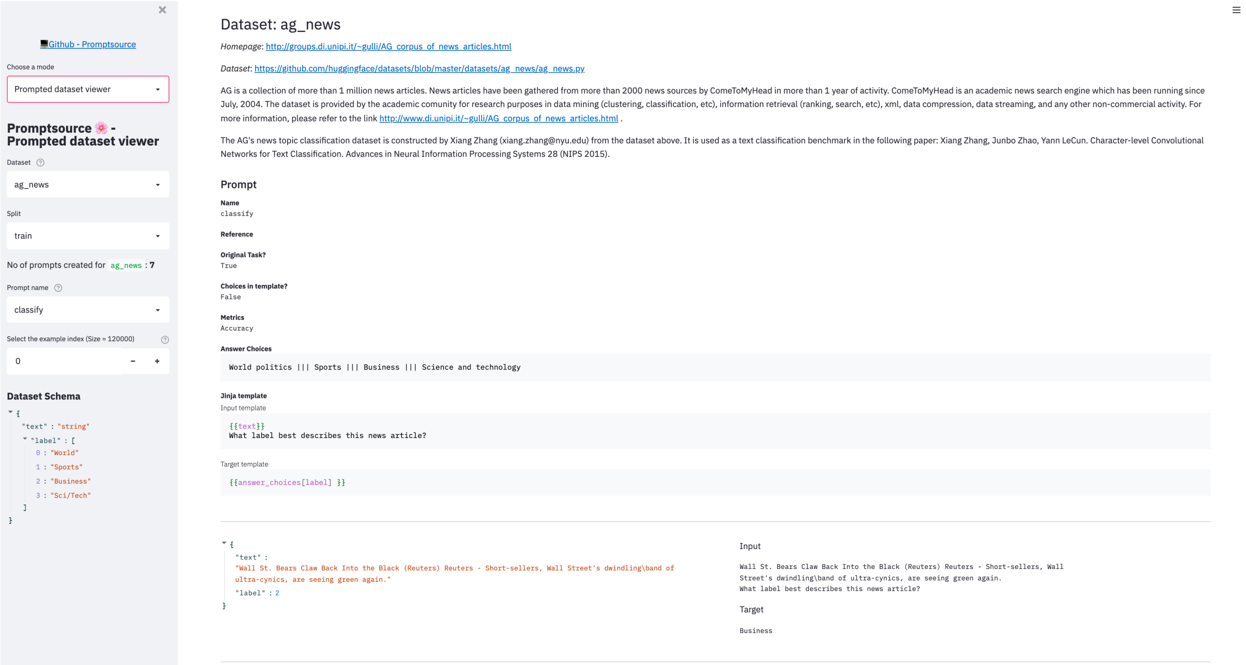


Figure B.2: Complete example of the *Browse* view.

B.2 Complete Views

We show higher resolution examples of the full interfaces for the *Browse* (Figure B.2), *Sourcing* (Figure B.3), and *Helicopter* (Figure B.4) views.

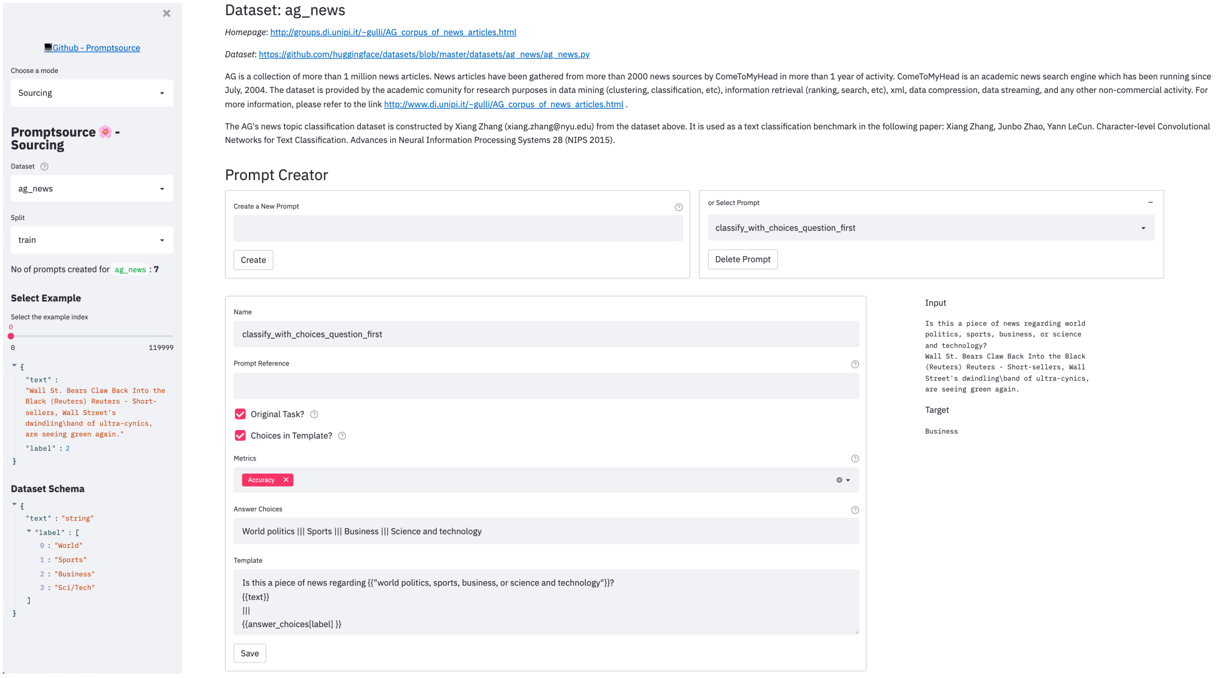


Figure B.3: Complete example of the *Sourcing* view.

