Grounded and Consistent Question Answering

Christopher Alberti

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2023

Abstract

Grounded and Consistent Question Answering

Christopher Alberti

This thesis describes advancements in question answering along three general directions: model architecture extensions, explainable question answering, and data augmentation. Chapter 2 describes the first state-of-the-art model for the Natural Questions dataset based on pretrained transformers. Chapters 3 and 4 describe extensions to the model architecture designed to accomodate long textual inputs and multimodal text+image inputs, establishing new state-of-the-art results on the Natural Questions and on the VCR dataset. Chapter 5 shows that significant improvements can be obtained with data augmentation on the SQuAD and Natural Questions dataset, introducing roundtrip consistency as a simple heuristic to improve the quality of synthetic data. In Chapters 6 and 7 we explore explainable question answering, demonstrating the usefulness of a new concrete kind of structured explanations, QED, and proposing a semantic analysis of why-questions in the Natural Questions, as a way of better understanding the nature of real world explanations. Finally, in Chapters 8 and 9 we delve into more exploratory data augmentation techniques for question answering. We look respectively at how straight-through gradients can be utilized to optimize roundtrip consistency in a pipeline of models on the fly, and at how very recent large language models like PaLM can be used to generate synthetic question answering datasets for new languages given as few as five representative examples per language.

Table of Contents

Acknowledgments xiv

Chapter 1: Introduction 1

1.1 Base Question Answering Architecture 3

1.2 Architectures for Question Answering Extensions 4

1.3 Explainable Question Answering 8

1.4 Improving Question Answering with Data Augmentation 10

Chapter 2: Answering Natural Questions with Transformers 15

2.1 Natural Questions: A Benchmark for Question Answering Research 15

2.2 A BERT Baseline for the Natural Questions Dataset 16

2.3 Data Preprocessing 17

2.4 Model 18

2.5 Experiments 20

Chapter 3: ETC: Encoding Long and Structured Inputs in Transformers 22

3.1 Overview 22

3.2 Background 24

3.3 Extended Transformer Construction 26

3.3.1 Relative Position Encoding 26

3.3.2 Global-Local Attention . . . 27
3.3.3 Long Inputs and Global-Local Attention . . . 29
3.3.4 Structured Inputs . . . 30
3.3.5 Pretraining Tasks . . . 31
3.3.6 Lifting Weights from Existing Models . . . 32
3.4 Empirical Evaluation . . . 32
3.4.1 Training Configuration . . . 34
3.4.2 Results on the Dev Set . . . 34
3.4.3 Official Leaderboard Results . . . 37
3.5 Conclusions . . . 38

Chapter 4: Fusion of Detected Objects in Text for VQA . . . 39
4.1 Overview . . . 39
4.2 Problem Formulation . . . 41
4.3 Models and Methods . . . 42
4.4 Dual Encoder . . . 42
4.5 B2T2 . . . 44
4.6 Loss . . . 45
4.7 Pretraining on Conceptual Captions . . . 45
4.8 Implementation Details . . . 46
4.9 Data . . . 48
4.10 Experimental Results . . . 49
4.11 VCR Task Performance . . . 49

4.12 Ablations . . . 50
4.13 Error Analysis . . . 52
4.14 Related Work . . . 54
4.15 Conclusion . . . 55

Chapter 5: QA Corpora Generation with Roundtrip Consistency . . . 56
5.1 Overview . . . 56
5.2 Model . . . 57
5.2.1 Question (Un)Conditional Extractive QA . . . 58
5.2.2 Question Generation: Fine-tuning Only . . . 59
5.2.3 Question Generation: Full Pretraining . . . 60
5.3 Experiments . . . 60
5.3.1 Experimental Setup . . . 60
5.3.2 Results . . . 63
5.4 Conclusion . . . 64

Chapter 6: QED: A Framework and Dataset for Explanations in Question Answering . . . 65
6.1 Introduction . . . 65
6.2 Motivation: The Need for Explanations in Question Answering . . . 67
6.3 Annotation Definition . . . 68
6.3.1 An Overview of the Approach . . . 68
6.3.2 A Formal Definition . . . 70
6.3.3 Extending Annotations to Include Bridging . . . 71
6.4 QED Annotations for the Natural Questions . . . 73

6.4.1 Agreement Statistics 74
6.4.2 Types of Referential Expressions 74
6.5 Tasks and Baseline Results 77
6.5.1 Four Tasks 77
6.5.2 A Baseline Model for Task 1 79
6.5.3 A Baseline Model for Task 2 81
6.6 Rater Study 82
6.6.1 Task Setup 83
6.6.2 Results 83
6.6.3 Effectiveness of explanations 85
6.7 Discussion 86
6.7.1 QED and strong explainability 86
6.7.2 Potential Extensions to the QED Framework 87
6.7.3 Future uses of QED representations 87
6.8 Conclusions 88

Chapter 7: A Semantic Analysis of Negative Why Questions in the Natural Questions Dataset 89
7.1 Overview 89
7.2 Reason and Purpose 90
7.3 The Syntactic View 92
7.4 Negative why-questions 95
7.5 A Modal Framework for why-questions 98
7.6 Other Types of Answers to why-questions 102

7.7 Conclusion . . . 103

Chapter 8: Towards Computationally Verifiable Semantic Grounding For Language Models . . . 105

8.1 Overview . . . 105

8.2 Background and Intuition . . . 106

8.3 Related Work . . . 108

8.4 Model . . . 109

8.4.1 Straight Through Approximation . . . 110

8.5 Data Set . . . 111

8.6 Experiments . . . 112

8.6.1 Automatic Evaluation . . . 113

8.6.2 Human Evaluation . . . 115

8.6.3 Generation Examples . . . 117

8.7 Conclusions . . . 118

8.8 Limitations and Future Work . . . 119

Chapter 9: QAMELEON: Multilingual QA with Only 5 Examples . . . 120

9.1 Overview . . . 120

9.2 Synthetic Data Generation . . . 123

9.2.1 Machine Translation (MT) . . . 123

9.2.2 Prompt Engineering (PE) . . . 124

9.2.3 QAMELEON (PT) . . . 126

9.2.4 Data Assumptions . . . 127

9.3 Experimental Setup 127
9.3.1 Datasets 127
9.3.2 Model Configuration 129
9.4 Results 131
9.5 Data Analysis 136
9.6 Related Work 137
9.7 Conclusions 138

Conclusion or Epilogue 139

References 141

List of Figures

1.1 Thesis outline. 2

1.2 Long and short answer to the question "when are hops added to the brewing process", from the Natural Questions dataset. The long answer is a paragraph from the Wikipedia page "Brewing". The short answer is the string "The boiling process". 3

1.3 Sparse self-attention from the ETC transformer, presented in Chapter 3, to handle long document contexts in the Natural Questions and other question answering datasets. 5

1.4 An input image and question from the VCR dataset. The image is augmented with bounding boxes produced by Mask-RCNN, an automatic object detection system. The VCR task consists of picking the correct answer and answer rationale out of four options for each. 6

1.5 Example of a structured explanation for question answering following the QED framework described in Chapter 6. 8

1.6 Data augmentation strategy for question answering explored in Chapter 5. 11

1.7 Verifiable language modeling intuition developed in Chapter 8. The approach seeks to re-frame language modeling as a semantic auto-encoder, where the intermediate representation is natural language. 12

1.8 Data augmentation methods explored in Chapter 9 for low resource language scenarios. 13

3.1 An illustration of mechanisms to scale attention to long inputs, including our proposed model, ETC. 24

3.2 Sparsity diagram showing which attention queries (rows) can attend to which attention keys (columns) a) for standard Transformer attention with input size n; b) for global-local attention with input sizes n_g, n_l, and radius r; c) how the l2l attention piece is reshaped into a much smaller attention matrix, limited by local radius. 27

3.3 Example attention patterns for handling (a) long inputs and (b) structured inputs. White background means attention is masked via M, and the other colors indicate different relative position labels. 29

4.1 An example from the VCR dataset. The tasks consists in picking an answer A_{1-4}, and then picking a rationale R_{1-4}. The data contains explicit pointers in the text to bounding boxes in the image. 40

4.2 Dual Encoder architecture with late fusion. The model extracts a single visual feature vector from the entire image. Bounding boxes are ignored. 43

4.3 B2T2 architecture with early fusion. Bounding boxes are inserted where they are mentioned in the text and at the end of the input, as described in Sec. 4.9. 43

4.4 How input embeddings are computed in our B2T2 architecture. 46

4.5 Mask-LM pretraining for B2T2. 47

4.6 Boxplot of dev $Q \rightarrow A$ accuracy on VCR with and without pretraining. Pretraining on Conceptual Captions lowers variance when fine-tuning on VCR, from a grid search on multiple random seeds, learning rates, and VCR training epochs. 52

5.1 Learning curves for pretraining using synthetic question-answering data (fine-tuning only setting). "no-RT" refers to omitting the roundtrip consistency check. Best exact match is reported after fine-tuning on SQuAD2. Performance improves with the amount of synthetic data. For a fixed amount of synthetic data, having a more diverse source (NQ+SQuAD vs. just SQuAD) yields higher accuracies. Roundtrip filtering gives further improvements. 62

6.1 QED explanations decompose the question-passage relationship in terms of referential equality and predicate entailment. 66

6.2 An example outside of QED's current scope, since multiple passage sentences contribute an answer. 73

6.3 Examples from the QED dataset, grouped according to different types of referential equalities. 75

6.4 Referential equalities from the QED corpus. 76

6.5 Counts for 100 randomly drawn referential equality annotations from the QED corpus, subcategorized by expression type in the question (Qu.) and passage (Ps.). P/N/A/G/Pn/B/M refer to Proper/Def(non-ana)/Def(ana)/Generic/Pronoun/Bridge/Misc. 76

6.6 Sorted, per-question evaluation accuracies from different rater study settings, with 95% binomial confidence intervals. Left three plots correspond to trials with incorrect answers highlighted; right three plots to trials with correct answers highlighted. Dashed red lines correspond to the average accuracy for each setting, identical to the numbers in Table 6.4. 84

8.1 Screenshots of our evaluation template: (1) fluency question, (2) data coverage questions, and (3) a question asking annotators if the triples cover all the information in the text. 114

9.1 Synthetic data generation for multilingual question-answering (QA). Left: Examples of the multilingual QA task. Translations are added for readability. Middle: Strategies for localizing QA models to new languages: 1. Using English QA data as a zero-shot approach, 2. with Machine Translation (MT) to approximate training data for supervised learning, and 3. few-shot approaches with a handful of multilingual examples. Right: Model performance on the multilingual QA task. We report average Exact Match (EM) across all languages on the TyDiQA-GoldP dataset [142]. 121

9.2 Effect of synthetic data size on downstream QA performance (Average EM on TyDiQA-GoldP evaluation set); results shown for mT5-XL QA model fine-tuned via Machine Translation (MT), Prompt Engineering (PE), Prompt Tuning (QAmeleon (PT)), and combinations thereof (PE + MT and QAmeleon (PT) + MT). 133

List of Tables

1.1 Several predictions made by [13] on admissible answers to different kinds of why-questions can be disproven by real world examples in the Natural Questions dataset. The prediction about the absence of purpose answers to negative why-questions however holds surprisingly well. 9

2.1 Our results on NQ compared to the baselines in the original dataset paper and to the performance of a single human annotator and of an ensemble of human annotators. The systems used in previous NQ baselines are DocumentQA [17], DecAtt [18], and Document Reader [19]. 20

3.1 Dataset stats (length in word piece tokens). 33

3.2 Empirical results on the dev sev set for the *Natural Questions* (NQ) dataset. Best results for *base* and *large* models highlighted. BERT-large results obtained from [8]. * although not visible due to rounding to the closest million, doubling the relative position encoding vocabulary adds about 600k parameters. 33

3.3 Empirical results on HotpotQA and WikiHop (dev set results). *Longformer parameter counts provided by the authors via personal communication. 35

3.4 Empirical results on OpenKP (dev set F1@3 results). 35

3.5 Official leaderboard results for ETC at the time of submission. 35

4.1 Glossary of mathematical symbols used in this chapter. 41

4.2 Experimental results on VCR, incorporating those reported by [11]. The proposed B2T2 model and the B2T2 ensemble outperform published and unpublished/undocumented results found on the VCR leaderboard at `visualcommonsense.com/leaderboard` as of May 22, 2019. 50

4.3 Ablations for B2T2 on VCR dev. The Dual Encoder and the full B2T2 models are the main models discussed in this work. All other models represent ablations from the full B2T2 model. 51

4.4 Examples of the $Q \rightarrow A$ task from the VCR dev set. The correct answer for every example is marked in bold. The answers picked by the text-only model, by the dual encoder and by B2T2 are indicated in parenthesis. 53

5.1 Example of how synthetic question-answer pairs are generated. The model's predicted answer (A') matches the original answer the question was generated from, so the example is kept. 57

5.2 Our results on SQuAD2. For our fine-tuning only setting, we compare a BERT baseline (BERT single model - Google AI Language on the SQuAD2 leaderboard) to similar models pretrained on our synthetic SQuAD2-style corpus and on a corpus containing both SQuAD2- and NQ-style data. For the full pretraining setting, we report our best single model and ensemble results. 61

5.3 Our results on NQ, compared to the previous best system and to the performance of a human annotator and of an ensemble of human annotators. $\text{BERT}_{\text{joint}}$ is the model described in [8]. 61

5.4 Comparison of question-answer pairs generated by NQ and SQuAD2 models for the same passage of text. 61

6.1 Referential link count frequency distribution in a random sample of 1000 instances. 74

6.2 SpanBERT model performance for Task 1: recovering QED annotations when the correct answer is given. 81

6.3 SpanBERT model performance for Task 2: recovering answer and QED annotations given a passage that is known to contain the answer. 82

6.4 Rater study results. Corr and Incorr are accuracies of raters in each group on correct and incorrect instances respectively, with incorrect instances further broken into Pred(icate) and Ref(erence) model errors. F1 is on the task of identifying incorrect instances. 84

6.5 Generalized linear mixed model fixed effect coefficients, showing mean and standard deviation of 10k MCMC samples. The Intercept corresponds to the Incorrect+None setting. 85

7.1 Question counts and why-question counts by answer type and presence of a negation in [7]. 96

8.1 Automatic evaluation of our method on "seen" test data (dev split of English WebNLG 3.0). Grayed rows have BLEU scores more than 1% below the baseline. The "Improved" column displays the ratio of generations with a higher triple F1 compared to the greedy baseline. 112

8.2 Automatic evaluation of our method on "mixed: seen and unseen" test data (test split of English WebNLG 3.0). Grayed rows have BLEU scores more than 1% below the baseline. The "Improved" column displays the ratio of generations with a higher triple F1 compared to the greedy baseline. 113

8.3 Exact match parsing performance of our T5 semantic parser (SP) on Text2RDF English WebNLG. 113

8.4 Human evaluation results. The row labeled "reference" has the evaluation of the human-generated reference text (provided as part of the test set). Triples is the total number of triples rated, not the number of triples generated by the parser for a particular system. See text for details. 115

8.5 Text generated by the two main approaches described in this work (greedy+sampling and greedy finetuned) compared to the a baseline greedy generation, for two different sets of input triples. Inserted and deleted triples are determined by human annotators as described in Section 8.6.2. 118

9.1 Number of question-answer pairs per language and data split for the datasets considered in this work. 128

9.2 Synthetic question-answering data generation methods for training multilingual reading comprehension systems on TyDiQA-GoldP. We report averages over 3 runs of fine-tuning mT5-XL on gold or synthetic data. Standard deviation is given in parentheses. Performance for individual languages (excluding English) is shown in Table 9.3. For comparison we also include recent few-shot prompting results with large language models on TyDiQA-GoldP: [171]§, [122]†, and [172]‡. . . . 129

9.3 QA performance (Average EM over three runs) for individual languages on the TyDiQA-GoldP evaluation set; the backbone of the QA model is mT5-XL fine-tuned on gold (Baseline, Supervised) or synthetically generated data. The final row displays the percent of tokens for each language in the PLM training data. 131

9.4 Comparison of QA performance from training mT5-XL on 5 or 50 examples (Baseline), on synthetic data generated with prompt tuning (PT), or on the full TyDiQA-GoldP training set. Results are averaged across languages. 132

9.5 BLEU scores and downstream QA performance on TyDiQA-GoldP for questions generated by mT5-XL and QAmeleon (Few-shot setting, 5 examples in the target language). 134

9.6 Downstream QA performance on the MLQA test set with an mT5-XL model trained on SQuAD English data (English-Only), SQuAD translated to all MLQA languages (MT), on synthetic data generated by QAmeleon (5-shot) in all MLQA languages, or on a combination of data generated by MT and QAmeleon. Results for [176] and [178] are taken from the respective papers. 135

9.7 Number of synthetic question-answer pairs per language generated via Prompt Engineering (PE) and QAmeleon (PT) with 5 human-labeled examples. 135

9.8 Examples of QA pairs from human-annotated TyDi QA and generated by QAmeleon (PT) on corresponding passages. English translations from Google Translate are added for readability. 136

9.9 QA pairs (random selection) generated by QAmeleon (PT) on Wikipedia passages. English translations from Google Translate are added for readability. 137

Acknowledgements

I would like to acknowledge all of the Google Research team in New York for supporting my academic pursuits over the entire course of this doctorate. The decision to support me in this effort still feels inconceivably generous. I hope I can return this debt to the team in some way in the future. I would also like to acknowledge the support of the Columbia Computer Science department, which had to maintain a special doctoral program essentially just for me. I would like to specially acknowledge the invaluable support of my advisors, Michael Collins and Julia Hirschberg, my Google managers Slav Petrov and Dipanjan Das, key supporters of my partecipation in this program, Michael Riley and Michiel Bacchiani, my collaborators (in no particular order), Shankar Kumar, Jared Lichtarge, Daniel Andor, Livio Baldini Soares, Kuzman Ganchev, Ciprian Chelba, Priyanka Agrawal, Fantine Huot, Joshua Meinez, Ji Ma, Mirella Lapata, Kenton Lee, Emily Pitler, Tom Kwiatkowski, Joshua Ainslie, Matthew Lamm, Jeffrey Ling, David Reitter, some of whom met with me a number of times that feels way too high to count. I would also like to acknowledge my family who put up with me being much more busy than I should have been for several years.

Chapter 1: Introduction

Computational question answering has been a crucial research area within the field of natural language processing for several decades [1]. Its progress has been accelerating rapidly in the last several years, with stunning improvements obtained by fine-tuning large language models, such as ELMo [2], BERT [3], T5 [4], PaLM [5], GPT-3 and GPT-4 [6]. As more data and computational resources have become accessible to researchers, the performance of models on question-answering tasks has surged, often surpassing human performance on benchmarks of increasing complexity, raising the question of whether sufficiently large language models are all we need to reach human performance in question answering in general.

This thesis describes several advances in neural approaches to question answering that give significant improvements over previous approaches. We have detailed these techniques across eight self-contained chapters, with their logical structure presented in Figure 1.1. The core method utilized in this thesis to tackle the question answering task is described in Chapter 2, namely fine-tuning pretrained transformers. The thesis work then develops along three complementary lines: extending the architecture of question answering models to support richer inputs (Chapters 3 and 4), adding explanatory capabilities to question answering models (Chapters 6 and 7) and, finally, improving the quality of question answering systems through different data augmentation techniques (Chapters 5, 8 and 9).

The general formulation for the problem of question answering is to learn a function of the form $(q, c) \rightarrow a$, where q is a question, c is the context based on which the question should be answered, and a is either a valid answer to the question or *null* if the question cannot be answered. The function is to be learned from a set of provided training examples for which the answer a is known, with the goal of maximizing generalization to unseen instances, sampled from the same distribution as the training examples or from a similar distribution. In this thesis, q and a are always textual,

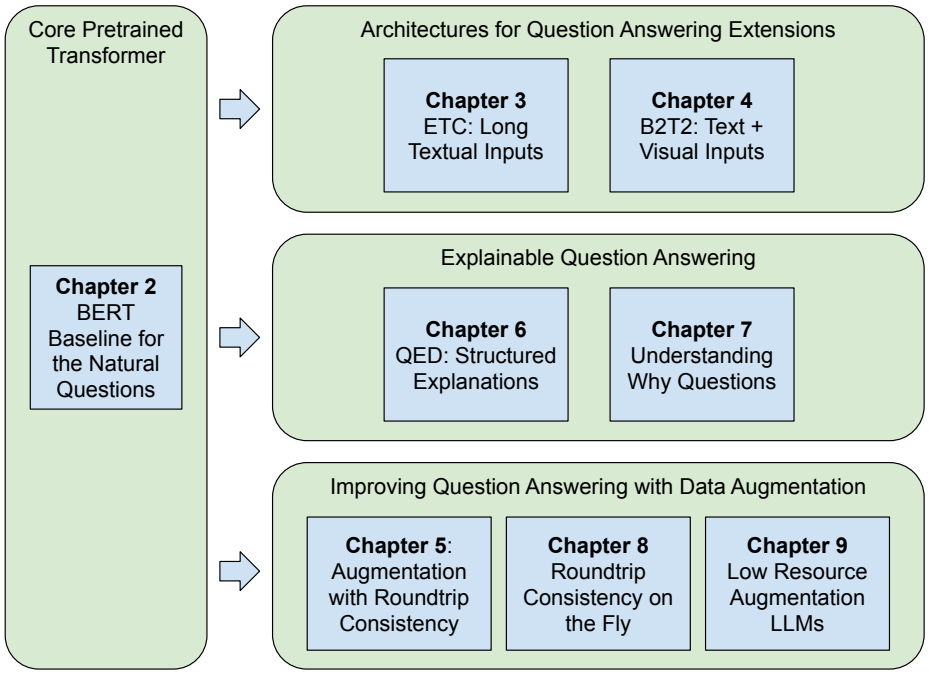


Figure 1.1: Thesis outline.

while c can be either textual or a combination of text and images. An additional categorization to consider is whether the answer a is extractive, i.e. a span in c, or abstractive, i.e. an unconstrained textual answer.

In practice, every task we consider in this thesis contains variations from the idealized question answering framework. For instance, in the Natural Questions dataset [7] the context c is the text of a full Wikipedia page, complete with sections, headings, lists and tables expressed in html. The answer in the Natural Questions is a pair $(a_{\text{long}}, a_{\text{short}})$. a_{long}, the *long answer*, is either a full paragraph from c that fully answers question q, or *null*. a_{short}, the *short answer*, is either a set of entities contained in a_{long}, the words 'yes' or 'no' for yes/no questions, or *null* when no valid short answer could be found. An example of long and short answer from the Natural Questions dataset is show in Figure 1.2.

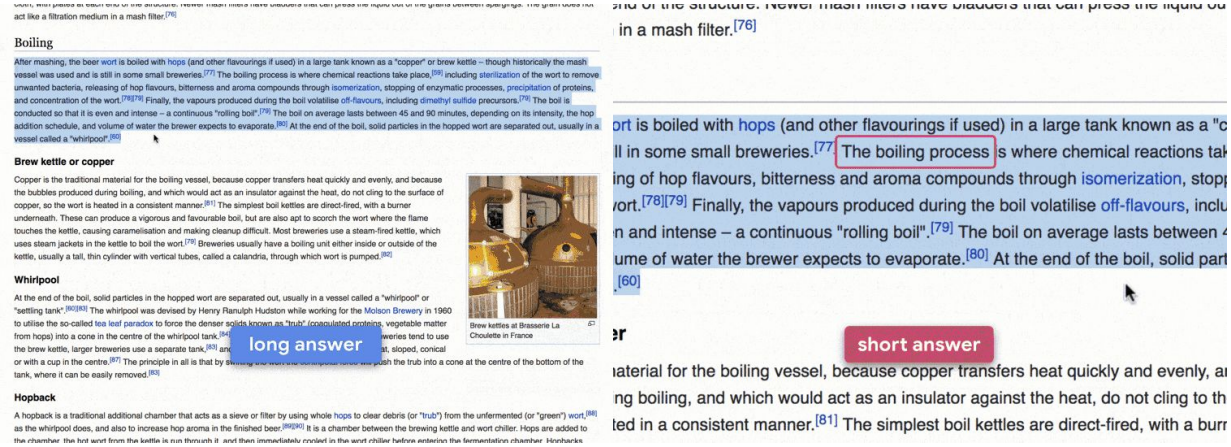


Figure 1.2: Long and short answer to the question "when are hops added to the brewing process", from the Natural Questions dataset. The long answer is a paragraph from the Wikipedia page "Brewing". The short answer is the string "The boiling process".

1.1 Base Question Answering Architecture

In Chapter 2, we discuss a strong baseline we built for the Natural Questions dataset. The baseline is a transformer model called "BERT-joint" [8] which obtained the initial best performance on this dataset by a large margin compared to other available methods. In BERT-joint, we tokenize the question and the context using a *word piece tokenizer*. We then join the tokenized question and context using special tokens [CLS] and [SEP], obtaining the following input sequence:

$$\mathcal{I} = [\text{CLS}], q_1, \ldots, q_n, [\text{SEP}], c_1, \ldots, c_m, [\text{SEP}]$$

where n is the length of the tokenized query and m is the length of the tokenized context. We then utilize the BERT-Large pretrained encoder [3] to obtain context dependent representations of this sequence of tokens, each token being represented by a vector in $\mathbb{R}^d$. A small feed forward net is used to map each context dependent representation to two scores, $f_{\text{start}}(\cdot)$ and $f_{\text{end}}(\cdot)$, used to predict the start and the end of the short and long answer. Each span is assigned the score

$$s(i, j) = f_{\text{start}}(c_i) + f_{\text{end}}(c_j) - f_{\text{start}}([\text{CLS}]) - f_{\text{end}}([\text{CLS}])$$

and the highest scoring span is found as $(i^*, j^*) = \text{argmax}_{(i,j):i<j}\, s(i, j)$. We finally return

$$a_{\text{short}} = (i^*, j^*) \text{ or } \textit{null} \text{ if } s(i, j) < t_{\text{short}} \forall i < j$$

$$a_{\text{long}} = \text{paragraph containing } (i^*, j^*) \text{ or } \textit{null} \text{ if } s(i, j) < t_{\text{long}} \forall i < j$$

where t_{short} and t_{long} are thresholds selected to optimize the F1 score on the evaluation set. Note that in BERT-joint we intentionally ignore yes/no answers as they did not make a large difference in the final evaluation metric.

This model, finetuned from BERT-large on NQ, is found to obtain substantially improved results compared to other contemporary approaches. The advantage of BERT-joint compared to competitor systems on NQ is due to the following factors: (1) our model is pretrained with the masked-LM loss on a large amount of unlabeled text, while competitor systems were trained on NQ only, (2) our model uses a transformer based architecture, where competitors used LSTMs, and (3) our model uses an end-to-end answer extraction system instead of a pipeline comprised of a retriever and reading comprehension model.

1.2 Architectures for Question Answering Extensions

ETC: Question Answering With Long Textual Inputs. One major drawback of the question answering model described in Chapter 2 is that it could only ingest sequences of at most 512 tokens, while the Wikipedia articles in the Natural Questions are often much longer, requiring an average of 30 windows of 512 tokens to be extracted from each article to be processed independently. This limitation affects both question answering accuracy and computational efficiency.

We address this limitation in Chapter 3, presenting ETC, a model we developed in collaboration with other Google researchers and that again obtained state-of-the-art performance on the Natural Questions dataset. The core idea of ETC is to recognize that self-attention is often the main memory bottleneck for long input sequences to transformer models. This is because, while all components of the transformers scale linearly with the length of the input, the self attention scales quadratically,

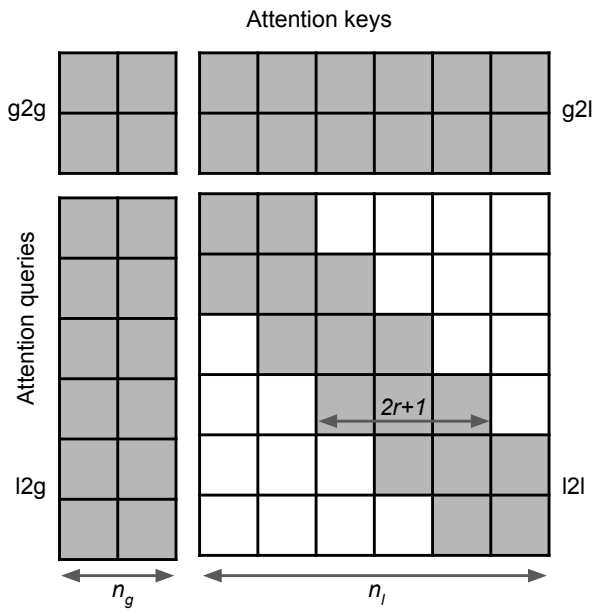


Figure 1.3: Sparse self-attention from the ETC transformer, presented in Chapter 3, to handle long document contexts in the Natural Questions and other question answering datasets.

since every token can attend to every other token. This could be addressed by limiting self-attention to tokens in a local radius r, however this would heavily degrade the model, intuitively because documents often encode important information in the relationship between far away tokens, and with a small local radius we would never be able to model long distance interactions. The ETC solution to this problem is depicted in Figure 1.3. The input of the model is modified with respect to the tokenized question and context in equation 1.1 by prepending n_g "global" tokens:

$$\mathcal{I} = g_1, \ldots, g_{n_g}, [\text{CLS}], q_1, \ldots, q_n, [\text{SEP}], c_1, \ldots, c_m, [\text{SEP}] \tag{1.1}$$

The original tokens from equation 1.1 are "local" and are only able to attend other local tokens if they are at a distance of at most r. Global tokens instead attend to and are attended to by all other tokens. All local tokens are therefore able to attend to all other local tokens through a global token as an intermediate step. This results in self-attention layers whose memory footprint grows linearly with the length of the input sequence like the rest of the transformer model. The additional model parameters required to represent the new global tokens as well as specialized cross-attention

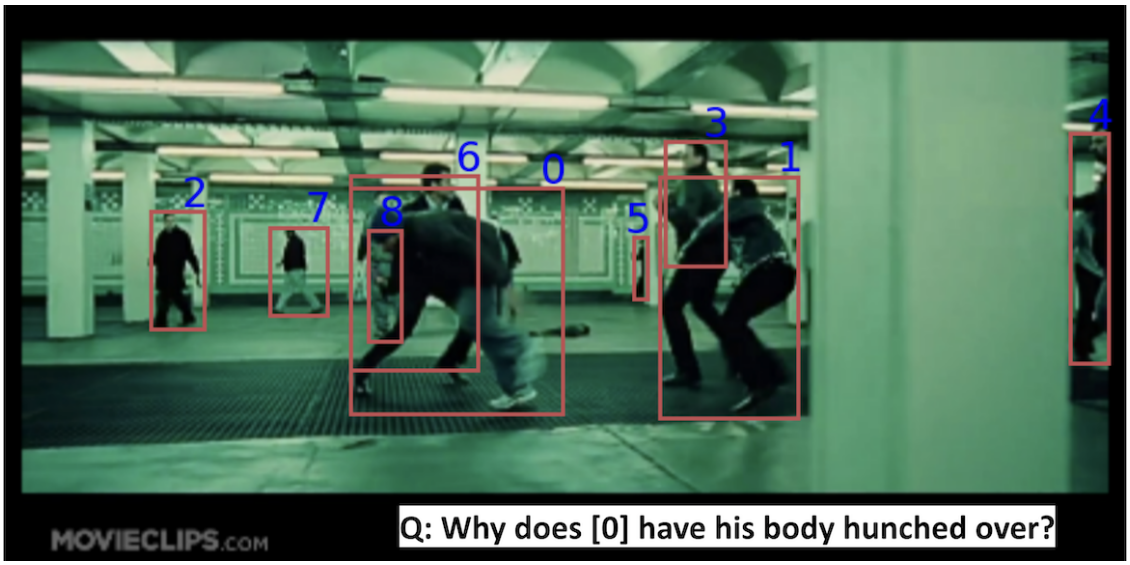


Figure 1.4: An input image and question from the VCR dataset. The image is augmented with bounding boxes produced by Mask-RCNN, an automatic object detection system. The VCR task consists of picking the correct answer and answer rationale out of four options for each.

weights are learned by "lifting" an existing pretrained transformer that can be trained without the sparse self-attention mechanism. Lifting consists of initializing all parameters of the ETC model from the quadratic attention pretrained transformer and then continuing pretraining the ETC model on unlabeled text with a masked-language-model objective.

The ETC model also introduces several additional improvements over the BERT baseline of Chapter 2, allowing it to obtain state of the art performance in the long answer task by a large margin on the Natural Questions. The additional improvements are to lift the more powerful RoBERTa model [9] instead of BERT, and to use contrastive predictive coding (CPC) [10] during pretraining to improve the global token representations. For CPC, the local tokens are partitioned in n_g contiguous sets and each partition p_i is assigned to a global token g_i. The hidden representation of global token g_i is used to predict the hidden representation of the local tokens in partition p_i. The loss for this prediction is added as a pretraining loss during the lifting process. A final refinement in ETC for question answering in the Natural Questions dataset is to further constrain the attention masks so that each global token can only attend to a section of the input Wikipedia page, so that global tokens can represent the high level structure of the input document.

B2T2: Question Answering with Text+Visual Inputs. In an effort to further advance performance of question answering models, we decided to look into the possibility of modeling images from Wikipedia pages together with the textual content. From initial analysis it seemed clear however that most questions in NQ were likely to benefit very little from image content. So we decided to instead experiment with extensions to the basic transformer model on a dataset with questions that would heavily rely upon the content of images. For this purpose we chose the VCR dataset [11]. An example from the VCR dataset is show in Figure 1.4.

The main idea in Chapter 4 is to introduce in the input to BERT special visual tokens, whose embeddings are obtained as a linear transformation of the output of a vision model, namely ResNet-152, applied to the pixels in each of the bounding boxes that appear in the input image. The tokens input to BERT in this case are

$$\mathcal{I} = [\text{CLS}], v_0, q_1, \ldots, q_n, [\text{SEP}], a_1, \ldots, a_m, [\text{SEP}], v_1, \ldots, v_k \tag{1.2}$$

where v_0 is a visual token corresponding to the entire input image, $q_1, \ldots, q_n$ is the tokenized question, $a_1, \ldots, a_m$ is the tokenized candidate answer, and $v_1, \ldots, v_k$ are the visual tokens for all the objects detected in the image. Additionally in this dataset, as shown in figure 1.4, some of the objects are referenced in the question and the answer by index. In this case we simply put the visual token corresponding to that mention in place of the reference. A score corresponding to whether an answer choice is correct or not is finally computed with a small feed-forward network applied to the contextual representation of the [CLS] token. The model is then fine-tuned on the training split of the VCR training set and tuned on the validation split.

Our experiments reported in Chapter 4 showed state-of-the-art results on VCR based on this multimodal modeling architecture. We found that additional improvements could also be obtained by pretraining the entire model on a synthetic task generated from the Conceptual Captions dataset [12], a collection of millions of images and corresponding captions mined from the web. The synthetic task we designed consisted of replacing some of the tokens in the caption with [MASK]

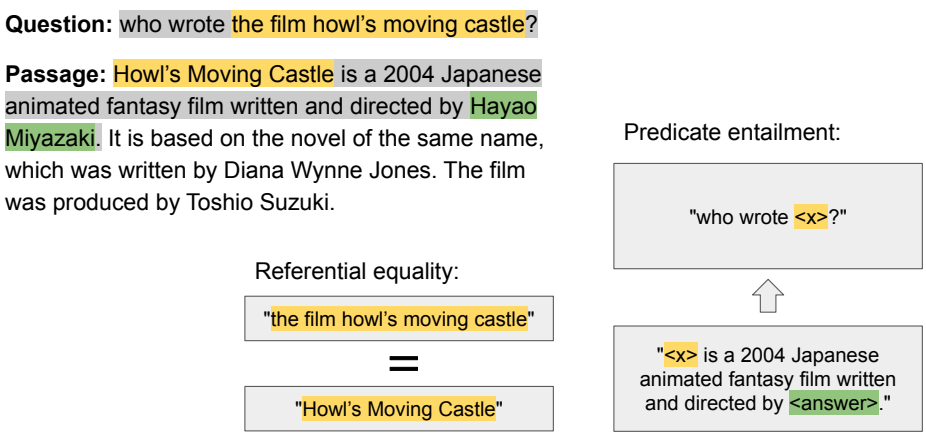


Figure 1.5: Example of a structured explanation for question answering following the QED framework described in Chapter 6.

tokens and occasionally replacing the image itself with a different image. Our multimodal model was then trained to predict back the masked tokens from the caption as well as predict whether the associated image had been replaced or not. We found that pretraining improved both question answering performance and training stability.

1.3 Explainable Question Answering

QED: Structured Explanations. All question answering systems seen so far are still very far from perfect. Frequent mistakes create a very real risk of misleading the user into believing false information. As a way of mitigating this risk and of increasing user trust in question answering systems, we decided to investigate the possibility of providing a certificate of accuracy together with predicted answers, so that users could easily verify the correctness of the answer based on the given textual context, or alternatively quickly realize that the model had output an invalid answer. The notion of a certificate of accuracy can be seen as a special kind of structured explanation that the question answering system provides together with the answer.

In Chapter 6 of this thesis, we present a concrete framework for constructing structured explanations of question answering. We named this framework QED. QED explanations consist of a list of referential equalities between question and context spans and a predicate entailment between the question and a sentence where corresponding entities and the extractive answer are replaced

Table 1.1 Several predictions made by [13] on admissible answers to different kinds of why-questions can be disproven by real world examples in the Natural Questions dataset. The prediction about the absence of purpose answers to negative why-questions however holds surprisingly well.

Prediction from [13]	**Counterexample in Natural Questions**
"Passives only admit reason"	Why were Luke and Leia separated at birth? To keep them hidden from Darth Vader.
"Locative-existentials only admit reason"	Why is the Angel of the North there? To serve as a focus for our evolving hopes and fears.
"Unaccusatives only admit reason"	Why does cooling water run through the condenser? To condense the steam coming out of the cylinders or turbines.
"Negative questions only admit reason."	$\emptyset$

by placeholders. An example of QED explanation can be seen in Figure 1.5: the noun phrases "the film howl's moving castle" and "Howl's Moving Castle" are found to be a referential equality, meaning that they refer to the same thing in the world, and the predicate "who wrote <x>?" is found to be entailed by the sentence "<x> is a 2004 Japanese animated fantasy film written and directed by <answer>". The referential equality and predicate entailment together make up the QED explanation, which certifies to the user the accuracy of the answer, which in this case is "Hayao Miyazaki".