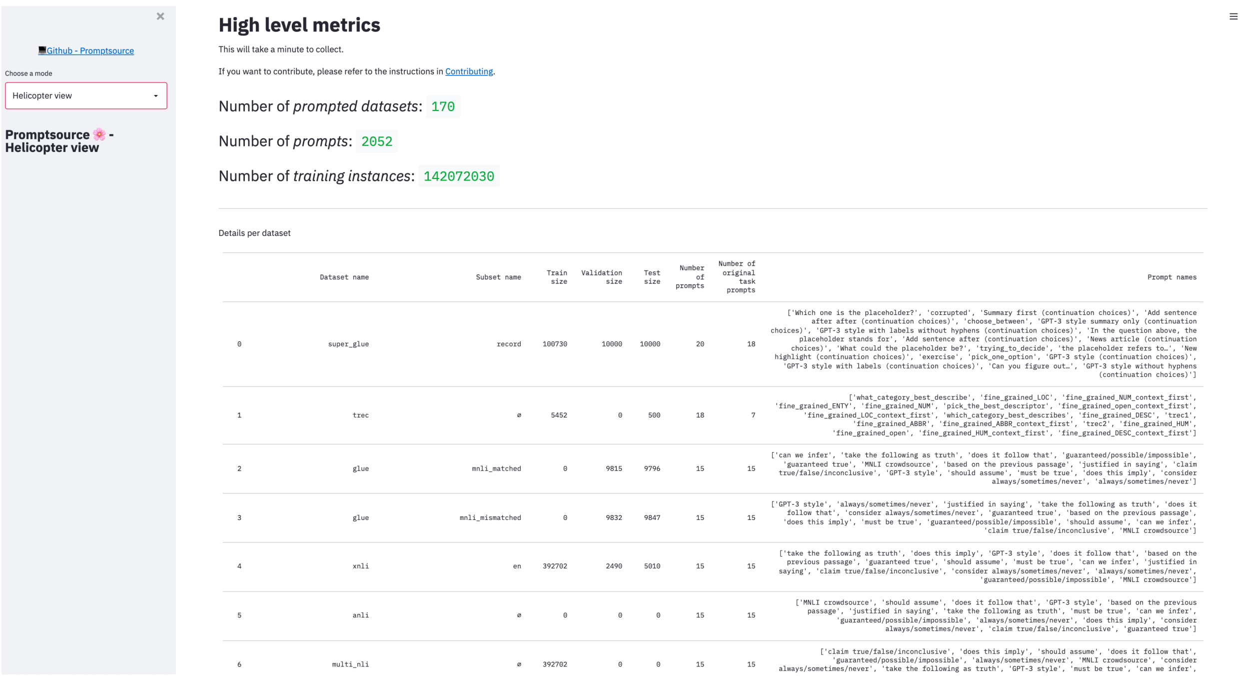


Figure B.4: Complete example of the *Helicopter* view.

Appendix C

Appendix “Accelerating Transformer Inference for Translation via Parallel Decoding”

C.1 Additional implementation details

We run Opus experiments in table 3.1 on an AMD EPYC Milan with 16 cores at 2.45 GHz and 64GB of RAM (accessible on Google Cloud - `c2d-standard-16`). For the scalability experiment in figure 3.3, we also used Google Cloud instances with an increasing number of cores (referred to as `c2d-standard-XX`, where `XX` is the number of used cores). Experiments with MBart50 on table 3.1, 3.2 and 3.3 are performed on a Desktop machine with Ubuntu 20.04.4 LTS, AMD Ryzen 9 3900X 12-Core Processor, 32GB of RAM, and a Palit Nvidia 3090 GPU. Additional experiments with Opus in table 3.3 are also performed on this machine. Models are implemented in Pytorch 1.11.0 [268] and the Huggingface Transformer library [269]. We used python 3.8 and NVIDIA-SMI Drivers 510.73.05 with CUDA version 11.6. For OPUS we used Huggingface models available on the hub under the tag `Helsinki-NLP/opus-mt-{src}-{tgt}` except for the language pair Ro-En where we used the model `Helsinki-NLP/opus-mt-roa-en` and the pair En-De where we used the checkpoint `opus-2021-02-22` [1]. For the model MBart50, we used the `facebook` pre-trained model available on the hub with the tag `mbart-large-50-many-to-many-mmt`. Since this is a multilingual model, we prepend the source and target language tag corresponding properly to the language pair to be translated. We report results for a single run over the test dataset since we found low variance in estimates with multiple runs which can be calculated by simply varying the corresponding parameter in the `config.yaml` file. For each dataset, we used the official test split via the Huggingface dataset library [68]. Datasets statistics are reported in table C.1.

C.2 FLOPs calculation details

We measured computational complexity using floating point operations (FLOPs), which, as the name imply, counts the number of floating point operation performed by a model. This is a standard metric used in literature to measure hardware-agnostic complexity. This means that hardware and software optimizations

[1] https://object.pouta.csc.fi/Tatoeba-MT-models/eng-deu/opus-2021-02-22.zip

Dataset	# Test
WMT 14 De-En [177]	3003
WMT 16 Ro-En [178]	1999
WMT 17 Fi-En [181]	3002
IWSLT 15 En-Vi [179]	1046
IITB En-Hi [180]	2507
FLORES-101 En-It [182]	1012
FLORES-101 En-Fr [182]	1012

Table C.1: Data Statistic

are not counted in the score [142, 143]. We used the ELECTRA flops calculator[2] inserting the number of parameters and the number of training step performed for each model analyzed in table 3.4 according to the training specification in each paper. For inference FLOPs, we computed the decoding cost of each sentence in the testset of WMT14 En-De for each model. For a scale reference, we report in here Table C.2 training flops of other well-known architecture. The code package contains the scripts to replicate all the experiments.

Model	Train FLOPs	Infer. FLOPs	Total FLOPs
Semi-NAT	1.55e17	2.08e13	1.55e17
Shallow Dec.	1.02e19	1.15e13	1.02e19
DSLP	1.93e19	1.58e13	1.93e19
F-VAE	4.06e19	1.58e13	4.06e19
DisCo	4.06e19	1.58e13	4.06e19
SUNDAE	5.27e21	1.58e14	5.27e21
BERT base	6.43e19	-	-
BERT large	1.92e20	-	-
RoBERTa	3.19e21	-	-

Table C.2: FLOPs comparison with other models.

C.3 Additional results

We propose here additional results to the experiments in the paper that were omitted due to limitations constraints. Table 3.3 shows the same experiments of Table 3.1 in the main paper, proposed here on a standard desktop CPU with also the speedup in terms of iterations. It is possible to observe that in the case of MBart50 and PGJ there is a speedup of $8 - 11\%$ in terms of iterations compare to a time speedup of $3 - 8\%$. This means that there is room for improvement for our algorithm. Furthermore, results show that the time speedups are consistent also with standard desktop hardware. Table 3.5 shows the BLEU scores for the cross-lingual experiment. It is possible to observe that parallel decoding algorithms guarantee quality compared to greedy autoregressive and are not so distant from beam search. We show also here in table C.1 some qualitative results for the experiments in table 3.2. Finally, we propose additional visualizations using DGG*viz* in Figure C.2.

[2] https://github.com/google-research/electra/blob/master/flops_computation.py

Example 1 - Wmt16 En-Ro

TARGET	Dl Corbyn va adresa primele dintre cele șase întrebări la care are dreptul la scurt timp după prânz; prestația sa va fi probabil analizată îndeaproape de mass-media și parlamentarii laburiști.	Times (s)	BLEU
A	Dl Corbyn va ridica pentru a adresa prima dintre cele şase întrebări alocate la scurt timp după miezul zilei, iar performanţa sa va fi probabil examinată îndeaproape de presă şi de parlamentarii laburişti.	0.51	19.71
PJ	Dl Corbyn va ridica pentru a adresa prima dintre cele şase întrebări alocate la scurt timp după miezul zilei, iar performanţa sa va fi probabil examinată îndeaproape de presă şi de parlamentarii laburişti.	0.56	19.71
PGJ	Dl Corbyn va ridica pentru a adresa prima dintre cele şase întrebări alocate la scurt timp după miezul zilei, iar performanţa sa va fi probabil examinată îndeaproape de presă şi de parlamentarii laburişti.	0.45	19.71
HGJ	Dl Corbyn va ridica pentru a adresa prima dintre cele şase întrebări alocate la scurt timp după miezul zilei, iar performanţa sa va fi probabil examinată îndeaproape de presă şi de parlamentarii laburişti.	**0.44**	19.71

Example 2 - Flores En-It

TARGET	Quando un piccolo gruppo di esseri viventi (una piccola popolazione) si separa dalla popolazione principale alla quale appartiene (per esempio se si sposta oltre una catena montuosa o un fiume, o si sposta su una nuova isola, rendendo quindi difficile un eventuale ritorno), esso si ritroverà probabilmente in un ambiente diverso da quello in cui si trovava prima.	Times (s)	BLEU
A	Quando un piccolo gruppo di esseri viventi si separa dalla popolazione principale da cui provengono, come se si muovano su una catena di montagne o su un fiume o se si trasferiscono su una nuova isola per non poter tornare facilmente, si troveranno spesso in un ambiente diverso da quello in cui erano prima.	0.61	31.69
PJ	Quando un piccolo gruppo di esseri viventi si separa dalla popolazione principale da cui provengono, come se si muovano su una catena di montagne o su un fiume o se si trasferiscono su una nuova isola per non poter tornare facilmente, si troveranno spesso in un ambiente diverso da quello in cui erano prima.	0.73	31.69
PGJ	Quando un piccolo gruppo di esseri viventi si separa dalla popolazione principale da cui provengono, come se si muovano su una catena di montagne o su un fiume o se si trasferiscono su una nuova isola per non poter tornare facilmente, si troveranno spesso in un ambiente diverso da quello in cui erano prima.	**0.58**	31.69
HGJ	Quando un piccolo gruppo di esseri viventi si separa dalla popolazione principale da cui provengono, come se si muovano su una catena di montagne o su un fiume o se si trasferiscono su una nuova isola per non poter tornare facilmente, si troveranno spesso in un ambiente diverso da quello in cui erano prima.	0.59	31.69