As detailed in Chapter 6, we introduce the QED dataset, a new resource derived from Natural Questions where expert linguists have annotated existing question answering examples with QED explanations. We report experimental results showing that QED explanations can be accurately recovered by models and that training a model on question answering and on the QED task jointly can improve question answering performance, even if the size of the QED dataset is small. We finally report results of a user study showing that QED explanations improve the ability of untrained raters to spot errors made by question answering systems.

A Semantic Analysis of Why Questions. As powerful as QED structured explanations are, they only cover a specific type of explanation for question answering. In a much more general sense, it can be productive to think of explanations for question answering as answers to questions of the type: "why does a answer question q in the context of c?". In other words explainable question answering can be re-framed as an instance of question answering itself, with the caveat that the

questions in this context are why-questions, which are often considered particularly challenging and open-ended. The Natural Questions dataset fortunately contains 1,220 why-questions that we were able to study in search of meaningful patterns.

In Chapter 7 of this thesis, we employ tools from semantics to obtain a deeper understanding of why-questions, based on the real world examples available in the Natural Questions dataset, hoping that this understanding will allow us to design more general explanations for question answering in the future. From semantics literature on why-questions we discover that there are two main types of answers to why-questions, *reason* answers (generally introduced by the preposition "because") and *purpose* answers (generally introduced by the preposition "to"). Indeed, in the Natural Questions dataset we find that roughly 250 questions have a purpose answer, while the remaining roughly 1,000 have a reason answer.

The question of how why-questions work from a technical semantic point of view does not appear to be widely studied, except for a few key papers. We did find however very interesting predictions in a work by Chapman and Kučerová [13]. The predictions appeared to be in most cases contradicted by examples in the Natural Questions, as shown in Table 1.1, but, even more interestingly, a specific prediction held with perfect accuracy on NQ, namely that negative why-questions are never answered with purpose answers. In Chapter 7 we propose a deeper analysis of this phenomenon and propose a new explanation for it based on semantic modalities.

1.4 Improving Question Answering with Data Augmentation

Even though question answering datasets are available up to considerable size, e.g. 300,000 examples in Natural Questions, or 150,000 examples in SQuaD 2.0 [14], it seems self evident that the capacity of language models could benefit from much larger datasets. In this thesis we show that performance improvements are achievable by significantly increasing the size of training data with synthetically generated examples. Increased accuracy can be obtained by augmenting English datasets (Chapter 5) as well as by augmenting datasets for low-resource languages (Chapter 9), provided that at least a handful of representative samples (5 or more) are provided for each new

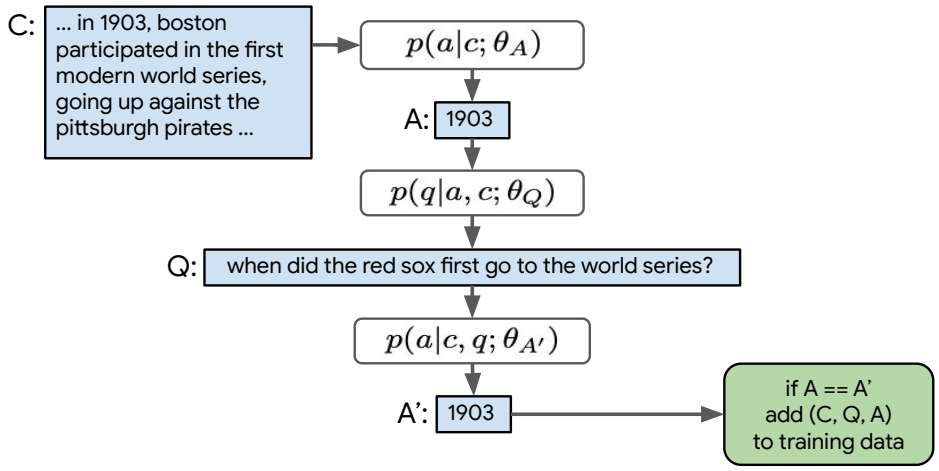


Figure 1.6: Data augmentation strategy for question answering explored in Chapter 5.

language.

Augmenting Question Answering Datasets with Roundtrip Consistency. The data augmentation strategy explored in Chapter 5 is sketched in Figure 1.6. The approach combines three models: (1) a question independent answer model $p(a|c;\theta_A)$ that predicts plausible extractive answers a given a textual context c, (2) a question generation model $p(q|a,c;\theta_Q)$ that generates a question given a context c and an extractive answer a, and (3) a question answering model $p(a|c,q;\theta_{A'})$ used to verify that the synthetic question-answer pair is valid. The final verification step is referred to as the "roundtrip consistency" check. In this work we showed that significant improvements can be obtained through this simple data augmentation approach both on Natural Questions and SQuAD 2.0, generating 4M synthetic questions for Natural Questions and 50M questions for SQuAD 2.0.

Both Natural Questions and SQuAD 2.0 contain unanswerable questions, i.e. QA examples where a model is expected to output a *null* answer since the context does not contain a valid answer to the question. For this reason, in Chapter 5 we employ a simple method to generate synthetic unanswerable questions that are however difficult to identify by the models as such: we generate a question based on a paragraph of Wikipedia page, but then associate that question with a different paragraph from the same page.

We additionally observed interesting differences in the style of questions generated by models

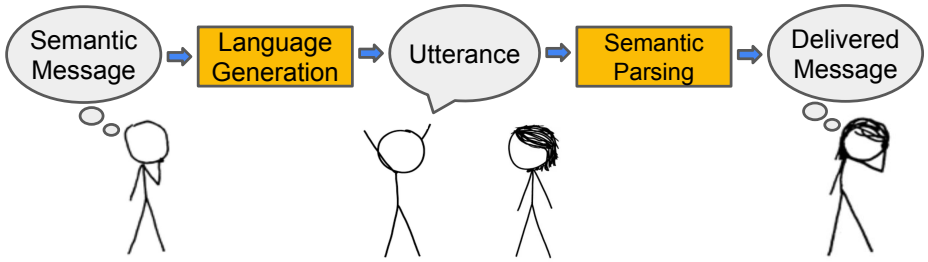


Figure 1.7: Verifiable language modeling intuition developed in Chapter 8. The approach seeks to re-frame language modeling as a semantic auto-encoder, where the intermediate representation is natural language.

trained on Natural Questions vs. SQuaD 2.0. For the same passage, the model trained on NQ generated the question-answer pair "what was the population of chicago in 1857?" → "over 90,000", whereas the model trained on SQuaD generated the question-answer pair "what was the weight of the brigg's hotel?" → "22,000 tons". The former generation reflects the fact that Natural Questions is generally focused on facts that could be of interest to the user of a search engine. SQuAD style questions instead appear to be constructed more artificially, aiming to extract specific entities from paragraphs without a notion of how likely generated questions are to be asked in real world scenarios.

Roundtrip Consistency as a Semantic Auto-encoder. We continued the roundtrip consistency line of work in Chapter 8. In Chapter 5 we generated question-answer pairs, then removed the ones that are deemed invalid by a question answering model, and finally trained on the filtered synthetic data. in Chapter 8 we aimed at performing data generation and consistency checking in a single chain of models that we could back-propagate through end-to-end. We decided to pick a dataset outside of question answering, the WebNLG dataset [15], to validate this approach. However, the method should apply identically to the question answering tasks considered in the rest of this thesis.

The main intuition considered in Chapter 8 is depicted in Figure 1.7. We attempt to conceptually re-frame the problem of language modeling from the usual task of predicting the next word given the preceding context, to the problem of encoding a semantic message from the mind of the speaker into a natural language utterance, with the objective of making the semantic message easy to reconstruct

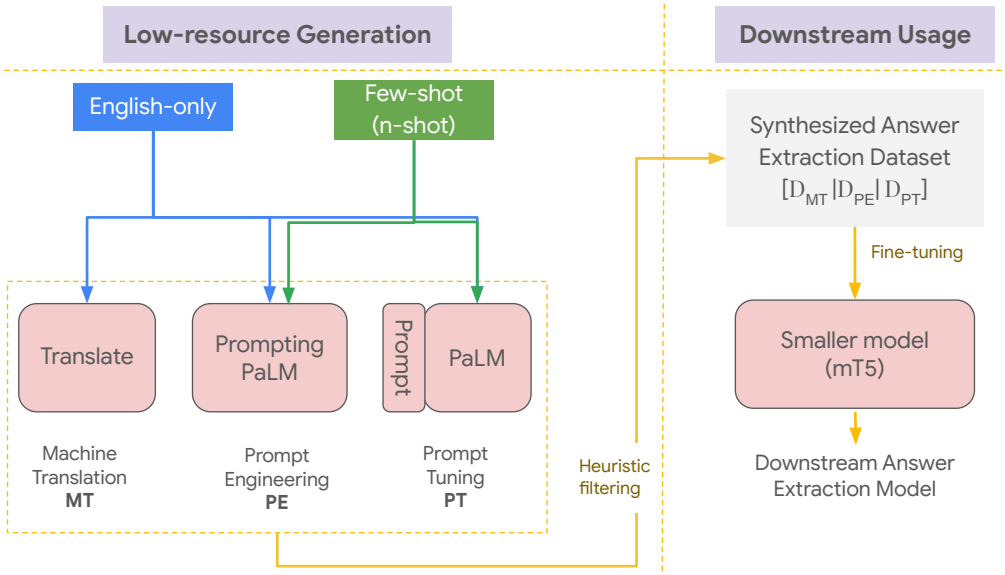


Figure 1.8: Data augmentation methods explored in Chapter 9 for low resource language scenarios.

by the listener through semantic parsing. The WebNLG dataset [15] is very appropriate to investigate these ideas, since it is designed to benchmark for the two tasks that make up our reframing, namely *language generation* and *semantic parsing*. The examples in WebNLG specifically contain lists of subject-relation-object triples from DbPedia [16] and corresponding human written verbalizations.

We train two T5 models to perform the language generation and semantic parsing tasks on the WebNLG dataset. We then stack the models and use a technique known as "straight-through gradient", to allow us to pass gradients through the non-differentiable decoding step on the interface between the two models. We then propose a decoding method that we name "greedy-finetuned", where we generate a verbalization and then perform a few steps of gradient descent on the language generation model to improve the probability of correct reconstruction with a frozen semantic parser. In experiments, we find that this method leads to increased semantic parsing accuracy at a small cost in BLEU score. We additionally corroborate these results with human evaluations.

Very Low Resource Question Answering with Large Language Models. We conclude our thesis with a final exploratory effort in the space of data augmentation for question answering. In the final chapter, Chapter 9, we look at the possibility of using large language models to bootstrap question answering systems in very low resource languages. We compare different bootstrapping

methods, which are schematically represented in Figure 1.8. We experiment with bootstrapping by translating English question answering datasets to new languages and we compare this to the more modern approach of generating synthetic question-answer pairs with large language models (PaLM), either through prompt engineering or prompt tuning. In all cases, we train downstream question answering models on the synthetic data and evaluate them on the TyDiQA dataset.

The experiments in Chapter 9 lead us to several interesting observations. We found that large language models can learn to generate useful training data for question answering in new languages from as little as 5 examples. We found that prompt tuning is significantly better than prompt engineering and machine translation when it comes to the quality of downstream question answering models. We additionally found that generating synthetic data and then training a question answering model is always consistently better than fine-tuning a model directly on a small number of labeled examples, confirming our previous findings that synthetic data generation is always a useful tool for improving accuracy of question answering systems. We finally found that large language models on the scale of PaLM (half a trillion parameters) are substantially better than smaller models like T5 (11B parameters) at generating synthetic data for training downstream models. From inspecting generated question answer pairs, we found that larger models seem to produce a higher variety in generated questions and we hypothesize that the diversity of generation is responsible for downstream QA improvements.

Chapter 2: Answering Natural Questions with Transformers

In this chapter we discuss the Natural Questions (NQ) [7] dataset and a baseline transformer model that obtained state-of-the-art performance on this dataset. We give an overview of how NQ was collected and how it compares to other popular question answering datasets. The model we present is BERT-joint [8]. BERT-joint is a BERT-based model which provides a substantial improvement over the original LSTM baseline system. The new ideas in BERT-joint were to use pretrained transformers instead of LSTMs as models and to predict short answer spans directly on the entire document rather than selecting the best paragraph first.

2.1 Natural Questions: A Benchmark for Question Answering Research

Questions in NQ consist of real anonymized, aggregated queries issued to the Google search engine. During data collection, annotators are presented with a question along with a Wikipedia page from the top 5 search results on google.com. The annotators select a long answer (typically a paragraph) and a short answer (one or more entities) if present on the page, or they mark null if no long/short answer is present. The dataset consists of 307,373 training examples with single annotations, 7,830 examples with 5-way annotations for development data, and a further 7,842 examples with 5-way annotated examples sequestered as test data. Evaluation on test data can only be performed by uploading a question answering system to the NQ website. Models are expected to provide two scores for every document in the NQ test set, a long-answer score and a short-answer score. The evaluation then computes the F1 score corresponding to the best possible choice of threshold. A significant challenge in NQ comes from the fact that only half of the examples have a long answer annotated and so the model has to learn not only to find answering paragraphs but also to abstain when no answering paragraph exists.

One of the most important objectives in the development of NQ was to provide researchers with a high quality and long-lived dataset for long term research efforts. In the time since its release, NQ seems to have delivered on this goal. While leaderboards for popular question answering tasks like SQuAD 2.0 and CoQA already show models scoring several percentage points better than humans on test set metrics, NQ still has non-negligible headroom. For NQ, human performance is estimated to be 87.2% and 75.7% on the long-answer and short-answer task respectively, while the best model results are 79.8% and 64.1%. It therefore appears that NQ is still a challenging benchmark for the NLP community.

The qualities that we think make NQ more challenging than other question answering datasets are the following: (1) the questions in NQ were formulated by people out of genuine curiosity or out of a real need for an answer to complete another task, (2) the questions were formulated by people before they had seen the document that might contain the answer, (3) the documents in which the answer is to be found are much longer than the textual evidence used in most other question answering challenges.

2.2 A BERT Baseline for the Natural Questions Dataset

The original NQ paper was published with an LSTM baseline result that was state-of-the-art in 2018. The LSTM baseline was a pipeline system that first made a long answer prediction with a lightweight model and then analyzed the best long answer with a more expensive LSTM model to find the best short answer. We were able to obtain a substantial improvement over this baseline by combining the following insights:

1. we make use of a pretrained BERT model and finetune on the NQ training split,
2. we jointly predict short and long answers in a single model rather than using a pipeline approach,
3. rather than relying on HTML markup, we split each document into multiple training instances by using fixed-size and partially overlapping windows of tokens,

4. we use the "[CLS]" token at training time to predict null instances and rank spans at inference time by the difference between the span score and the "[CLS]" score.

We picked the name "BERT-joint" for our system to emphasize the fact that we are modeling short and long answers in a single model rather than in a pipeline of two models. This approach was very successful. It reduced the gap between the F1 scores reported in the original NQ paper and the human upper bound by 30% and 50% relative for the long and short answer tasks respectively. In the rest of this section we give further details on how the NQ dataset was pre-processed, we explain the modeling choices we made to adapt BERT to the NQ task, and we finally present our results.

2.3 Data Preprocessing

Following [3] we tokenize every example in NQ using a vocabulary comprised of 30,522 word segments, then generate multiple instances per example by concatenating a "[CLS]" token, the tokenized question, a "[SEP]" token, tokens from the content of the document, and a final "[SEP]" token, limiting the total size of each instance to 512 tokens. For each document we generate all possible instances, by listing the document content starting at multiples of 128 tokens, effectively sliding a 512 token size window over the entire length of the document with a stride of 128 tokens. On average we generate 30 instances per NQ example. Each instance is then processed independently by BERT.

For each training instance we compute start and end token indices to represent the target answer span. If all annotated short spans are contained in the instance, we set the start and end target indices to point to the smallest span containing all the annotated short answer spans. If there are no annotated short spans but there is an annotated long answer span completely contained in the instance, we set the start and end target indices to point to the entire long answer span. If no short or long span can be found in the current instance, we set the target start and end indices to point to the "[CLS]" token. In the following, we refer to the instances in the last category as "null instances".

Given the large size of documents in NQ and the fact that 51% of the documents are annotated as not having an answer to the query at all, we find that about 98% of generated instances are null;

therefore for training we down-sample null instances by 50 times in order to obtain a training set that has roughly as many null instances as non-null instances. This leads to a training set that has approximately 500,000 instances of 512 tokens each.

We introduce special markup tokens in the document to give the model a notion of which part of the document it is reading. The special tokens we introduced are of the form "[Paragraph=N]", "[Table=N]", and "[List=N]" at the beginning of the N-th paragraph, list and table respectively in the document. This decision was based on the observation that the first few paragraphs and tables in the document are much more likely than the rest of the document to contain the annotated answer and so the model could benefit from knowing whether it is processing one of these passages. Special tokens are atomic, meaning that they are not split further into word segments.

We finally compute for each instance a target answer type as one of five values: "short" for instances that contain all annotated short spans, "yes" and "no" for yes/no annotations where the instance contains the long answer span, "long" when the instance contains the long answer span but there is no short or yes/no answer, and "no-answer" otherwise. Null instances correspond to the set of instances with the "no-answer" target answer type.

2.4 Model

Formally, we define a training set instance as a four-tuple

$$(c, s, e, t)$$

where c is a context of 512 word segment ids (including question, document tokens and markup), $s, e \in \{0, 1, \ldots, 511\}$ are inclusive indices pointing to the start and end of the target answer span, and $t \in \{0, 1, 2, 3, 4\}$ is the annotated answer type, corresponding to the labels "short", "long", "yes", "no", and "no-answer".

We define the loss of our model for a training instance to be

$$\begin{aligned} L &= -\log p(s, e, t|c) \\ &= -\log p_{\text{start}}(s|c) - \log p_{\text{end}}(e|c) \\ &\quad - \log p_{\text{type}}(t|c), \end{aligned}$$

where each probability p is obtained as a softmax over scores computed by the BERT model as follows:

$$\begin{aligned} p_{\text{start}}(s|c) &= \frac{\exp(f_{\text{start}}(s, c; \theta))}{\sum_{s'} \exp(f_{\text{start}}(s', c; \theta))}, \\ p_{\text{end}}(e|c) &= \frac{\exp(f_{\text{end}}(e, c; \theta))}{\sum_{e'} \exp(f_{\text{end}}(e', c; \theta))}, \\ p_{\text{type}}(t|c) &= \frac{\exp(f_{\text{type}}(t, c; \theta))}{\sum_{t'} \exp(f_{\text{type}}(t', c; \theta))}, \end{aligned}$$

where θ represents the BERT model parameters and f_{start}, f_{end}, f_{type} represent three different outputs derived from the last layer of BERT.

At inference time we score all the contexts from each document and then rank all document spans (s, e) by the score

$$\begin{aligned} g(c, s, e) = f_{\text{start}}(s, c; \theta) \\ + f_{\text{end}}(e, c; \theta) \\ - f_{\text{start}}(s = [\text{CLS}], c; \theta) \\ - f_{\text{end}}(e = [\text{CLS}], c; \theta) \end{aligned}$$

and return the highest scoring span in the document as the predicted short answer span. Note that $g(c, s, e)$ is exactly the log-odds between the likelihood of an answer span (defined by the product $p_{\text{start}} \cdot p_{\text{end}}$) and the "[CLS]" span.

We select the predicted long answer span as the DOM tree top level node containing the

Table 2.1 Our results on NQ compared to the baselines in the original dataset paper and to the performance of a single human annotator and of an ensemble of human annotators. The systems used in previous NQ baselines are DocumentQA [17], DecAtt [18], and Document Reader [19].

	Long Answer Dev			Long Answer Test			Short Answer Dev			Short Answer Test		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
DocumentQA	47.5	44.7	46.1	48.9	43.3	45.7	38.6	33.2	35.7	40.6	31.0	35.1
DecAtt + DocReader	52.7	57.0	54.8	54.3	55.7	55.0	34.3	28.9	31.4	31.9	31.1	31.5
BERT-joint [8]	**61.3**	**68.4**	**64.7**	**64.1**	**68.3**	**66.2**	**59.5**	**47.3**	**52.7**	**63.8**	**44.0**	**52.1**
Single Human	80.4	67.6	73.4	-	-	-	63.4	52.6	57.5	-	-	-
Super-annotator	90.0	84.6	87.2	-	-	-	79.1	72.6	75.7	-	-	-

predicted short answer span, and assign to both long and short prediction the same score equal to the maximum value of $g(c, s, e)$ for the document.

We opted to limit the complexity of this baseline model by always outputting a single short answer as prediction and we rely on the official NQ evaluation script to set thresholds to decide which of our predictions should be changed to having only a long answer or no answer at all. We expect that improvements can be obtained by combining start/end and answer type outputs to sometimes predict yes/no answers instead of always predicting a span as the short answer. We also expect additional improvements to be achievable by extending the model to be able to emit short answers comprised of multiple disjoint spans.

2.5 Experiments

We initialized our model from a BERT model already finetuned on SQuAD 1.1 [20]. We then further finetuned the model on the training instances precomputed as described in Section 2. We trained the model by minimizing loss L from Section 3 with the Adam optimizer [21] with a batch size of 8. As is common practice for BERT models, we only tuned the number of epochs and the initial learning rate for finetuning and found that training for 1 epoch with an initial learning rate of $3 \cdot 10^{-5}$ was the best setting. Evaluation completed in about 5 hours on the NQ dev and test set with a single Tesla P100 GPU.

The results obtained by our model are shown in Table 2.1. Our BERT model for NQ performed

dramatically better than the models presented in the original NQ paper. Our model closes the gap between the F1 score achieved by the original baseline systems and the super-annotator upper bound by 30% for the long answer NQ task and by 50% for the short answer NQ task. However NQ appeared to be still far from being solved, with more than 20 F1 points of headroom for both the long and short answer tasks.

Chapter 3: ETC: Encoding Long and Structured Inputs in Transformers

In this chapter we describe ETC, a BERT-style extractive question answering model with a sparse attention mechanism that allows it to scale up the length of the input and represent structure in the input document. The work described in this chapter is the result of a collaboration with other researcher at Google. My contribution is the adaptation of the ETC model to the Natural Questions task. Natural Questions was also the first task on which we saw substantial improvements from using this technique. Natural Questions seemed particularly well suited to a methods that allow long input lengths because many of the input documents in NQ have many thousands of tokens, well above the 512 token limit that BERT had at the time.

3.1 Overview

Models based on Transformers [22], such as BERT [3], or other variants [23, 24, 25] have yielded state-of-the-art results in many NLP tasks such as language modeling [26, 27, 28, 29], question answering [24, 30], and summarization [31]. In this chapter we present the *Extended Transformer Construction* (ETC) architecture[1], targeting two limitations of the original models: (1) scaling input length, (2) encoding structured inputs.

The computational and memory complexity of attention in the original Transformer scales quadratically with the input length, typically limiting input length to around 512 tokens. While 512 might be enough for some tasks (e.g., co-reference resolution seems to benefit from even smaller input lengths [32]), this is problematic in others. Consider question answering (QA) tasks that require reasoning across multiple documents (e.g., the *HotpotQA* dataset [33]) all of which must simultaneously fit in the model input. Other examples are summarization or QA on long documents.

[1]Source code and pretrained checkpoints for ETC can be found at `http://goo.gle/research-etc-model`

Many approaches have been proposed to address this, like hierarchical processing [31], sparse attention [26], and segment-level recurrence [34].

A second limitation is that few models focus on *structured inputs*, by which we refer to any underlying graph or hierarchical structure among the input tokens. Although ETC can encode more general graph structure, in this work we focus on representing hierarchical structure in NLP tasks, not usually modeled by Transformer models. For example, text is organized into sentences and paragraphs, and while these have a sequential order, different input documents might not hold any order between them (e.g., the HotpotQA dataset). Additionally, web text contains markup and is laid out using a DOM tree, giving additional structure. We show ETC can represent these and other types of structure, like linking different entity mentions.

To address these challenges, we present a novel attention mechanism called *global-local attention*, which divides the input into two sequences (which we call the *global* input and the *long* input). This mechanism introduces local sparsity to reduce the quadratic scaling of the attention mechanism. When this is coupled with relative position encodings [35], it allows for handling structured inputs in a natural way. Additionally, unlike previous Transformer extensions, ETC can be initialized from existing pretrained standard BERT models (which together with a GPU/TPU-friendly implementation, allows for efficient model training)[2]. Our results show that initializing from RoBERTa [9] significantly improves performance. Finally, we show that by adding a pretraining *Contrastive Predictive Coding* (CPC) task [10], performance improves even further for tasks where structure is important, as CPC plays the role of a masked language model (MLM) task, but at a sentence level of granularity.

We report experiments on four datasets: *Natural Questions* (NQ) [7], HotpotQA [33], WikiHop [36], and OpenKP (part of MS MARCO) [37], which have long and/or structured inputs. We set a new state of the art in all of them.

Moreover, although in this chapter we strictly focus on ETC, in a related model called BigBird [38], we experimented with an alternative set of ideas to handle long inputs and its extensions

[2]An exception to this is *Longformer* [30], a new model developed concurrently to ETC, which also allows initialization from BERT/RoBERTa.

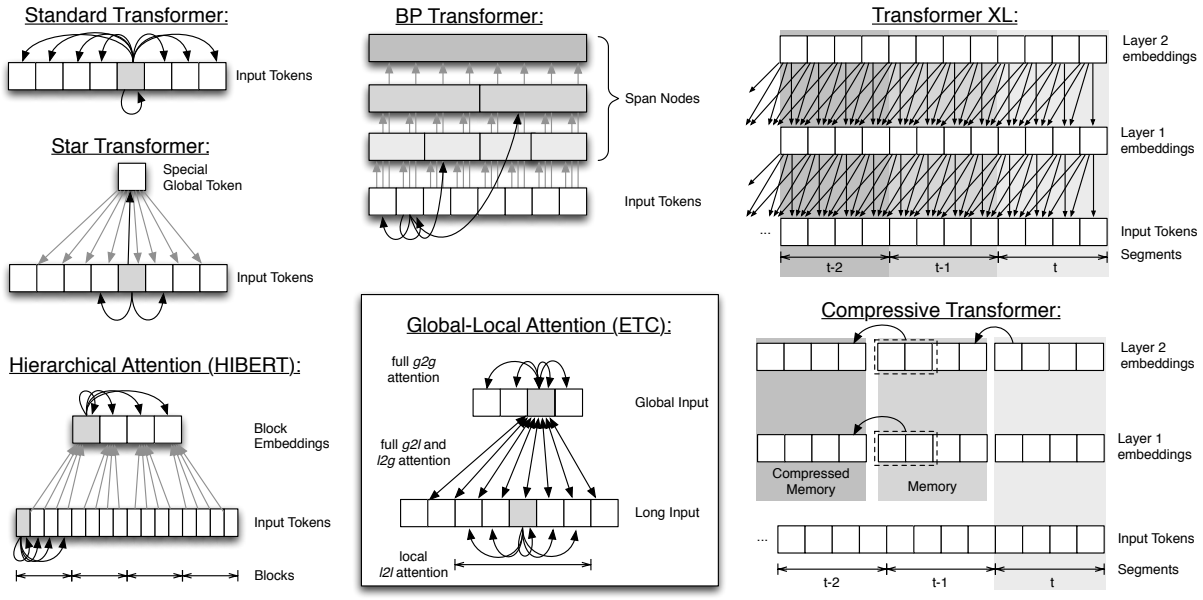


Figure 3.1: An illustration of mechanisms to scale attention to long inputs, including our proposed model, ETC.

to a decoder for text generation. The focus of BigBird is on the idea of adding random sparse attention patterns to global-local attention, and on showing under which conditions models like BigBird/ETC are universal approximators of sequence functions and are Turing complete. While the key ideas and techniques required to achieve the state-of-the-art results mentioned above for QA tasks are the focus of this chapter, the reader is referred to the BigBird work for a joint evaluation of ETC (referred to as BigBird-ETC in that work) and the idea of random sparse attention patterns.

3.2 Background

Many variants of the original Transformer model [22] have been proposed for scaling up training [9], the internal representation [24], or both [25], outperforming BERT [3] in tasks such as GLUE [39], SQuAD [20] or RACE [40]. However, these models typically limit inputs to $n = 512$ tokens due to the $O(n^2)$ cost of attention. We classify prior approaches to scale up attention into four categories: sparse attention, recurrence, hierarchical mechanisms, and compressed attention.

Sparse Attention involves limiting each token to attend to a subset of the other tokens. For example, the *Sparse Transformer* [26] used predefined attention patterns for both text and image

generation. They showed that attending only to previous pixels in the same row or column was enough to generate high quality images, while keeping attention cost at $O(n\sqrt{n})$. In the *Adaptive Attention Span Transformer* [27] each attention head is associated with a decaying learnable masking function, which limits the number of tokens it can attend to. They show that lower layers learn to use short attention spans, and only in higher layers are attention spans longer. Sparse attention has also been used to increase the interpretability of attention heads by allowing attention to assign exactly zero weight to certain input tokens [41]. The *Reformer* [29] model finds the nearest neighbors of the attention query (those input tokens that would result in the highest attention weights) using locality sensing hashing [42] and only uses those for attention. This reduces attention cost to $O(n\ \log(n))$. The *Routing Transformer* [43] learns dynamic sparse attention patterns using online k-means, reducing complexity to $O(n^{1.5})$. Finally, the most related approach to the work presented in this paper is *Longformer* [30], developed concurrently to ETC, and which features a very similar global-local attention mechanism as ETC's but does not directly encode graph or hierarchical structure (more detailed comparison in Section 3.3).

Recurrence incorporates elements of recurrent neural networks into Transformer models to lengthen their attention span. *Transformer-XL* [34] takes this approach, dividing the input sequence into segments and then processing these segments one at a time. At each layer, the model attends to the layer immediately below for both the current and previous input segments. The effect is that layer k is influenced by the current segment and the $k-1$ previous segments, as shown in the top-right of Figure 3.1.

In **Hierarchical Mechanisms** the input sequence is split into *blocks* that are ingested independently to produce summary embeddings that represent the whole block. Then, separate layers ingest the concatenation of these embeddings. For example, HIBERT [31] uses this idea at the sentence level for extractive summarization (illustrated in the bottom-left of Figure 3.1). Hierarchical attention in Transformers has also been applied to other NLP tasks such as neural machine translation [44]. Moreover, notice that this idea of processing the input hierarchically is not specific to Transformer models, and it has been applied to recurrent neural network models both at the level

of sentences [45, 46] and blocks [47].

Compressed Attention takes the idea of hierarchical attention one step further by selectively compressing certain parts of the input. The *BP-Transformer* [48] model builds a binary partitioning tree over the input, and only lets the model attend to the leaves (the raw tokens) for nearby tokens, and higher nodes in the tree (summaries of groups of tokens) as tokens grow more distant (see Figure 3.1, middle top). Other ideas include *memory compressed attention* [49] where groups of k tokens are compressed via a convolution filter before they are attended to, and the *Star Transformer* [50], where each token can attend only to its immediate left/right neighbors and to a separate special auxiliary token that represents a summary of the whole input (see Figure 3.1, left). The *Compressive Transformer* [28] integrates this idea into Transformer-XL by compressing tokens in the input that are far away. The model benefits from detailed attention to nearby tokens, while using summarized information for more distant tokens (see Figure 3.1, lower right).

3.3 Extended Transformer Construction

Our model follows the original Transformer architecture [22], with key modifications to tackle long and structured inputs: relative position encoding, global-local attention, and a CPC pretraining task, explained below. In this paper, we consider only the encoder side of the Transformer, and leave the decoder for future work.

3.3.1 Relative Position Encoding

Inspired by the work of [35], ETC replaces absolute position encodings with relative position encodings, which provide information about the relative position of tokens in the input sequence with respect to one another. Given the input sequence $x = (x_1, ..., x_n)$, we can see it as a labeled fully connected and directed graph, where l_{ij} is the label of the edge that connects x_i to x_j. Given a maximum clipping distance k, Shaw *et al.* define $2k + 1$ *relative position labels*: $l_{-k}, ..., l_k$. The label of the edge between two input tokens depends only on their relative position $j - i$. For input pairs with $j - i \geq k$, label l_k is given, and with $j - i \leq -k$, l_{-k} is given. Each label then becomes a

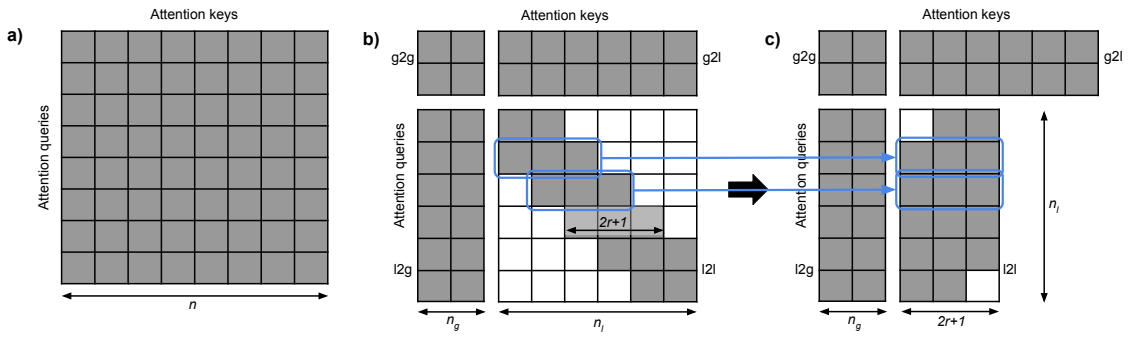


Figure 3.2: Sparsity diagram showing which attention queries (rows) can attend to which attention keys (columns) a) for standard Transformer attention with input size n; b) for global-local attention with input sizes n_g, n_l, and radius r; c) how the l2l attention piece is reshaped into a much smaller attention matrix, limited by local radius.

learnable vector a_l^K, which modifies the attention mechanism (equations in the next section)[3].

Relative position encodings are independent of input length, so it is easy to adapt a model to greater input lengths than seen during pretraining. As other recent work [51], ETC's attention mechanism uses relative position labels not just for relative positions in a sequence but also to express arbitrary pairwise token relations useful for structured inputs, as explained below.

3.3.2 Global-Local Attention

Global-local attention is a generalization of several of the models presented above. ETC receives two separate input sequences: the *global input* $x^g = (x_1^g, ..., x_{n_g}^g)$ and the *long input* $x^l = (x_1^l, ..., x_{n_l}^l)$. Typically, the long input contains the input a standard Transformer would receive, while the global input contains a much smaller number of auxiliary tokens ($n_g \ll n_l$). Attention is then split into four separate pieces: *global-to-global* (g2g), *global-to-long* (g2l), *long-to-global* (l2g), and *long-to-long* (l2l). Attention in the l2l piece (the most computationally expensive piece) is restricted to a fixed radius $r \ll n_l$. To compensate for this limited attention span, the tokens in the global input have unrestricted attention, and thus long input tokens can transfer information to each other through global input tokens. Accordingly, g2g, g2l, and l2g pieces of attention are unrestricted.

This concept is illustrated in Figure 3.2, where each cell (row i, column j) is shaded grey if

[3]In the work of Shaw et al., a second a_l^V vector was used, but their ablations showed it may not affect performance.

token x_i can attend to token x_j. As we can see, in a regular Transformer, attention is unrestricted (full $n \times n$ attention). ETC, illustrated in Figure 3.2b, however, restricts the l2l piece to a local radius, significantly reducing computational and memory complexity for very long inputs. Conceptually, the l2l attention piece is reshaped into a $n_l \times (2r + 1)$ matrix as illustrated in Figure 3.2c.[4]

If $r = 1$ and $n_g = 1$, we recover exactly the Star Transformer (Section 3.2). Similarly, placing all the tokens in the global input and setting $n_l = 0$ yields standard Transformer attention. Attention in ETC is $O(n_g(n_g + n_l) + n_l(n_g + 2r + 1))$. If we assume $n_g = O(2r + 1)$, we see attention is linear in the size of the long input: $O(n_g^2 + n_g n_l)$.

To provide flexible attention and help with structured inputs, per-instance Boolean attention matrices M^{g2g}, M^{g2l}, M^{l2g}, and M^{l2l} exist, with zeroes for those pairs of tokens that should not attend to one another. Each g2g attention head works as follows. Given the global input $x^g = (x_1^g, ..., x_{n_g}^g)$, which is a sequence of token representations $x_i^g \in \mathbb{R}^{d_x}$, the output of attention is $z^g = (z_1^g, ..., z_{n_g}^g)$, where each $z_i^g \in \mathbb{R}^{d_z}$ is calculated as follows:

$$z_i^g = \sum_{j=1}^{n_g} \alpha_{ij}^{g2g} x_j^g W^V$$

$$\alpha_{ij}^{g2g} = \frac{\exp(e_{ij}^{g2g})}{\sum_{\ell=1}^{n} \exp(e_{i\ell}^{g2g})}$$

$$e_{ij}^{g2g} = \frac{x_i^g W^Q (x_j^g W^K + a_{ij}^K)^T}{\sqrt{d_z}} - (1 - M_{ij}^{g2g})C$$

where: M^{g2g} is a binary attention mask, W^Q, W^K, and W^V are learnable weight matrices, and a_{ij}^K are learnable vectors representing the relative position labels, and C is a large constant ($C = 10000$ in our experiments to follow the same convention as BERT). Attention for the other 3 pieces is analogous. We experiment with having separate W^K and W^V across all four attention pieces, or sharing them. And for W^Q, we experiment with having one for g2g and g2l, and a separate one for l2g and l2l; or sharing them also. To recover BERT as a special case when r is large enough to

[4]In practice, for GPU/TPU efficiency, a different reshaping occurs that yields identical outputs (see the appendices).

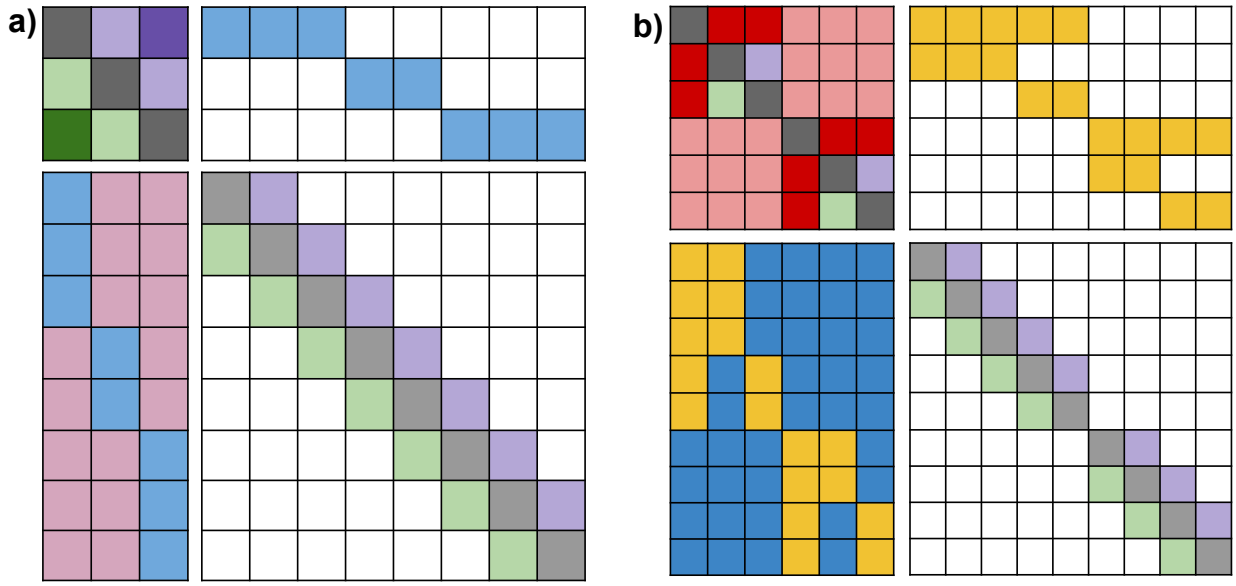


Figure 3.3: Example attention patterns for handling (a) long inputs and (b) structured inputs. White background means attention is masked via M, and the other colors indicate different relative position labels.

remove sparsity, attention is actually only split into 2 pieces internally instead of 4, as g2g+g2l can be computed jointly (top half of Figure 3.2c), and l2g+l2l can also be computed jointly (bottom half of Figure 3.2c). A single softmax is used to jointly calculate α_{ij}^{g2g} and α_{ij}^{g2l}, and another for α_{ij}^{l2g} and α_{ij}^{l2l}.

Thus, the output of global-local attention is a sequence of length n_g and one of length n_l. These sequences go through a layer normalization and feed forward layer in the same way as in the standard transformer.

3.3.3 Long Inputs and Global-Local Attention

Let us illustrate how ETC can be used to encode long inputs. A general way to handle long inputs in ETC is to place the entire sequence of input tokens (e.g., word pieces) in the long input and then, assuming some sort of division into segments (e.g., sentences), place one auxiliary token in the global input per segment in the long input. We then use one relative position label to link the global segment tokens with the word piece tokens that belong to them, and a different label for those that do not. Moreover, as we will show in the experiments below, we have seen that using the

M^{g2l} attention masks to perform hard masking in one direction (g2l) can bring performance gains in some datasets. This last asymmetric hard-masking is illustrated in Figure 3.3a, where we used different colors to indicate different relative position labels. In this way, although tokens in the long input can only attend to the local neighborhood defined by the radius k, they can indirectly attend to all other tokens in the input sequence via the global tokens.

3.3.4 Structured Inputs

A standard Transformer resembles a graph neural network [52] over a fully connected graph g; see [48]. Thanks to the combination of global-local attention and relative position labels, ETC exploits this relation to encode *structured inputs*. Given the input $x = (x_1, ..., x_n)$, we use the term *structure* to refer to the relations that exist between the tokens in x. When x is a plain ordered sequence, the only relation is the *sequential order* of tokens, which is the only structure captured by BERT (encoded by absolute position encodings, used to modify attention). We define *structured inputs* as those that have additional relations between the tokens beyond sequential order. In principle, we could think of inputs with arbitrary graph structure (such as chemical molecule graphs), but here we focus on structure in NLP tasks.

ETC is particularly well suited to capture hierarchical structure thanks to three mechanisms. First, as originally conceived, the vocabulary of relative position labels is used to represent token relative positions. However, seeing a Transformer as a graph neural network over a graph g (with one vertex per token in x, and edges representing their relations), we can expand this vocabulary to label some edges with labels for relations such as *is-a*, *part-of*, or others. Second, the division between long and global input induces a natural structure where the global input contains summary tokens of sets of tokens in x (a 2-level hierarchy). However, we can also have tokens summarizing sets of summary tokens (constructing a 3-level hierarchy, or beyond). Third, if some pairs of tokens should not have an edge between them, this can be captured with the M^{g2g}, M^{g2l}, M^{l2g}, M^{l2l} masks. An illustration of all these concepts is shown in Figure 3.3b, which uses masking and relative position labels to represent a context-sentence-token hierarchy that includes within-context order

of sentences but no order between contexts. Another example would be social community graphs structure, where we could partition the graph into components, use M^{l2l} to constrain attention to within components, and add per-component global tokens, linked to allow information to propagate from one component to another in a hierarchical way.

3.3.5 Pretraining Tasks

We use two pretraining tasks: (1) a masked language model (MLM) task with *whole word masking* (if one word piece token is masked, then all other tokens of the same word are masked); and (2) instead of using BERT's next sentence prediction (NSP), we adapt Contrastive Predictive Coding (CPC) [10] for ETC.

The goal of CPC is to predict subsequent inputs in latent space, i.e., to predict internal hidden representations of blocks of tokens. We adapted this idea in ETC by using global input sentence summary tokens. Given an input sequence containing n sentences, we mask all the tokens corresponding to a subset of sentences (but leave the sentence summary tokens in the global input). We then train the model to minimize the difference between the hidden representation of the global sentence summary tokens for the masked sentences with respect to that of a global summary token that can see the unmasked sentence and nothing else. We use a Noise Contrastive Estimation (NCE) loss as in the work of Oord et al. (2018) (details in the appendices).

Having described ETC, we can now compare it with Longformer [30], which uses a similar attention mechanism, except that Longformer has a single input sequence with some tokens marked as *global* (the only ones that use full attention). The key differences are that (1) ETC's combination of global-local attention with relative position encodings and flexible masking enables it to encode structured inputs in a similar way as graph neural networks do; (2) global tokens in Longformer are never pretrained with anything like our CPC loss, and thus their use must be learned during fine-tuning.

3.3.6 Lifting Weights from Existing Models

ETC and BERT share enough similarities that BERT parameters are useful to perform a warm start. The parameters are compatible because the global-local attention mechanism includes BERT as a special case if the input is small enough or the local radius is large enough to eliminate sparsity. Moreover, when lifting weights from BERT into an ETC model with separate W^Q, W^K, and W^V projection matrices, BERT's parameters are just copied over to the different matrices of ETC.

Although pretraining is still required to adapt the weights to use global tokens and relative position encodings, we show that initializing from RoBERTa results in significant performance improvements compared to pretraining from scratch. Specifically, we initialized from the RoBERTa checkpoints reported in the work of Rothe et al. [53].

3.4 Empirical Evaluation

This section focuses on evaluating our two main contributions: (1) long inputs, and (2) structure in text inputs, as well as initialization from existing BERT models. We chose four datasets (Table 3.1) with long inputs or interesting input structure.

NQ [7]: in Google's *Natural Questions* (NQ) dataset the input consists of a question and a full Wikipedia article. The task is to identify both a *short answer* (a few words from the article) and a *long answer* (e.g., a whole paragraph), if they exist within the article (and otherwise, return null answers). Performance is measured based on the F1 score of the model predictions with respect to the human generated answers.

HotpotQA [33] is a question answering dataset where the goal is to combine evidence from multiple contexts. We use the *distractor* setting, where 10 paragraphs are provided: two of them contain useful information to answer the question, and the rest are distractors. The task is both to answer the question, and also to identify the supporting facts that are relevant to answer the questions (at a sentence granularity).

Table 3.1 Dataset stats (length in word piece tokens).

	Instances		*Instance length*		
Dataset	*Training*	*Dev*	*Median*	*95%*	*Max*
NQ	307373	7830	4004	17137	156551
HotpotQA	90447	7405	1227	1810	3560
WikiHop	43738	5129	1541	3994	20337
OpenKP	133724	6610	761	4546	89183

Table 3.2 Empirical results on the dev sev set for the *Natural Questions* (NQ) dataset. Best results for *base* and *large* models highlighted. BERT-large results obtained from [8]. * although not visible due to rounding to the closest million, doubling the relative position encoding vocabulary adds about 600k parameters.

Model	*Input length*	*Configuration*	*#Params*	*Long answer F1*	*Short answer F1*
BERT-base	512		110M	0.634	0.475
BERT-large	512		340M	0.647	0.527
RikiNet	512	lifting from $\text{RoBERTa}_{\text{large}}$	-	0.753	**0.593**
ETC	512	shared, no CPC, no hard g2l	109M	0.645	0.478
ETC	4096	shared, no CPC, no hard g2l	109M	0.692	0.497
ETC	4096	fixed blocks, shared, no CPC, no hard g2l	109M	0.697	0.508
ETC	4096	shared, no hard g2l	109M	0.717	0.524
ETC	4096	shared	109M	0.721	0.514
ETC	4096	-	166M	0.725	0.522
ETC	8192		166M	0.740	0.542
ETC	4096	2x local radius	166M	0.737	0.530
ETC	4096	2x relative vocab	166M*	0.733	0.532
ETC	4096	2x pretraining	166M	**0.746**	**0.558**
ETC-large	4096		539M	0.761	0.565
ETC-large	4096	lifting from RoBERTa	558M	**0.782**	0.585

WikiHop [36] is similar in structure to HotpotQA. The contexts correspond to portions of Wikipedia articles, and the goal is to answer about properties of an entity that cannot be found in the entity's article. Each instance contains a query, a collection of candidate answers, and a collection of contexts from which to obtain information to select among the candidate answers.

OpenKP [37] is a keyphrase extraction dataset. Each document contains up to 3 short keyphrases to be identified. We selected this dataset as the input is not flat text sequences, but websites, including the hierarchical and spatial relations between the different DOM elements on the website, as well as other visual properties.

3.4.1 Training Configuration

We use two basic configurations: *base* and *large*. Base uses 12 layers, 768 hidden size, 12 attention heads, local attention radius $r = 84$, and relative position maximum distance $k = 12$. Large uses 24 layers, 1024 hidden size, 16 heads, $r = 169$, and $k = 24$. We used 128, 230 and 460 global tokens for models with 512, 4096 and 8192 long input size respectively in NQ[5], 256 global tokens in HotpotQA, 430 in WikiHop, and 512 in OpenKP.

Pretraining: We place all word piece tokens in the long input and add one auxiliary token per sentence to the global input. We defaulted to BERT's 30k English uncased word piece vocabulary. Models were pretrained using the original BERT datasets, except that documents with fewer than 7 sentences were filtered out. Unless stated otherwise, base models were pretrained with the same total number of tokens as the original BERT, and for large models, twice as many. We used the *LAMB* optimizer [54] with learning rate set to $\sqrt{8} \times 10^{-3}$.

Fine-tuning: we put all input tokens in the long input (CLS, question, and context tokens for QA datasets), and use relative position labels to encode structure (see Section 3.3.4). Global input has a CLS token, tokens mirroring the question tokens in long, and one summary token per paragraph/sentence (or VDOM block in OpenKP). OpenKP had no CLS nor question tokens. For WikiHop, we also add one global token per candidate answer, and used a different relative position label to link these tokens to their string-matched mentions in the text (more details in the appendices).

3.4.2 Results on the Dev Set

NQ: We used NQ to study the different parts of ETC via ablations. Results are shown in Table 3.2. The first three rows show baseline models: BERT-base, BERT-large, and RikiNet [55] (one of the best models in the NQ leaderboard). BERT's performance is comparable to ETC using input length of 512. The smaller local radius of ETC (84) puts ETC at a disadvantage with respect

[5]With gradient check-pointing, ETC can scale beyond this, but we limit our experiments to 8192 tokens for this paper.

Table 3.3 Empirical results on HotpotQA and WikiHop (dev set results). *Longformer parameter counts provided by the authors via personal communication.

Model	*Input length*	*Configuration*	*#Params*	*HotpotQA* Ans. F1 / Sup. F1	*WikiHop* Acc.
Longformer	4096		149M*	0.743 / 0.844	75.0
Longformer-large	4096		435M*	0.788 / 0.860[6]	77.6
ETC	4096	flat structure, no CPC, no hard g2l	166M	0.722 / 0.857	70.0
ETC	4096	flat structure	166M	0.748 / **0.870**	70.7
ETC	4096	no CPC	166M	0.747 / 0.866	73.0
ETC	4096	no hard g2l	166M	0.743 / 0.864	**75.9**
ETC	4096	shared	109M	0.733 / 0.866	73.7
ETC	4096	-	166M	**0.751** / 0.869	73.2
ETC-large	4096		539M	0.798 / 0.890	77.0
ETC-large	4096	lifting from RoBERTa	558M	**0.813** / **0.894**	**79.8**

Table 3.4 Empirical results on OpenKP (dev set F1@3 results).

Model	*Input length*	*Configuration*	*#Params*	*OpenKP F1@3*
RoBERTa-JointKPE	512		-	0.398
ETC	512	fixed blocks, no CPC, no hard g2l, no visual features	166M	0.399
ETC	4096	fixed blocks, no CPC, no hard g2l, no visual features	166M	0.400
ETC	4096	no CPC, no hard g2l, no visual features	166M	0.400
ETC	4096	no hard g2l, no visual features	166M	0.400
ETC	4096	no visual features	166M	0.402
ETC	4096	-	166M	0.409
ETC	4096	shared	109M	0.409
ETC	4096	max loss	166M	**0.416**
ETC-large	4096	max loss	539M	0.419
ETC-large	4096	max loss, lifting from RoBERTa	558M	**0.423**

to BERT, but other ETC improvements, such as dynamic whole word masking seem to compensate.

The rest of Table 3.2 shows performance under different ablations. Our default configuration (marked with a "-" in the configuration column) is ETC-base with long input length of 4096 tokens, using CPC, hard g2l masking, and separate W^Q, W^K, and W^V matrices for long/global inputs. We tested the following ablations: *shared* (sharing all model parameters for attention across both the

Table 3.5 Official leaderboard results for ETC at the time of submission.

Leaderboard	Result	Position
NQ long answer	0.7778	1st
NQ short answer	0.5786	18th
HotpotQA Sup. F1	0.8909	1st
HotpotQA Overall	0.7362	3rd
WikiHop	0.8225	1st
OpenKP	0.4205	1st

global and long inputs), *no CPC* (removing the CPC pretraining task), *no hard g2l* (not having a hard g2l mask), and *fixed blocks* (which configures the global input to just have one global token per 97 long input tokens, to keep the same proportion as without fixed-blocks, ignoring sentence boundaries, and not having any other tokens in the global input for pretraining or fine-tuning). Sharing W^Q, W^K, and W^V and removing CPC significantly hurt the performance of ETC in NQ[7]. Using fixed blocks, surprisingly, seems to slightly help without CPC.

Increasing long input from 512 to 4096 significantly helped performance, and going to 8192 increased performance further to 0.740 / 0.542, highlighting the importance of longer inputs. Increasing the local radius, relative position vocabulary, or the amount of pretraining all helped performance (especially the latter, reaching 0.746 / 0.558). Moving to a large model also helped, especially when lifting from RoBERTa (both large models used the RoBERTa vocabulary). Lifting from RoBERTa achieved our best scores: 0.782 / 0.585, beating the best dev scores in the literature for long answer (compare with 0.754 / 0.593 for RikiNet). For short answer, we still lag behind RikiNet.

HotpotQA, WikiHop: Table 3.3 shows our results in HotpotQA and WikiHop. We show two Longformer models as baselines (currently the state-of-the-art model in WikiHop), as well as ablations to study the effect of structure in the results. In particular, we consider a *flat structure* ablation where: (1) we do not break long input attention by context boundaries, (2) we limit relative position labels between global and long tokens to representing only sentence-level relationships (this removes any special attention in WikiHop between candidate answers and their mentions).

Our results show that both our base and large models outperform their corresponding Longformer models in both HotpotQA and WikiHop. Besides parameter counts, the main factors that can explain this difference in performance are the different pretraining strategies and the different handling of structure in ETC and Longformer. Removing the CPC pretraining task, and not using a hard g2l mask significantly hurt the performance of the model in HotpotQA, going from a performance of 0.751 / 0.869 for the baseline model to 0.722 / 0.857 using none of those features. Using a flat structure (but

[7]Separate projection matrices were also found to be helpful in other models, like Longformer [30].

keeping CPC and hard g2l) did not seem to hurt in HotpotQA. WikiHop shows a slightly different picture, as it seems that hard g2l masking and especially flat structure hurt performance in this dataset. Our best model is the base configuration without hard g2l masking, which achieves an accuracy of 75.9. Interestingly, sharing W^Q, W^K, and W^V seems to help performance in WikiHop. This is our smallest dataset, and maybe the added capacity of the model without sharing parameters leads it to overfit.

OpenKP: Table 3.4 shows our results on the OpenKP dataset, using RoBERTa-JointKPE [56] as the baseline, which is currently #1 in the leaderboard. This is an interesting structured dataset, and thus, we performed additional ablations to investigate the effect of removing such structural information. Our results show that even the most constrained ETC model already achieves very good performance (0.399), and scaling to 4096 length seems to give a slight boost. Using hard g2l also helps, and adding the visual features brings the largest benefit. Finally, we see that using a large model, and especially lifting weights from RoBERTa improve results significantly. As with WikiHop, sharing W^Q, W^K, and W^V does not hurt performance. Our default model uses the first occurrence of a key-phrase, but we saw that using the maximum logit of all occurrences (*max loss*) improved results.

3.4.3 Official Leaderboard Results

Finally, Table 3.5 shows official results on the leaderboards of each dataset. The model submitted to the leaderboards was the model with best dev set results (shown at the bottom of the respective results tables, lifting weights from RoBERTa). We set a new state of the art in WikiHop and OpenKP, NQ long answer, and HotpotQA Support F1. Remarkably, our submissions were all single model, outperforming the leaderboard ensemble models.

[7] Better results were reported for Longformer-large using a 2 stage approach, reaching 81.0 / 85.8 [30], but our table shows single-model results only, for comparison.

3.5 Conclusions

This chapter described the Extended Transformer Construction (ETC), an architecture designed to (1) scale up the input length (linearly with input), and (2) encode structured inputs. ETC allows lifting weights from existing BERT models, improving results significantly. The key ideas are a new global-local attention mechanism, coupled with relative position encodings and a CPC pretraining task.

We showed that significant gains can be obtained thanks to increased input sequence length. The ability to represent dataset structure in ETC further improves the model quality. We hypothesize that CPC helps the model train the usage of the higher-level global input summary tokens, as CPC plays a role akin to MLM, but at the global input level. Notice that although our datasets contain a limited amount of structure (compared to graph datasets), our experiments show that ETC was able to exploit this existing structure.

As future work, we would like to investigate complementary attention mechanisms like those of Reformer [29] or Routing Transformer [43], push scalability with ideas like those from RevNet [57], and study the performance of ETC in datasets with even richer structure.

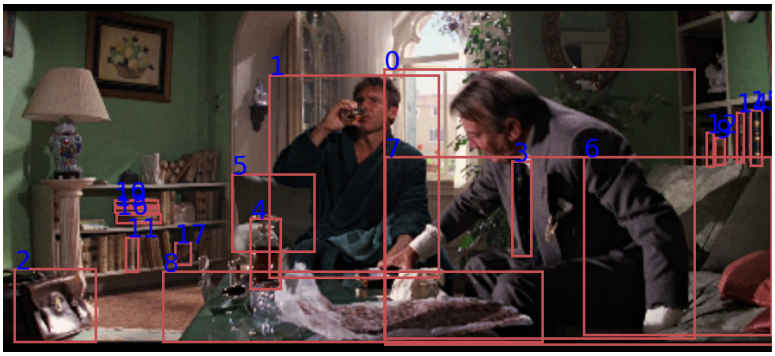
Chapter 4: Fusion of Detected Objects in Text for VQA

This chapter describes B2T2 [58] ("Bounding Boxes in Text Transformer"), a technique to ground mentions in text to objects in images while solving a question answering task. B2T2 introduces special visual tokens in the textual context of the task. These tokens are treated as regular word tokens by the transformer model solving the QA task, but their representation is generated by an upstream vision model. B2T2 is pretrained on a large scale dataset of images with captions and then finetuned on the Visual Commonsense Reasoning (VCR) benchmark[1], achieving a new state-of-the-art performance with a 25% relative reduction in error rate compared to published baselines and obtaining the best performance to date on the public leaderboard (as of May 22, 2019). In this chapter we additionally present an ablation study showing that multimodal pretraining not only improves the results of finetuning, but also its stability. We also find that the early integration of the visual features into the text analysis is key to the effectiveness of the new architecture.

4.1 Overview

The challenge we consider in this chapter is Visual Question Answering (VQA), a natural instance of grounding text in a multimodal context. More specifically we look at an instance of VQA where a model has to solve a multiple choice question related to a given image, where objects from the image are explicitly pointed to in the text. Several research questions are investigated in this work. How is visual and verbal information best encoded in a neural architecture? How should we represent text entities bound to objects seen in images? Are text and image best integrated late, allowing for independent analysis (*late fusion*), or should the processing of one be conditioned on the analysis of the other (*early fusion*)? Can neural representations from single-modality text and vision models be made compatible and combined in an effective multi-modal model?

[1] `https://visualcommonsense.com`



Q: What was [1] doing before he sat in his living room?
A_1: He was reading [10].
A_2: He was taking a shower. ✓
A_3: [0] was sleeping until the noise [1] was making woke him up.
A_4: He was sleeping in his bedroom.
R_1: His clothes are disheveled and his face is glistening like he's sweaty.
R_2: [0] does not look wet yet, but [0] looks like his hair is wet, and bathrobes are what you wear before or after a shower.
R_3: He is still wearing his bathrobe. ✓
R_4: His hair appears wet and there is clothing hanging in front of him on a line as if to dry.

Figure 4.1: An example from the VCR dataset. The tasks consists in picking an answer A_{1-4}, and then picking a rationale R_{1-4}. The data contains explicit pointers in the text to bounding boxes in the image.

In this work we gather evidence to answer these questions by designing the Bounding Boxes in Text Transformer, B2T2 for short, a neural architecture for multimodal encoding of natural language and images, and we evaluate B2T2 on the Visual Commonsense Reasoning benchmark (VCR, [11]). Figure 4.1 shows an illustrative example from the VCR benchmark. VCR is well suited to test rich multimodal representations because it requires the analysis of images depicting people engaged in complex activities; it presents questions, answers and rationales created by human annotators rather than automatic generation; it has a clean multiple-choice interface for evaluation; and yet it is still challenging thanks to a careful selection of answer choices through adversarial matching. VCR has much longer questions and answers compared to other popular Visual Question Answering (VQA) datasets, such as VQA v1 [59], VQA v2 [60] and GQA [61], requiring more modeling capacity for language understanding.

In our experiments, we found that early fusion of co-references between textual tokens and visual features of objects was the most critical factor in obtaining improvements on VCR. We found that the more visual object features we included in the model's input, the better the model

Table 4.1 Glossary of mathematical symbols used in this chapter.

Symbol	Type	Description
m	$\mathbb{N}$	number of extracted bounding boxes
n	$\mathbb{N}$	number of tokens input to BERT
k	$\mathbb{N}$	number of positional embeddings for image coordinates, usually 56
d	$\mathbb{N}$	visual features dimension, usually 2048
h	$\mathbb{N}$	hidden dimension of BERT, usually 1024
l	$\{0, 1\}$	a binary label
I	$\mathbb{R}^{\cdot\times\cdot\times 3}$	an image
B	$\mathbb{R}^{m\times 4}$	rectangular bounding boxes on I, as coordinates of opposite corners
R	$\{0, 1\}^{m\times n}$	matrix encoding which bounding boxes in B correspond to which tokens in T
T	$\mathbb{N}^{n\times 2}$	input tokens, each expressed as word piece id and token type
Φ	$\mathbb{R}^{\cdot\times\cdot\times 3} \to \mathbb{R}^d$	a function to extract visual feature vectors from an image
π	$\mathbb{R}^4 \to \mathbb{R}^d$	a function to embed the position and shape of a bounding box
Ψ	$\mathbb{R}^{n\times h} \to \mathbb{R}^h$	a function to compute a passage embedding from per-token embeddings
E	$\mathbb{N}^{n\times 2} \to \mathbb{R}^{n\times h}$	non-contextualized token embeddings, encoding word piece ids, token types and positions

performed, even if they were not explicitly co-referent to the text. Positional features of objects, i.e. the pixel position and dimensions of the object's bounding box, also provided a small performance boost. We finally discovered that our models for VCR could be trained much more reliably when they were initialized from pretraining on Conceptual Captions [12], a public dataset of about 3M images with captions. From the combination of these modeling improvements, we obtained a new model for visual question answering that achieves state-of-the-art on VCR, reducing error rates by more than 25% relative to the best published and documented model [11].

4.2 Problem Formulation

In this work, we assume data comprised of 4-tuples (I, B, T, l) where

1. I is an image,
2. $B = [b_1, \ldots, b_m]$ is a list of bounding boxes referring to regions of I, where each b_i is identified by the lower left corner, height and width,
3. $T = [t_1, \ldots, t_n]$ is a passage of tokenized text, with the peculiarity that some of the tokens are not natural language, but explicit references to elements of B, and
4. l is a binary label in $\{0, 1\}$.

While it might seem surprising to mix natural text with explicit references to bounding boxes, this is actually a quite natural way for people to discuss objects in images and the VCR dataset is annotated in exactly this way.

We assume an image representation function Φ that converts an image, perhaps after resizing and padding, to a fixed size vector representation of dimension d.

We similarly assume a pretrained textual representation capable of converting any tokenized passage of text, perhaps after truncating or padding, into a vector representation of dimension h. We assume a context independent token representation E in the shape of a vector of dimension h for each token and a passage level representation Ψ which operates on $E(T)$ and returns a passage level vector representation of dimension h.

We refer the reader to Table 4.1 for an overview of the notation used in this chapter. Full details on how the VCR dataset is encoded into this formalism are given in Section 4.9.

4.3 Models and Methods

We evaluate two main architectures: "Dual Encoder", a late fusion architecture where image and text are encoded separately and answer scores are computed as an inner product, and the full B2T2 model, an early fusion architecture where visual features are embedded on the same level as input word tokens. Section 4.12 will summarize experiments with model variants to answer the research questions laid out in the introduction and to analyze what works, and why.

4.4 Dual Encoder

Dual Encoders, discussed for example by [62] and [63], are models that embed objects of potentially different types into a common representation space where a similarity function can be expressed e.g. as a dot product or a cosine similarity. A notable example of a dual encoder for image classification is WSABIE, proposed by [64].

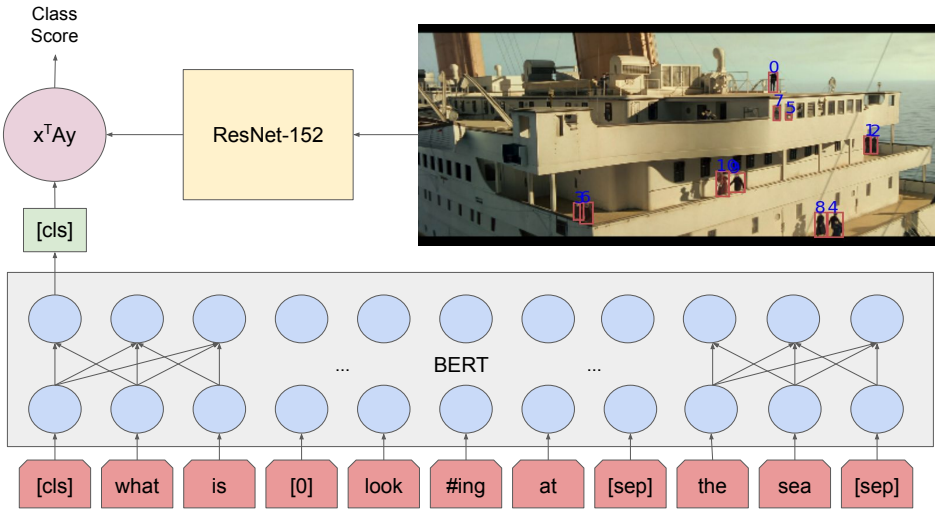


Figure 4.2: Dual Encoder architecture with late fusion. The model extracts a single visual feature vector from the entire image. Bounding boxes are ignored.

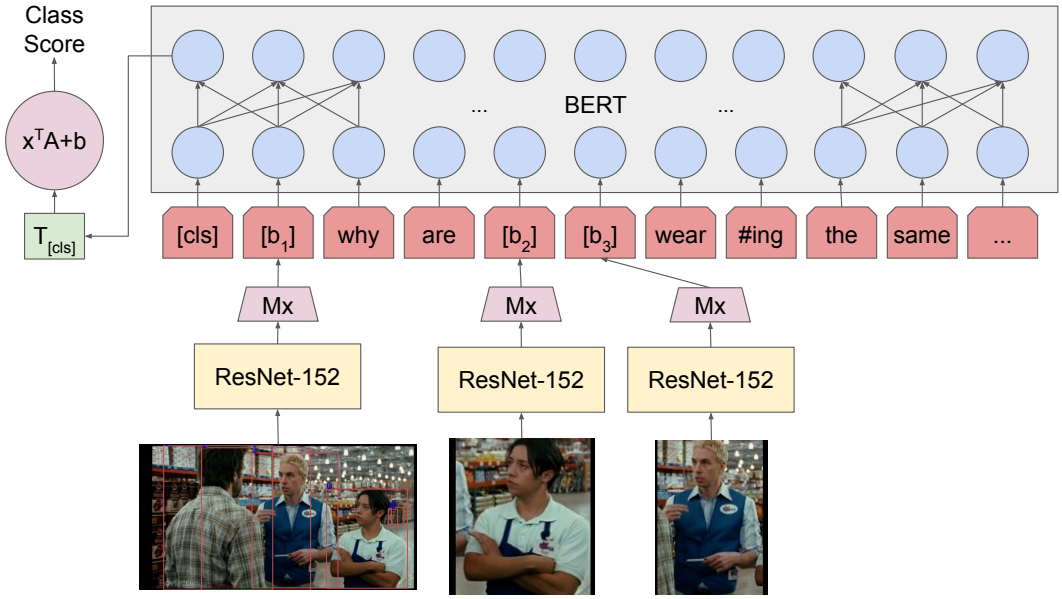


Figure 4.3: B2T2 architecture with early fusion. Bounding boxes are inserted where they are mentioned in the text and at the end of the input, as described in Sec. 4.9.

Our Dual Encoder architecture is shown in Figure 4.2. We model the class distribution as

$$p(l = 1|I, T) = \frac{1}{1 + e^{-\Psi(E(T))^\top D\Phi(I)}}$$

where D is a learned matrix of size $d \times h$. In this model, co-reference information is completely

ignored, and the model must rely on fixed dimensional vectors for the late fusion of textual and visual contexts. However, we found this to be surprisingly competitive on VCR compared to published baselines, perhaps due to our choice of powerful pretrained models.

4.5 B2T2

Our B2T2 architecture is shown in Figure 4.3. We model the class distribution as

$$p(l|I, B, R, T) = \frac{e^{\Psi(E'(I,B,R,T))\cdot a_l + b_l}}{\sum_{l'} e^{\Psi(E'(I,B,R,T))\cdot a_{l'} + b_{l'}}}$$

where $a_l \in \mathbb{R}^h$ and $b_l \in \mathbb{R}$ for $l \in \{0, 1\}$ are learned parameters. $E'(I, B, R, T)$ is a non-contextualized representation for each token and of its position in text, but also of the content and position of the bounding boxes. The key difference from "Dual Encoder" is that text, image and bounding boxes are combined at the level of the non-contextualized token representations rather than right before the classification decision.

The computation of $E'(I, B, R, T)$ is depicted in Figure 4.4. More formally, for a given example, let matrix $R \in \{0, 1\}^{m\times n}$ encode the references between the bounding boxes in B and the tokens in T, so that R_{ij} is 1 if and only if bounding box i is referenced by token j. Then

$$E'(I, B, R, T) = \\ E(T) + \sum_{i=1}^{m} R_i \left[M(\Phi(\text{crop}(I, b_i)) + \pi(b_i))\right]^\top$$

where M is a learned $h \times d$ matrix, $\Phi(\text{crop}(I, b_i))$ denotes cropping image I to bounding box b_i and then extracting a visual feature vector of size d, and $\pi(b_i)$ denotes the embedding of b_i's shape and position information in a vector of size d.

To embed the position and size of a bounding box b, we introduce two new learnable embedding matrices X and Y of dimension $k \times \frac{d}{4}$. Let the coordinates of the opposite corners of b be (x_1, y_1) and (x_2, y_2), after normalizing so that a bounding box covering the entire image would have $x_1 = y_1 = 0$

and $x_2 = y_2 = k$. Position embeddings are thus defined to be

$$\pi(b) = \text{concat}(X_{\lfloor x_1 \rfloor}, Y_{\lfloor y_1 \rfloor}, X_{\lfloor x_2 \rfloor}, Y_{\lfloor y_2 \rfloor})$$

4.6 Loss

All of our models are trained with binary cross entropy loss using label l. Denoting $p := p(l = 1|I, B, R, T)$, we have for each example

$$\mathcal{L}_{\text{BCE}} = l \log p + (1 - l) \log(1 - p)$$

4.7 Pretraining on Conceptual Captions

Before training on VCR, we pretrain B2T2 on image and caption pairs using a Mask-LM pretraining technique like the one used in BERT [3]. The setup used during pretraining is shown in Figure 4.5, where the model uses the image as additional context when filling in the mask.

We use two tasks for pretraining: (1) impostor identification and (2) masked language model prediction. For the impostor task, we sample a random negative caption for each image and ask the model to predict whether the caption is correctly associated. For mask-LM, we randomly replace tokens in the caption with the `[MASK]` token, and the model must predict the original token (see [3] for more details).

Formally, the pretraining data consist of images I and captions T. We do not consider bounding boxes during pretraining, so $B = \emptyset$. The binary label l indicates whether the caption is an impostor or not. The loss for impostor identification is binary cross entropy $\mathcal{L}_{\text{BCE}}$ with label l as in 4.6. We denote the loss for mask-LM as $\mathcal{L}_{\text{MLM}}$, which is the summed cross entropy of the predicted token distributions against the true tokens.

To ensure that our model correctly grounds the language to the image with the mask LM loss, we only use it for positive captions, zeroing it out for negative captions. Our final objective is the

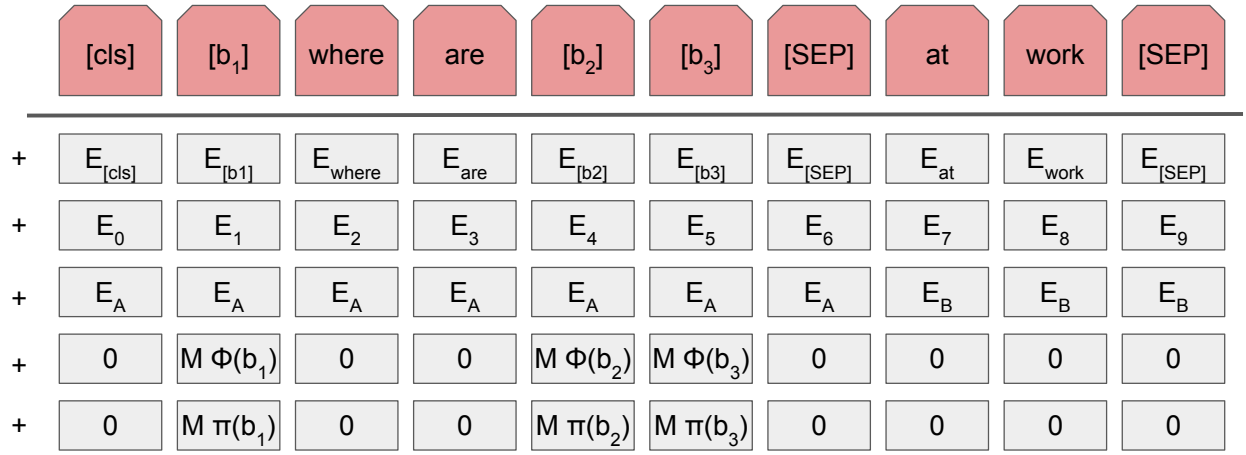


Figure 4.4: How input embeddings are computed in our B2T2 architecture.

sum of the losses:

$$\mathcal{L} = \mathcal{L}_{\text{BCE}} + I[l = 1] \cdot \mathcal{L}_{\text{MLM}}$$

where $I[l = 1]$ is an indicator for the label l being positive for the image and caption pair.

We pretrain on Conceptual Captions [12], a dataset with over 3M images paired with captions.[2] We found empirically that pretraining improves our model slightly on VCR, but more importantly, allows our model to train stably. Without pretraining, results on VCR exhibit much higher variance. We refer the reader to Section 4.12 for an ablation analysis on the effect of pretraining.

4.8 Implementation Details

We use ResNet-152[3] [66] pretrained on ImageNet for Φ, which yields a vector representation of size $d = 2048$. BERT-Large [3] provides both E and Ψ. The latter is a pretrained Transformer with 24 layers, 16 attention heads, and hidden size 1024. For BERT, E corresponds to its token embeddings, Ψ to the `[CLS]` token representation in the final layer, and so $\Psi(E(T))$ corresponds to the BERT passage representation of size $h = 1024$.

We found empirically that it was slightly better to keep Φ fixed rather than fine-tuning it, but

[2]We also tried pretraining on MS-COCO images and captions [65], but found this to be ineffective. This could be because MS-COCO is smaller (with around 80k images, 400k captions).