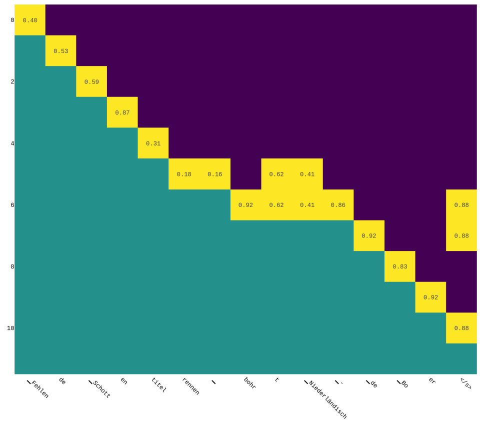
Example 3 - Wmt14 En-De

TARGET	Bei der diesjährigen Veranstaltung gibt es Auftritte von Wanda Sykes, Kathy Griffin und Bill Maher sowie auch von „Stand Up for Heroes“, einer jährlichen Musik- und Comedy-Benefizveranstaltung für Armeeveteranen im Madison Square Garden, bei der unter anderem Bruce Springsteen, Jon Stewart, Roger Waters und Bill Cosby auftreten.	Times (s)	BLEU
A	Zu den diesjährigen Veranstaltungen gehören Auftritte von Wanda Sykes, Kathy Griffin und Bill Maher sowie ”Stand Up for Heroes”, ein jährlicher Musik- und Komödie-Vorteil für Militärveteranen, im Madison Square Garden, mit u.a. Bruce Springsteen, Jon Stewart, Roger Waters und Bill Cosby.	1.30	47.04
PJ	Zu den diesjährigen Veranstaltungen gehören Auftritte von Wanda Sykes, Kathy Griffin und Bill Maher sowie ”Stand Up for Heroes”, ein jährlicher Musik- und Komödie-Vorteil für Militärveteranen, im Madison Square Garden, mit u.a. Bruce Springsteen, Jon Stewart, Roger Waters und Bill Cosby.	2.43	47.04
PGJ	Zu den diesjährigen Veranstaltungen gehören Auftritte von Wanda Sykes, Kathy Griffin und Bill Maher sowie ”Stand Up for Heroes”, ein jährlicher Musik- und Komödie-Vorteil für Militärveteranen, im Madison Square Garden, mit u.a. Bruce Springsteen, Jon Stewart, Roger Waters und Bill Cosby.	1.09	47.04
HGJ	Zu den diesjährigen Veranstaltungen gehören Auftritte von Wanda Sykes, Kathy Griffin und Bill Maher sowie ”Stand Up for Heroes”, ein jährlicher Musik- und Komödie-Vorteil für Militärveteranen, im Madison Square Garden, mit u.a. Bruce Springsteen, Jon Stewart, Roger Waters und Bill Cosby.	**1.08**	47.04

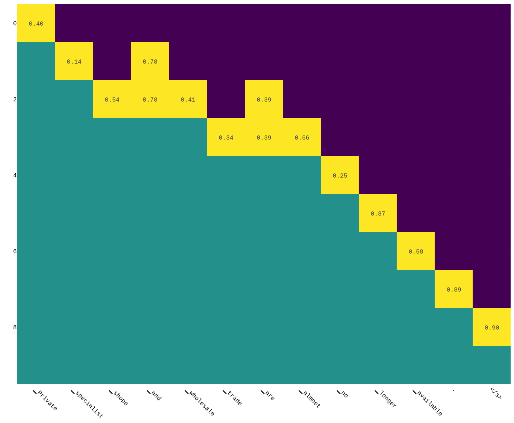
Example 4 - Flores En-Fr			
TARGET	Cinq minutes après le début de l'exposition, un vent se met à souffler pour atteindre, environ une minute plus tard, la vitesse de 70km/h... puis la pluie arrive, mais si forte et si grosse qu'elle frappe votre peau comme une aiguille, puis la grêle tombe du ciel, les gens paniquent, crient et se roulent dessus.	Times (s)	BLEU
A	Cinq minutes après l'exposition, le vent commence à tourner, environ un minute plus tard, le vent atteint 70 km/h, puis la pluie arrive, mais si forte et si grande qu'elle vous frappe la peau comme une aiguille, puis le hail tombe du ciel, les gens paniquent, s'expriment et se courent l'un sur l'autre.	0.82	39.90
PJ	Cinq minutes après l'exposition, le vent commence à tourner, environ un minute plus tard, le vent atteint 70 km/h, puis la pluie arrive, mais si forte et si grande qu'elle vous frappe la peau comme une aiguille, puis le hail tombe du ciel, les gens paniquent, s'expriment et se courent l'un sur l'autre.	0.94	39.90
PGJ	Cinq minutes après l'exposition, le vent commence à tourner, environ un minute plus tard, le vent atteint 70 km/h, puis la pluie arrive, mais si forte et si grande qu'elle vous frappe la peau comme une aiguille, puis le hail tombe du ciel, les gens paniquent, s'expriment et se courent l'un sur l'autre.	0.73	39.90
HGJ	Cinq minutes après l'exposition, le vent commence à tourner, environ un minute plus tard, le vent atteint 70 km/h, puis la pluie arrive, mais si forte et si grande qu'elle vous frappe la peau comme une aiguille, puis le hail tombe du ciel, les gens paniquent, s'expriment et se courent l'un sur l'autre.	**0.72**	39.90

Example 5 - IWSLT15 En-Vi			
TARGET	Tôi yêu sức mạnh của tiến hoá, và tôi nhận ra một điều rất cơ bản đối với mọi sự sống trong những sinh vật đơn bào, mỗi tế bào chỉ đơn giản là phân chia, và mọi thông tin di truyền trong tế bào đó được truyền sang hai tế bào con.	Times (s)	BLEU
A	Tôi đã yêu thích sức mạnh của sự tiến hoá và tôi nhận ra một điều rất căn bản trong hầu hết sự tồn tại của sự sống trong các sinh vật đơn bào mỗi tế bào đơn giản là chia ra và tất cả năng lượng di truyền của tế bào đó được vận hành trong cả hai tế bào con.	0.61	31.45
PJ	Tôi đã yêu thích sức mạnh của sự tiến hoá và tôi nhận ra một điều rất căn bản trong hầu hết sự tồn tại của sự sống trong các sinh vật đơn bào mỗi tế bào đơn giản là chia ra và tất cả năng lượng di truyền của tế bào đó được vận hành trong cả hai tế bào con.	0.71	31.45
PGJ	Tôi đã yêu thích sức mạnh của sự tiến hoá và tôi nhận ra một điều rất căn bản trong hầu hết sự tồn tại của sự sống trong các sinh vật đơn bào mỗi tế bào đơn giản là chia ra và tất cả năng lượng di truyền của tế bào đó được vận hành trong cả hai tế bào con.	0.54	31.45
HGJ	Tôi đã yêu thích sức mạnh của sự tiến hoá và tôi nhận ra một điều rất căn bản trong hầu hết sự tồn tại của sự sống trong các sinh vật đơn bào mỗi tế bào đơn giản là chia ra và tất cả năng lượng di truyền của tế bào đó được vận hành trong cả hai tế bào con.	**0.53**	31.45

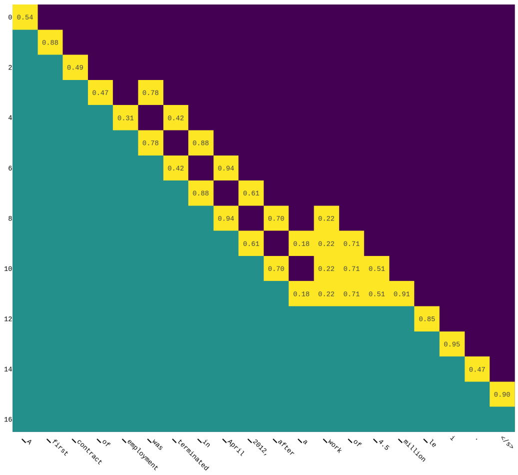
Table 7: Translation examples generated with the autoregressive (A) and the different decoding algorithms proposed (PJ, PGJ, HGJ) on Opus (WMT datasets) and MBart50. The decoding time is shown in seconds.