[3]Publicly available at `tfhub.dev`

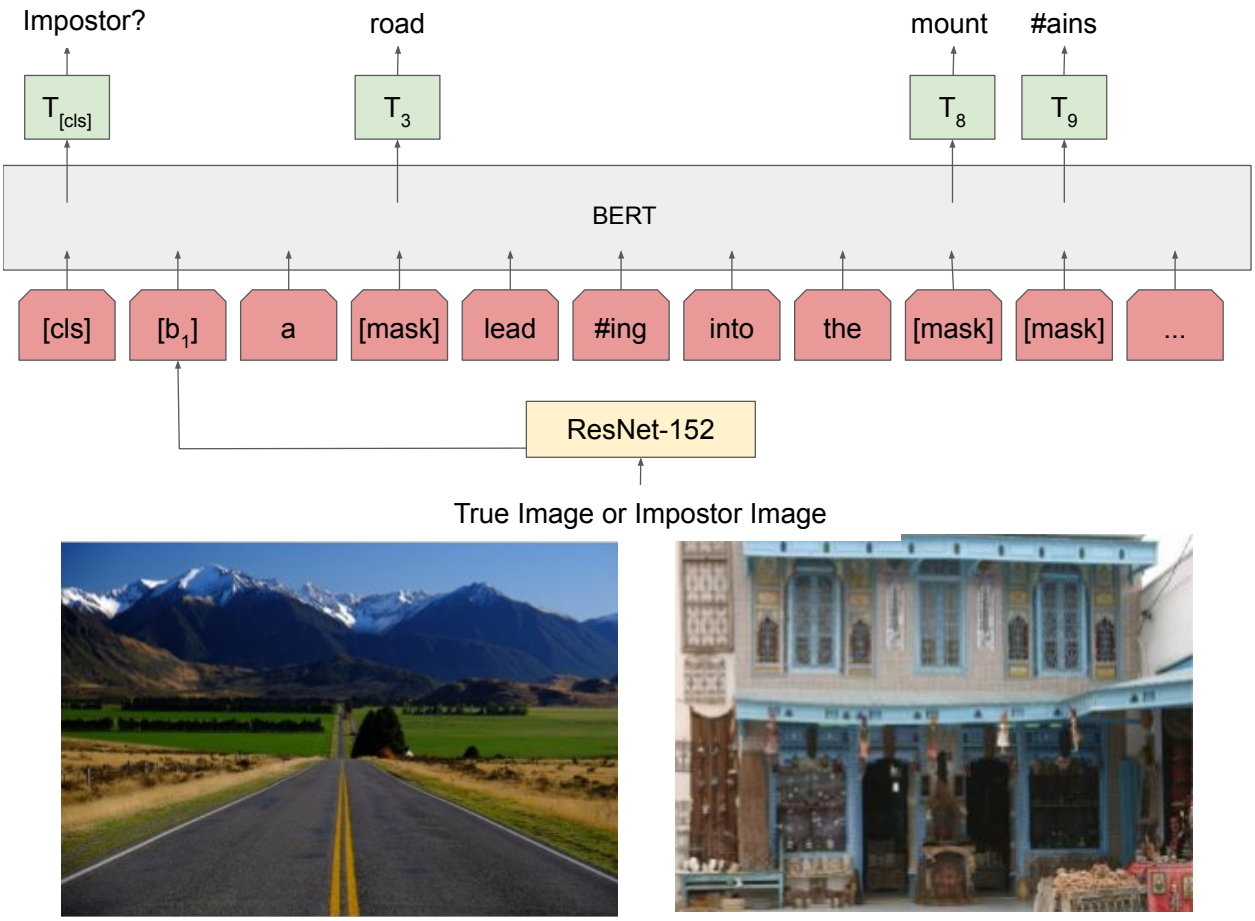


Figure 4.5: Mask-LM pretraining for B2T2.

that it was of critical importance to fine-tune Ψ and E for the new task.

In all of our finetuning experiments we use the Adam optimizer [21] and trained our models with a grid of hyperparameters: a learning rate of $2 \cdot 10^{-5}$ and $3 \cdot 10^{-5}$, for 3, 4, and 5 epochs with a linear learning rate decay, and two random seed for initialization. To maximize performance on VCR, we also evaluate an ensemble of B2T2 models. Our ensemble is comprised of 5 identical B2T2 models, trained for 3 epochs with an initial learning rate of $2 \cdot 10^{-5}$, but initialized with 5 different random seeds. The resulting class logits are then summed to obtain the ensemble scores.

4.9 Data

Visual Commonsense Reasoning (VCR, `visualcommonsense.com`, [11]) is a corpus that contains a sample of stills from movies. Questions and answers revolve around conclusions or assumptions that require knowledge external to the images. The associated task is to not only select a correct answer but also provide reasoning in line with common sense. Matching our problem formulation given before, a VCR sample is defined as a tuple (I, O, Q, A, R). Here, I is the image, and O is a sequence of objects identified in the image. A question $Q = [q_0, \ldots, q_k]$ is given, where tokens are either textual words or references to objects in O. Each question contains a set of four answers $A = \{A_1, A_2, A_3, A_4\}$, with exactly one correct answer A^*. Each response follows the schema of the queries. Finally, there is a set of four rationales $R = \{R_1, R_2, R_3, R_4\}$, with exactly one rationale R^* identified as correct in supporting A^*.

Each of the objects in $O = [(b_1, l_1), \ldots, (b_{|O|}, l_{|O|})]$ is identified in the image I by bounding boxes b_i. The objects are also labeled with their classes with a text token l_i.

The $Q \rightarrow A$ task is to choose A^* given (I, O, Q, A). The $QA \rightarrow R$ task is to choose R^* given (I, O, Q, A^*, R). Finally, the $Q \rightarrow AR$ task is a pipeline of the two, where a model must first correctly choose A^* from A, then correctly choose R^* given A^*.

We adapt VCR to our problem formulation by converting each VCR example to four instances for the $Q \rightarrow A$ task, one per answer in A, and four instances for the $QA \rightarrow R$ task, one per rationale in R. We construct the text for the instances in the $Q \rightarrow A$ task as

$$[[\text{CLS}], [b_0], q_0, \ldots, [\text{SEP}],$$
$$a_0, \ldots, [\text{SEP}], l_1, [b_1], \ldots, l_p, [b_p]]$$

and in the $QA \rightarrow R$ task as

$$[[\text{CLS}], [b_0], q_0, \ldots, [\text{SEP}], a_0^*, \ldots,$$
$$r_0, \ldots, [\text{SEP}], l_1, [b_1], \ldots, l_p, [b_p]].$$

where [CLS], [SEP] are special tokens for BERT.

Here, $[b_0]$ is a bounding box referring to the entire input image. $q_0, \ldots$ are all question tokens, $a_0, \ldots$ answer tokens, $a_0^*, \ldots$ answer tokens for the correct answer, and $r_0, \ldots$ rationale tokens. We append the first p bounding boxes in O with class labels to the end of the sequence (in our experiments, we use $p = 8$), and for objects referenced in Q, A, R, we prepend the class label token (i.e. $[b_i]$ becomes $l_i, [b_i]$). We assign the binary label l to every instance to represent whether the answer or rationale choice is the correct one.

4.10 Experimental Results

4.11 VCR Task Performance

Our final results on the VCR task are shown in Table 4.2. Our Dual Encoder model worked surprisingly well compared to [11], surpassing the baseline without making use of bounding boxes. We also evaluate a Text-Only baseline, which is similar to the Dual Encoder model but ignores the image. The ensemble of B2T2 models, pretrained on Conceptual Captions, obtained absolute accuracy improvements of 8.9%, 9.8% and 13.1% compared to the published R2C baseline for the $Q \rightarrow A$, $QA \rightarrow R$, and $Q \rightarrow AR$ tasks respectively. At the time the B2T2 paper was written (May 22, 2019), both our single B2T2 and ensemble B2T2 models outperformed all other systems in the VCR leaderboard.

Table 4.2 Experimental results on VCR, incorporating those reported by [11]. The proposed B2T2 model and the B2T2 ensemble outperform published and unpublished/undocumented results found on the VCR leaderboard at `visualcommonsense.com/leaderboard` as of May 22, 2019.

	$Q \rightarrow A$		$QA \rightarrow R$		$Q \rightarrow AR$	
Model	Val	Test	Val	Test	Val	Test
Chance	25.0	25.0	25.0	25.0	6.2	6.2
Text-Only BERT (Zellers et al.)	53.8	53.9	64.1	64.5	34.8	35.0
R2C (Zellers et al.)	63.8	65.1	67.2	67.3	43.1	44.0
HCL HGP (unpub.)	-	70.1	-	70.8	-	49.8
TNet (unpub.)	-	70.9	-	70.6	-	50.4
B-VCR (unpub.)	-	70.5	-	71.5	-	50.8
TNet 5-Ensemble (unpub.)	-	72.7	-	72.6	-	53.0
Text-Only BERT (ours)	59.5	-	65.6	-	39.3	-
Dual Encoder (ours)	66.8	-	67.7	-	45.3	-
B2T2 (ours)	71.9	72.6	76.0	75.7	54.9	55.0
B2T2 5-Ensemble (ours)	**73.2**	**74.0**	**77.1**	**77.1**	**56.6**	**57.1**
Human		91.0		93.0		85.0

4.12 Ablations

To better understand the reason for our improvements, we performed a number of ablation studies on our results, summarized in Table 4.3. We consider ablations in order of decreasing impact on the VCR dev set $Q \rightarrow A$ accuracy.

Use of Bounding Boxes. The bounding boxes considered by our model turn out to be the most important factor in improving the accuracy of our model. Without any bounding boxes we obtain 67.5% accuracy, just above the accuracy of the dual encoder. With 4 instead of 8 appended bounding boxes we obtain 71% accuracy. With 8 bounding boxes, but no textual labels from the bounding boxes in the text we obtain 70.9% accuracy, showing that our model can make use of labels for detected objects. Example 1 in Table 4.4 shows an example that our models can only get right if bounding box 5 is available.

Late Fusion vs. Early Fusion. The second most important architectural choice in our model is to combine visual information at the level of context independent token embeddings, rather than at the highest levels of the neural representation. If in the the full B2T2 model we add visual

Table 4.3 Ablations for B2T2 on VCR dev. The Dual Encoder and the full B2T2 models are the main models discussed in this work. All other models represent ablations from the full B2T2 model.

	$Q \rightarrow A$
Dual Encoder	66.8
No bboxes	67.5
Late fusion	68.6
BERT-Base	69.0
ResNet-50	70.4
No bbox class labels	70.9
Fewer appended bboxes ($p = 4$)	71.0
No bbox position embeddings	71.6
Full B2T2	71.9

embeddings in the last layer of BERT rather than in the first, we lose 3.3% accuracy.

Effect of Textual Model Size. The original VCR work by [11] made use of BERT-base, while we use BERT-large to initialize our models. To test how much of our improvements are simply due to our model being larger, we retrained B2T2 models using BERT-base and found that we lose 2.9% accuracy.

Effect of Visual Model Size. How important is the choice of the visual model in the performance of B2T2? As further discussed in the error analysis section of this work, we suspect that B2T2 could be significantly improved by extending the visual features to represent more than just objects, but also activities, expressions and more. However it appears that even the size of the object detection model is important. If we swap out ResNet-152 for ResNet-50, accuracy decreases by 1.5%.

Pretraining. We found that performance improvements from pretraining are quite small, around 0.4% accuracy, but initializing from a pretrained model heavily reduces variance of results. We show this effect in Figure 4.6 over the grid of learning rates, random seeds, and training epochs described in Section 4.8.

Position of Bounding Boxes We additionally investigated the effect of removing position information from the model. The benefit of having bounding box positional embeddings is the smallest of the ones we considered. A model trained without positional embeddings only loses 0.3% accuracy compared to the full model.

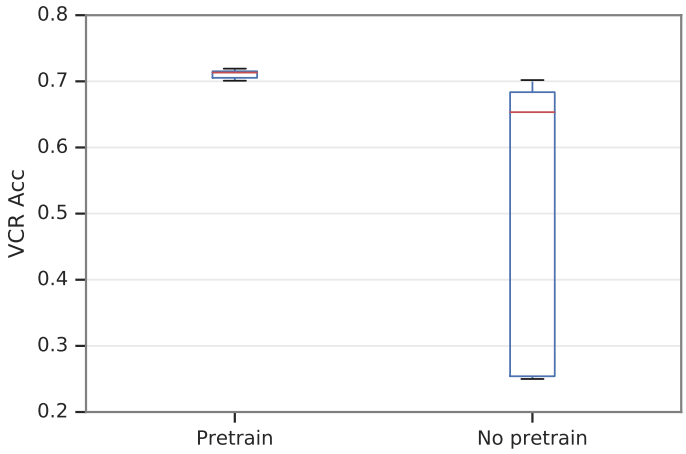


Figure 4.6: Boxplot of dev $Q \rightarrow A$ accuracy on VCR with and without pretraining. Pretraining on Conceptual Captions lowers variance when fine-tuning on VCR, from a grid search on multiple random seeds, learning rates, and VCR training epochs.

4.13 Error Analysis

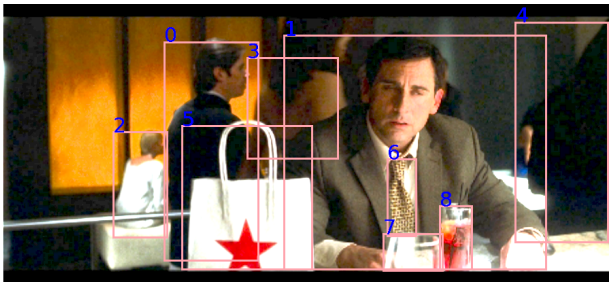
We picked some examples, shown in Table 4.4, to illustrate the kinds of correct and incorrect choices that B2T2 is making, compared to our dual encoder and to a text only model.

In Example 1 we show an example of how our model picks the right answer only when it is able to make use of all provided bounding boxes. Bounding box 5 in particular contains the clue that allows the observer to know that the man in the picture might have just gone shopping.

In Examples 2 and 3, no specific bounding box appears to contain critical clues for answering the question, but B2T2 outperforms models without access to the image or without access to bounding boxes. It is possible that B2T2 might be gaining deeper understanding of a scene by combining information from important regions of the image.

In Examples 4 and 5, we see failure cases of both the dual encoder and B2T2 compared to the text only-model. Both these examples appear to point to a limitation in the amount of information that the we are able to extract from the image. Indeed our vision model is trained on ImageNet, and so it might be very good at recognizing objects, but might be unable to recognize human expressions and activities. Our models could have correctly answered the question in Example 4 if they were

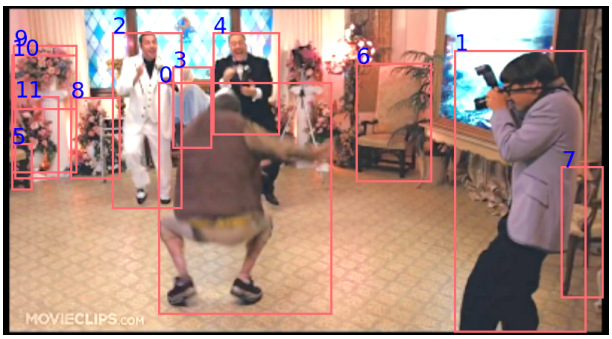
Table 4.4 Examples of the $Q \rightarrow A$ task from the VCR dev set. The correct answer for every example is marked in bold. The answers picked by the text-only model, by the dual encoder and by B2T2 are indicated in parenthesis.



Example 1

Q: What did [1] do before coming to this location?

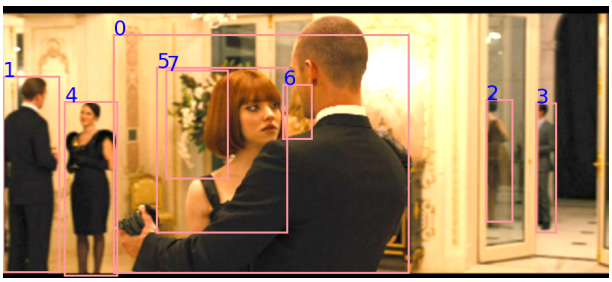
A_1: He took horse riding lessons. (text-only)
A_2: He was just shopping. (B2T2)
A_3: He found a skeleton.
A_4: He came to buy medicine. (dual encoder)



Example 2

Q: How are [2, 4] related?

A_1: [2, 4] are partners on the same mission.
A_2: [2, 4] are a recently married gay couple. (B2T2)
A_3: They are likely acquaintances.
A_4: They are siblings. (text-only, dual encoder)



Example 3

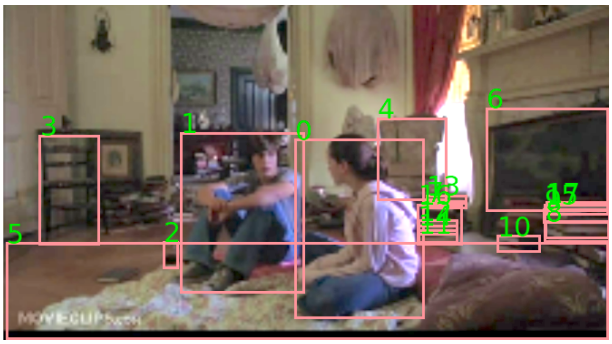
Q: What are [0] and the woman doing?

A_1: Their husbands are doing something dumb.
A_2: They are observing the results of an experiment. (text-only, dual encoder)
A_3: They are dancing. (B2T2)
A_4: They are acting as nurses for the rescued people.

Example 4

Q: How is [2] feeling?

A_1: [2] is feeling shocked. (B2T2, dual encoder)
A_2: [0] is feeling anxious.
A_3: [2] is not feeling well.
A_4: [2] is feeling joy and amusement. (text-only)



Example 5

Q: Why is [1] on the floor talking to [0]?

A_1: The man on the floor was assaulting [1].
A_2: He is asking her to help him stand up. (B2T2, dual encoder)
A_3: [1] just dropped all his books on the floor.
A_4: [1] looks like he is telling [0] a secret. (text-only)

able to recognize smiles. Similarly our models could have ruled out the incorrect answer they picked for the question in Example 5 if they were able to see that both people in the picture are sitting down and are not moving.

4.14 Related Work

Our B2T2 model is similar to the Bottom-Up Top-Down attention model [67] in how bounding boxes generated at pre-processing time are attended to by the VQA model. "Bottom-Up" refers to the idea of attending from the text to the bounding boxes of objects detected in the image, while "Top-Down" refers to the idea of attending to regions constructed as a regular grid over the image. The Bottom-Up Top-Down model however reduces the text to a fixed length vector representation before attending to image regions, while B2T2 instead treats image regions as special visual tokens mixed in the text. In this sense, Bottom-Up Top-Down model is a late fusion model, while B2T2 is early fusion. The Neuro-Symbolic Concept Learner [68] also uses bounding boxes to learn visually grounded concepts through language. The Neuro-Symbolic Concept Learner however relies on a semantic parser to intepret language, while B2T2 uses a Transformer to construct a joint representation of textual tokens and visual tokens. Another recently proposed model for VQA is MAC [69]. As presented, MAC does not make use of bounding boxes, which makes it a Top-Down model in the nomenclature of [67]. MAC also reduces the textual information to a vector of fixed length. However MAC makes use of a new neural architecture designed to perform an explicit multi-step reasoning process and is reported to perform better than [67] on the GQA dataset [61]. After the submission of the B2T2 paper, several new papers were published with excellent results on VCR, in some cases exceeding the performance of our system. In particular we mention ViLBERT [70], VL-BERT [71], Unicoder-VL [72], and VisualBERT [73]. VCR is only one of several recent datasets pertaining to the visual question answering task. VQA [59, 74, 60] contains photos and abstract scenes with questions and several ground-truth answers for each, but the questions are less complex than VCR's. CLEVR [75] is a visual QA task with compositional language, but the scenes and language are synthetic. GQA [61] uses real scenes from Visual Genome, but the language is

artificially generated. Because VCR has more complex natural language than other datasets, we consider it the best evaluation of a model like B2T2, which has a powerful language understanding component.

4.15 Conclusion

In this chapter we contrasted different ways of combining text and images when powerful text and vision models are available. We picked BERT-Large [3] as our text model, ResNet-152 [66] as our vision model, and the VCR dataset [11] as our main benchmark. The early-fusion B2T2 model, which encodes sentences along with links to bounding boxes around identified objects in the images, produced the best available results in the visual question answering tasks. A control model, implementing late fusion (but the same otherwise), performed substantively worse. Thus, grounding words in the visual context should be done early rather than late.

We demonstrated competitive results with a Dual Encoder model, matching state-of-the-art on the VCR dataset even when textual references to image bounding boxes are ignored. We then showed that our Dual Encoder model can be substantially improved by deeply incorporating in the textual embeddings visual features extracted from the entire image and from bounding boxes. We finally showed that pretraining our deep model on Conceptual Captions with a Mask-LM loss yields a small additional improvement as well as much more stable fine-tuning results.

Chapter 5: QA Corpora Generation with Roundtrip Consistency

In this chapter we report another first author contribution [76], where we look at a specific way of pretraining question answering models to obtain better logical consistency, namely roundtrip consistency. The proposed method, illustrated in Table 5.1), consists of generating synthetic question answering corpora with a question generation model, and then in filtering generated questions by verifying that a question answering model would return the same answer that was used to generate the question. By pretraining on the resulting corpora we obtain significant improvements on SQuAD2 [14] and NQ, establishing a new state-of-the-art on the latter. We also describe a more powerful variant that does full sequence-to-sequence pretraining for question generation, obtaining exact match and F1 at less than 0.1% and 0.4% from human performance on SQuAD2.

5.1 Overview

Automatically generating tasks from vast amounts of unlabeled text has been one of the key ideas behind the recent dramatic advancements in question answering and natural language processing. BERT [3] in particular relies on two such tasks: masked language modeling, i.e. masking some words from each passage and training the model to predict them back, and next sentence prediction, i.e. classifying a sentence as the next one to appear after the current text or not. It seems plausible however that other auxiliary tasks might exist that are better suited for QA, but can still be constructed from widely available natural text. It also seems intuitive that such auxiliary tasks will be more helpful the closer they are to the particular QA task we are attempting to solve. Based on this intuition we construct auxiliary tasks for QA, generating millions of synthetic question-answer-context triples from unlabeled passages of text, pretraining a model on these examples, and finally finetuning on a particular labeled dataset. Our auxiliary tasks are illustrated in Table 5.1.

Table 5.1 Example of how synthetic question-answer pairs are generated. The model's predicted answer (A') matches the original answer the question was generated from, so the example is kept.

Input (C)		... in 1903, boston participated in the first modern world series, going up against the pittsburgh pirates ...
(1)	$\boldsymbol{C \rightarrow A}$	1903
(2)	$\boldsymbol{C, A \rightarrow Q}$	when did the red sox first go to the world series
(3)	$\boldsymbol{C, Q \rightarrow A'}$	1903
(4)	$\boldsymbol{A \stackrel{?}{=} A'}$	Yes

For a given passage C, we sample an extractive short answer A (Step (1) in Table 5.1). In Step (2), we generate a question Q conditioned on A and C, then (Step (3)) predict the extractive answer A' conditioned on Q and C. If A and A' match we finally emit (C, Q, A) as a new synthetic training example (Step (4)). We train a separate model on labeled QA data for each of the first three steps, and then apply the models in sequence on a large number of unlabeled text passages. We show that pretraining on synthetic data generated through this procedure provides us with significant improvements on two challenging datasets, SQuAD2 [14] and NQ [77], achieving a new state of the art on the latter.

5.2 Model

Given a dataset of contexts, questions, and answers: $\{(c^{(i)}, q^{(i)}, a^{(i)}) : i = 1, \ldots, N\}$, we train three models: (1) answer extraction: $p(a|c; \theta_A)$, (2) question generation: $p(q|c, a; \theta_Q)$, and (3) question answering: $p(a|c, q; \theta_{A'})$.

We use BERT [3][1] to model each of these distributions. Inputs to each of these models are fixed length sequences of wordpieces, listing the tokenized question (if one was available) followed by the context c. The answer extraction model is detailed in §5.2.1 and two variants of question generation models in §5.2.2 and §5.2.3. The question answering model follows [8].

[1]Some experiments use a variant of BERT that masks out whole words at training time, similar to [78]. See `https://github.com/google-research/bert` for both the original and whole word masked versions of BERT.

5.2.1 Question (Un)Conditional Extractive QA

We define a question-unconditional extractive answer model $p(a|c;\theta_A)$ and a question-conditional extractive answer model $p(a|q,c;\theta_{A'})$ as follows:

$$p(a|c;\theta_A) = \frac{e^{f_J(a,c;\theta_A)}}{\sum_{a''} e^{f_J(a'',c;\theta_A)}}$$
$$p(a|c,q;\theta_{A'}) = \frac{e^{f_I(a,c,q;\theta_{A'})}}{\sum_{a''} e^{f_I(a'',c,q;\theta_{A'})}}$$

where a, a'' are defined to be token spans over c. For $p(a|c;\theta_A)$, a and a'' are constrained to be of length up to L_A, set to 32 word piece tokens. The key difference between the two expressions is that f_I scores the start and the end of each span independently, while f_J scores them jointly.

Specifically we define $f_J : \mathbb{R}^h \rightarrow \mathbb{R}$ and $f_I : \mathbb{R}^h \rightarrow \mathbb{R}$ to be transformations of the final token representations computed by a BERT model:

$$f_J(a,c;\theta_A) =$$
$$\text{MLP}_J(\text{CONCAT}(\text{BERT}(c)[s], \text{BERT}(c)[e]))$$
$$f_I(a,q,c;\theta_{A'})) =$$
$$\text{AFF}_I(\text{BERT}(q,c)[s]) + \text{AFF}_I(\text{BERT}(q,c)[e]).$$

Here h is the hidden representation dimension, $(s,e) = a$ is the answer span, $\text{BERT}(t)[i]$ is the BERT representation of the i'th token in token sequence t. MLP_J is a multi-layer perceptron with a single hidden layer, and AFF_I is an affine transformation.

We found it was critical to model span start and end points jointly in $p(a|c;\theta_A)$ because, when the question is not given, there are usually multiple acceptable answers for a given context, so that the start point of an answer span cannot be determined separately from the end point.

5.2.2 Question Generation: Fine-tuning Only

Text generation allows for a variety of choices in model architecture and training data. In this section we opt for a simple adaptation of the public BERT model for text generation. This adaptation does not require any additional pretraining and no extra parameters need to be trained from scratch at finetuning time. This question generation system can be reproduced by simply finetuning a publicly available pretrained BERT model on the extractive subsets of datasets like SQuAD2 and NQ.

Fine-tuning We define the $p(q|c, a; \theta_Q)$ model as a left-to-right language model

$$\begin{aligned} p(q|a, c; \theta_Q) &= \prod_{i=1}^{L_Q} p(q_i|q_1, \ldots, q_{i-1}, a, c; \theta_Q) \\ &= \prod_{i=1}^{L_Q} \frac{e^{f_Q(q_1,\ldots,q_i,a,c;\theta_Q)}}{\sum_{q'_i} e^{f_Q(q_1,\ldots,q'_i,a,c;\theta_Q)}}, \end{aligned}$$

where $q = (q_1, \ldots, q_{L_Q})$ is the sequence of question tokens and L_Q is a predetermined maximum question length, but, unlike the more usual encoder-decoder approach, we compute f_Q using the single encoder stack from the BERT model:

$$\begin{aligned} &f_Q(q_1, \ldots, q_i, a, c; \theta_Q) = \\ &\quad \text{BERT}(q_1, \ldots, q_{i-1}, a, c)[i-1] \cdot W_{\text{BERT}}^{\intercal}, \end{aligned}$$

where W_{BERT} is the word piece embedding matrix in BERT. All parameters of BERT including W_{BERT} are finetuned. In the context of question generation, the input answer is encoded by introducing a new token type id for the tokens in the extractive answer span, e.g. the question tokens being generated have type 0 and the context tokens have type 1, except for the ones in the answer span that have type 2. We always pad or truncate the question being input to BERT to a constant length L_Q to avoid giving the model information about the length of the question we want it to generate.

This model can be trained efficiently by using an attention mask that forces to zero all the attention weights from c to q and from q_i to $q_{i+1} \ldots q_{L_Q}$ for all i.

Question Generation At inference time we generate questions through iterative greedy decoding, by computing $\text{argmax}_{q_i} \; f_Q(q_1, \ldots, q_i, a, c)$ for $i = 1, \ldots, L_Q$. Question-answer pairs are kept only if they satisfy roundtrip consistency.

5.2.3 Question Generation: Full Pretraining

The prior section addressed a restricted setting in which a BERT model was fine-tuned, without any further changes. In this section, we describe an alternative approach for question generation that fully pretrains and fine-tunes a sequence-to-sequence generation model.

Section 5.2.2 used only an encoder for question generation. In this section, we use a full sequence-to-sequence Transformer (both encoder and decoder). The encoder is trained identically (BERT pretraining, Wikipedia data), while the decoder is trained to output the next sentence. Fine-tuning is done identically as in Section 5.2.2, where the input is (C, A) and the output is Q from tuples from a supervised question-answering dataset (e.g., SQuAD). To get examples of synthetic (C, Q, A) triples, we sample from the decoder with both beam search and Monte Carlo search. As before, we use roundtrip consistency to keep only the high precision triples.

5.3 Experiments

5.3.1 Experimental Setup

We considered two datasets in this work: SQuAD2 [14] and the Natural Questions (NQ) [77]. SQuAD2 is a dataset of QA examples of questions with answers formulated and answered by human annotators about Wikipedia passages. NQ is a dataset of Google queries with answers from Wikipedia pages provided by human annotators. We used the full text from the training set of NQ (1B words) as a source of unlabeled data.

In our fine-tuning only experiments (Section 5.2.2) we trained two triples of models $\left(\theta_A, \theta_Q, \theta_{A'}\right)$

Table 5.2 Our results on SQuAD2. For our fine-tuning only setting, we compare a BERT baseline (BERT single model - Google AI Language on the SQuAD2 leaderboard) to similar models pretrained on our synthetic SQuAD2-style corpus and on a corpus containing both SQuAD2- and NQ-style data. For the full pretraining setting, we report our best single model and ensemble results.

	Dev		Test	
	EM	F1	EM	F1
Fine-tuning Only				
BERT-Large (Original)	78.7	81.9	80.0	83.1
+ 3M synth SQuAD2	80.1	82.8	-	-
+ 4M synth NQ	81.2	84.0	82.0	84.8
Full Pretraining				
BERT (Whole Word Masking)[2]	82.6	85.2	-	-
+ 50M synth SQuAD2	85.1	87.9	85.2	87.7
+ ensemble	86.0	88.6	86.7	89.1
Human	-	-	86.8	89.5

Table 5.3 Our results on NQ, compared to the previous best system and to the performance of a human annotator and of an ensemble of human annotators. $\text{BERT}_{\text{joint}}$ is the model described in [8].

	Long Answer Dev			Long Answer Test			Short Answer Dev			Short Answer Test		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
BERT-joint	61.3	68.4	64.7	64.1	68.3	66.2	59.5	47.3	52.7	**63.8**	44.0	52.1
+ 4M synth NQ	**62.3**	**70.0**	**65.9**	**65.2**	**68.4**	**66.8**	**60.7**	**50.4**	**55.1**	62.1	**47.7**	**53.9**
Single Human	80.4	67.6	73.4	-	-	-	63.4	52.6	57.5	-	-	-
Super-annotator	90.0	84.6	87.2	-	-	-	79.1	72.6	75.7	-	-	-

Table 5.4 Comparison of question-answer pairs generated by NQ and SQuAD2 models for the same passage of text.

	Question	Answer
NQ	what was the population of chicago in 1857?	over 90,000
SQuAD2	what was the weight of the brigg's hotel?	22,000 tons
NQ	where is the death of the virgin located?	louvre
SQuAD2	what person replaced the painting?	carlo saraceni
NQ	when did rick and morty get released?	2012
SQuAD2	what executive suggested that rick be a grandfather?	nick weidenfeld

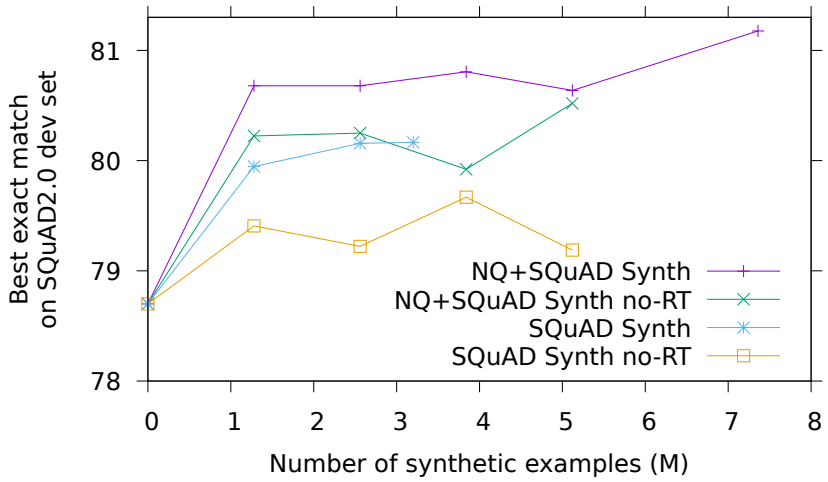


Figure 5.1: Learning curves for pretraining using synthetic question-answering data (fine-tuning only setting). "no-RT" refers to omitting the roundtrip consistency check. Best exact match is reported after fine-tuning on SQuAD2. Performance improves with the amount of synthetic data. For a fixed amount of synthetic data, having a more diverse source (NQ+SQuAD vs. just SQuAD) yields higher accuracies. Roundtrip filtering gives further improvements.

on the extractive subsets of SQuAD2 and NQ. We extracted 8M unlabeled windows of 512 tokens from the NQ training set. For each unlabeled window we generated one example from the SQuAD2-trained models and one example from the NQ-trained models. For A we picked an answer uniformly from the top 10 extractive answers according to $p(a|c;\theta_A)$. For A' we picked the best extractive answer according to $p(a|c,q;\theta_{A'})$. Filtering for roundtrip consistency gave us 2.4M and 3.2M synthetic positive instances from SQuAD2- and NQ-trained models respectively. We then added synthetic unanswerable instances by taking the question generated from a window and associating it with a non-overlapping window from the same Wikipedia page. We then sampled negatives to obtain a total of 3M and 4M synthetic training instances for SQuAD2 and NQ respectively. We trained models analogous to [8] initializing from the public BERT model, with a batch size of 128 examples for one epoch on each of the two sets of synthetic examples and on the union of the two, with a learning rate of $2 \cdot 10^{-5}$ and no learning rate decay. We then fine-tuned the the resulting models on SQuAD2 and NQ.

In our full pretraining experiments (Section 5.2.3) we only trained $(\theta_A, \theta_Q, \theta_{A'})$ on SQuAD2.

[2] https://github.com/google-research/bert

However, we pretrained our question generation model on all of the BERT pretraining data, generating the next sentence left-to-right. We created a synthetic, roundtrip filtered corpus with 50M examples. We then fine-tuned the model on SQuAD2 as previously described. We experimented with both the single model setting and an ensemble of 6 models.

5.3.2 Results

The final results are shown in Tables 5.2 and 5.3. We found that pretraining on SQuAD2 and NQ synthetic data increases the performance of the fine-tuned model by a significant margin. On the NQ short answer task, we reduce by 50% the gap between model performance and the performance of a single annotator. We additionally found that pretraining on the union of synthetic SQuAD2 and NQ data is very beneficial on the SQuAD2 task, but does not improve NQ results. The full pretraining approach with ensembling obtains the highest EM and F1 listed in Table 5.2. This result is only $0.1 - 0.4\%$ from human performance and is the third best model on the SQuAD2 leaderboard as of this writing (5/31/19).

Roundtrip Filtering Roundtrip filtering appears to be consistently beneficial. As shown in Figure 5.1, models pretrained on roundtrip consistent data outperform their counterparts pretrained without filtering. From manual inspection, of 46 (C, Q, A) triples that were roundtrip consistent 39% were correct, while of 44 triples that were discarded only 16% were correct.

Data Source Generated question-answer pairs are illustrative of the differences in the style of questions between SQuAD2 and NQ. We show a few examples in Table 5.4, where the same passage is used to create a SQuAD2-style and an NQ-style question-answer pair. The SQuAD2 models seem better at creating questions that directly query a specific property of an entity expressed in the text. The NQ models seem instead to attempt to create questions around popular themes, like famous works of art or TV shows, and then extract the answer by combining information from the entire passage.

5.4 Conclusion

This chapter presented a novel method to generate synthetic QA instances and demonstrated improvements from this data on SQuAD2 and on NQ. We additionally proposed a possible direction for formal grounding of this method, which we hope to develop more thoroughly in future work.

Chapter 6: QED: A Framework and Dataset for Explanations in Question Answering

A question answering system that, in addition to providing an answer, provides an *explanation* of the reasoning that leads to that answer, has potential advantages in terms of debuggability, extensibility and trust. To this end, we propose QED, a linguistically informed, extensible framework for explanations in question answering. A QED explanation specifies the relationship between a question and answer according to formal semantic notions such as referential equality, sentencehood, and entailment. We describe and publicly release an expert-annotated dataset of QED explanations built upon a subset of the Google Natural Questions dataset, and report baseline models on two tasks – post-hoc explanation generation given an answer, and joint question answering and explanation generation. In the joint setting, a promising result suggests that training on a relatively small amount of QED data can improve question answering. In addition to describing the formal, language-theoretic motivations for the QED approach, we describe a large user study showing that the presence of QED explanations significantly improves the ability of untrained raters to spot errors made by a strong neural QA baseline.

6.1 Introduction

Question Answering (QA) systems can enable efficient access to the vast amount of information that exists as text [20, 7, 79, 80, i.a.]. Modern neural systems have made tremendous progress in QA accuracy in recent years [81]. However, they generally give no explanation or justification of how they arrive at an answer to a question. Models that, in addition to providing an answer, can explain their reasoning may have significant benefits pertaining to trust and debuggability [82, 83].

Critical questions then, are: what constitutes an *explanation* in question answering? and how can

Question: who wrote the film howl's moving castle?
Passage: Howl's Moving Castle is a 2004 Japanese animated fantasy film written and directed by Hayao Miyazaki. It is based on the novel of the same name, which was written by Diana Wynne Jones. The film was produced by Toshio Suzuki.
Answer: Hayao Miyazaki

---

(1) Sentence Selection
Howl's Moving Castle is a 2004 Japanese animated fantasy film written and directed by Hayao Miyazaki.
(2) Referential Equality
the film howl's moving castle = Howl's Moving Castle
(3) Entailment
X is a 2004 Japanese animated fantasy film written and directed by ANSWER.; ⊢ ANSWER wrote X.

Figure 6.1: QED explanations decompose the question-passage relationship in terms of referential equality and predicate entailment.

we enable models to provide such explanations? In an effort to make progress on these questions, in this paper we make the following contributions: (1) we introduce QED[1], a linguistically grounded definition of QA explanations; and (2) we describe a corpus of QED annotations based on the Natural Questions [84]. The QED corpus has been released publicly.[2]

Figure 6.1 shows a QED example. Given a question and a passage, QED represents an explanation as a combination of discrete, human-interpretable steps: (1) identification of a sentence implying an answer to the question, (2) identification of noun phrases in both the question and answering sentence that refer to the same thing, and (3) confirmation that the predicate in the sentence entails the predicate in the question once referential equalities are abstracted away.

This choice of explanation makes use of core semantic relations—referential equality and entailment—and thus has well-understood formal properties. (See Section 6.2 for further discussion.) In addition, we found that this way of decomposing explanations has high coverage (77% on the Natural Questions corpus[3]). Since QED decomposes the QA process into distinct subproblems, we also believe that it should enable research directions aimed at extending or improving upon extant QA systems.

[1] QED stands for the Latin "quod erat demonstrandum" or "that which was to be shown".
[2] `https://github.com/google-research-datasets/QED`.
[3] Instances with annotated short answers, omitting table passages.

In what follows, after contextualizing the present work in the broader discussion on explainability, we present a formal definition of QED explanations. We then describe the dataset of QED annotations (7638/1353 train/dev examples), including discussion of the distribution of linguistic phenomena exhibited in the data. We move to propose four potential tasks, of varying complexity, related to the QED framework, and use the QED annotations to train and evaluate baseline models on two of these. Additionally, we describe a rater study which shows how the presence of QED explanations can help users identify errors made by an automated QA system.

6.2 Motivation: The Need for Explanations in Question Answering

We take as our departure point the following passage from [85] concerning explainable AI:

> *Explainability is important in situations where human operators work alongside autonomous and semi-autonomous systems because it can help build rapport, confidence, and understanding between the agent and its operator. In the event that an autonomous system fails to complete a task or completes it in an unexpected way, explanations help the human collaborator understand the circumstances that led to the behavior, which also allows the operator to make an informed decision on how to address the behavior.*

This quote refers to AI and ML systems in general, but is highly relevant to QA systems. Explanations can help users understand and trust a QA system, and can help them work in tandem with a QA system to fulfill their information needs. Explanations can also help system builders understand and debug QA systems and also t extend them.

QED makes a particular choice about the form of explanations for QA. In particular, it decomposes the question-answer relationship according to known semantic and syntactic categories – sentence, reference (and referential equality), predicate, and entailment. The explanations provided in QED are discrete structured objects, as opposed, for example, to “heat map"-style explanations (attention distributions, or other real-valued, word-level feature importance measures) [86].

One major goal in developing QED is to define models which provide *faithful* explanations; that

is, explanations that in some sense truly reflect the underlying computation or reasoning performed by a question-answering model. (See Section 6.7 for more discussion.) Another major goal, which is closely related to faithfulness, is to develop models that have a sound basis in concepts from cognitive science and linguistics and are thus closer to human reasoning. For example reference, a core component of QED, is fundamental to semantics and cognition [87, 88, 89].

6.3 Annotation Definition

We now describe the form of QED annotations. Section 6.3.1 gives an overview of the annotation process. Section 6.3.2 then gives a formal definition, which is extended in Section 6.3.3.

6.3.1 An Overview of the Approach

We will use the following example to illustrate the approach:

Question: how many seats in university of michigan stadium **Passage:** Michigan Stadium, nicknamed "The Big House", is the football stadium for the University of Michigan in Ann Arbor, Michigan. It is the largest stadium in the United States and the second largest stadium in the world. Its official capacity is 107,601.

The annotator is presented with a question/passage pair. Annotation then proceeds in the following four steps:

(1) Single Sentence Selection. The annotator identifies a single sentence in the passage that entails an answer to the question assuming that co-reference and bridging anaphora (see Section 6.3.3) have been resolved in the sentence.[4]

In the above example, the following sentence entails an answer to the question, and would be selected by the annotator:

Its official capacity is 107,601.

[4]If it is not possible to find a sentence that satisfies these properties—typically because the answer requires inference beyond co-reference/ bridging that involves multiple sentences—the annotator marks the example as not possible. See Section 6.4.

This follows because given the passage context, “Its” refers to the same thing as the NP “university of michigan stadium” in the question, and the predicate in the sentence, “X’s official capacity is 107,601”, entails the predicate in the question “how many seats in X”.

(2) Answer Selection. The annotator highlights a short answer span (or spans) in the answer sentence. In the above example the annotator would mark the following (answer shown with [=A...]):

Its official capacity is [=A 107,601].

In addition if the answer appears in the sentence in the form of a pronoun, bridged reference or underspecified NP, the annotator resolves the underlying co-reference within the passage (see Section 6.3.3 for more discussion).

(3) Identification of Question-Sentence Noun Phrase Equalities. The annotator marks referentially equivalent noun phrases, or noun phrases that refer to the same thing, in the question and the answer sentence. This includes reference not only to individuals and other proper nouns, but also to generic concepts.

In our example the annotator would mark the following two noun-phrases (marked with the [=1 ...] annotations) as referentially equivalent:

how many seats in [=1 university of michigan stadium]
[=1 Its] official capacity is [=A 107,601]

(4) Extraction of an Entailment Pattern. As a final, automatic step, an entailment pattern can be extracted from the annotated example by abstracting over referentially equivalent noun phrases, and the answer. In the above example the entailment pattern would be as follows:

how many seats in X

X’s official capacity is ANSWER

6.3.2 A Formal Definition

An annotator is presented with a question q that consists of m tokens $q_1 \ldots q_m$, along with a passage c consisting of n tokens $c_1 \ldots c_n$.

The QED annotation is a triple $\langle s, e, a \rangle$ where:

- s is a sentence within the context c. Specifically s is a pair s_0, s_1 indicating that the sentence spans words $c_{s_0} \ldots c_{s_1}$ inclusive.
- e is a sequence of 0 or more "referential equality annotations", $e_1 \ldots e_{|e|}$. Each member of e specifies that some noun phrase within the question refers to the same item in the world as some noun phrase within the sentence s.
- a is one or more answer annotations $a_1 \ldots a_{|a|}$.

We now describe the form of the e and a annotations. As a preliminary step, given the paragraph c and sentence s, we use $\mathcal{S}$ to refer to the set of all phrases within s. Our initial definition of $\mathcal{S}$ is

$$\mathcal{S} = \{(i, j) : s_0 \leq i \leq j \leq s_1\}$$

We also define the set of question phrases $\mathcal{Q}$ and passage phrases C to be

$$\begin{aligned} \mathcal{Q} &= \{(i, j) : 1 \leq i \leq j \leq m\} \\ C &= \{(i, j) : 1 \leq i \leq j \leq n\} \end{aligned}$$

We can then give the following definitions:

Definition 1 *Each referential equality annotation e_k for $k = 1 \ldots |e|$ is a pair $(\phi_k, \pi_k) \in \mathcal{Q} \times \mathcal{S}$, specifying that the phrase ϕ_k in the query refers to the same thing in the world as the phrase π_k within s.*

Definition 2 *Each answer annotation a_k for $k = 1 \ldots |a|$ is a pair $(\pi_k, \xi_k) \in \mathcal{S} \times C$ specifying that the answer is given by phrase π_k, and the full string corresponding to π_k after co-reference is resolved is the phrase ξ_k. If no co-reference resolution is required then $\pi_k = \xi_k$.*

To illustrate the treatment of co-reference resolution within answers, consider the following:

Question: who won wimbledon in 2019
Passage: Simona Halep is a female tennis player. She won Wimbledon in 2019.

In this case the single sentence *She won Wimbledon in 2019* would be selected by the annotator in step 1, as once co-reference is resolved, this entails the answer to the question. The QED annotation would be as follows

who won [=1 wimbledon] in [=2 2019]
[=A She] won [=1 Wimbledon] in [=2 2019]

However, the answer "She" is not sufficient, as it involves an unresolved anaphor. Because of this, the annotator would mark the fact that "She" refers to "Simona Halep" earlier in the passage. In this case the answer is a pair (π, ξ) where π corresponds to "She" within the sentence, and ξ corresponds to the earlier phrase "Simona Halep".

6.3.3 Extending Annotations to Include Bridging

Bridging anaphora [90] are frequently encountered in the QA passages in our data, and in Wikipedia more broadly. This section describes an extension to include annotations of bridging anaphora. Consider the following:

Question: who won america's got talent season 11

Passage: The 11th season of America's Got Talent, an American talent show competition, began broadcasting in the United States during 2016. Grace VanderWaal was announced as the winner on September 14, 2016.

It is clear from context surrounding the sentence "Grace VanderWaal was announced as the winner on September 14, 2016" that the noun phrase "the winner" refers to "the winner of America's Got Talent Season 11", and hence the sentence provides an answer to the question. It is helpful to imagine that there is an implicit prepositional phrase "of America's Got Talent Season 11" modifying "the winner".

Another motivating example is the following:

Question: who sang the national anthem at the first game of 2017 world series

Passage: Game 1 of the 2017 World Series: The ceremonial first pitch was thrown out by members of former Dodger Jackie Robinson's family, including his widow Rachel. The game marked the 45th anniversary of Robinson's death. Keith Williams Jr., a gospel singer, performed "The Star-Spangled Banner", the national anthem.

In this case it is clear that the sentence "Keith Williams Jr., a gospel singer, performed "The Star-Spangled Banner", the national anthem" is referring to a performance at Game 1 of the 2017 World Series, and hence that this sentence provides an answer to the question. In some sense there is an implicit prepositional phrase "at the first game of 2017 world series" modifying the entire sentence.

Recall that the set of phrases within the sentence s was previously defined as $\mathcal{S} = \{(i, j) : s_0 \leq i \leq j \leq s_1\}$. We extend QED by redefining $\mathcal{S}$ to include implicit phrases introduced in the form of implicit prepositional phrases, as in the "winner [of ...]" and "[at the first game ...]" examples above. The modified definition of $\mathcal{S}$ includes all phrases of the following form: (1) Any pair (i, j) such that $s_0 \leq i \leq j \leq s_1$ indicating the subsequence of words $c_i \ldots c_j$ within the sentence. (2) Any triple (i, j, p) such that $s_0 \leq i \leq j \leq s_1$ and p is a preposition, indicating the implicit noun phrase in the

Question: where did they film and then there were none
Wikipedia page: And_Then_There_Were_None
Passage: Filming began in July 2015. Cornwall was used for many of the harbour and beach scenes, including Holywell Bay, Kynance Cove, and Mullion Cove. Harefield House in Hillingdon, outside London, served as the location for the island mansion. Production designer Sophie Beccher decorated the house in the style of 1930s designers like Syrie Maugham and Elsie de Wolfe. The below stairs and kitchen scenes were shot at Wrotham Park in Hertfordshire. Railway scenes were filmed at the South Devon Railway between Totnes and Buckfastleigh.

Figure 6.2: An example outside of QED's current scope, since multiple passage sentences contribute an answer.

sentence that modifies the phrase $c_i \ldots c_j$ through the preposition p. (3) Any pair (NULL, p) such that p is a preposition, indicating the implicit noun phrase modifying the entire sentence $c_{s_0} \ldots c_{s_1}$ through the preposition p.

6.4 QED Annotations for the Natural Questions

We now describe QED annotations over the Natural Questions (NQ) dataset [84]. We first describe the annotation process; then describe agreement statistics; finally we describe statistics of types of referential expression.

We focus on questions in the NQ corpus that have both a passage and short answer marked by the NQ annotator. We exclude examples where the passage is a table. A QED annotator was presented with a question/paragraph pair. In a first step they determine whether: (1) there is a valid short answer within the paragraph (note that they can overrule the original NQ judgment), and there is a valid QED explanation for that answer; (2) there is a valid short answer within the paragraph, but there is no valid QED explanation for that answer. (See Figure 6.2 for a representative example in this category, in which multiple sentences are required to justify an answer, thus violating the single-sentence assumption of QED); (3) there is no valid short answer within the passage (hence the original NQ annotation is judged to be an error). 10% of all examples fell into category (3). Of

Table 6.1 Referential link count frequency distribution in a random sample of 1000 instances.

	Referential Link Count			
	0	1	2	3
Instances	54	649	294	6

the remaining 90% of examples which contained a correct short answer, 77% fell into category (1), and 23% fell into category (2).

Three QED annotators[5] annotated 7638 training examples (5154/1702/782 in categories 1/2/3 respectively), and 1353 dev examples (1019/183/151 in categories 1/2/3).

6.4.1 Agreement Statistics

Each of the three annotators marked a common set of 100 examples drawn from the development set. Average accuracy of classification of instances was 73.9.%[6] Average pairwise F1 on mention identification/mention alignment, conditioned on both annotators labeling instances as amenable to QED, was 88.4 and 84.1 respectively.

6.4.2 Types of Referential Expressions

The referential equality annotations are a major component of QED. Figure 6.3 shows some full QED examples from the corpus, and Figure 6.4 shows some example equalities from the corpus. In this section, in an effort to gain insight about the types of phenomena present, we describe statistics on types of referential equalities. We subcategorize referring expressions into the following types:[7]

Proper Names Examples are “How I met your Mother” or “the cbs television sitcom how i met your mother”.

[5]Three of the authors of this paper.

[6]One annotator was more conservative interpreting the single sentence assumption. Pairwise accuracy breakdown was thus 81.2/72.3/68.1%. Given the high number of “debatable" instances reported in the Natural Questions paper, this divergence is however unsurprising.

[7]For formal discussion, see [91, 92, 93, 94] among others.

Pronominal reference

Question: how many blocks in the great pyramid of giza$_1$

Wikipedia page: Great_Pyramid_of_Giza

Passage: Based on these estimates, building the pyramid in 20 years would involve installing approximately 800 tonnes of stone every day. Additionally, since **it**$_1$ consists of an estimated 2.3 million$_A$ blocks, completing the building in 20 years would involve moving an average of more than 12 of the blocks into place each hour, day and night. The first precision measurements of the pyramid were made by Egyptologist Sir Flinders Petrie in 1880–82 and published as The Pyramids and Temples of Gizeh. Almost all reports are based on his measurements[...]

Inexact match

Question: where does **the term sixes and sevens**$_1$ originate

Wikipedia page: At_sixes_and_sevens

Passage: An ancient dispute between the Merchant Taylors and Skinners livery companies$_A$ is the probable origin of **the phrase**$_1$. The two trade associations, both founded in the same year (1327), argued over sixth place in the order of precedence. In 1484, after more than a century and a half of bickering, the Lord Mayor of London Sir Robert Billesden ruled that at the feast of Corpus Christi, the companies would swap between sixth and seventh place and feast in each other's halls[...]

Answer bridging/coref

Question: what is whitney houston's mother$_1$'s name

Wikipedia page: Cissy_Houston

Passage: **Emily "Cissy" Houston**$_A$ (née Drinkard; born September 30, 1933) is an American soul and gospel singer. After a successful career singing backup for such artists as Dionne Warwick, Elvis Presley and Aretha Franklin, Houston embarked on a solo career, winning two Grammy Awards for her work. **Houston** is the mother of singer Whitney Houston$_1$, grandmother of Whitney's daughter, Bobbi Kristina Brown, aunt of singers Dionne and Dee Dee Warwick, and a cousin of opera singer Leontyne Price.

Entity bridging

Question: who sang the national anthem$_1$ at **the first game of 2017 world series**$_2$

Wikipedia page: 2017_World_Series

Passage: **Game 1**: The ceremonial first pitch was thrown out by members of former Dodger Jackie Robinson's family, including his widow Rachel. The game marked the 45th anniversary of Robinson's death, and the 2017 season was the 70th anniversary of his breaking of the baseball color line. [. . .]$_2$ Keith Williams Jr.$_A$, a gospel singer, performed "The Star-Spangled Banner", the national anthem$_1$.

Generic reference

Question: what is the function of **a paints binder**$_1$

Wikipedia page: Paint

Passage: **The binder**$_1$ is the film-forming$_A$ component of paint. It is the only component that is always present among all the various types of formulations. Many binders are too thick to be applied and must be thinned. The type of thinner, if present, varies with the binder.

Figure 6.3: Examples from the QED dataset, grouped according to different types of referential equalities.

Non-Anaphoric Definite NPs These are expressions such as "the president of the United States" or "the next Maze Runner film". The majority involve one or more common nouns (e.g., "president", "film") together with a proper name, thereby defining a new entity that is in some sense a "derivative"

Question Expression	Passage Expression
how i.met your mother	the CBS television sitcom How I Met Your Mother
the most wins in the nfl	most wins
mantis	Mantis
the nashville sound	Countrypolitan - a smoother sound typified through the use of lush string arrangements with a real orchestra and often, background vocals provided by a choir
a permit driver	a driver operating with a learner 's permit
god's not dead a light in the darkness	it
the current president of un general assembly	the United Nations General Assembly President of its 72nd session beginning in September 2017
the new maze runner movie	Runner : The Death Cure
a box lacrosse team	a team

Figure 6.4: Referential equalities from the QED corpus.

Qu. \ Ps.	P	N	A	G	Pn	B	M	T
Proper	44	0	16	0	9	4	0	73
Def. (Non-Ana)	4	6	4	0	0	1	1	16
Def. (Ana)	0	1	1	0	0	0	0	2
Generic	0	0	0	6	0	0	0	6
Pronoun	0	0	0	0	0	0	0	0
Bridge	0	0	0	0	0	0	0	0
Misc	2	0	0	0	0	0	1	3
Total	50	7	21	6	9	5	2	100

Figure 6.5: Counts for 100 randomly drawn referential equality annotations from the QED corpus, subcategorized by expression type in the question (Qu.) and passage (Ps.). P/N/A/G/Pn/B/M refer to Proper/Def(non-ana)/Def(ana)/Generic/Pronoun/Bridge/Misc.

of the underlying proper name.

Anaphoric Definite NPs These are definite NPs, most often from within the passage rather than the question, that require context to be interpreted. Examples are "the series" referring to an earlier mention of "the Vampire Diaries" within the passage, or "the winner" referring to "the winner of America's got Talent Season 11".

Generics Examples are "a dead zone" in the question "what causes a dead zone in the ocean", or "Dead zones" in the passage sentence "Dead zones are low-oxygen areas caused by ...".

Pronouns Examples are it, they, he, she.

Bridging Referential expressions in the passage sentence that use bridging (see Section 6.3.3).

Miscellaneous All referential expressions not included in the categories above.

Table 6.1 shows the frequency distribution of per-instance referential equality counts. Figure 6.5 shows an analysis of 100 referential equality annotations from QED, with a breakdown by type of referring expression in the question and passage. Proper names, non-anaphoric definites, and generics dominate expression types in the question (73, 16, and 6 examples respectively). Expressions in the sentence are more diverse, with a much greater proportion of anaphoric definites, pronouns, and bridging examples (21, 9, and 5 cases respectively).

Finally, as an indication of the difficulty of the referential equality task, we note that in only 12% of all referential equalities in the 100 examples in Figure 6.5 is there an exact string match (after lower-casing of both question and passage) between the question and passage referential expression.

6.5 Tasks and Baseline Results

We release the QED dataset with the intention to spur research into QED-based tasks and models. In this section, we introduce four potential modeling tasks using the data and describe baseline approaches and results for the first two tasks.

6.5.1 Four Tasks

Each QED example is a (q, d, c, a, e) tuple where q is a question from the NQ corpus, d is a Wikipedia page, c is a long answer (typically a paragraph) within d, a is a short answer within c, and e is a QED explanation. We use $\mathcal{E}$ to refer to set of evaluation examples (either the development or test set).

Such data could potentially be used in many different ways. We highlight the following four tasks, in order of increasing complexity:

Task 1 Given a (q, d, c, a) 4-tuple, make a prediction $\hat{e} = f(q, d, c, a)$ where f is a function that maps a (q, d, c, a) tuple to an explanation. We might for example define $f(q, d, c, a) = \arg\max_e p(e|q, d, c, a; \theta)$ under some model $p(\ldots)$. The evaluation measure is then

$$\frac{1}{|\mathcal{E}|} \sum_{(q,d,c,a,e)\in\mathcal{E}} l_1(e, f(q, d, c, a))$$

where $l_1(e, \hat{e})$ is a per-example evaluation measure indicating how close $\hat{e}$ is to e.

Task 2 Given a (q, d, c) triple, predict $(\hat{a}, \hat{e}) = f(q, d, c)$, where f is a function that maps a (q, d, c) pair to a short-answer/explanation triple. We might for example define $f(q, d, c) = \arg\max_{a,e} p(a, e|q, d, c; \theta)$ under some model $p(\ldots)$. The evaluation measure is $\sum_{(q,d,c,a,e)\in\mathcal{E}} l_2((a, e), f(q, d, c))$ where l_2 is some per-example measure.

Task 3 Given a (q, d) pair, predict $(\hat{c}, \hat{a}, \hat{e}) = f(q, d)$, where f is a function that maps a (q, d) pair to a long-answer/short-answer/explanation triple. We might for example define $f(q, d) = \arg\max_{c,a,e} p(c, a, e|q, d; \theta)$ under some model $p(\ldots)$. The evaluation measure is $\sum_{(q,d,c,a,e)\in\mathcal{E}} l_3((c, a, e), f(q, d)$ where l_3 is some per-example measure.

Task 4 As in Task 3, given a (q, d) pair, predict $(\hat{c}, \hat{a}, \hat{e}) = f(q, d)$. One part of the evaluation is the same as in Task 3. But in addition, we require the explanations generated by $f(\ldots)$ to be *faithful* with respect to the reasoning process of the underlying model. This will require an evaluation measure for faithfulness, which is an open question beyond the scope of this paper.

Accurate models for Tasks 1, 2, and 3 even if they do not generate faithful explanations (Task 4), may still have considerable utility. However, faithful models have several desirable characteristics (see Section 6.7); we view them as a major avenue for future work.

In the remainder of this section we describe results for baseline models on Tasks 1 and 2. The intention here is to establish baseline results as a reference point for future work on QED models and to get an idea of tractability of recovery of QED annotations.

6.5.2 A Baseline Model for Task 1

Our baseline model for Task 1 is an extension of the recently proposed co-reference resolution model of [95] and [96]. We present two variations on the model, the first trained on co-reference data alone, the second trained on co-reference data with fine-tuning on QED annotations

The co-reference Resolution Model

We give a brief recap of the approach of [95] and [96]. Given some document d and a candidate mention x, corresponding to a span within d, define $\mathcal{Y}(x)$ to be the set of potential antecedents for x. Each antecedent is either a span in the document with start-point before x in the document, or ϵ signifying that x does not have an antecedent. We can then define a distribution over the antecedent spans $\mathcal{Y}(x)$ as $p(y|x, D) = \frac{e^{s(x,y)}}{\sum_{y' \in \mathcal{Y}(x)} e^{s(x,y')}}$ where

$$s(x, y) = \begin{cases} 0 & \text{if } y = \epsilon; \\ s_m(x) + s_m(y) + s_c(x, y) & \text{o.t.} \end{cases}$$

$$s_m(x) = \texttt{FFNN}_m(g_x)$$

$$s_c(x, y) = \texttt{FFNN}_c(g_x, g_y)$$

where g_x and g_y are span representations obtained by concatenating the SpanBERT representation of the first and last token in each mention span. The scoring functions s_m and s_c represent mention and joint span match scores respectively.

[96] describe a method for training the model based on log-likelihood, and a beam search method that uses the scores $s_m(\ldots)$ to filter mentions and antecedents. The final output from the model is a hard clustering of the potential mentions into co-reference clusters.

The Model Applied to Task 1

Assume an example contains a question q of m tokens $q_1 \ldots q_m$ and a passage c consisting of n tokens $c_1 \ldots c_n$. We denote the title of the Wikipedia page separately as the sequence t of k tokens

$t_1 \ldots t_k$. The model considers the concatenation of these token sequences,

$$\texttt{[CLS]}t_1 \ldots t_k \texttt{[S1]} q_1 \ldots q_m \texttt{[S2]} c_1 \ldots c_n \texttt{[SEP]},$$

as an input document.[8] The model is tasked with predicting the referential equality annotations $e = e_1 \ldots e_k$ in the QED annotation. We do assume that the NQ short answer is also an input to the model, used to restrict the position of referential equality annotations in the passage; we describe this restriction below.

QED referential equality annotations are of two types: (1) coreferential links between noun phrases in the question and in the passage, and (2) coreferential links between a noun phrase in the question and an implicit argument in the passage. We observe that many implicit arguments link to the title of the passage, so we model the latter annotation type as a coreferential link between the question mention and the title span $t_1 \ldots t_k$. In the untrained baseline, we restrict s_m to only score mentions in the sentence containing the answer. In both models we restrict s_c to only score coreferential links between the query and the passage or between the query and the title (all other values for s_m or s_c are set to $-\infty$).

We finally post-process the cluster outputs as follows: for each cluster we output the first cluster mention in the question paired with the first cluster mention in the passages. If there is no cluster mention in the passage, then we output the question mention paired with an implicit argument.

For the untrained baseline, we did not use expert annotated QED data but instead used the CoNLL OntoNotes co-reference dataset [97] to train the pretrained SpanBERT model. For the fine-tuned baseline, we further trained the model with the training portion of QED data converted into co-reference format. We used SpanBERT "large", with a maximum span width of 16 tokens, a top span ratio of 0.2, 30 max antecedents per mention. In fine-tuning, we used an initial learning rate of $3 \cdot 10^{-4}$ and trained for 3 epochs on the QED training set.

We evaluate both mention identification (the identification of individual referential expressions in the question and passage) and referential equality detection (the identification of pairs of referential

[8] We simply use [S1] = "." and [S2] = "?" as separators.

Table 6.2 SpanBERT model performance for Task 1: recovering QED annotations when the correct answer is given.

	Mention Identification			Mention Alignment		
	P	R	F1	P	R	F1
zero-shot	59.0	35.6	44.4	47.7	28.8	35.9
fine-tuned	76.8	68.8	72.6	68.4	61.3	64.6

expressions). We compute precision, recall, and F1 measure in both cases. Evaluation results are reported in Table 6.2. The table shows results for both the zero-shot model, trained on co-reference data alone, and a fine-tuned model, which is fine-tuned on QED annotations.[9]

6.5.3 A Baseline Model for Task 2

Our baseline model for Task 2 is a straightforward extension of the baseline model for Task 1. We build a model of the form

$$\begin{aligned} & p(a, e|q, d, c; \theta) \\ = \; & p^{(1)}(a|q, d, c; \theta^{(1)}) p^{(2)}(e|a, q, d, c; \theta^{(2)}) \end{aligned}$$

where $p^{(1)}$ is an existing QA model (similar to [8]), and $p^{(2)}$ is the baseline model for Task 1. Thus we simply compose an existing question-answering model with an answer agnostic model that recovers explanations.

The answer scoring component of the model computes answer candidate representations g_z in the same way as the Task 1 baseline computes mention representations. The score of an answer z is then computed as

$$s_a(z) = \texttt{FFNN}_a(g_z).$$

Mention representations are shared between $p^{(1)}$ and $p^{(2)}$, so the only new parameters belong to

[9] Official evaluation code will be released with the dataset.

Table 6.3 SpanBERT model performance for Task 2: recovering answer and QED annotations given a passage that is known to contain the answer.

	Mention Identification			Mention Alignment			Answer Accuracy
	P	R	F1	P	R	F1	
QED-only	74.1	63.8	68.6	63.6	54.9	58.9	-
QA-only	-	-	-	-	-	-	73.4
QA+QED	77.5	64.6	70.5	68.6	57.3	62.4	74.5

a single hidden layer feed-forward net $\texttt{FFNN}_a$ that computes the answer score for each mention. No further dependence is introduced between the answer and explanation predictions. We train $p^{(1)}$ and $p^{(2)}$ in a multitask fashion, by minimizig the weighted sum of the question answering and co-reference cross entropy losses. Our best results are obtained with a weight of 5 on the co-reference loss and 2 epochs of training. The best answer accuracy and QED F1 are obtained for different base learning rates of $2 \cdot 10^{-5}$ and $5 \cdot 10^{-5}$ respectively.

Results

In Table 6.3 we report results for Task 2 for three separate variations of the approach described in the previous section. QED-only fine-tunes $p^{(2)}$ on the QED training set only. QA-only fine-tunes $p^{(1)}$ on all the paragraphs of the NQ dataset that contain a short answer. QA+QED fine-tunes both $p^{(1)}$ and $p^{(2)}$ on all NQ and QED data. We obtain the encouraging result that both QA and QED metrics improve significantly in the final multitask setting, despite the fact that the QED training data (5154 examples) amounts to only 6% of the data available for QA (91632 examples).

6.6 Rater Study

A system which makes use of QED explanations to answer a question is one which decomposes its reasoning process into human-interpretable chunks. We hypothesize that exposing QED explanations should improve a user's ability to spot errors made by an automated QA system. To this end, we evaluate QED explanations using a rater study.

6.6.1 Task Setup

Given a question, passage, and a candidate answer span, raters were tasked with assessing whether the candidate answer was correct or incorrect, and indicating the confidence of their assessment.

We obtained the data for the study by taking a random set of 50 correct answers and 50 incorrect guesses from the NQ baseline model on the Natural Questions dev set. So as to ensure that the task was sufficiently challenging, correct instances were the *gold* answer spans on question/passage pairs where the model produced a false negative.[10] Incorrect instances were false positive guesses from the model.

A total of 354 raters, all of whom are US-residents and native English speakers, were divided into three disjoint pools to perform the task in three distinct test settings: The **None** group of raters (n=121) was presented with a question, passage, and a highlighted answer span. The **Sentence** group (n=117) was provided with additional highlighting of the sentence containing the answer, with no distinction made between referential equalities and predicates. The **QED** group (n=116) was provided with additional highlighting to indicate referential equalities between spans in the question and spans in the passage. On average, a given rater provided judgments for 41 questions.

In each case, raters were told that highlighting was the output of "an automated question answering system" that was incorrect "about half of the time." Where explanations were present, they were manually imputed to simulate the inferences of a hypothetical model that used a QED-style reasoning process. Additionally, raters were told that the system made use of the highlighted information to produce its candidate answers.

6.6.2 Results

Average rater accuracies for each test setting are presented in Table 6.4. We see that, in aggregate, QED explanations improved accuracy on the task over and above the other test settings, and gave the most improvement on the identification of answers that were incorrect. These improvements

[10] That is, where an answer existed in the passage, but the model was not confident about it.

Table 6.4 Rater study results. Corr and Incorr are accuracies of raters in each group on correct and incorrect instances respectively, with incorrect instances further broken into Pred(icate) and Ref(erence) model errors. F1 is on the task of identifying incorrect instances.

	Accuracy			F1
	All	Corr	Incorr/Pred/Ref	Incorr
None	67.5	90.4	44.3/43.9/44.7	57.6
Sentence	69.7	**92.4**	47.1/46.1/48.0	60.9
QED	**70.2**	90.6	**49.7/48.2/51.0**	**62.5**

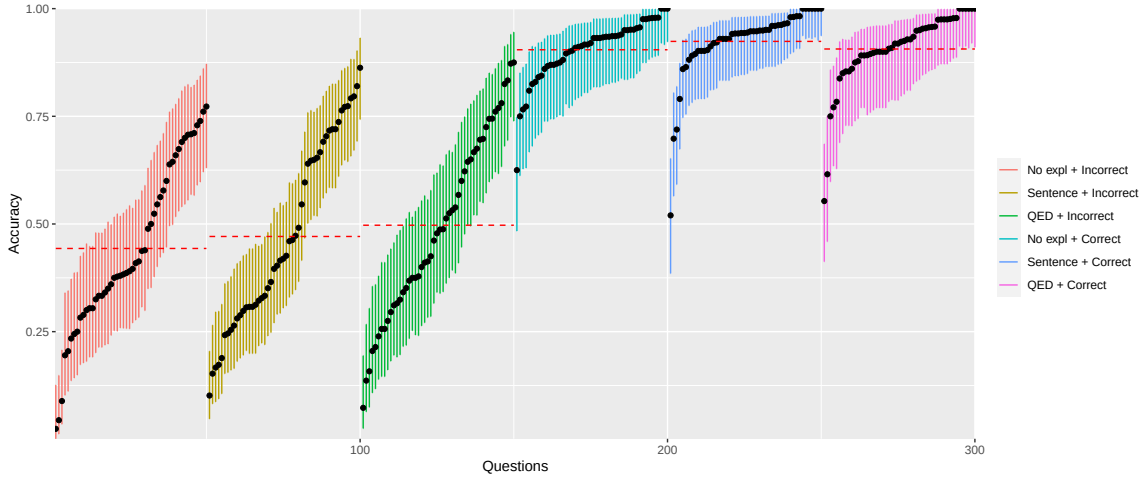


Figure 6.6: Sorted, per-question evaluation accuracies from different rater study settings, with 95% binomial confidence intervals. Left three plots correspond to trials with incorrect answers highlighted; right three plots to trials with correct answers highlighted. Dashed red lines correspond to the average accuracy for each setting, identical to the numbers in Table 6.4.

translate to incorrect answers resulting from both predicate and reference model errors.

Somewhat surprisingly, highlighting just the sentence containing the answer improved accuracy more than including referential equality highlighting on instances that were correct. This is likely because raters' propensity to mark instances correct decreases as the complexity of explanations increases, from None (73.1%) to Sentence (72.6%) to QED (70.5%).

Also clear from Table 6.4 is that rater accuracy is much lower on incorrect instances. Even though raters were told that the answers presented were incorrect half of the time, they marked the model guess as correct roughly 71% of the time.[11]

[11]While this confirmation bias presents an interesting challenge for future work, it is not a shortcoming of our results: Raters were not trained to do well on the task, as we aimed to approximate how users interact with automated QA systems.

Table 6.5 Generalized linear mixed model fixed effect coefficients, showing mean and standard deviation of 10k MCMC samples. The Intercept corresponds to the Incorrect+None setting.

Parameter	Coefficient (SD)
(Intercept)	-0.31 (0.15)
+ Incorrect+Sentence	0.15 (0.11)
+ Incorrect+QED	0.25 (0.11)
+ Correct+None	2.94 (0.21)
+ Correct+Sentence	3.04 (0.13)
+ Correct+QED	2.69 (0.13)

Figure 6.6 provides another perspective on the disparity in judgments on correct/incorrect instances summarized in Table 6.4. The instances receiving the highest accuracy in the incorrect pool are harder for raters on average than most of the correct instances, and the lowest accuracy on incorrect instances is far lower than that of any of the correct instances. The wide distribution of accuracies on incorrect instances ($\sigma \approx 0.50$) seen in Figure 6.6 was also reflected in the rater pool ($\sigma \approx 0.45$). The challenging nature of incorrect instances speaks to the promise of improvements from QED explanations.

6.6.3 Effectiveness of explanations

How statistically significant are the results reported in Table 6.4? The 14,115 test instances were spread across 354 raters and 100 questions. To control for the correlations induced by the rater and question groups, we fit a generalized linear mixed model (GLMM) using the `rstanarm` R package [98]. We used the formula `a ~ c * e + (1|r) + (1|q)`, where `a` is whether or not the rater accurately marked the instance; `c` is whether the instance was Correct or Incorrect; `e` is the explanation test setting of None, Sentence, or QED; `r` is the rater id; and `q` is the question id. This formula specifies a regression of the log-odds of the rater accuracy on the fixed effects of instance correctness (`c`) and explanation setting (`e`), while allowing for random effects in the raters (`r`) and questions (`q`). Ultimately we are interested in the magnitude and statistical properties of `e` under the various test settings.

Table 6.5 shows the fixed effect coefficient and standard deviations for each setting. The presence

of QED explanations in the Incorrect setting increased the log-odds of rater accuracy by 0.25, with a posterior predictive p-value of 0.015 that this effect is greater than zero. The comparable effect for Sentence explanations was 0.15, with a posterior predictive p-value of 0.08. The rater and question random effects had standard deviations of 0.63 and 0.90 respectively, reflecting again the high variance of questions shown in Figure 6.6.

As we saw earlier, the effects of explanations in the Correct setting was reversed: the Sentence explanations caused a small, statistically insignificant increase in log-odds, while QED explanations caused a statistically significant drop in log-odds.

6.7 Discussion

6.7.1 QED and strong explainability

It is an open question as to what constitutes a good explanation [99]. A major inflection point in the discussion is the notion of *faithfulness* [100, 101]. We say a model's explanations are faithful when there is a causal relationship between an explanation and a prediction. That is, when an explanation changes, the outputs change accordingly. When this is not true, we say a model generates *rationales*, which have the appearance of justifying its outputs, but without causal guarantees [85].

While the models described in Section 6.5 fall into the latter category, we believe QED is a promising framework for strongly explainable QA. This is due in large part to its commitment to the cognitive reality of reference and entailment. We can say, definitively, that in order for a sentence to answer a question about a thing, its meaning must involve that thing in a very particular sense. Posed counterfactually, when you break referential equality, you break answerhood, and the same argument follows for predicate entailment. Unlike other intelligent behavior that may permit of post-hoc rationalization at best [83], certain forms of high-level linguistic reasoning are in fact amenable to strong explanation.

6.7.2 Potential Extensions to the QED Framework

QED exists in between relatively unstructured explanation forms on the one hand, such as attention distributions [102, 103, 104] or sequential outputs [105, 106, 107, 108] and more elaborate, discrete semantic representations that can in theory be applied to explainable QA [109, 110].

The version of QED presented here is a broad coverage, yet limited instantiation of a framework, in which explanations are semantic relations whose substructures are defined in terms of formally motivated linguistic categories. However, in keeping with its modularity, we can extend QED to account for these by looking to semantic relations beyond referential equality and predicate entailment, such as set-membership noun phrase [111] and interclausal [112, 113, 114] relations.

6.7.3 Future uses of QED representations

Our hope is that QED representations may be useful in a variety of extensions to extant QA systems. Some examples are as follows:

Ambiguous Questions. Consider again the question in Figure 6.1, "who wrote the film howl's moving castle". Now consider the question "who wrote howl's moving castle". In this case there are two possible answers, depending on whether the author of the question is referring to the film or novel. It would be natural for a system to provide two possible answers [see, e.g. 115], with two possible QED explanations highlighting the differing assumptions underlying each answer. Such referential ambiguities are common, and the centrality of referential equality in QED annotations should mean that they are useful in this scenario.

Complex Referential Equalities. Consider the question "meaning of whiskey in the jar by metallica". The Wikipedia page for "Whiskey in the Jar" says the following:

Passage: "Whiskey in the Jar" is an Irish traditional song set in the southern mountains of Ireland. The song, about a rapparee (highwayman) who is betrayed by his wife or lover, is one of the most widely performed traditional Irish songs and has been recorded by numerous artists since the 1950s.

A good answer could be that the song is "about a rapparee . . . who is betrayed by his wife or lover", assuming that the Metallica song is a variant of the Irish traditional song. Thus the validity of this answer hinges on a complex referential equality, between the Metallica version and the original. Examples that require this type of complex referential reasoning are quite common, and the centrality of reference in QED should be relevant.