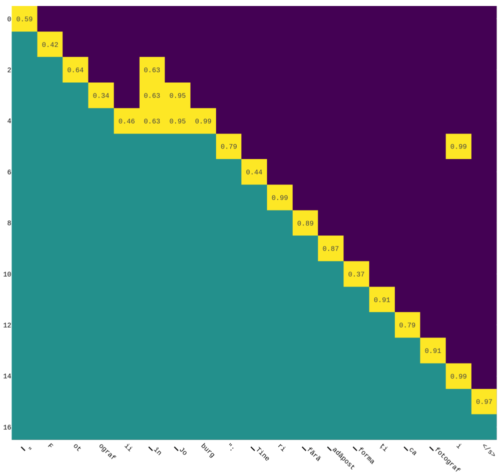
(a) En-De: "Lack of Scots title race bores Dutch - de Boer"→"Fehlende Schottentitelrennen bohrt Niederlandisch - de Boer"



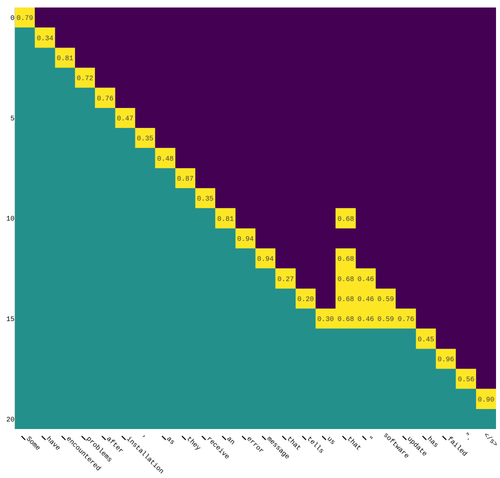
(b) De-En: "Private Fachgeschafte und auch den Großhandel gibt es fast nicht mehr."→"Private specialist shops and wholesale trade are almost no longer available."



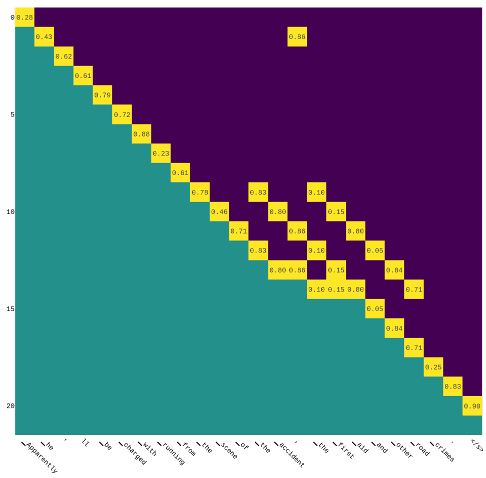
(c) Ro-En: "Un prim contract de lucrări a fost reziliat în aprilie 2012, după ce se efectuaseră lucrări de 4,5 milioane lei."→ "A first contract of employment was terminated in April 2012, after a work of 4.5 million lei."



(d) En-Ro: "'Shot in Joburg': Homeless youth trained as photographers"→ ""Fotografii in Joburg": Tineri fără adăpost formaţi ca fotografi"



(e) De-En: "Einige sind nach der Installation auf Probleme gestoßen, da sie eine Fehlermeldung erhalten, die mitteilt, dass die "Software-Aktualisierung fehlgeschlagen" ist."→"Some have encountered problems after installation, as they receive an error message that tells us that "software update has failed"."



(f) Ro-En: "Se pare că va fi acuzat de fugă de la locul accidentului, neoferirea primului ajutor și alte infracțiuni rutiere."→ "Apparently he'll be charged with running from the scene of the accident, the first aid and other road crimes."

Figure C.2: DGG*viz* additional visualizations

Appendix D

Appendix “Camoscio: An italian Instruction-tuned LLaMA”

D.1 Implementation Details

The model was trained with the LoRA Parameter-efficient Finetuning technique [209], using the Hugging Face Transformers, PEFT, Datasets libraries [68, 270, 271] and the library Alpaca-LoRA [225]. Specifically, it was trained for 3 epochs with int8 quantization [272] on a standard desktop GPU Nvidia 3090 on a machine with Ubuntu 20.04.4 LTS, AMD Ryzen 9 3900X 12-Core Processor and 32GB of RAM. The model was trained with batches of dimension 4 and gradient accumulation to obtain a final “virtual batch” of 128. The maximum length used for training is 256 tokens. The learning rate is set to 3×10^{-4} with AdamW [273] and a total of 100 warmup steps are performed. We used a *lora_r* (i.e., the dimensionality of the low-rank update of the matrices) equals to 8, *lora_alpha* equals to 16 and *lora_dropout* equals to 0.05. We used LoRA adapters just for the matrices *Query* and *Value* in all the attention layers in the LLaMA model, following the original LoRA paper. We used the LLaMA 7 billion checkpoint by loading it from the Hugging Face Hub repository *“decapoda-research/llama-7b-hf”*.

D.1.1 Exact Match via ChatGPT

Exact Match via ChatGPT is a metric we introduced to evaluate the performance of Camoscio in the zero-shot setting on the question-answering task. This metric assesses whether the answer provided by a model is correct or not, compared to a ground-truth answer, without the need to have an exact string match (Exact Match). Specifically, we used an external LM (in our case *gpt-3.5-turbo*) that acts as a judge with the scope of verifying the correctness of the answer. We used a prompt similar to the following to compute this metric[1]:

“Given the context below and the corresponding question, please indicate whether the answer is correct (1) or not (0). Use a dict format in the response.