6.8 Conclusions

We have described QED, a framework for explanations in question answering, and we have introduced a corpus of QED annotations. The framework is grounded in referential equality, and entailment. In addition we have described baseline models for two QED-based tasks, and a rater study utilizing QED annotations.

Future work should consider the development of models that provide faithful explanations based on QED; extensions of QED, for example to handle multi-sentence inference or referential phenomena going beyond equality; and applications of QED, for example to sentences with multiple potential answers, to questions that are vague or underspecified, or to referential equalities that require significant inference to be justified.

Chapter 7: A Semantic Analysis of Negative Why Questions in the Natural Questions Dataset

In this chapter we temporarily depart from question answering modeling with pretrained transformers, which is the topic of all the remaining chapters in this thesis, and we attempt a particular semantic investigation of the data of the Natural Questions dataset. We look at the specific subset of why-questions, and we make the interesting discovery that there are no attested cases of why-questions containing a negation with an answer that describes a purpose rather than a reason. We discovered in existing literature a proposed syntax-based explanation for this phenomenon, but we find examples in the Natural Questions that invalidate that explanation. We therefore put forward a new semantics-based explanation for the absence of purpose answers to negative why-questions.

7.1 Overview

Several authors have noted that there are two fundamental types of answers to *why*-questions: *reason* and *purpose*. They can be easily exemplified as follows:

- Why did John tear down the wall?

  Reason: because Mary asked him to.

  Purpose: to show off.

Reasons are past-oriented answers to why-questions. They are often introduced by the conjunction *because*, and provide a proposition that constitutes a cause for the *explicandum*, i.e. the proposition modified by *why* in the question. On the other hand, purposes are future-oriented answers to why-questions. They are introduced by the preposition *to*, and provide an *intentional* result of the explicandum.

This brief analysis of why-questions is a good starting point, but many issues remain to be explored. Are there types of answers to why-questions other than reason and purpose? Is the presence of *to* or *because* as a definition of purpose and reason consistent with our intuition? Can something be said about which types of explanation are admissible by which types of why-questions? In this work we explore these questions empirically using a dataset of question-answer pairs ([7]) and attempt to provide a formal framework that systematizes our empirical observations.

We start our investigation in section 7.2, by introducing a new notion of purpose and reason which can be more generally used in empirical judgements. In section 7.3, where we look at a syntactic account of why-questions and analyze a series of interesting predictions due to [13]. We find that many of the predictions, though plausible, are in disagreement with empirical data. In section 7.4, we focus on a particular prediction about negative why-questions from the same syntactic account, which instead has a surprisingly high level of empirical support. In section 7.5, we propose a formal framework for the analysis of why-questions based on the ideas described by [116] and guided by our empirical observations. This framework will provide a justification for the success of the prediction in section 7.4. Finally in section 7.6, we show how our framework naturally makes room for additional types of answers to why-questions, showing that connections are possible with taxonomies of explanations.

7.2 Reason and Purpose

While answers starting with *because* and *to* seem to closely reflect the intuitive idea of reason and purpose respectively, we cannot take them to be a definition of these answer types. One practical reason for this is that many answers found empirically do not start with either of these words and so will require more flexible classification criteria. Another reason stems from the observation that it would nearly always be possible to convert a purpose into a reason with a minimal modification to the meaning:

- Why did John tear down the wall?

  Purpose: To show off.

~~Reason~~ *Still purpose*: Because he wanted to show off.

This simple operation is almost always possible in practice: [1]

- ‡Why did the United States declare war on Spain?

  Purpose: To help Cuba gain independence.

  Still purpose: [Because the US wanted t]o help Cuba gain independence.

- ‡Why did Chandler Massey leave days of our lives?

  Purpose: To return to school.

  Still purpose: [Because he wanted t]o return to school.

This empirical preponderance of examples that admit this rephrasing suggests that the compatibility with the *bouletic* modality (*want*) might be a much better definition of purpose. To capture this fact, we propose the following operational definitions, which will serve us well for most of this work. In the context of a why-question:

- 1. a *purpose* is an answer that expresses a want.

  2. a *reason* is an answer that is not a purpose.

The conjunction *because* and the preposition *to* continue to be strong features to identify reasons and purposes. Strictly speaking however we have to apply the definition above to determine the type of each explanation, even if they start with *to* or *because*.

Future vs. Past. An alternative definition of reason and purpose could have been given in terms of whether an explanation is past-oriented or future-oriented respectively. We could imagine a formalization such as the following:

- Why x?

  Reason: Because y. (if $y \implies x$)

  ?*Purpose*: To y. (if $x \implies y$)

[1] We always indicate with ‡ data derived from [7].

While we do not see anything fundamentally wrong with the the above definition of reason, we note that the definition of purpose is incomplete. It is not enough for y to be entailed by x: a desire for y has to also be true in order to obtain a purpose, and such desire would have to logically or temporally precede x. Therefore we would still have to include some component related to the bouletic modality, even if we decided to adopt a future vs. past definition.

7.3 The Syntactic View

In a traditional interrogative semantics approach, such as [117], the meaning of *why* questions might be represented as the following set of propositions:

- $[\![\text{Why did John tear down the wall?}]\!]^w$
  $= \left\{\text{that John tore down the wall because } x : x \in D_{st}\right\}$
  $\cup\left\{\text{that John tore down the wall to } x : x \in D_{st}\right\}$

As detailed by [13] (henceforth C&K), reason why and purpose why have distinct attachment positions in the syntactic structure of the question. Purpose why is base-generated within CP (the *higher* why), while purpose why is adjoined to vP (the *lower* why). C&K predict that only dynamic predicates with agentive subjects have a syntactic structure with both these attachment sites available, and so only questions with those predicates will be able to support both reason and purpose answers. They base this prediction on the fact that only dynamic predicates are propositionally complex, a finding due to [118]. We find however that why-questions are much more flexible than what this account suggests and provide counterexamples for several of the predictions made by C&K.

A first prediction is that passive why-questions only admit reasons as answers because the agent is implicit. C&K present the following example:

- Why was that student cheated out of a grade?

  # *Purpose*: To help him get a better grade on the final.

  Reason: Because the instructor felt like it.

C&K explicitly disagree in their judgement with [119], who give the opposite prediction: that passive why-questions only admit purposes and not reasons. The example due to [119] is as follows:

- Why was the ship sunk?

  Purpose: In order to collect the insurance money.

  # *Reason*: Because they wanted to collect the insurance money.

Notably the reason answer in the last example would be classified as a purpose according to our proposed definition. The fact that this example was likely intended to be a minimal pair exemplifies how natural it seems to be to impute an implicit bouletic modality in most purpose answers.

Empirically this issue is of simple resolution: both types of answers are attested. Example 7.3 shows that it is easily possible to obtain a purpose answer even though the agent is implicit. Example 7.3 shows that a reason answer is also obtainable.

- ‡ Why were Luke and Leia separated at birth?

  Purpose: To keep them hidden from Darth Vader.

- ‡ Why was the constitution of India adopted on 26 Jan 1950?

  Reason: Because it was on this day in 1930 when the Declaration of Indian Independence (Purna Swaraj) was proclaimed by the Indian National Congress.

A second prediction by C&K is that locative-existential why-questions prefer reason readings. C&K present the following example, with the purpose answer marked with (??) rather than (#) to indicate that the answer is questionable rather than conclusively infelicitous.

- Why are there three engineers on the NSF linguistics committee?

  ?? *Purpose*: To block funding of theoretical research.

  Reason: Because they were asked to help out.

Again however we are easily able to find counterexamples empirically. Purpose answers seem to be obtainable despite the absence of an explicit agent in either the question or the answer, and despite the predicate in the question not being propositionally complex.

- ‡ Why is the Angel of the North there?

  Purpose: To serve as a focus for our evolving hopes and fears.

- ‡ Why is there a ball in Guinness beer?

  Purpose: To manage the characteristics of the beer's head.

A prediction that seems more difficult to refute at first is the one about un-accusative why-questions, which C&K also predict to be only answerable with purposes. The example provided by C&K is the following:

- Why did the butter melt?

  # *Purpose*: To make more room in our fridge.

  Reason: Because I left it out in the sun.

One might initially think that metaphorical animacy would have to be ascribed to the butter to obtain a barely acceptable purpose answer. For example we might imagine a children's story where food ingredients in a kitchen are magically collaborating on Saturday morning so that a child might wake up to a delicious breakfast.

- ?? Even the stubborn butter helped by melting on the pancakes to help them taste better.

We must admit that this example is indeed quite far-fetched. The data however provides us with a much more compelling example:

- ‡Why does cooling water run through the condenser?

  Purpose: To condense the steam coming out of the cylinders or turbines.

Examples 7.3, 7.3 and 7.3 also rule out a prediction that one might be tempted to make: that an animate agent is in some way required in the question for a purpose answer to be obtainable. What does seem to be true however is that an animate subject will be required in the answer if we rephrase it as a want as described in our operative definition of purpose.

An interesting type of why-question that C&K do not make predictions about is the case of bi-clausal why-questions. For example

- Why did you ask her to resign?

  Short construal: why modifies *ask*.

  Long construal: why modifies *resign*.

C&K note that interestingly the issue of bi-clausal why-questions has only ever been analyzed with reason answers in mind. Unfortunately the vast majority of why-questions in our dataset appear to be mono-clausal. We found only two bi-clausal examples with the verb *ask* and both of them had reason answers.

7.4 Negative why-questions

Hope might be running low at this point about the possibility of predicting possible answer types of why-questions. Perhaps why-questions are so flexible and open ended that no strong prediction is possible about their answer types, if a sufficiently complex world is allowed. It was quite a pleasant surprise for us to discover that one of the predictions made by C&K held with surprising strength: that purpose why is sensitive to negation. C&K justify their prediction as a weak island effect. Their proposal is that negation attaches in the syntax below reason-why (introduced by the conjunction *because*), but above purpose-why (introduced by the preposition *to*), and therefore wh-movement only encounters negation as an intervener in the case of purpose.

While this syntactic explanation might be true, we believe that further analysis is required for at least two reasons. First, we observe that it is not only answers starting with the preposition *to* that are missing from negative why-questions in our data. Purposes are missing even when our more general definition of purpose as an answer expressing a want is adopted. If the problem with purpose answers was purely syntactic, then we would expect to find other ways of expressing purpose readings to get around the limitations of syntax. Instead what we find in the data is that negative questions actually block the purpose meaning entirely, regardless of the syntax in which it might be expressed. This suggest that purposes are rejected not only based on syntactic constraints, but that there is a more general semantic or pragmatic reason for the this phenomenon.

Secondly, it does not seem particularly difficult, upon some reflection, to *make up* purpose

Table 7.1 Question counts and why-question counts by answer type and presence of a negation in [7].

	Example count
Total question with answers	100,000
why-questions	1,220
with a purpose answer	~ 250
with a *not*	52
with a *not* and a purpose answer	0

answers to why-questions, particularly if the purpose expresses the desire to avoid or prevent something from happening. For example:

- Why did Mary not shake John's hand?

  Purpose: To avoid spreading the flu.

For some non-trivial reason, the examples that we make up do not give us the correct intuition about the most likely answer types, perhaps because we neglect to construct a realistic context where the imaginary exchange is taking place. This suggests that purposes are actually syntactically admissible, but that some other mechanism is systematically discarding them as possible answers in practice.

The results of our empirical analysis are summarized in Table 7.1. The fraction of purpose answers was estimated by manually annotating 100 randomly selected why-questions, where we found exactly 20 answers of type purpose. We found that while the incidence of purpose answer types in why-questions is generally on the order of 20%, there were no purpose answers at all in the set of 52 why-questions containing negation. The sensitivity of why-questions to negation in this dataset is thus very statistically significant.

Since the syntactic account seems unreliable in our setting, we turn to the data for some insight on the causes of this phenomenon. We found it to be particularly useful for our intuition to look at near misses, i.e. answers to negative why-questions that look suspiciously close to purposes, but do not actually fit our definition. The closest example in the data to a negative why-question with a purpose answer is the following:

- Why did the East Side not let people go up to the Berlin Wall?

#*Purpose:* The wall served to prevent the massive emigration and defection that had marked East Germany and the communist Eastern Bloc during the post-World War II period.

It is alas incorrect. The answer describes the purpose of the Berlin Wall, not the purpose that East Germany had for not allowing people close to the wall. A possible purpose reading for this question could have been the following:

- Why did the East Side not let people go up to the Berlin Wall?

  ?*Purpose:* To prevent people from scaling the wall.

Again we notice that it is quite natural to come up with a purpose answer to a negative why-question using a verb like *avoid* or *prevent*. What could be wrong in this purpose answer though? It seems that this answer could have been just as good if the question had been about the West Side of the Berlin Wall rather than the East Side. A much more reasonable answer would have been to explain that people on the East Side were much more likely than those on the West Side to try to scale the Berlin Wall. This however would have been a reason and not a purpose.

In our judgement, this seems to be an utterly common pattern in negative why-questions: *a purpose answer is possible, but the answerer chooses to instead to give a more in depth reason for why that purpose came into being or became relevant*. We will attempt to formalize this notion in section 7.5.

Other examples that could be easily modified into having a purpose answer, but instead provide a reason answer are the following:

- ‡Why do we not eat egg with a silver spoon?

  Reason: Because the sulfur in eggs causes silver to tarnish.

  ?*Purpose:* To avoid tarnishing silver.

- ‡Why do Jehovah's Witnesses not celebrate birthdays or Christmas?

  Reason: Because they believe that these continue to involve "false religious beliefs or activities."

  ?*Purpose:* To avoid practicing false religious beliefs or activities

- ‡Why does john a. powell not capitalize his name?

  Reason: Based on the idea that we should be "part of the universe, not over it, as capitals signify."

  ?*Purpose:* To promote the idea that we are "part of the universe, not over it, as capitals signify."

In example 7.4 the attested reason provides a fact that causes a presumably preexisting desire "to avoid tarnishing silver" to become relevant as a purpose. In examples 7.4 and 7.4 the answer provides as reason a belief or an idea that justifies the purpose that we list as potentially admissible.

The main intuitions that we are attempting to convey with these examples are two: First, that potentially admissible purpose answers to negative why-questions are often given in terms of avoiding or preventing some effect from happening. Second, that this purpose is systematically overridden by a reason that makes some *presupposed* purpose relevant.

7.5 A Modal Framework for why-questions

Given the evidence in section 7.2, we are convinced that an account of reason and purpose cannot escape having some notion of modality. We therefore decided to give a representation of why-questions and their answers in the classical modality framework described by [116].

In this view, we assume that the questioner and answerer both have a *world view* represented by two conversational contexts: a modal base and a bouletic ordering source. Both conversational contexts are represented as sets of propositions. The modal base contains hard requirements or known facts, that cannot be violated or contradicted in any possible world. Ordering sources generally contain preferences, desires, or other rules that can violated without making a world impossible, but just less likely. The bouletic ordering source in particular contains only desires.

We propose a model of why-question-answering exchanges where the questioner expresses curiosity about a proposition, and the answerer replies with a proposition from one of their conversational contexts. Note that, if a successful why-question-answering exchange has taken place, we can safely assume that the views of the world of questioner and answerer were different before

the exchange and have become more similar after. Otherwise the answer must have contained a proposition that the questioner already knew and so no useful information has been conveyed.

We formally define our model as follows:

- A why-question is a request for an addition to a conversational context.
- Reasons are additions to the modal base s.t.:

  1. Why x? Because y.

     $\wedge$[modal base] $\not\Longrightarrow$ x

     $\wedge$[$\{y\} \cup$ modal base] $\Longrightarrow$ x

- Purposes are additions to the bouletic ordering source s.t.:

  1. Why x? To y.

     $\wedge$[modal base] $\not\Longrightarrow$ y

     $\wedge$[$\{x\} \cup$ modal base] $\Longrightarrow$ y

     y is not already likely (todo: write this formally)

- Negative why-questions presuppose the bouletic ordering sources of the questioner and the answerer are the same. (Maybe it is enough to presuppose that if the question-answer pair is "Why $!x$? To y" then x is presupposed to be likely, and y is a proposition entailed by $!x$ that can be made less likely.)

In rule 1, note that adding y to the bouletic ordering source will necessarily make x more likely, because the set of worlds where x is true is a subset of the accessible worlds where y is true. In rule 7.5, the informal interpretation is that we assume a presupposition that the bouletic ordering source was previously worked out by the questioner, and so some condition outside of it must have changed.

To see how this representation works in practice, consider the following toy example as the world view of the questioner:

- Modal base:

    1. If Mary asks him, John will tear down the wall.
    2. If John tears down the wall, he is showing off.
    3. John doesn't have time to tear down the wall and also play video games.

- Bouletic ordering source:

    1. John wants to play video games.

We start our analysis by noting that, in the toy world as it stands, John is likely to play video games and not tear down the wall. What happens to this world view if a successful why-question-answering exchange takes place?

- *Positive question:* Why did John tear down the wall?
  Reason: Because Mary asked him to.

In this case the answer provides an addition to the modal base. Initially the proposition x = "John tore down the wall" was possible but unlikely. After the addition of y = "Mary asked John to tear down the wall" to the modal base, the proposition x is guaranteed to be true and the ordering source has become irrelevant.

- *Positive question:* Why did John tear down the wall?
  Purpose: To show off.

In this case the answer provides an addition to the bouletic ordering source. The proposition x = "John tore down the wall" was unlikely before the addition. After the addition of y = "John wants to show off." has been added to the ordering source, and so x has become more likely, specifically just as likely as "John played video games." The reason x has become more likely is that any world where "John tore down the wall" also must have "John showed off" (by 2) and so if "John showed off" has become likely, then "John tore down the wall" has become more likely as well.

- *Negative question:* Why did John **not** play video games?

  Reason: Because Mary asked him to tear down the wall.

This case works in essentially the same way as example 7.5. The bouletic ordering constraint introduced by the negation is not relevant since the answer only affects the modal base.

- *Negative question:* Why did John **not** play video games?

  #*Purpose:* To show off.

  Reason: Because he suddenly felt like he wanted to show off.

Our modeling of negations is critical in this case: we know that the questioner the bouletic ordering sources of questioner and answerer are identical. As a result, an answer that directly adds to the ordering source would not be as desirable for the questioner as a reply that gives an addition to the modal base that in turn affects the composition of the bouletic ordering source or the relevance of existing rules within it.

Why do negative why-questions contain this presupposition? In an ordinary, information seeking intent, a questioner is unlikely to go through the trouble of posing their question in negative form. They have observed that a certain proposition is true and they are simply seeking to increase its likelihood in their world view. If the questioner chooses to add a negation, it must be because they have a degree of surprisal associated with discovering that a certain proposition that they thought was likely is actually not likely. This therefore presupposes that the likelihood of that proposition had already been computed by the questioner.

Why does the surprisal presupposition not cover the modal base? Essentially, it seems to fit the empirical data well and it is a simple constraint to add to our model of why-questions. However if a surprisal presupposition is to be made, then the bouletic ordering source is the only place where it makes sense with the data.

7.6 Other Types of Answers to why-questions

Another way of viewing answers to why-questions is that they are linguistic counterparts of explanations. However, there seems to be substantial misalignment between our two-way reason and purpose classification, and the much richer taxonomies of explanations provided by philosophers. As a notable example, [99] provides us with five types of explanations:

- 1. Reason ('Why P?' = 'Why should I believe that P?')
  2. Familiarity (reduction to the familiar)
  3. Unification (special case of a general pattern)
  4. Necessity (the event in question *had* to occur)
  5. Causation (the answer is actually a cause)

It is difficult to make a connection, but one first observation to make is that in these explanation types, the modality appears in the why-question and not in the answer. For example for the *reason* explanation type (in Lipton's terminology), we might be able to examine examples such as the following:

- Why is the butler the killer?

  They found his fingerprints on the murder weapon.

This can be related to the *doxastic* modality, where, however, it is the question that presents a proposition from the ordering source, rather than the answer.

Similarly an example of *necessity* could be the following:

- Why do engines need to be broken in?

  To produce the last small bit of size and shape adjustment that will settle them into a stable relationship for the rest of their working life.

- Why were the Athenians required to send fourteen sacrificial maidens and young men to king Minos?

  To be devoured in retribution for the death of Minos' son Androgeos.

Again the fact that this question is a necessity is determined by the presence of *need* and *require* in the question rather than in the answer.

In connection to unification and familiarity, potentially related to the stereotypical modality, there might be a promising connection to be made with answers to why-questions introduced by conjunctions or prepositions other than *to* and *because*. We found a small but interesting set of why-questions whose answers start with *as*, *in*, and *by*:

- Why did people collect the blood of king Charles I?

  As a memento.

- Why was alchemy important to the development of chemistry as a science?

  By performing experiments and recording the results.

- Why do we have a big feast on Christmas day?

  In the tradition of the Christian feast day.

While it might be possible to coerce the answer types in these examples as *purpose* or *reason*, it might be more natural to think of the propositions in the answers as general rules for which the proposition in the question is a special case.

7.7 Conclusion

In this work, we find that several predictions based on syntactic accounts of why-questions do not seem to hold in a real world dataset. Surprisingly, one specific prediction about the incompatibility of purpose answers with negative why-questions holds with a surprisingly high level of statistical significance.

We argue that the syntactic justification for this prediction is not satisfactory and we propose a new analysis based on modalities where why-questions are defined as requests for additions to conversational contexts. Within this analysis we show that reasons and purposes can be distinguished based on the type of conversational context they are added to. We argue that purposes are incompatible with why-questions because of a presupposition on the ordering source having already

been worked out. Finally we show how some other known types of explanations might within reach for this analysis by looking at the type of the ordering source that the proposition in the question belongs to.

Chapter 8: Towards Computationally Verifiable Semantic Grounding For Language Models

The chapter presents an approach to semantic grounding of language models (LMs) that conceptualizes the LM as a conditional model generating text given a desired semantic message formalized as a set of entity-relationship triples. It embeds the LM in an auto-encoder by feeding its output to a semantic parser whose output is in the same representation domain as the input message. Compared to a baseline that generates text using greedy search, we demonstrate two techniques that improve the fluency and semantic accuracy of the generated text. The first technique samples multiple candidate text sequences from which the semantic parser chooses. The second trains the language model while keeping the semantic parser frozen to improve the semantic accuracy of the auto-encoder. We carry out experiments on the English WebNLG 3.0 data set, using BLEU to measure the fluency of generated text and standard parsing metrics to measure semantic accuracy. We show that our proposed approaches significantly improve on the greedy search baseline. Human evaluation corroborates the results of the automatic evaluation experiments.

8.1 Overview

A statistical language model (LM) in its standard formulation assigns a probability to sequences of tokens that constitute an (ideally lossless) representation of text units such as sentences, paragraphs or larger. The original use for LMs was as prior probability in source-channel models for automatic speech recognition, machine translation and other similar tasks. Such use cases for LMs have largely disappeared with the shift to seq2seq models [120] as conditional LMs (direct models) for text given speech, or text in a foreign language, or other topics.

Very large LMs trained on huge amounts of text have demonstrated bewildering capabilities in

dealing with a large array of natural language processing and understanding tasks, e.g. GPT-3 [121], PaLM [122], Gopher [123], or Lamda [124]. Displaying nearly flawless fluency, the text sampled from LMs turns out to also be semantically adequate to an ad-hoc prompt setup in one- or a few-"shot" manner on a wide range of NLP tasks [121]. This is surprising since the LM is trained strictly as a generative model without any specific context other than the preceding text; as pointed out by many, the text sampled from LMs trained on large amounts of surface text cannot be expected to come with any weak or strong guarantees in terms of semantic adequacy, e.g. [125].

In this work, we approach semantic grounding of language models (LMs) by conceptualizing the LM as a conditional model generating text given a desired semantic message formalized as a set of entity-relationship triples. The LM is embedded in a semantic auto-encoder by feeding its output to a semantic parser whose output is in the same representation domain as the input message.

We propose two techniques for improving the semantic adequacy of the text generated by the LM. We find that both approaches improve the semantic $F1$ score significantly over the greedy baseline while preserving the same text fluency as measured using BLEU or METEOR scores against the reference text for a given S.

In order to mitigate the limitations of the semantic parser (SP) in our evaluation, we also conduct human evaluations. We collect a high-recall set of possible meaning triples represented by a piece of text and ask human raters to evaluate whether each one is present in the text. Starting with a high-recall set of triples mitigates recall failures of the SP, while collecting human ratings mitigates precision failures in the high-recall set. Human evaluation corroborates the conclusions from experiments using automatic metrics, showing that our proposed techniques lead to a small but significant improvement in semantic grounding of generated text.

8.2 Background and Intuition

The use case for decoder-only auto-regressive LMs—input text ("prompt") is fed to the LM and then output text is sampled from the LM given the state induced by the "prompt"—follows the original use of seq2seq models [120] and is best described by conceptualizing the LM as a

conditional model $P(W|\text{prompt})$, a probability distribution on output text W given an input textual prompt.

In our work we take a similar view on LMs, framing them as conditional models whose purpose is to encode meaning S as text W, $P(W|S;\theta)$, except that the semantic message S is formally defined and different from surface form text. Communication thus entails the exchange of semantic messages $S_1, \ldots S_n$ between conversation partners, or reader and writer, encoded as utterances $W_1, \ldots, W_n$. Besides possessing a LM that is used to verbalize the semantic message S into words W, each speaker is able to "decode" a semantic meaning $S^+ = \text{argmax}_S\, R(S|W;\phi)$ using a semantic parser (SP).

In this view, training a LM from surface text alone is no longer possible. To be able to do so we first need a SP that is able to recover the semantic message S in a unit of text W. Assuming the SP is available, we can use it to generate training pairs (S, W) for a semantic LM $P(W|S;\theta)$. The question of whether such a LM is *semantically grounded* is now well posed: we can use held-out semantic messages S^* and then sample text W^* from our LM $P(W|S^*;\theta)$ and compare the output $S^+ = \text{argmax}_S\, R(S|W^*;\phi)$ of the SP to the desired input message S^* and thus check the semantic accuracy of text generated by the LM.

However, the exact definition of semantic messages S and finding a SP that is able to extract them from unrestricted text is infeasible with current methods. For a proof of concept we settle on using the highly constrained setup in the WebNLG challenge [15]. In this setup, a large text-to-text seq2seq model such as T5 [4] is incrementally trained as a WebNLG SP and its accuracy in producing entity-relationship (E-R) triples as defined in WebNLG is measured on available test data. Using this SP and a semantic LM $P(W|S;\theta)$, also bootstrapped from T5 and incrementally trained on the WebNLG training data, we can measure the extent to which the LM produces "semantically grounded" text by computing Precision and Recall between the input set of E-R triples fed to the LM and that output by running the SP on the text sampled from the LM.

In the absence of a perfect SP (with Precision/Recall = 1.0) we can no longer computationally verify the semantic grounding of the LM: a SP with Recall < 1.0 misses E-R triples present in the

text generated by the LM and we can no longer strictly guarantee its semantic adequacy. We do note however that a SP that does come close to Recall = 1.0 at a Precision < 1.0 value can still guarantee semantic adequacy of the text as long as the set of E-R triples produced is a subset or equal to the desired input set.

Our work evaluates the semantic adequacy of LM generated text $W^* = \text{GENERATE}(P(W|S^*))$ and investigates algorithms that improve it while preserving fluency.

A family of simple inference-time techniques consists of sampling different word sequences in ways that preserve fluency and picking the one that produces the highest semantic $F1$ score with respect to the input S, without re-estimation of the LM parameters. Another approach conceptualizes the LM followed by the SP as a semantic auto-encoder and estimates the LM, while keeping the SP "frozen" such that the output sequence W^* maximizes the probability of the correct semantic parse $R(S^*|W^*)$. The latter estimation approach can be applied in both training and at inference time; in this work we only investigate the latter.

8.3 Related Work

The idea of building auto-encoders for language where the latent variable itself is language was previously explored by [126], where the authors propose an inference approach for sentence compression based on variational auto-encoders. Unlike [126], our work draws the latent representation from a probability distribution conditioned on the input semantic parse, rather than a background language model. The optimization strategy is also different, since we perform a small number of gradient descent steps instead of employing variational inference.

[127] tackle a very similar problem in a reinforcement learning setup, training both LM ("narrator") and open information extraction models jointly. Our work fixes the semantic parser while still taking advantage of the ability to back-propagate the error of reconstructing the correct semantic message.

The approach of data-to-text-to-data is also a somewhat common evaluation strategy. Starting with [128], people have been using information extraction to match information in text with that

of the input. More recent work like [129] tries to parse text into AMRs. In this work we go a step further and attempt to directly improve generation quality by sampling or by backpropagating parsing signals into the generation component.

8.4 Model

Let W be a text unit (a sentence in the WebNLG data) and S be a representation of the semantic message of W (a sequence of WebNLG triples). We define a semantic LM and a semantic parser as two sequence-to-sequence models:

- $P(W|S;\theta)$: a *semantic LM* trained to encode a semantic message S into text units W;
- $R(S|W;\phi)$: a *semantic parser (SP)*, trained to decode the semantic message S from a text unit W.

Our objective is to develop a method that generates computationally verifiable text: given a semantic message S^*, we would like to find a verbalization W^* such that running the semantic parser R on W^* will result in a semantic message S^+ as close as possible to the original S^*. In this work we compare a baseline to two methods designed to achieve this objective: a sampling approach and a new approach we refer to as "greedy finetuned".

In the baseline approach we decode greedily with $P(W|S;\theta)$ to encode the input semantic message S^* into the text unit W^*. In the sampling approach, we sample text units from the $P(W|S = S^*;\theta)$ model using temperature or nucleus sampling [130], then run the semantic parser on every sampled text unit, finally select as W^* the text unit that maximizes the $F1$ score between the semantic parse $S^+ = \text{argmax}_s\, R(S|W^*;\phi)$ and the input semantic message S^*.

In "greedy finetuned" we take a more complex approach and re-estimate the LM based on feedback from the semantic parser. We iteratively perform greedy decoding with the semantic LM P to obtain a text unit W_i, then adjust the semantic LM P while keeping the SP model R "frozen" in order to increase the probability of reconstruction of the correct semantic message. After k steps,

we pick the final text unit W^* from the set $\{W_1, \ldots, W_k\}$, choosing the one that leads to the best F1 score between the decoded semantic message S^+ and the original message S^*.

The semantic LM update in "greedy finetuned" is performed by taking a gradient ascent step on the following objective function:

$$J(S^*, S^+; \theta, \phi) = \mathop{\mathbb{E}}_{W \sim P(\cdot|S^*;\theta)} [R(S^+|W; \phi)],$$

where keeping the semantic parser frozen corresponds to keeping ϕ constant and only updating θ.[1]

Since both P and R are neural networks, the gradient ascent step for θ can *mostly* be computed by backpropagation. However, the objective J is not differentiable because it requires auto-regressively decoding from the model P to sample values for W and estimate the expectation of correct reconstruction. We address this difficulty by sampling the text unit W using greedy decoding and then employing the straight-through gradient estimator ([132], [133]). The estimator is illustrated in more detail in section 8.4.1.

In addition to (1) greedy (baseline), (2) sampling and (3) greedy finetuned, we consider two ensemble methods: (4) greedy+sampling and (5) +sampling. Greedy+sampling will run both methods (1) and (2), and then pick the verbalization W^* that leads to the reconstruction with the highest F1 score. Greedy finetuned+sampling will similarly pick the best output after running methods (2) and (3).

8.4.1 Straight Through Approximation

To propagate the gradient of J through the SP into the language model we need to compute

$$\frac{\partial R(S|W^*; \phi)}{\partial \theta},$$

where $W^* = \text{argmax}\, P(W|S^+; \theta)$ is computed using greedy search G_SEARCH.

[1] We note that greedy finetuned is expensive since it requires updating all the parameters of the semantic LM k times for every inference batch. The approach could be made substantially more efficient, however, by employing parameter efficient methods such as prompt tuning [131].

Following [133], during the forward pass we transmit the text tokens W^*, and during the backward pass we pretend that for every token the real value distribution over the vocabulary was transmitted instead. For this purpose, we can regard the W^* sequence passed to the SP as a binary tensor

$$W^* = \text{ONE\_HOT}(\text{G\_SEARCH}(\log P(W|S^+;\theta)))$$

of shape (b, l, v) where b is the batch size, l is the maximum length in tokens or word-pieces for text sequences including padding, and v is the size of the vocabulary.

In the straight-through (ST) approximation with automatic differentiation, we replace the W^* tensor in the computational graph with the following:

$$\begin{aligned} W^*_{\text{ST}} &= P(W^*|S^+;\theta) + \\ &\quad \text{S\_G}[W^* - P(W^*|S^+;\theta)] \end{aligned}$$

where the S_G$[\cdot]$ stop-gradient operator is the familiar pass-through in the forward direction and none-shall-pass in back-propagation. Thus in the forward direction the computation will proceed with $W^*_{\text{ST}} = W^*$, but in the backward direction the computation will behave as if $W^*_{\text{ST}} = P(W^*|S^+;\theta)$.

8.5 Data Set

The WebNLG corpus [134] comprises sets of triplets describing facts (entities and relations between them) and the corresponding facts rendered in the form of natural language text. The corpus contains sets of up to 7 triplets alongside one or more reference texts for each set. As explained in Section 2 of [15], the English dev set consists of categories and entities seen in the training data; the English test set (D2T) consists of a mixture of seen entities and categories, as well as unseen entities and 5 unseen categories (28%, 22% and 50%, respectively), making it hard to expect correct text generation and/or semantic parsing from our models. Nevertheless, we experiment in this condition as well.

The dataset can be used for work in both natural language generation and the reverse task of

Table 8.1 Automatic evaluation of our method on "seen" test data (dev split of English WebNLG 3.0). Grayed rows have BLEU scores more than 1% below the baseline. The "Improved" column displays the ratio of generations with a higher triple F1 compared to the greedy baseline.

Method	BLEU	METEOR	chrF++	Triple P / R / F1	Improved
greedy (baseline)	0.64	0.46	0.76	0.93 / 0.87 / 0.90	-
sampling	0.51	0.43	0.71	0.97 / 0.92 / 0.95	-
sampling (t=0.3, p=0.95)	0.63	0.46	0.76	0.96 / 0.91 / 0.94	-
greedy+sampling	0.60	0.46	0.75	0.98 / 0.93 / 0.95	0.20
greedy+sampling (t=0.3, p=0.95)	0.63	0.47	0.76	0.96 / 0.91 / 0.94	0.14
greedy finetuned	0.62	0.46	0.76	0.96 / 0.91 / 0.93	0.13
greedy f.+sampl. (t=0.3, p=0.95)	0.63	0.46	0.76	**0.97 / 0.92 / 0.95**	**0.17**

triplets extraction. Its main use was for the WebNLG natural language generation challenge with the goal of mapping the sets of triplets to text, including referring expression generation, aggregation, lexicalization, surface realization, and sentence segmentation.

The initial (2017) WebNLG shared task required participating systems to generate English text from a set of DBpedia triples [16], whereas the more recent (2020) WebNLG 3.0 data set used for the WebNLG+ challenge [15] additionally includes generation into Russian and semantic parsing of English and Russian texts, encompassing four tasks: RDF-to-English, RDF-to-Russian, English-to-RDF and Russian-to-RDF.

We only use WebNLG 3.0 English data, starting from a pre-trained language model [4] and fine-tuning it on the WebNLG training data for either LM or semantic (RDF) parsing. Since the model uses an encoder-decoder architecture, we feed the conditioning information (RDF triples when incrementally training the LM or natural language text when training the SP) to the encoder and then let the decoder generate the output sequence (natural language text or RDF triples, respectively). We note that the exact order of the RDF triples for a given reference text is a degree of freedom that we could experiment with.

8.6 Experiments

As semantic LM and semantic parser we train T5 XXL models on the training set split of WebNLG for 100 steps with a batch size of 256 examples. We then compare three main methods:

Table 8.2 Automatic evaluation of our method on "mixed: seen and unseen" test data (test split of English WebNLG 3.0). Grayed rows have BLEU scores more than 1% below the baseline. The "Improved" column displays the ratio of generations with a higher triple F1 compared to the greedy baseline.

Method	BLEU	METEOR	chrF++	Triple P / R / F1	Improved
Amazon AI (2020 1st)	0.54	0.42	0.69		
bT5	0.52	0.41	0.68		
greedy (baseline)	0.54	0.41	0.69	0.58 / 0.50 / 0.54	-
sampling	0.44	0.39	0.64	0.66 / 0.58 / 0.62	-
sampling (t=0.3, p=0.95)	0.53	0.41	0.68	0.65 / 0.57 / 0.60	-
greedy+sampling	0.51	0.41	0.67	0.67 / 0.59 / 0.63	0.31
greedy+sampling (t=0.3, p=0.95)	0.54	0.41	0.69	0.65 / 0.57 / 0.61	0.23
greedy finetuned	0.53	0.41	0.68	0.65 / 0.57 / 0.61	0.20
greedy f.+sampl. (t=0.3, p=0.95)	0.53	0.41	0.68	**0.69 / 0.61 / 0.65**	**0.34**

Table 8.3 Exact match parsing performance of our T5 semantic parser (SP) on Text2RDF English WebNLG.

Model	F1	P	R
Amazon AI (Shanghai)	0.67	0.69	0.69
bt5	0.68	0.67	0.70
Our T5	0.63	0.63	0.64

(1) the greedy decoding baseline, (2) sampling with different temperatures and nucleus probability mass, (3) greedy finetuned. We additionally report ensembles of these methods. (1)+(2) is "greedy+sampling", where we perform both greedy decoding and sampling and then pick the generated text with highest semantic parsing F1 score. Similarly (2)+(3) is "greedy finetuned+sampling".

For the sampling approach we alway pick 4 samples, we set the temperature to either 1.0 or 0.3 and the nucleus probability mass to either 1.0 or 0.95. For the greedy finetuned approach we perform gradient descent steps on θ with AdaFactor, with a learning rate of 10^{-3}. We perform 4 iterations and return the hypothesis with the best semantic reconstruction. Greedy finetuned is performed on batches of 64 examples.

8.6.1 Automatic Evaluation

As an automatic metric of semantic grounding, we measure the standard Precision, Recall and F1 metrics used for evaluating semantic parsing accuracy, fixing the input WebNLG triples as

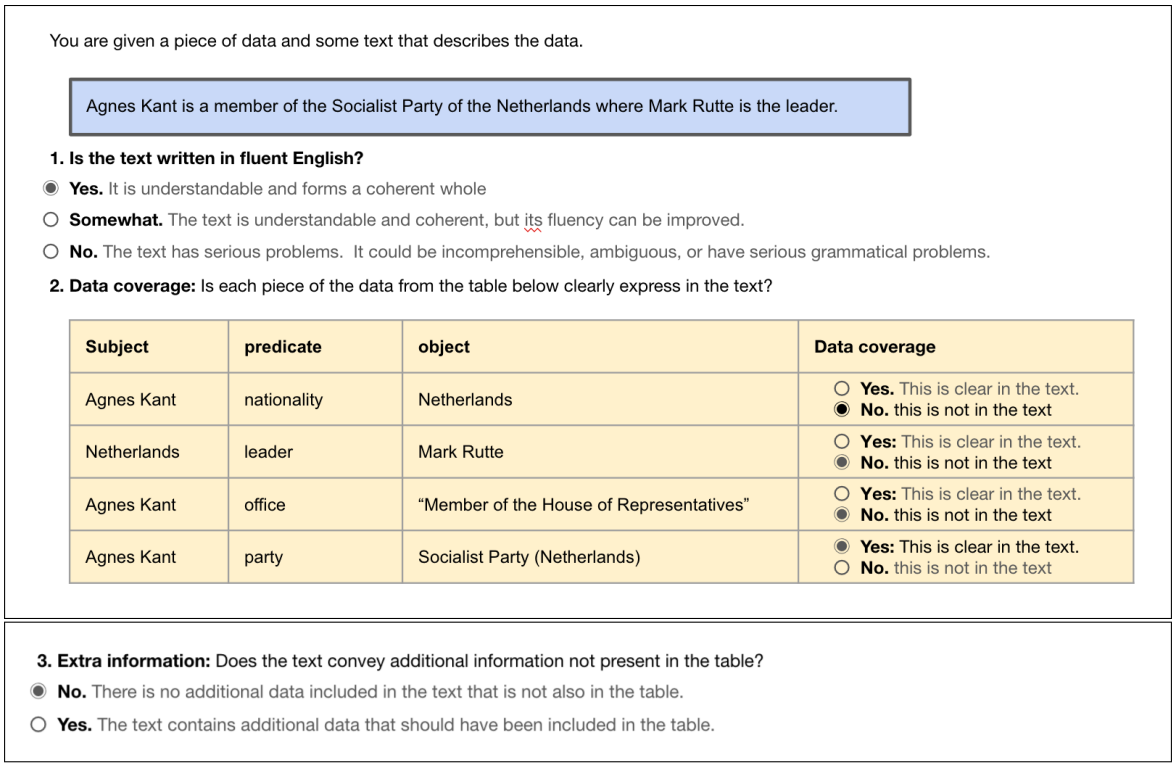


Figure 8.1: Screenshots of our evaluation template: (1) fluency question, (2) data coverage questions, and (3) a question asking annotators if the triples cover all the information in the text.

reference and the triples reconstructed by the semantic parser as hypothesis. As for the fluency of the text generated by the LM, we measure it using BLEU [135], METEOR [136] and chrF++ [137] against the human references in the D2T data sets. We note that there are between one and five references for each RDF triple, frequently two or three.

The results of automatic evaluation on the English section of WebNLG 3.0 are shown in Tables 8.1 and 8.2. We first note that our greedy baseline matches the performance of the 2020 WebNLG winning system on BLEU and chrF++. Sampling and greedy finetuned increase triple F1 by 6% and 7% absolute on the test set while keeping the BLEU score and other metrics within one point from the greedy baseline, showing that the semantic match with the desired input meaning can be significantly increased at minimal cost in the fluency of the generated text. The greedy finetuned+sampling ensemble gives us the highest performance, increasing triple F1 by 11%, again keeping the fluency metrics within 1% from the greedy baseline. The same overall trends can be observed for the WebNLG dev set in Table 8.1. We additionally note that the sampling approach

Table 8.4 Human evaluation results. The row labeled "reference" has the evaluation of the human-generated reference text (provided as part of the test set). Triples is the total number of triples rated, not the number of triples generated by the parser for a particular system. See text for details.

Method	Examples	Triples	Fluency	Precision	Recall	F-Score
reference	206	1278	1.00	0.77	0.91	0.84
greedy (baseline)	206	1278	**1.00**	0.74	0.84	0.78
greedy+sampling	115	719	0.99	0.74	**0.87**	**0.80**
greedy finetuned	112	675	0.98	**0.76**	0.84	**0.80**

only performs well in this evaluation if temperature and nucleus probability mass are tuned.

We additionally report for both dev and test set the fraction of generations with improved triple F1 compared to the greedy baseline. We only report the improved ratio for methods that are ensembled with greedy and so are guaranteed to always improve on automatically measured triple F1 compared to the baseline. We find that our best method, greedy finetuned+sampling, improves 34% of our generations according to our automated metric of semantic verification.

As a sanity check for our semantic parser model, we report in Table 8.3 the performance of our T5 semantic parser on the official evaluation metrics of WebNLG Text2RDF. Since we only fine-tune T5 on the training split of WebNLG and we do not employ any additional techniques, the performance of our parser is competitive but lower than SotA systems.

8.6.2 Human Evaluation

Because our inference method relies on the SP, one could reasonably question whether it is appropriate to use the same SP also in the automatic evaluation. For example, it is possible that our method is learning to cheat by finding text with incorrect semantics that the text-to-parse model incorrectly parses as the desired semantics. In order to alleviate such concerns we augment the automatic evaluations with human ratings.

Ideally, expert annotators would encode the DBPedia semantics of reference text as well as system outputs. However, due to the size of DBPedia with tens of thousands of possible relations, it is infeasible for a human to perform the parsing task. Instead, we present annotators with a text as well as candidate triples and ask them to evaluate whether each candidate triple is represented in the

given text.

We evaluate the text generated by our models in the following way:

1. Use the SP model to generate possible triples for the reference text as well as all model outputs.

2. For the reference text and each system output, we present the raters with that text and the union of parse triples generated in step 1.

3. For each triple, the raters verify whether it is implied by the text.

4. We compute precision and recall for the reference text and each system with respect to the union of generated triples and the human ratings.

We evaluate samples where the greedy decoding baseline did not produce the same result as greedy+sampling and greedy finetuned. We see from Table 8.1 that this happens in 23% and 20% of examples respectively. We include the exact phrasing of the questions as well as a screenshot of the rating interface in Figure 8.1. Note that in addition to the semantic annotation we also ask the raters to judge the fluency of the generated text.

The results are shown in Table 8.4. The row labeled reference refers to the human-generated reference text which has a surprisingly low precision and recall. The remaining rows have the same definitions as in Table 8.1. We obtained 3-way annotations for 112 examples for greedy finetuned and 115 examples for greedy+sampling (and the corresponding 206 examples for greedy decoding). Overall, we see that sampling improves recall while greedy finetuned improves precision, while not significantly changing the fluency of the text. Using bootstrap resampling on the examples, we find that greedy+sampling has significantly higher recall than greedy (p value 0.97), but greedy finetuned is not better at the 95% level in terms of precision (p value 0.948 with 10k samples). The other metrics are not significantly different. In addition to corroborating the results from automatic evaluation, the human evaluation shows strikingly low precision and recall of the "reference" text. We show two illustrative examples of losses. First, we display an example below of a recall loss, where the annotator did not adhere strictly to the semantics.

Input: `Zaoyang | isPartOf | Hubei`

`Nie_Haisheng | birthPlace | Zaoyang`

Target: Nie Haisheng is from Zaoyang in Hubei province.

In this example, the rater chose to realize the `birthPlace` relation using the phrase *is from*, but those were judged by our annotators as not having the same meaning.

Below is an example of a precision error:

Input: `Zaoyang | isPartOf | Hubei`

`Nie_Haisheng | mission | Shenzhou_6`

`Nie_Haisheng | birthPlace | Zaoyang`

Target:

Born in Zaoyang city, Hubei, Nie Haisheng participated in the Shenzhou 6 mission.

Implied:

`Nie_Haisheng | birthPlace | Hubei`

In the example, `Nie_Haisheng | birthPlace | Zaoyang` and `Zaoyang | isPartOf | Hubei` together imply `Nie_Haisheng | birthPlace | Hubei` and the SP identified this by generating the implied relation. This was accepted by our annotators, and our metrics show it as a precision loss – the generated text includes a triple that was not in the desired parse.[2]

8.6.3 Generation Examples

Table 8.5 displays example generated text for the two main approaches described in this paper (greedy+sampling and greedy finetuned), compared to the greedy baseline. As noted in the previous section, we find that greedy+sampling has a larger effect on reducing triple deletion, corresponding to the recall gain in Table 8.4, while greedy finetuned is better at mitigating triple insertion, corresponding to a precision gain.

As can be seen in the third row of Table 8.5, removing inserted triples with greedy finetuned

[2]It may be possible in theory to enumerate implied relations by writing rules to avoid penalizing the text presented to the raters for including the semantics implied by the desired triples. However, this would make results more difficult to reproduce, so instead note that this is a shortcoming of the evaluation of the generated semantics.

Table 8.5 Text generated by the two main approaches described in this work (greedy+sampling and greedy finetuned) compared to the a baseline greedy generation, for two different sets of input triples. Inserted and deleted triples are determined by human annotators as described in Section 8.6.2.

Method	Generated Text	Inserted Triples	Deleted Triples
greedy (baseline)	The University of Burgundy is located in Dijon, France. It has 2900 staff and 16800 students. Dijon has the postal code 21000.	`UniversityOfBurgundy | country | France : UniversityOfBurgundy | numberOfStudents | 16800 : Dijon | country | France`	`UniversityOfBurgundy | numberOfUndergraduate Students | 16800`
greedy+ sampling	The University of Burgundy is located in Dijon, France and has 2900 employees and 16800 undergraduate students. The postal code of Dijon is 21000.	`UniversityOfBurgundy | country | France : Dijon | country | France`	–
greedy finetuned	The University of Burgundy is located in Dijon, 21000. It has 2900 staff and 16800 undergraduate students.	–	–
greedy (baseline)	Alan Shepard was born in New Hampshire, United States on November 18th, 1923. ...	–	`AlanShepard | nationality | UnitedStates`
greedy+ sampling	Alan Shepard was born in New Hampshire on November 18th, 1923. He was a United States national and ...	–	–
greedy finetuned	Alan Shepard was an American born in New Hampshire on November 18 1923. ...	–	`AlanShepard | nationality | UnitedStates`

can at times come at the cost of fluency. In this case the model is led to output "Dijon, 21000" to indicate the postal code and avoid mentioning the country in which Dijon is found.

8.7 Conclusions

We have presented an approach to semantic grounding of language models that conceptualizes the LM as a conditional model generating text given a desired semantic message and embeds it into a semantic auto-encoder model by feeding the LM output to a semantic parser whose output is in the same representation domain as the input message. We evaluate a simple baseline that generates text using greedy search and show that one can improve the semantic accuracy of generated text by

two simple techniques. The first is a sampling method that generates a few candidate text sequences and lets the semantic parser choose the better one. The second trains the language model (while keeping the semantic parser frozen) with the aim of improving the auto-encoder semantic accuracy. We carry out experiments on the English WebNLG 3.0 data set, using BLEU to measure the fluency of generated text and standard semantic parsing metrics to measure the match between the output of the parser and the desired (input) semantic message. We show that our proposed approaches improve the semantic accuracy of generated text significantly over the greedy search baseline and are partially additive. Human evaluation corroborates the results of automatic evaluation experiments and suggests that the approaches are complementary, greedy+sampling improving triple recall and greedy finetuned improving triple precision.

8.8 Limitations and Future Work

The main challenge of this work has been finding a suitable representation for the desired semantics of the generated text along with a semantic parser that could generate such representations from unconstrained text. The suitability of RDF triples is questionable for open domain semantics representation in unconstrained text, so an approach aimed at widening the scope of the current work would have to first address this obstacle.

Assuming the parser operating point on the Precision/Recall curve can be brought close enough to perfect Recall, the framework proposed would lend itself to computationally verifiable semantic grounding, namely being able to guarantee the fact that the generated text does not convey meaning outside the intended one.

Chapter 9: QAMELEON: Multilingual QA with Only 5 Examples

The availability of large, high-quality datasets has been a major driver of recent progress in question answering (QA). Such annotated datasets, however, are difficult and costly to collect and rarely exist in languages other than English, rendering QA technology inaccessible to underrepresented languages. An alternative to building large monolingual training datasets is to leverage pre-trained language models (PLMs) under a few-shot learning setting. In this chapter we present QAMELEON, an approach that uses a PLM to automatically *generate* multilingual data upon which QA models are trained, thus avoiding costly annotation. Prompt tuning the PLM with only five examples per language delivers accuracy superior to translation-based baselines; it bridges nearly 60% of the gap between an English-only baseline and a fully-supervised upper bound trained on almost 50,000 hand-labeled examples and consistently leads to improvements compared to directly fine-tuning a QA model on labeled examples in low resource settings. Experiments on the TYDIQA-GOLDP and MLQA benchmarks show that few-shot prompt tuning for data synthesis scales across languages and is a viable alternative to large-scale annotation.

9.1 Overview

Question answering (QA) has seen impressive progress in recent years enabled by the use of large pre-trained language models [138, 139, 140], and the availability of high-quality benchmarks [20, 141, 7]. Many QA datasets frame the task as reading comprehension where the question is about a paragraph or document and the answer is a span therein. Advances in QA modeling have been primarily reported for English, which offers a considerable amount of high-quality training data compared to other languages. More recently, efforts have focused on the creation of *multilingual* QA benchmarks such as TYDI QA (10 languages; [142]), MLQA (6 languages;

[143]), and XQuAD (10 languages; [144]). Among these, only TyDi QA is genuinely large-scale; MLQA and XQuAD are limited to an evaluation set due to the high cost and labor required to collect data across languages.

As a result, efforts to localize QA models to new languages have been primarily focusing on *zero-shot* approaches. Recent proposals include using machine translation to approximate training data for supervised learning [143], and data augmentation via generating synthetic questions for new languages [145, 146]. Both approaches rely on transfer from English, which leads to a dependence on translation artifacts [147, 148] and a bias towards the linguistic characteristics of English, which is not the best source for all target languages [149]. However, annotating a minimally-sized data sample can potentially overcome these limitations while incurring significantly reduced costs compared to full dataset translation [150].

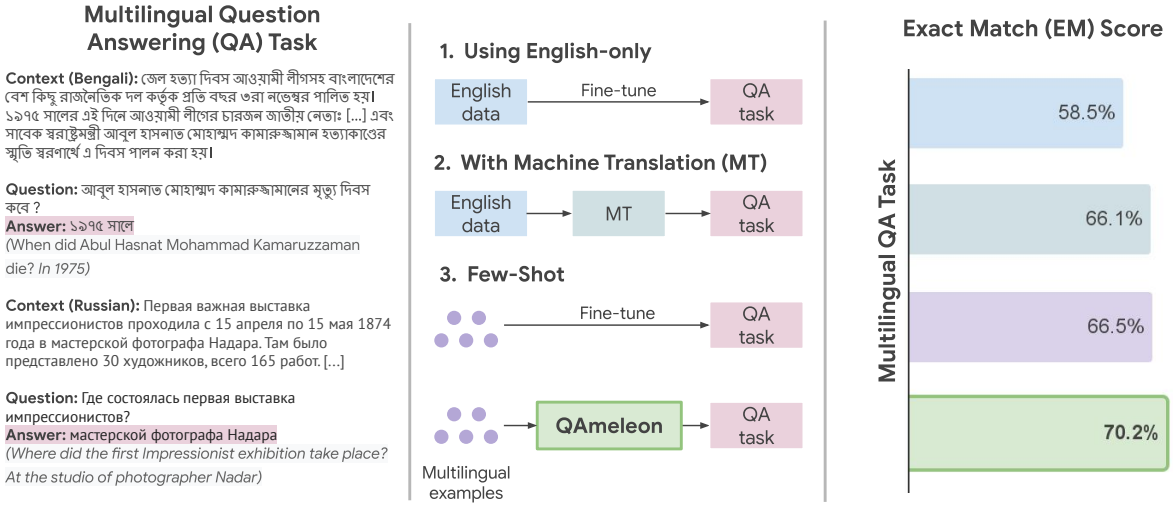


Figure 9.1: Synthetic data generation for multilingual question-answering (QA). Left: Examples of the multilingual QA task. Translations are added for readability. Middle: Strategies for localizing QA models to new languages: 1. Using English QA data as a zero-shot approach, 2. with Machine Translation (MT) to approximate training data for supervised learning, and 3. few-shot approaches with a handful of multilingual examples. Right: Model performance on the multilingual QA task. We report average Exact Match (EM) across all languages on the TyDiQA-GoldP dataset [142].

In this paper, we argue that a few-shot approach in combination with synthetic data generation and existing high-quality English resources can mitigate some of the above mentioned artifacts. Beyond question answering, multilingual approaches have succeeded at leveraging a small number

of annotations within a variety of tasks [151] including natural language inference, paraphrase identification, and semantic parsing [152]. Existing work [121, 153] has further shown that prompting pre-trained large language models (PLMs) can lead to strong performance on various tasks, including question answering [154, 122] and open-ended natural language generation [155, 156]. Investigations of prompting in multilingual settings have also shown strong few-shot performance in classification tasks [157], natural language inference [158], common sense reasoning [159], machine translation [160], and retrieval [161].

We synthesize these directions into QAMELEON, an approach for bootstrapping multilingual QA systems, with as few as five examples in a new target language (see Figure 9.1). We use gold annotations to prompt-tune a PLM in order to automatically generate multilingual QA data, which is then used to train a QA model. We find that QAMELEON delivers accuracy superior to zero-shot methods and competitive translation-based baselines and in some cases competes with the fully supervised upper bound. [1] Experiments on the TYDI QA [142] and MLQA [143] benchmarks show that few-shot prompt tuning [163] scales across languages, significantly outperforms prompt engineering [121] with the same number of labeled examples, and is a viable alternative to large-scale annotation.

Our contributions include (a) a new approach to bootstrapping a multilingual QA system; QAMELEON prompt-tunes a PLM with as few as five gold examples to automatically generate multilingual QA data which is then used to train a QA model; (b) a series of experimental results showing significant improvements over existing approaches in the few-shot regime, ranging from 12% absolute accuracy on TYDIQA-GOLDP [142] over an English-only baseline and 4% absolute accuracy over a competitive translate-train baseline; (c) extensive analysis of the behavior of QAMELEON in zero shot and low resource regimes, on different multilingual QA datasets, and in comparison to prompt-engineering.

[1] This is noteworthy as multilingual models fine-tuned on translated data—also known as translate-train—form the state of the art on most multilingual datasets [162].

9.2 Synthetic Data Generation

Let $\mathcal{D}_l$ denote a QA dataset with examples provided by human annotators, where l is a *target* language in a set L of languages of interest. $\mathcal{D}_l$ consists of samples $(c, q, a)_l$, where c is a paragraph of text, q is a question, and a is an answer extracted from c (see Figure 9.1 left). We further use $\mathcal{D}_{l,n}$ to denote a dataset $\mathcal{D}_l$, but making n explicit, with n referring to the number of examples it contains. For instance, $\mathcal{D}_{\text{fr},5}$ denotes a French QA dataset with 5 examples. Finally, let $\mathcal{U}_l$ denote sets of *unlabeled* paragraphs in language l; we assume these are in-domain with respect to the paragraphs in $\mathcal{D}_l$ but are not accompanied by questions or answers.

Throughout this work, we will assume the availability of $\mathcal{D}_{\text{en}}$, a large QA dataset in English (*source* language). This assumption corresponds to the observation that most large-scale QA datasets [20, 164, 165, 7] contain examples exclusively in English. For languages other than English, we assume that only small datasets $\mathcal{D}_{l,n}$ are available for training (e.g., $n = 5$) ("Few-Shot" scenario) or no data at all ("English-Only" scenario). We will also assume that sets $\mathcal{U}_l$ of unlabeled passages are available for all target languages. Our task will be to synthesize QA data in each *target* language l in order to train QA models on l directly.

In the rest of this section we formally describe three ways of synthesizing QA data and give further details on the two scenarios we consider, "English-Only" and "Few-Shot".

9.2.1 Machine Translation (MT)

A widely adopted approach [143, 166] makes use of a machine translation system $\mathcal{T}$ to automatically translate text from one language into another. Let $\mathcal{T}_{l'}(\mathcal{D}_l)$ denote the translation of dataset $\mathcal{D}_l$ from language l to language l'. The translation is performed by independently applying $\mathcal{T}$ to context c, question q, and answer a for each example in the source dataset (see approach 2 in Figure 9.1). A synthetic QA dataset $\mathcal{D}_{\text{MT}}$ is generated by translating the entire English dataset into each language of interest:

$$\mathcal{D}_{\text{MT}} = \mathcal{D}_{\text{en}} \cup \bigcup_{l \in L - \{\text{en}\}} \mathcal{T}_l(\mathcal{D}_{\text{en}}).$$

The approach described here is known as "translate-train". An alternative is "translate-test", where translation is employed during inference instead of training. Multilingual inputs are translated to English and inference is done via an English QA model. The English predictions are then translated back to the respective target language. We experimentally found "translate-test" to perform poorly on our task in comparison to translate-train due to its reliance on multiple noisy translation steps.

Note that training on $\mathcal{D}_{\text{MT}}$ still relies on the support of the high-quality $\mathcal{D}_{\text{en}}$. Previous work [167, 168] has highlighted various limitations with multilingual approaches based on MT including (a) their dependence on the quality of available MT systems in a given language and, in turn, the availability of high-quality (expensive) parallel data, (b) a potential misalignment of answer spans after translation in context to the passage vs. translation of answers independently, and (c) translationese artifacts and English-centric content topics [142].

9.2.2 Prompt Engineering (PE)

PLMs [121, 122] have recently shown unprecedented performance on a vast number of tasks, including natural language generation, without the need for modifying any of the model's parameters, simply by hand-designing a textual prompt that instructs the model to perform a certain task. Following [121], we consider a class of hand-designed prompts referred to as "prompting" or "in-context learning". The prompt starts with a free form instruction, followed by a small number of instances exemplifying how the task is solved. An incomplete instance is then appended to this prompt and the PLM performs the task by completing that instance. We refer to this approach as "prompt engineering" (PE) since the input to the PLM has to be hand-engineered based on human intuition about the target task (see approach 3 in Figure 9.1).

In order to hand-engineer prompts for our task, we use a small set of parallel examples $C_{l,n}$ consisting of passages, questions, and their answers in the English source and target language l. We discuss how we construct these examples shortly. For now, suffice it to say that we create two

prompts for answer and question generation, respectively.[2] Our first prompt is used to obtain an answer a_l in the target language l from passage c_l:

```
I will write potential answers
for the following passages.
  Passage: c_l
  Answer in English: a_en
  Answer in the original language: a_l
...
```

The second prompt generates question q_l, utilizing passage c_l and the previously predicted answer a_l:

```
I will write questions and answers
for the following passages.
  Passage: c_l
  Answer: a_l
  Question in English: q_en
  Question in the original language: q_l
...
```

We generate synthetic data instances $(c, q, a)_l$ where a and q are inferred by applying our two prompts consecutively on each passage $c_l \in \mathcal{U}_l$ (recall $\mathcal{U}_l$ is the set of unlabeled passages in target language l).

In the English-Only scenario, neither questions nor answers are available in target language; we obtain these by resorting to machine translation:

$$C_{l,n}^{\text{en-only}} = \{(\mathcal{T}_l(c), q, a, \mathcal{T}_l(q), \mathcal{T}_l(a)) | (c, q, a) \in \mathcal{D}_{\text{en},n}\},$$

In the "Few-Shot" setting, we have access to n-labeled examples (questions and answers) in the

[2]We find that joint answer and question generation using single-stage prompting performs worse in comparison to two-stage generation.

target language, and translate these into English:

$$C_{l,n}^{n-\text{shot}} =$$
$$\{(c, \mathcal{T}_{\text{en}}(q), \mathcal{T}_{\text{en}}(a), q, a) | (c, q, a) \in \mathcal{D}_{l,n}\}.$$

Let $\mathcal{P}_l^e$ denote this prompting based generation. We can write the generated synthetic dataset as:

$$\mathcal{D}_{\text{PE}} = \mathcal{D}_{\text{en}} \cup \bigcup_{l \in L - \{\text{en}\}} \mathcal{P}_l^e(\mathcal{U}_l).$$

Note that, in the composition of the prompt, we always include English as an intermediate or "bridge", i.e., asking the model to predict questions and answers in English in addition to the ones in the target language, as we experimentally found it improves the quality of the generated data. The use of a bridge for this task can be thought of as an example of multilingual "chain of thought" prompting [169].

9.2.3 QAMELEON (PT)

In this approach, an optimizer is utilized to minimize the cross-entropy loss by updating the PLM's parameters for $P(a, q|c, l)$ over a training set containing examples $(c, q, a)_l$ for the languages in L. As with PE, we generate the training set for the PLM in two ways. For "English-Only" we construct the dataset as $\bigcup_{l \in L} \mathcal{T}(\mathcal{D}_{\text{en}})$, while for "Few-Shot" we use $\bigcup_{l \in L} \mathcal{D}_{l,n}$.

Given the small size of the training set in the "Few-Shot" setting and the large size of current models, we opt for using prompt tuning [PT; 163], a parameter-efficient fine-tuning variant where only the embeddings of the first m tokens in the input of the PLM are allowed to be modified by the optimizer (see approach 3 in Figure 9.1). We note that in prompt tuning, like in prompt engineering, the parameters of the PLM remain unchanged. What is trained is only a short soft prompt that is prepended to the input embeddings at inference time.

We use $\mathcal{P}_l^t$ to denote the operation of generating question-answer pairs through greedy decoding on the prompt-tuned PLM, by taking an unlabeled passage $c_l \in \mathcal{U}_l$ as input, preceded by a few

tokens of encoding language l. We finally obtain the synthetic QA dataset $\mathcal{D}_{\text{PT}}$ as:

$$\mathcal{D}_{\text{PT}} = \mathcal{D}_{\text{en}} \cup \bigcup_{l \in L - \{\text{en}\}} \mathcal{P}_l^t(\mathcal{U}_l).$$

9.2.4 Data Assumptions

English-Only In this scenario, only training data in English is available, denoted as $\mathcal{D}_{\text{en}}$. Prompt Engineering (PE) assumes parallel exemplars are available, while Prompt Tuning (PT) requires exemplars in the target language only. Both are possible by translating examples of the English data $\mathcal{D}_{\text{en}}$ into each target language. Machine Translation (MT) approaches in this work follow this scenario only.

Few-Shot In this scenario, a small number of examples (n-shot) are available in each target language, denoted as $\mathcal{D}_{\text{l},n}$. In this scenario, parallel exemplars for Prompt Engineering (PE) can be obtained by translating the target language data into English. Prompt Tuning (PT) only requires exemplars in the target language, which are readily available in this setting.

9.3 Experimental Setup

We evaluate the synthetic data generation approaches presented in Section 9.2 across various languages on two benchmark datasets, which we discuss below. We also describe various model configurations, and comparison systems before presenting our results.

9.3.1 Datasets

TyDi QA [142] is a multilingual extractive question answering dataset designed to represent a typologically diverse set of languages. Annotators were given a Wikipedia passage in the target language and asked to write a question that could not be answered by that passage. For each question, the top-ranked Wikipedia article was then retrieved via Google Search. Annotators were subsequently asked to answer the question given the retrieved Wikipedia article. As a result of this

Table 9.1 Number of question-answer pairs per language and data split for the datasets considered in this work.

	TyDiQA-GoldP		MLQA	
Language	Train	Eval	Dev	Test
Arabic	14,805	921	517	5,335
Bengali	2,390	113	—	—
Chinese	—	—	504	5,137
English	3,696	440	1,148	11,590
Finnish	6,855	782	—	—
German	—	—	512	4,517
Hindi	—	—	507	4,918
Indonesian	5,702	565	—	—
Kiswahili	2,755	499	—	—
Korean	1,625	276	—	—
Russian	6,490	812	—	—
Spanish	—	—	500	5,253
Telugu	5,563	669	—	—
Vietnamese	—	—	511	5,495
Total	49,881	5,077	4,199	42,245

information-seeking task design, questions in TyDi QA are often without an answer. In this work we consider TyDiQA-GoldP: the Gold Passage version of TyDi QA where only questions with answers in the Wikipedia page are given and the model has to identify the answer in the passage that contains it (see Table 9.1 for statistics on this dataset).

MLQA [143] is an extractive question answering dataset, designed for evaluating multilingual and cross-lingual question answering models. MLQA does not publish a training split, but only development and test partitions. MLQA was created by aligning sentences in Wikipedia passages across different languages. Annotators then created questions based on English sentences, professional translators translated these questions to other languages, and, finally, annotators selected answers from passages containing sentences aligned to the translated questions. As in TyDiQA-GoldP, the task is to extract the answer from a passage given a question (dataset statistics are shown in Table 9.1).

Unlabeled Data We obtained paragraphs $\mathcal{U}_l$ in each target language from Wikipedia. Specifically, we pre-processed Wikipedia pages using WikiExtractor [170]. Paragraphs were sampled uniformly,

Table 9.2 Synthetic question-answering data generation methods for training multilingual reading comprehension systems on TyDiQA-GoldP. We report averages over 3 runs of fine-tuning mT5-XL on gold or synthetic data. Standard deviation is given in parentheses. Performance for individual languages (excluding English) is shown in Table 9.3. For comparison we also include recent few-shot prompting results with large language models on TyDiQA-GoldP: [171]§, [122]†, and [172]‡.

		English-Only		Few-Shot		
Method	Translate	Avg EM	Avg F1	n-Shot	Avg EM	Avg F1
Baseline		$58.5_{(\pm 3.1)}$	$74.2_{(\pm 2.6)}$	5	$66.5_{(\pm 0.7)}$	$79.8_{(\pm 0.4)}$
MT	✓	$66.1_{(\pm 2.1)}$	$79.5_{(\pm 1.8)}$	5	—	—
PE	✓	$64.4_{(\pm 1.4)}$	$76.9_{(\pm 1.1)}$	5	$62.6_{(\pm 1.8)}$	$77.6_{(\pm 1.2)}$
PE + MT	✓	$\mathbf{69.4}_{(\pm 0.4)}$	$\mathbf{81.4}_{(\pm 0.4)}$	5	$67.9_{(\pm 0.2)}$	$80.5_{(\pm 0.6)}$
QAmeleon (PT)	✓	$65.5_{(\pm 0.7)}$	$79.4_{(\pm 0.7)}$	5	$70.2_{(\pm 0.2)}$	$81.7_{(\pm 0.1)}$
QAmeleon (PT)+MT	✓	$68.1_{(\pm 0.8)}$	$80.9_{(\pm 0.7)}$	5	$\mathbf{70.7}_{(\pm 0.9)}$	$\mathbf{82.2}_{(\pm 0.8)}$
code-davinci-002§		—	—	1	48.1	—
PaLM-540B†		—	—	1–10	60.0	—
Flan-U-PaLM-540B‡		—	—	1	68.3	—

with a length between 200 and 510 characters. The target language was determined based on the language code of the Wikipedia edition.

9.3.2 Model Configuration

Synthetic Data Generation In our TyDi QA experiments, we treat the English training data as the English source. For MLQA, we employ the English SQuAD [20] training data as the source. In the Few-Shot scenario, our human-annotated target-language examples $\mathcal{D}_{l,n}$ are taken from the training split of TyDiQA-GoldP and the validation split of MLQA.

For machine translation (MT), we employ the public Google Translate API [173] while the PLM utilized in this work is Anonymous et al.[3] We perform heuristic checks to clean synthetic datasets $\mathcal{D}_{\text{PE}}$ and $\mathcal{D}_{\text{PT}}$. We only preserve a question-answer pair if the generated answer a is a substring of the given context c, but not a substring of the query q. We perform the first check as both TyDiQA-GoldP and MLQA are extractive QA datasets. We perform the latter check because we empirically found that some of the low quality generated question-answer pairs were trivially answered based on the content of the question alone, for example, q: "where is X?", a: "X".

[3]Omitted to preserve anonymity.

In the construction of $\mathcal{D}_{\text{PE}}$, we additionally perform round-trip filtering [174] as qualitative analysis of random QA pairs suggested a higher level of noise in the PE-generated data. This round-trip consistency check is done by comparing the originally generated answer a in $(c, q, a)_l$ with the predicted answer. This predicted answer is obtained by prompting the PLM to answer question q based on passage c. We also tried round-trip filtering for PT generated data; however, we did not observe any gains. We report detailed statistics of the synthetically generated datasets in Section 9.5.

In the construction of $\mathcal{D}_{\text{PT}}$, we prompt-tune the PLM on $\bigcup_{l \in L} \mathcal{T}(\mathcal{D}_{\text{en}})$ or $\bigcup_{l \in L} \mathcal{D}_{l,n}$ as detailed earlier. Prompt tuning is performed with the AdaFactor optimizer [175]. We tune a prompt of length 50 tokens for up to 1,000 steps, evaluating every 50 steps, with a batch size of 16 examples, and a learning rate of 0.3 with a linear warmup of 200 steps. We use early stopping to select the best prompt per language based on BLEU [135] on a held-out dataset from the English TyDiQA-GoldP, translated to each target language.

Question Answering We trained an mT5-XL model [176] for question-answering to evaluate different synthetic data generation methods ($\mathcal{D}_{\text{MT}}$, $\mathcal{D}_{\text{PE}}$, and $\mathcal{D}_{\text{PT}}$). As a baseline, we further use mT5-XL fine-tuned on available training data. Specifically, in the English-Only scenario, Baseline mT5-XL is trained on the English QA data $\mathcal{D}_{en}$. In the Few-shot scenario, Baseline mT5-XL is fine-tuned on n human annotated examples in the target languages (the same number given to PE and PT). We conducted experiments on TyDiQA-GoldP [142] and MLQA [143], see Section 9.3.1.

During downstream QA evaluation, mT5-XL was fine-tuned with AdaFactor, with a learning rate of 0.0002, a batch size of 64, and up to 5,000 steps of training evaluating every 50 steps. We measure QA performance with Exact Match (EM) and F1, and report the unweighted average across languages (excluding English). For TyDiQA-GoldP, we report results on the development split, which is commonly used as an evaluation set since the test split is unavailable. We select mT5 checkpoints per language using EM and report the average of 3 runs. For MLQA, we present results on the test split, selecting the best mT5 checkpoint based on the average EM on the MLQA dev set.

Table 9.3 QA performance (Average EM over three runs) for individual languages on the TYDIQA-GOLDP evaluation set; the backbone of the QA model is mT5-XL fine-tuned on gold (Baseline, Supervised) or synthetically generated data. The final row displays the percent of tokens for each language in the PLM training data.

Method	n-shot	Ar	Bn	Fi	Id	Ko	Ru	Sw	Te	Avg
Baseline	5	65.9	68.4	65.1	71.3	68.4	57.6	60.1	75.4	66.5
MT	0	66.3	62.2	65.2	72.4	63.9	61.1	70.5	67.0	66.1
PE	0	60.4	66.7	63.5	63.6	65.1	53.8	74.5	67.3	64.4
PE + MT	0	**68.1**	70.5	68.2	73.6	**68.5**	61.0	**78.4**	66.9	69.4
QAMELEON (PT)	5	65.4	**76.7**	**69.4**	69.0	67.6	61.5	75.6	**76.7**	70.2
QAMELEON (PT)+MT	5	67.9	72.6	69.2	**73.8**	65.1	**62.8**	77.7	76.1	**70.7**
Supervised	Multi-k	75.7	81.4	74.5	79.8	77.2	72.8	82.6	83.0	78.4
% tokens in PLM	—	*0.15*	*0.03*	*0.42*	*0.16*	*0.19*	*0.53*	*0.01*	*0.02*	—

9.4 Results

QAMELEON (PT) Delivers the Best QA System Table 9.2 summarizes our results on TYDI QA for both English-only and Few-Shot scenarios. Overall, we find that a low resource setting with 5 human-annotated examples in the target language ($\mathcal{D}_{l,5}$) is useful for scaling QA to multiple languages. More specifically, 5-shot prompt tuning gives an EM improvement of 11.7% absolute (58.5% → 70.2%) in exact match answer accuracy on the TYDIQA-GOLDP evaluation set over mT5 fine-tuned on English data only (Baseline), 3.7% (66.5% → 70.2%) over mT5 trained on 5 examples per language (Few-shot Baseline), and 4.1% (66.1% → 70.2%) over mT5 fine-tuned on the data obtained with the MT approach.

QAMELEON further improves over the few-shot results obtained by prompting code-davinci-002 [171], PaLM-540B [122], and Flan-U-PaLM-540B [172], with a similar number of available labeled examples. These approaches directly employ extremely large PLMs for the task of QA, whereas QAMELEON leverages data synthesis to distill a PLM into a much smaller mT5-XL model. It also is important to note that QAMELEON as an approach is orthogonal and possibly complementary to any improvements due to more performant QA models and more sophisticated PLMs (e.g., Flan-U-PaLM-540B).

In both English-only and Few-shot resource scenarios, QAMELEON outperforms the other two

Table 9.4 Comparison of QA performance from training mT5-XL on 5 or 50 examples (Baseline), on synthetic data generated with prompt tuning (PT), or on the full TyDiQA-GoldP training set. Results are averaged across languages.

Method	n-Shot	Avg EM
Baseline	5	66.5
QAmeleon (PT)	5	70.2
Baseline	50	69.3
QAmeleon (PT)	50	73.7
Supervised	Multi-k	78.4

data generation approaches, Machine Translation (MT), and Prompt Engineering (PE). Despite employing PE in two stages, chain-of-thought style, we observe that the generated data leads to lower QA performance. Moreover, we see better performance when using English-Only data in comparison to the Few-Shot scenario, suggesting that the PLM is able to better utilize high-quality English data rather than small amounts of labeled data (in other languages). Finally, augmenting PLM generated data (either via PE or PT) with data generated via MT leads to gains in QA performance over using any of these methods independently. This could be due to the coupling of diverse QA data i.e., language-specific content and task-specific English-centric translated content.

Table 9.3 shows QA performance in individual languages, for each of the methods in Table 9.3 in their best performing setting: Few-shot Baseline, Machine Translation (MT), Prompt Engineering (PE), Prompt Tuning (PT) and augmenting PE and PT with MT. Data generated by QAmeleon (PT) using 5 target examples provides the best performance in Bengali, Finnish and Telugu. A boost can be seen for Arabic, Indonesian, Russian and Swahili when QAmeleon data is combined with MT data. Language distribution listed under '% tokens in PLM' reflects the extremely low representation of many languages in the pre-training corpora of the PLM used in this work. As an upper bound, we additionally show the performance of supervised mT5-XL fine-tuned on large amounts of gold training data (see Table 9.1) to illustrate the remaining gap, which could potentially be bridged by increasing the number of labeled examples or by improved (e.g. more multilingual or FLAN-tuned) PLMs.