Context: `{Context}`
Question: `{Question}`
Correct gold answer: `{Correct_answer}`

[1]Evaluation script available here

Answer: `{Answer}`"

This prompt returns 1 if the answer is correct or zero otherwise. These values are then used to compute the *Exact Match via ChatGPT* with the same procedure of the Exact Match score.

D.1.2 Generation Examples

We propose here some generation examples. We also provide in the GitHub repository the generated text for all the examples in the test datasets, together with other open-ended generation examples.

D.2 NewSum-IT ("Il Post")

Input text:
L'esame di maturità è sempre un momento importante, quando lo si vive, anche se poi forse a distanza di anni lo si ricorderà con distacco e pensando che non era poi così difficile, o memorabile. Per molti continuerà a farsi vivo di notte, nei sogni. Quello di quest'anno, che inizia oggi, di certo sarà un po' diverso: senza tema e dizionari, senza attese di gruppo fuori dalla scuola prima di iniziare, ma con mascherine e gel igienizzanti. Agli studenti della Scuola Holden, la scuola di scrittura di Torino fondata da Alessandro Baricco, è stato chiesto di scrivere una lettera ai maturandi del 2020, per incoraggiarli e dare loro sostegno. Quella scritta da Nausica Anerdi, che ha frequentato il primo anno di Academy, il corso di laurea triennale in scrittura della Holden, è stata pubblicata oggi sul Post e su Repubblica, La Stampa e sulla pagina di Torino del Corriere della Sera. Tutte le altre potete leggerle qui.

Camoscio generated text:
Nausica Anerdi, che ha frequentato il primo anno di Academy, il corso di laurea triennale in scrittura della Holden, ha scritto una lettera ai maturandi del 2020, per incoraggiarli e dare loro sostegno.

mT5-base generated text:
La lettera ai maturandi del 2020. È stata scritta da Nausica Anerdi, che ha frequentato il primo anno di Academy, la scuola di scrittura di Torino fondata da Alessandro Baricco.

Ground Truth:
Una lettera a chi sta per fare la maturità. Da parte di una studentessa della Scuola Holden, del corso di laurea triennale in scrittura.

D.3 NewSum-IT ("Fanpage.it")

Input text:
Continuano ad aumentare i nuovi casi di coronavirus nel nostro Paese. Sono stati 2.800 i contagi registrati ieri: numeri che preoccupano il governo e che ricordano quelli delle fasi più critiche dell'emergenza. Domani l'esecutivo si riunirà e valuterà se sia il caso di rendere più severe le norme anti-contagio attualmente in vigore. Entro la prossima settimana si attende il nuovo Dpcm contenente le misure di contrasto all'epidemia, mentre si valuta la proroga dello stato di emergenza fino al prossimo 31 gennaio 2021. Ma vediamo quindi

quali sono queste nuove regole che il governo sta pensando di introdurre per frenare la curva dei contagi. L'obbligo di portare la mascherina all'aperto, già introdotto nei giorni scorsi in alcune zone, sarà esteso a tutto il territorio nazionale. Oltre quindi a confermare la necessità di indossare sempre il dispositivo di protezione nei luoghi chiusi, di igienizzare frequentemente le mani e di rispettare le distanze di sicurezza e il divieto di assembramento, il governo studia se rendere alcune misure più stringenti. In particolare, saranno potenziati i controlli nei luoghi della movida o dove è più facile che si vadano a costituire affollamenti. Le operazioni di vigilanza saranno affidate anche ai militari impegnati nel progetto "Strade secure". Il ministro della Salute, Roberto Speranza, si sarebbe detto favorevole all'estensione dell'obbligo di mascherina all'aperto a tutto il Paese, mantenendosi per quella linea di prudenza sostenuta fin dallo scoppio dell'emergenza. La stessa considerazione, d'altronde, è condivisa anche dalla autorità locali, dal momento che diversi governatori hanno già emesso delle ordinanze che impongono di coprire naso e bocca all'aperto, con multe salate per chi non rispetta le nuove regole. Il governo continua ad escludere la possibilità di tornare a una situazione di lockdown generale. Ma se i contagi continuassero a crescere a questi ritmi, potrebbero essere imposte alcune limitazioni in specifiche zone. In questo senso si sta anche considerando di introdurre un chiusura per bar e locali alle 22 o alle 23: una chiusura anticipata affinché non si ripetano situazioni di movida, come abbiamo visto anche quest'estate, in cui si favorisce la diffusione dell'infezione. All'aperto non sarà inoltre consentito riunire più di mille persone alla volta (ovviamente sempre secondo le regole del distanziamento): questa regola sarà valida sia per le manifestazioni che per gli eventi sportivi. Anche il limite per gli stadi resta a mille spettatori. Negli spazi chiusi, invece, il tetto rimane a 200 persone: questo vale sia per cinema, teatri e sale da concerto, ma anche per le celebrazioni private come i matrimoni. Le discoteche resteranno sicuramente chiuse, nei ristoranti si dovrà continuare a rispettare il distanziamento e anche nei negozi gli ingressi saranno contingentati.

Camoscio generated text:
Il governo sta valutando l'estensione dell'obbligo di mascherina all'aperto a tutto il Paese.

mT5-base generated text:
Il governo sta pensando di introdurre nuove regole anti-contagio per frenare la curva dei contagi. Il provvedimento è stato esteso a tutto il territorio nazionale, mentre si valuta la proroga dello stato di emergenza fino al prossimo 31 gennaio 2021. Vediamo quindi quali sono le nuove regole che il governo sta pensando di introdurre per frenare la curva dei contagi.

Ground Truth:
Con i nuovi casi di coronavirus in continuo aumento, il governo pensa a inasprire le regole anti-contagio per contrastare la diffusione dell'infezione. Si va verso l'obbligo di indossare la mascherina anche all'aperto e la chiusura a bar e ristoranti alle 22 o alle 23. Anche l'esercito sarà impiegato nei controlli.

D.4 XFORMAL-IT (F→I)

Input text:
Buongiorno, quella su E-Bay è molto bella.

Camoscio generated text:
Ciao, quella su E-Bay è davvero bella!

mT5-base generated text:
Buongiorno, quella su E-Bay è molto bella.

Ground Truth:
!ciaooooooooquella su e-bay è troppo carina!!

D.5 XFORMAL-IT (I→F)

Input text:
no mai , però ora per recuperare vado subito a lavare i piatti!

Camoscio generated text:
È ora per lavare i piatti.

mT5-base generated text:
No, ma ora per recuperare, vado subito a lavare i piatti.

Ground Truth:
Giammai! Ora però per recuperare vado subito a lavare i piatti

D.6 SQuAD-IT

Input text:
La crisi petrolifera del 1973 iniziò nell' ottobre 1973 quando i membri dell' Organizzazione dei Paesi esportatori di petrolio arabo (OAPEC, composta dai membri arabi dell' OPEC più Egitto e Siria) proclamarono un embargo petrolifero. Alla fine dell' embargo, nel marzo 1974, il prezzo del petrolio era salito da 3 dollari al barile a quasi 12 dollari a livello mondiale; i prezzi americani erano notevolmente più elevati. L' embargo ha causato una crisi petrolifera, o "shock", con molti effetti a breve e lungo termine sulla politica globale e sull' economia globale. Più tardi fu chiamato il "primo shock petrolifero", seguito dalla crisi petrolifera del 1979, definita il "secondo shock petrolifero". Domanda: Quando è stata la seconda crisi petrolifera?

Camoscio generated text:
La seconda crisi petrolifera è stata nel 1979.

mT5-base generated text:
1979

Ground Truth:
1979

Acknowledgements

I would like to express my gratitude to my advisor, Prof. Emanuele Rodolà, who took a chance on me as his PhD student at Sapienza University in 2020 in a moment of high uncertainty during the COVID-19 pandemic and gave me the freedom and flexibility to explore topics that I liked. My gratitude extends to Prof. Marco Baroni, who hosted me at Universitat Pompeu Fabra (UPF) during my visiting period at the Computational Linguistic and Linguistic Theory Lab and trusted me from the very first interaction with a cold email, and to Prof. Fabio Massimo Zanzotto, who sparked my interest in pursuing a research career path in NLP.

I am deeply grateful to my fellow researchers at GLADIA lab of Sapienza University: Silvio Severino, Giovanni Trappolini, Antonio Norelli, Emilian Postolache, Michele Mancusi, Giorgio Mariani, Donato Crisostomi, Marco Fumero, Luca Moschella, Michele Miranda, Valentino Maiorca, Irene Cannistraci, Antonio Pio Ricciardi, Irene Tallini, Adrian Robert Minut, Filippo Maggioli, Marco Pegoraro, Daniele Baieri, Giorgio Strano, Arianna Rampini, Riccardo Marin, Simone Melzi, and Luca Cosmo. In addition to this, I would like to extend my gratitude to friends and colleagues at Sapienza University: Andrea Bacciu, Marzia Riso, Viviana Arrigoni, Daniele Izzi, Giacomo Nazzaro, Bardh Prenkaj, Lorenzo Madeddu, Leonardo Picchiami, Federico Trombetti, Emanuele Giona, Marco Esposito; and to collaborators over different projects: Leonardo Ranaldi, Elena Sofia Ruzzetti, Fabrizio Silvestri, Giuseppe Attanasio, Moreno La Quatra, Beatrice Savoldi, Giambattista Parascandolo, Alon Halevy, Sébastien Bratières, Simone Scardapane, Iacopo Masi.

I am profoundly grateful to Gemma Boleda and Marco Baroni for hosting me during my six-month research visit at Universitat Pompeu Fabra and to friends and colleagues met during this time: Roberto Dessì, Mateo Mahaut, Eleonora Gualdoni, Nathanaël Carraz Rakotonirina, Francesca Franzon, Lucas Weber, Corentin Kervadec, Germán Kruszewski, Emily Cheng, Xixian Liao, Andrea Zaninello, Daniela Mihai, Ionut-Teodor Sorodoc.

Special thanks to my former colleagues at Apple for their incredible support: Adam Goliński, Luca Zappella, Xavier Suau, Miguel Sarabia, Pau Rodríguez López, Federico Danieli, Arno Blaas, Nicholas Apostoloff, Miao Xiong, Michael Kirchhof, Sinead Williamson, Gerard Gállego, and Federico Scozzafava.

I would like to thank Hugging Face and everyone involved in organizing the BigScience Workshop, especially those in the Prompt Engineering working group. The first two works of this thesis would not have been possible without this workshop and represent not just individual efforts but the power of collaborative research and open science. I am truly grateful to everyone who has contributed to these works, directly or indirectly. I would like to acknowledge the support of various institutions that have made this research possible, including the Italian Ministry of University and Research (MUR) which funded my PhD scholarship, Translated which awarded me an Imminent Research Grant, Sapienza Univerisity that awarded me two research grants "Avvio alla ricerca", and Apple for enabling me to pursue a research internship.

On a personal note, I owe profound thanks to my partner, Livia, for her unwavering love and support throughout the highs and lows of this PhD journey, and to my family, especially my parents Fiorella and Eliseo, for their encouragement and belief in me over the years.

To my friends, both within and beyond academia, thank you for being an integral part of this journey: Paolo Carta, Alessandro Beasley, Alessio Fanelli, Tiziano Carulli, Gianluca Galli, Marta Giacomelli, Emanuele Rossi, Maria Giulia Casucci, Giovanni Ciampi, Marco Del Tredici, Gabriele Sarti, Valerio Carletti, Daniele Santinelli, Marco Siliquini, and Marta Raggi. Your support has been invaluable.

Lastly, I extend my gratitude to everyone whose insights and conversations have left a lasting impression on my work. I am thankful to Prof. Pasquale Minervini (University of Edinburgh), Prof. Alessandro Raganato

(University of Milano-Bicocca), and Prof. Felice Dell'Orletta (Institute of Computational Linguistics, CNR, Pisa) for their review of this thesis.

Andrea Santilli, Rome, 15 January 2025

Ringraziamenti

Vorrei esprimere la mia gratitudine al mio relatore, il Prof. Emanuele Rodolà, che nel 2020 ha scommesso su di me come suo studente di dottorato alla Sapienza Università di Roma, in un momento di grande incertezza durante la pandemia di COVID-19, offrendomi la libertà e la flessibilità di esplorare argomenti che mi interessavano. La mia gratitudine si estende al Prof. Marco Baroni, che mi ha ospitato presso l'Universitat Pompeu Fabra (UPF) durante il mio periodo di ricerca al Computational Linguistic and Linguistic Theory Lab, fidandosi di me fin dal primo contatto via email, e al Prof. Fabio Massimo Zanzotto, che ha acceso il mio interesse per intraprendere una carriera di ricerca nell'ambito del NLP.

Sono profondamente grato ai miei colleghi ricercatori del laboratorio GLADIA della Sapienza Università di Roma: Silvio Severino, Giovanni Trappolini, Antonio Norelli, Emilian Postolache, Michele Mancusi, Giorgio Mariani, Donato Crisostomi, Marco Fumero, Luca Moschella, Michele Miranda, Valentino Maiorca, Irene Cannistraci, Antonio Pio Ricciardi, Irene Tallini, Adrian Robert Minut, Filippo Maggioli, Marco Pegoraro, Daniele Baieri, Giorgio Strano, Arianna Rampini, Riccardo Marin, Simone Melzi e Luca Cosmo.

Inoltre, vorrei estendere la mia gratitudine agli amici e colleghi della Sapienza Università: Andrea Bacciu, Marzia Riso, Viviana Arrigoni, Daniele Izzi, Giacomo Nazzaro, Bardh Prenkaj, Lorenzo Madeddu, Leonardo Picchiami, Federico Trombetti, Emanuele Giona, Marco Esposito; e ai collaboratori di vari progetti: Leonardo Ranaldi, Elena Sofia Ruzzetti, Fabrizio Silvestri, Giuseppe Attanasio, Moreno La Quatra, Beatrice Savoldi, Giambattista Parascandolo, Alon Halevy, Sébastien Bratières, Simone Scardapane, Iacopo Masi.

Sono profondamente grato a Gemma Boleda e Marco Baroni per avermi ospitato durante i sei mesi di ricerca presso l'Universitat Pompeu Fabra e agli amici e colleghi conosciuti in questo periodo: Roberto Dessì, Mateo Mahaut, Eleonora Gualdoni, Nathanaël Carraz Rakotonirina, Francesca Franzon, Lucas Weber, Corentin Kervadec, Germán Kruszewski, Emily Cheng, Xixian Liao, Andrea Zaninello, Daniela Mihai, Ionut-Teodor Sorodoc.

Un ringraziamento speciale ai miei ex colleghi di Apple per il loro incredibile supporto: Adam Goliński, Luca Zappella, Xavier Suau, Miguel Sarabia, Pau Rodríguez López, Federico Danieli, Arno Blaas, Nicholas Apostoloff, Miao Xiong, Michael Kirchhof, Sinead Williamson, Gerard Gállego e Federico Scozzafava.

Vorrei ringraziare Hugging Face e tutti coloro che hanno organizzato il BigScience Workshop, in particolare il gruppo di lavoro Prompt Engineering. I primi due lavori di questa tesi non sarebbero stati possibili senza questo workshop e rappresentano non solo sforzi individuali, ma il potere della ricerca collaborativa e della scienza aperta. Sono davvero grato a tutti coloro che hanno contribuito a questi lavori, direttamente o indirettamente.

Vorrei riconoscere il supporto di varie istituzioni che hanno reso possibile questa ricerca, tra cui il Ministero Italiano dell'Università e della Ricerca (MUR) che ha finanziato la mia borsa di dottorato, Translated che mi ha assegnato un Imminent Research Grant, la Sapienza Università di Roma che mi ha assegnato due borse di ricerca "Avvio alla ricerca" e Apple per avermi permesso di svolgere uno stage di ricerca.

A livello personale, devo un ringraziamento profondo alla mia compagna, Livia, per il suo amore e supporto incrollabili durante gli alti e bassi di questo percorso di dottorato, e alla mia famiglia, in particolare ai miei genitori Fiorella ed Eliseo, per il loro incoraggiamento e la loro fiducia in me nel corso degli anni.

Ai miei amici, sia all'interno che al di fuori dell'accademia, grazie per essere stati una parte integrante di questo viaggio: Paolo Carta, Alessandro Beasley, Alessio Fanelli, Tiziano Carulli, Gianluca Galli, Marta

Giacomelli, Emanuele Rossi, Maria Giulia Casucci, Giovanni Ciampi, Marco Del Tredici, Gabriele Sarti, Valerio Carletti, Daniele Santinelli, Marco Siliquini e Marta Raggi. Il vostro supporto è stato inestimabile.

Infine, estendo la mia gratitudine a tutti coloro le cui intuizioni e conversazioni hanno lasciato un'impressione duratura sul mio lavoro. Ringrazio il Prof. Pasquale Minervini (University of Edinburgh), il Prof. Alessandro Raganato (University of Milano-Bicocca) e il Prof. Felice Dell'Orletta (Istituto di Linguistica Computazionale, CNR, Pisa) per la loro revisione di questa tesi.

Andrea Santilli, Roma, 15 Gennaio 2025