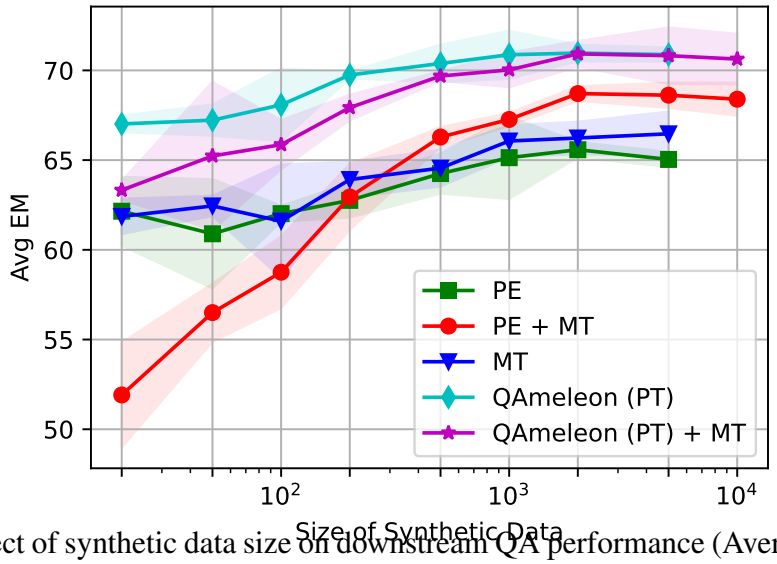


Figure 9.2: Effect of synthetic data size on downstream QA performance (Average EM on TyDiQA-GoldP evaluation set); results shown for mT5-XL QA model fine-tuned via Machine Translation (MT), Prompt Engineering (PE), Prompt Tuning (QAmeleon (PT)), and combinations thereof (PE + MT and QAmeleon (PT) + MT).

Increasing Labeled Examples Improves QA Performance So far, we have tested QAmeleon in an extremely low resource setting, using only 5 examples in the target language. We next examine its performance when we assume a 10-fold increase in the number of annotated examples. Table 9.4 compares the performance of mT5-XL trained on 5 or 50 examples (Baseline), on synthetic QA datasets generated by QAmeleon using 5 or 50 examples, and as an upper bound on the entire TyDiQA-GoldP dataset. As can be seen, increasing the number of examples from 5 to 50 improves performance. It is important to note that in both settings, significant improvements in multilingual QA can be obtained by generating data with QAmeleon instead of fine-tuning the QA model directly on labeled data.

The Larger the Synthetic Data, the Better the QA Model We now study the impact of varying the size of the automatically generated datasets on QA performance. As shown in Figure 9.2, when larger amounts of synthetic data are used for training the QA model, absolute accuracy increases. This improvement is higher when combining PLM-generated data with Translation data in comparison to individual datasets. This can be explained due to the increased diversity of

Table 9.5 BLEU scores and downstream QA performance on TYDIQA-GOLDP for questions generated by mT5-XL and QAMELEON (Few-shot setting, 5 examples in the target language).

Method	n-Shot	Avg BLEU	Avg EM
mT5-XL	5	24.74	57.3
QAMELEON (PT)	5	24.29	70.2

the combined data, which include English-centric translated content and target language-specific content obtained from the PLM. Eventually, we observe a saturation effect: i.e., beyond $O(1,000$ QA pairs in the target language improvements are limited.

BLEU Does not Correlate with Downstream QA Performance An interesting question is whether improvements in QA performance are due to better (e.g., more grammatical or diverse) questions. We assessed the quality of questions generated by QAMELEON (PT) on TYDIQA-GOLDP by measuring their similarity to gold standard questions. We compare this with an mT5-XL model for question generation fine-tuned in a Few-shot setting. Both QAMELEON (PT) and mT5-XL question generation models were given the same number of examples in each language. Table 9.5 reports BLEU [135] scores for these two models; we additionally report question answering performance (in terms of EM) via another set of mT5-XL models fine-tuned on the synthetic data generated by the respective models.

Even though mT5-XL produces questions with slightly higher BLEU score, QAMELEON generates QA data that leads to much higher QA performance. The result underscores the need for better trustworthy automatic evaluation metrics [177] across languages.

Our Results Generalize to MLQA To validate the general applicability of our approach, we evaluate QAMELEON on MLQA [143]. We prompt-tune the PLM on 5 examples per language taken from the MLQA development set, since MLQA does not provide training partitions. We generate synthetic datasets in all of the MLQA languages and compare an English-only baseline, MT, and QAMELEON (PT) approaches as we did previously for TYDIQA-GOLDP. We report results (EM and F1) using mT5-XL as the QA model in Table 9.6, where English is included in the average performance.

Table 9.6 Downstream QA performance on the MLQA test set with an mT5-XL model trained on SQuAD English data (English-Only), SQuAD translated to all MLQA languages (MT), on synthetic data generated by QAmeleon (5-shot) in all MLQA languages, or on a combination of data generated by MT and QAmeleon. Results for [176] and [178] are taken from the respective papers.

Method	Avg EM	Avg F1
English-Only	53.1	71.8
MT	56.4	74.8
QAmeleon (PT)	55.0	74.3
QAmeleon (PT) + MT	**56.8**	**75.3**
mT5-XL [176]	54.5	73.5
XLM-E-XL [178]	57.9	76.2

Table 9.7 Number of synthetic question-answer pairs per language generated via Prompt Engineering (PE) and QAmeleon (PT) with 5 human-labeled examples.

	TyDiQA-GoldP		MLQA
Language	PE	QAmeleon	QAmeleon
Arabic	5,219	8,499	14,738
Bengali	5,948	8,036	—
Chinese	—	—	14,669
Finnish	8,062	5,943	—
German	—	—	11,186
Hindi	—	—	12,036
Indonesian	6,487	7,810	—
Kiswahili	8,003	8,041	—
Korean	5,229	7,906	—
Russian	5,619	7,441	—
Spanish	—	—	10,134
Telugu	2,742	5,222	—
Vietnamese	—	—	13,333
Total	47,309	52,955	89,344

We find that the MT approach is very effective on MLQA, which is not surprising since MLQA questions are translated from English. QAmeleon (PT), however, still delivers an improvement in combination with MT synthetic data. Table 9.6 further reports comparisons with the state-of-the-art models of [176] and [178]. The former is mT5-XL (3.7B parameters) trained on English data only, whereas XLM-E-XL (2.2B parameters) benefits from a different language model pretraining technique. The latter approach is orthogonal and potentially complementary to QAmeleon.

Table 9.8 Examples of QA pairs from human-annotated TyDi QA and generated by QAmeleon (PT) on corresponding passages. English translations from Google Translate are added for readability.

Lang	Human Annotated	QAmeleon (PT)
ar	**Q:** متى تأسست جامعة فرايبورغ؟ **A:** 1457	**Q:** متى تأسست جامعة فرايبورغ؟ **A:** 1457
	Q: When was the University of Freiburg established? A: 1457	*Q: When was the University of Freiburg founded? A: 1457*
sw	**Q:** Je, Kifaru ana urefu kiasi gani? **A:** "mita 3.5-4.6	**Q:** Je , faru mweupe ana uzito gani? **A:** kilogramu 3,500
	Q: How tall is a Rhino? A: 3.5-4.6 meters	*Q: How much does a white rhino weigh? A: 3,500 kilograms*
ru	**Q:** Когда был подписан Георгиевский трактат? **A:** 24 июля (4 августа) 1783 года	**Q:** В какой крепости был заключен Георгиевский трактат? **A:** Георгиевск (Северный Кавказ)
	Q: When was the Treaty of Georgievsk signed? A: July 24 (August 4), 1783	*Q: In what fortress was the Treaty of St. George concluded? A: Georgievsk (North Caucasus)*

9.5 Data Analysis

Table 9.7 shows the size of synthetic data resources generated via Prompt Engineering (PE) and QAmeleon (PT), per language and in total. These were in the range of 47,000-53,000 QA examples for TyDiQA-GoldP, and 89,000 for MLQA. The varying size of the data across languages is due to the filtering described in Section 9.3. In some languages (e.g., Telugu) generation is more noisy, leading to fewer data points. We conjecture that this is due to the PLM being exposed to less data representing these languages during pre-training. We further hypothesize that a more multilingual pre-training of PLMs could potentially lead to better quality data across all languages.

Machine translation (MT) creates the same number of data points as the source training set. For TyDiQA-GoldP, the English training contains 3,696 data points (Table 9.1), leading to approximately 29,000 QA examples across 8 languages. In MLQA, machine translation (MT) uses SQuAD as the English dataset, consisting of ~ 87,000 data points, leading to ~ 522,000 QA examples across 6 languages.

Table 9.8 illustrates randomly picked examples of QA pairs generated by QAmeleon (PT) for passages in the TyDiQA-GoldP eval set. For these passages, we also have access to the human annotated QA pairs. As can be seen, QA pairs generated by QAmeleon are of similar quality and at times diverse in comparison to the human-annotated dataset. Table 9.9 illustrates examples of QA pairs generated by QAmeleon from randomly sampled Wikipedia passages.

Table 9.9 QA pairs (random selection) generated by QAMELEON (PT) on Wikipedia passages. English translations from Google Translate are added for readability.

Language	QAMELEON (PT)	
fi	**Q:** Milloin Tullintori valmistui?	**A:** *1990*
	Q: When was Tullintori completed?	*A: 1990*
ru	**Q:** Какой отраслью экономики преимущественно занято население Узбекистана? **A:** сфера обслуживания и туризма	
	Q: What sector of the economy is predominantly employed by the population of Uzbekistan? A: service and tourism	
sw	**Q:** Nyoka birisi iko katika familia gani?	**A:** Typhlopidae
	Q: Which family is the Birsi snake in?	*A: Typhlopidae*
ar	**Q:** في أي ولاية تقع بلدة فرانكلين ، مقاطعة مانيتووك؟	**A:** ويسكونسن
	Q: In which state is Franklin Township, Manitowoc County?	*A: Wisconsin*
id	**Q:** Siapa pencipta manga DN Angel?	**A:** Yukiru Sugisaki
	Q: Who created the manga DN Angel?	*A: Yukiru Sugisaki*

9.6 Related Work

Data Generation for QA Prior work on the generation of QA data has mostly focused on English and typically divides the task into answer extraction/generation and question generation, followed by some type of filtering. [174] employ round-trip consistency for filtering with BERT-based models. Other work [166] uses BART to jointly generate a question and its answer given an input passage, employing likelihood-based filtering. [179] use a RoBERTa-based passage selection model to identify interesting passages. [180] additionally train the generation models on an adversarial QA dataset, while [181] integrate a QA-pair ranking module.

The above approaches generally require large amounts of labeled QA data in the form of SQUAD [20] or Natural Questions [7] to train passage selection and question generation models. In contrast, we only assume access to a few question-answer pairs per language.

Multilingual QA In this work we used mT5-XL [176] as our reference QA model. We note that a slightly more performant choice could have been ByT5 [182], which reports improvements on TYDIQA-GOLDP by operating directly on raw text instead of sentence pieces. Existing work on low resource multilingual QA has been relatively limited. [183] propose to use automatically translated high-confidence QA examples for training, while other approaches [184, 185] only

generate questions and require supervised training data in the target language. Other approaches [145, 146, 167] focus on zero-shot transfer, i.e., a multilingual model trained on QA data generation on SQuAD (and optionally automatically translated SQuAD data) is applied to other languages. Our work shows that few-shot settings result in better multilingual generation quality in comparison to zero-shot models.

Prompting Existing work [121, 153] has shown that prompting pre-trained large language models can lead to strong performance in a wide range of tasks including natural language generation and common sense reasoning. In the context of multilingual QA, [122] employ a single prompt and a few labeled examples in the target language. In contrast, we employ chain-of-thought prompting, and English answers and questions as a bridge. Moreover, our experiments with QAmeleon demonstrate that prompt tuning is superior and a viable alternative to large-scale annotation. Prompting in multilingual settings has achieved the best performance using English prompts and target language exemplars [157, 160, 159]. We demonstrate that parameter-efficient methods such as prompt tuning using target language exemplars [163] is a superior choice.

9.7 Conclusions

In this chapter, we examined the ability of pre-trained language models to generate synthetic data for bootstrapping multilingual QA systems, with as few as five examples in a new target language. We introduced QAmeleon, a parameter efficient approach which uses prompt tuning to automatically create multilingual QA data. Extensive experiments under different resource scenarios demonstrate that QAmeleon is superior to prompt engineering and competitive baselines based on machine translation. In future, we would like to extend this approach to other multilingual tasks, including retrieval, summarization, and semantic parsing.

Epilogue

At the beginning of this work, we set out to explore new ways to advance question answering, focusing on challenging and realistic benchmarks such as Natural Questions. While major strides have been made in the field over the last several years, question answering remains an unsolved problem with many open research questions. Even if today we have approaches capable of outperforming humans on many English extractive question answering benchmarks, we are still below human performance when it comes to factuality, grounding and logical consistency of generative question answering systems, as well as extractive question answering for low resource languages.

In this work, we described a number of techniques that can be used to improve question answering systems, well beyond simple baselines based on pretrained transformers (Chapter 2). We found that sparse attention mechanisms (Chapter 3) can be used to include much more textual context for answering questions. Visual tokens (Chapter 4) can be used to ground question answering to objects detected within images. Combinations of learned models (Chapter 5) can be used to verify the output of question answering models and to improve quality through uptraining. We showed how structured explanations (Chapter 6) can be utilized to improve question answering accuracy as well as the ability of users to quickly judge whether an answer is correct. And most recently, we found that improvements can also be achieved with prompt-tuned LLMs in low resource languages (Chapter 9), requiring very few hand labeled examples.

We also studied question answering and the problem of generating grounded text from a semantic perspective. We found that tools from formal semantics (Chapter 7) can help develop richer conceptualizations of question answering and explain aspects of the distribution of question

answer pairs in datasets. We also found that representations from open information extraction (Chapter 8) can be used to reformulate language models as encoders of semantic messages into text, and that the factuality of generated text can be improved by increasing the probability of correct reconstruction of the intended semantic content.

Some of the findings discussed in this work are now relatively well known methods. It is not surprising that sparse attention and long contexts can lead to improvements in many question answering tasks, or that early fusion of visual feature works well in visual question answering, or that uptraining with critic models can lead to improvements, and that LLMs allow this approach to work even in cases where data is extremely scarce. What might be surprising is how very few of these findings have found their way into some of the latest generations of LLMs. PaLM [122] in 2022, for instance, has been trained in almost the same way as T5 [25] in 2019, just scaling size and training data, including very little of the large array of modeling improvements that have been found by researchers over the years. This seems to suggest that a good deal of performance is being left on the table and that we are likely to see many improvements in the coming years just from scaling up some of the more successful techniques that have been found in the last few years.

We also believe that some of the connections we explored with semantics and open information extraction are potentially fruitful areas of future research. For example, it might be suboptimal to train large language models without giving them any access to collections of facts or to knowledge graphs, leaving them to learn information extraction at the same time as they are learning language itself. We argue that the framing and technique we described in Chapter 8 could help shape powerful pretraining tasks for future large language models.

We hope that some of the techniques and challenges described in this work will inspire new research in question answering, and, more generally, in language understanding and language generation, helping create newer and better system for everyone to access useful and accurate information.

References

[1] W. Lehnert, "Human and computational question answering", *Cognitive Science*, vol. 1, no. 1, pp. 47–73, 1977.

[2] M. E. Peters *et al.*, "Deep contextualized word representations", *arXiv preprint arXiv:1802.05365*, 2018.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding", *arXiv preprint arXiv:1810.04805*, 2018.

[4] C. Raffel *et al.*, *Exploring the limits of transfer learning with a unified text-to-text transformer*, 2019.

[5] A. Chowdhery *et al.*, "Palm: Scaling language modeling with pathways", *CoRR*, 2022.

[6] L. Ouyang *et al.*, "Training language models to follow instructions with human feedback", *arXiv preprint arXiv:2203.02155*, 2022.

[7] T. Kwiatkowski *et al.*, "Natural questions: A benchmark for question answering research", *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 453–466, 2019.

[8] C. Alberti, K. Lee, and M. Collins, "A bert baseline for the natural questions", *arXiv preprint arXiv:1901.08634*, 2019.

[9] Y. Liu *et al.*, "Roberta: A robustly optimized bert pretraining approach.", *arXiv preprint arXiv:1907.11692*, 2019.

[10] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding.", *arXiv preprint arXiv:1807.03748*, 2018.

[11] R. Zellers, Y. Bisk, A. Farhadi, and Y. Choi, "From recognition to cognition: Visual commonsense reasoning", in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[12] P. Sharma, N. Ding, S. Goodman, and R. Soricut, "Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning", in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018, pp. 2556–2565.

[13] C. Chapman and I. Kučerová, "Structural and semantic ambiguity of why-questions: An overlooked case of weak islands in english", *Proceedings of the Linguistic Society of America*, vol. 1, pp. 15–1, 2016.

[14] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for squad", *arXiv preprint arXiv:1806.03822*, 2018.

[15] T. Castro Ferreira *et al.*, "The 2020 bilingual, bi-directional WebNLG+ shared task: Overview and evaluation results (WebNLG+ 2020)", in *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, Dublin, Ireland (Virtual): Association for Computational Linguistics, Dec. 2020, pp. 55–76.

[16] C. Gardent, A. Shimorina, S. Narayan, and L. Perez-Beltrachini, "The WebNLG challenge: Generating text from RDF data", in *Proceedings of the 10th International Conference on Natural Language Generation*, Santiago de Compostela, Spain: Association for Computational Linguistics, Sep. 2017, pp. 124–133.

[17] C. Clark and M. Gardner, "Simple and effective multi-paragraph reading comprehension", *arXiv preprint arXiv:1710.10723*, 2017.

[18] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference", *arXiv preprint arXiv:1606.01933*, 2016.

[19] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading wikipedia to answer open-domain questions", *arXiv preprint arXiv:1704.00051*, 2017.

[20] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text", in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 2383–2392.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

[22] A. Vaswani *et al.*, "Attention is all you need", in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[23] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding.", *Advances in neural information processing systems*, 2019.

[24] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations.", *arXiv preprint arXiv:1909.11942*, 2019.

[25] C. Raffel *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer", *arXiv preprint arXiv:1910.10683*, 2019.

[26] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers.", *arXiv preprint arXiv:1904.10509*, 2019.

[27] S. Sukhbaatar, E. Grave, P. Bojanowski, and A. Joulin, "Adaptive attention span in transformers.", *arXiv preprint arXiv:1905.07799*, 2019.

[28] J. W. Rae, A. Potapenko, S. M. Jayakumar, and T. P. Lillicrap, "Compressive transformers for long-range sequence modelling.", *arXiv preprint arXiv:1911.05507*, 2019.

[29] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The efficient transformer.", *arXiv preprint arXiv:2001.04451*, 2020.

[30] M. E. Peters and A. Cohan, "Longformer: The long-document transformer.", *arXiv preprint arXiv:2004.05150v1*, 2020.

[31] X. Zhang, F. Wei, and M. Zhou, "Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization.", *arXiv preprint arXiv:1905.06566*, 2019.

[32] M. Joshi, O. Levy, D. S. Weld, and L. Zettlemoyer, "Bert for coreference resolution: Baselines and analysis.", *arXiv preprint arXiv:1908.09091*, 2019.

[33] Z. Yang *et al.*, "Hotpotqa: A dataset for diverse, explainable multi-hop question answering.", *arXiv preprint arXiv:1809.09600*, 2018.

[34] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context.", *arXiv preprint arXiv:1901.02860*, 2019.

[35] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations.", *arXiv preprint arXiv:1803.02155*, 2018.

[36] J. Welbl, P. Stenetorp, and S. Riedel, "Constructing datasets for multi-hop reading comprehension across documents.", *Transactions of the Association for Computational Linguistics*, 2018.

[37] L. Xiong, C. Hu, C. Xiong, D. Campos, and A. Overwijk, "Open domain web keyphrase extraction beyond language modeling.", *arXiv preprint arXiv:1911.02671*, 2019.

[38] M. Zaheer *et al.*, "Big bird: Transformers for longer sequences.", *arXiv preprint arXiv:2007.14062*, 2020.

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

[40] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy, "Race: Large-scale reading comprehension dataset from examinations.", *arXiv preprint arXiv:1704.04683*, 2017.

[41] G. M. Correia, V. Niculae, and A. F. Martins, "Adaptively sparse transformers.", *arXiv preprint arXiv:1909.00015*, 2019.

[42] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt, "Practical and optimal lsh for angular distance.", *Advances in neural information processing systems*, 2015.

[43] A. Roy, M. Saffar, A. Vaswani, and D. Grangier, "Efficient content-based sparse attention with routing transformers.", *arXiv preprint arXiv:2003.05997*, 2020.

[44] S. Maruf, A. F. Martins, and G. Haffari, "Selective attention for context-aware neural machine translation.", *arXiv preprint arXiv:1903.08788*, 2019.

[45] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification.", *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 2016.

[46] L. Miculicich, D. Ram, N. Pappas, and J. Henderson, "Document-level neural machine translation with hierarchical attention networks.", *arXiv preprint arXiv:1809.01576*, 2018.

[47] T. Shen, T. Zhou, G. Long, J. Jiang, and C. Zhang, "Bi-directional block self-attention for fast and memory-efficient sequence modeling.", *arXiv preprint arXiv:1804.00857*, 2018.

[48] Z. Ye, Q. Guo, Q. Gan, X. Qiu, and Z. Zhang, "Bp-transformer: Modelling long-range context via binary partitioning.", *arXiv preprint arXiv:1911.04070*, 2019.

[49] P. J. Liu *et al.*, "Generating wikipedia by summarizing long sequences.", *arXiv preprint arXiv:1801.10198*, 2018.

[50] Q. Guo, X. Qiu, P. Liu, Y. Shao, X. Xue, and Z. Zhang, "Star-transformer.", *arXiv preprint arXiv:1902.09113*, 2019.

[51] P. Shaw, P. Massey, A. Chen, F. Piccinno, and Y. Altun, "Generating logical forms from graph representations of text and entities.", *arXiv preprint arXiv:1905.08407*, 2019.

[52] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model.", *IEEE Transactions on Neural Networks*, 2008.

[53] S. Rothe, S. Narayan, and A. Severyn, "Leveraging pre-trained checkpoints for sequence generation tasks.", *Transactions of the Association for Computational Linguistics*, 2020.

[54] Y. You *et al.*, "Large batch optimization for deep learning: Training bert in 76 minutes.", *International Conference on Learning Representations*, 2019.

[55] D. Liu *et al.*, "Rikinet: Reading wikipedia pages for natural question answering.", *arXiv preprint arXiv:2004.14560*, 2020.

[56] C. Xiong, Z. Liu, Z. Liu, and J. Bao, "Joint keyphrase chunking and salience ranking with bert.", *arXiv preprint arXiv:2004.13639*, 2020.

[57] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse, "The reversible residual network: Backpropagation without storing activations.", *Advances in neural information processing systems*, 2017.

[58] C. Alberti, J. Ling, M. Collins, and D. Reitter, "Fusion of detected objects in text for visual question answering", 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)*, 2019, pp. 2131–2140.

[59] S. Antol *et al.*, "VQA: Visual Question Answering", in *International Conference on Computer Vision (ICCV)*, 2015.

[60] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh, "Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering", in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[61] D. A. Hudson and C. D. Manning, "Gqa: A new dataset for compositional question answering over real-world images", *arXiv preprint arXiv:1902.09506*, 2019.

[62] L. Y. Wu, A. Fisch, S. Chopra, K. Adams, A. Bordes, and J. Weston, "Starspace: Embed all the things!", in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[63] D. Gillick, A. Presta, and G. S. Tomar, "End-to-end retrieval in continuous space", *arXiv preprint arXiv:1811.08008*, 2018.

[64] J. Weston, S. Bengio, and N. Usunier, "Wsabie: Scaling up to large vocabulary image annotation", in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[65] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context", in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[66] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks", in *European conference on computer vision*, Springer, 2016, pp. 630–645.

[67] P. Anderson *et al.*, "Bottom-up and top-down attention for image captioning and visual question answering", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6077–6086.

[68] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu, "The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision", in *ICLR*, 2019.

[69] D. A. Hudson and C. D. Manning, "Compositional attention networks for machine reasoning", in *ICLR*, 2018.

[70] J. Lu, D. Batra, D. Parikh, and S. Lee, "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks", *arXiv preprint arXiv:1908.02265*, 2019.

[71] W. Su *et al.*, "Vl-bert: Pre-training of generic visual-linguistic representations", *arXiv preprint arXiv:1908.08530*, 2019.

[72] G. Li, N. Duan, Y. Fang, D. Jiang, and M. Zhou, "Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training", *arXiv preprint arXiv:1908.06066*, 2019.

[73] L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang, "Visualbert: A simple and performant baseline for vision and language", *arXiv preprint arXiv:1908.03557*, 2019.

[74] P. Zhang, Y. Goyal, D. Summers-Stay, D. Batra, and D. Parikh, "Yin and Yang: Balancing and answering binary visual questions", in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[75] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C Lawrence Zitnick, and R. Girshick, "Clevr: A diagnostic dataset for compositional language and elementary visual reasoning", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2901–2910.

[76] C. Alberti, D. Andor, E. Pitler, J. Devlin, and M. Collins, "Synthetic qa corpora generation with roundtrip consistency", in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 6168–6173.

[77] T. Kwiatkowski *et al.*, "Natural questions: A benchmark for question answering research", *Transactions of the Association of Computational Linguistics*, 2019.

[78] Y. Sun *et al.*, "Ernie: Enhanced representation through knowledge integration", *CoRR*, vol. abs/1904.09223, 2019.

[79] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova, "Boolq: Exploring the surprising difficulty of natural yes/no questions", *arXiv preprint arXiv:1905.10044*, 2019.

[80] S. Reddy, D. Chen, and C. D. Manning, "Coqa: A conversational question answering challenge", *arXiv preprint arXiv:1808.07042*, 2018.

[81] J. Devlin, M.-W. Chang, K. Lee, and K. 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)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[82] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning", *arXiv preprint arXiv:1702.08608*, 2017.

[83] U. Ehsan, P. Tambwekar, L. Chan, B. Harrison, and M. O. Riedl, "Automated rationale generation: A technique for explainable ai and its effects on human perceptions", in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, ACM, 2019, pp. 263–274.

[84] T. Kwiatkowski *et al.*, "Natural questions: A benchmark for question answering research", *Transactions of the Association of Computational Linguistics*, 2019.

[85] U. Ehsan, B. Harrison, L. Chan, and M. O. Riedl, "Rationalization: A neural machine translation approach to generating natural language explanations", in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, 2018, pp. 81–87.

[86] A. Jacovi and Y. Goldberg, "Towards faithfully interpretable nlp systems: How should we define and evaluate faithfulness?", *arXiv preprint arXiv:2004.03685*, 2020.

[87] B. Russell, "On denoting", *Mind*, vol. 14, no. 56, pp. 479–493, 1905.

[88] H. H. Clark and C. R. Marshall, "Definite knowledge and mutual knowledge", *Elements of Discourse Understanding*, 1981.

[89] M. Tomasello, M. Carpenter, and U. Liszkowski, "A new look at infant pointing", *Child development*, vol. 78, no. 3, pp. 705–722, 2007.

[90] H. H. Clark, "Bridging", in *Theoretical issues in natural language processing*, 1975.

[91] G. N. Carlson, "A unified analysis of the english bare plural", *Linguistics and philosophy*, vol. 1, no. 3, pp. 413–457, 1977.

[92] M. Krifka, "Bare nps: Kind-referring, indefinites, both, or neither?", in *Semantics and linguistic theory*, vol. 13, 2003, pp. 180–203.

[93] B. Abbott, "Definiteness and indefiniteness", *The handbook of pragmatics*, vol. 122, 2004.

[94] L. Mikkelsen, "Copular clauses", in *Semantics: An International Handbook of Natural Language Meaning*, C. Maienborn, K. von Heusinger, and P. Portner, Eds., vol. 2, Berlin: Mouton De Gruyter, 2011, pp. 1805–1829.

[95] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "Spanbert: Improving pre-training by representing and predicting spans", *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 64–77, 2019.

[96] K. Lee, L. He, M. Lewis, and L. Zettlemoyer, "End-to-end neural coreference resolution", in *EMNLP*, 2017.

[97] S. Pradhan, A. Moschitti, N. Xue, O. Uryupina, and Y. Zhang, "Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes", in *EMNLP-CoNLL Shared Task*, 2012.

[98] B. Goodrich, J. Gabry, I. Ali, and S. Brilleman, *Rstanarm: Bayesian applied regression modeling via Stan.* R package version 2.19.3, 2020.

[99] P. Lipton, "What good is an explanation?", in *Explanation*, Springer, 2001, pp. 43–59.

[100] A. S. Ross, M. C. Hughes, and F. Doshi-Velez, "Right for the right reasons: Training differentiable models by constraining their explanations", *arXiv preprint arXiv:1703.03717*, 2017.

[101] Z. C. Lipton, "The mythos of model interpretability", *arXiv preprint arXiv:1606.03490*, 2016.

[102] S. Wiegreffe and Y. Pinter, "Attention is not not explanation", *arXiv preprint arXiv:1908.04626*, 2019.

[103] S. Jain and B. C. Wallace, "Attention is not explanation", *arXiv preprint arXiv:1902.10186*, 2019.

[104] A. K. Mohankumar, P. Nema, S. Narasimhan, M. M. Khapra, B. V. Srinivasan, and B. Ravindran, *Towards transparent and explainable attention models*, 2020. arXiv: 2004.14243 [cs.CL].

[105] O.-M. Camburu, T. Rocktäschel, T. Lukasiewicz, and P. Blunsom, "E-snli: Natural language inference with natural language explanations", in *Advances in Neural Information Processing Systems*, 2018, pp. 9539–9549.

[106] O.-M. Camburu, B. Shillingford, P. Minervini, T. Lukasiewicz, and P. Blunsom, *Make up your mind! adversarial generation of inconsistent natural language explanations*, 2019. arXiv: 1910.03065 [cs.CL].

[107] S. Narang, C. Raffel, K. Lee, A. Roberts, N. Fiedel, and K. Malkan, *Wt5?! training text-to-text models to explain their predictions*, 2020. arXiv: 2004.14546 [cs.CL].

[108] S. Kumar and P. Talukdar, *Nile : Natural language inference with faithful natural language explanations*, 2020. arXiv: 2005.12116 [cs.CL].

[109] L. Abzianidze *et al.*, "The parallel meaning bank: Towards a multilingual corpus of translations annotated with compositional meaning representations", *arXiv preprint arXiv:1702.03964*, 2017.

[110] T. Wolfson *et al.*, "Break it down: A question understanding benchmark", *Transactions of the Association for Computational Linguistics*, vol. 8, 183–198, 2020.

[111] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora", in *Proceedings of the 14th conference on Computational linguistics-Volume 2*, Association for Computational Linguistics, 1992, pp. 539–545.

[112] E. Miltsakaki, R. Prasad, A. K. Joshi, and B. L. Webber, "The penn discourse treebank.", in *LREC*, 2004.

[113] M. Lamm, A. T. Chaganty, C. D. Manning, D. Jurafsky, and P. Liang, "Textual analogy parsing: What's shared and what's compared among analogous facts", *arXiv preprint arXiv:1809.02700*, 2018.

[114] N. Tandon, B. Dalvi, K. Sakaguchi, P. Clark, and A. Bosselut, "Wiqa: A dataset for "what if..." reasoning over procedural text", 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)*, 2019, pp. 6078–6087.

[115] S. Min, J. Michael, H. Hajishirzi, and L. Zettlemoyer, *Ambigqa: Answering ambiguous open-domain questions*, 2020. arXiv: 2004.10645 [cs.CL].

[116] A. Kratzer *et al.*, "Modality", *Semantics: An international handbook of contemporary research*, vol. 7, pp. 639–650, 1991.

[117] I. Heim, "Notes on interrogative semantics", *Class notes*, 2000.

[118] A. C. Bale, "Quantifiers and verb phrases: An exploration of propositional complexity", *Natural Language & Linguistic Theory*, vol. 25, pp. 447–483, 2007.

[119] N. Kim, M. Yoshida, and A. Wellwood, "Two whys in english", *Talk presented at SWAMP*, 2015.

[120] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks", *Advances in neural information processing systems*, vol. 27, 2014.

[121] T. Brown *et al.*, "Language models are few-shot learners", *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[122] A. Chowdhery *et al.*, "Palm: Scaling language modeling with pathways", *arXiv preprint arXiv:2204.02311*, 2022.

[123] J. W. Rae *et al.*, "Scaling language models: Methods, analysis & insights from training gopher", *arXiv preprint arXiv:2112.11446*, 2021.

[124] R. Thoppilan *et al.*, "Lamda: Language models for dialog applications", *arXiv preprint arXiv:2201.08239*, 2022.

[125] E. M. Bender, T. Gebru, A. McMillan-Major, and S. 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*, ser. FAccT '21, New York, NY, USA: Association for Computing Machinery, 2021, 610–623, ISBN: 9781450383097.

[126] Y. Miao and P. Blunsom, "Language as a latent variable: Discrete generative models for sentence compression", in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 319–328.

[127] M. Sun, X. Li, and P. Li, "Logician and orator: Learning from the duality between language and knowledge in open domain", in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 2119–2130.

[128] S. Wiseman, S. M. Shieber, and A. M. Rush, "Challenges in data-to-document generation", in *EMNLP*, 2017.

[129] E. Manning and N. Schneider, "Referenceless parsing-based evaluation of amr-to-english generation", in *Proceedings of the 2nd Workshop on Evaluation and Comparison of NLP Systems*, 2021, pp. 114–122.

[130] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration", in *International Conference on Learning Representations*, 2019.

[131] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning", in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 3045–3059.

[132] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation", *arXiv preprint arXiv:1308.3432*, 2013.

[133] G. Hinton, N. Srivastava, and K. Swersky, *Neural networks for machine learning*, Lecture 15d, 3:05-3:35, 2012.

[134] C. Gardent, A. Shimorina, S. Narayan, and L. Perez-Beltrachini, "Creating training corpora for NLG micro-planners", in *Proceedings of the 55th Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers), Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 179–188.

[135] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation", in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[136] S. Banerjee and A. Lavie, "Meteor: An automatic metric for mt evaluation with improved correlation with human judgments", in *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, 2005, pp. 65–72.

[137] M. Popović, "Chrf++: Words helping character n-grams", in *Proceedings of the second conference on machine translation*, 2017, pp. 612–618.

[138] J. Devlin, M.-W. Chang, K. Lee, and K. 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)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[139] M. Lewis *et al.*, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension", in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 7871–7880.

[140] C. Raffel *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer", *Journal of machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[141] A. Trischler *et al.*, "NewsQA: A machine comprehension dataset", in *Proceedings of the 2nd Workshop on Representation Learning for NLP*, Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 191–200.

[142] J. H. Clark *et al.*, "TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages", *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 454–470, 2020.

[143] P. Lewis, B. Oguz, R. Rinott, S. Riedel, and H. Schwenk, "MLQA: Evaluating cross-lingual extractive question answering", in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 7315–7330.

[144] M. Artetxe, S. Ruder, and D. Yogatama, "On the cross-lingual transferability of monolingual representations", in *Proceedings of the 58th Annual Meeting of the Association for*

Computational Linguistics, Online: Association for Computational Linguistics, Jul. 2020, pp. 4623–4637.

[145] A. Riabi, T. Scialom, R. Keraron, B. Sagot, D. Seddah, and J. Staiano, "Synthetic data augmentation for zero-shot cross-lingual question answering", in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 7016–7030.

[146] S. Shakeri, N. Constant, M. Kale, and L. Xue, "Towards zero-shot multilingual synthetic question and answer generation for cross-lingual reading comprehension", in *Proceedings of the 14th International Conference on Natural Language Generation*, Aberdeen, Scotland, UK: Association for Computational Linguistics, Aug. 2021, pp. 35–45.

[147] M. Koppel and N. Ordan, "Translationese and its dialects", in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA: Association for Computational Linguistics, Jun. 2011, pp. 1318–1326.

[148] M. Artetxe, G. Labaka, and E. Agirre, "Translation artifacts in cross-lingual transfer learning", in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 7674–7684.

[149] Y.-H. Lin *et al.*, "Choosing Transfer Languages for Cross-Lingual Learning", in *Proceedings of ACL 2019*, 2019. arXiv: 1905.12688.

[150] D. Garrette and J. Baldridge, "Learning a part-of-speech tagger from two hours of annotation", in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, Georgia: Association for Computational Linguistics, Jun. 2013, pp. 138–147.

[151] M. Zhao *et al.*, "A closer look at few-shot crosslingual transfer: The choice of shots matters", 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)*, Online: Association for Computational Linguistics, Aug. 2021, pp. 5751–5767.

[152] T. Sherborne and M. Lapata, *Meta-learning a cross-lingual manifold for semantic parsing*, 2022.

[153] T. Schick and H. 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*, Online: Association for Computational Linguistics, Jun. 2021, pp. 2339–2352.

[154] D. Khashabi *et al.*, "UNIFIEDQA: Crossing format boundaries with a single QA system", in *Findings of the Association for Computational Linguistics: EMNLP 2020*, Online: Association for Computational Linguistics, Nov. 2020, pp. 1896–1907.

[155] T. Tang, J. Li, W. X. Zhao, and J.-R. Wen, *Context-tuning: Learning contextualized prompts for natural language generation*, 2022.

[156] K. Yang, Y. Tian, N. Peng, and D. Klein, *Re3: Generating longer stories with recursive reprompting and revision*, 2022.

[157] G. I. Winata, A. Madotto, Z. Lin, R. Liu, J. Yosinski, and P. Fung, "Language models are few-shot multilingual learners", in *Proceedings of the 1st Workshop on Multilingual Representation Learning*, Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 1–15.

[158] M. Zhao and H. Schütze, "Discrete and soft prompting for multilingual models", in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 8547–8555.

[159] F. Shi *et al.*, *Language models are multilingual chain-of-thought reasoners*, 2022.

[160] X. V. Lin *et al.*, "Few-shot Learning with Multilingual Language Models", in *Proceedings of EMNLP 2022*, 2022. arXiv: 2112.10668.

[161] Z. Dai *et al.*, "Promptagator: Few-shot dense retrieval from 8 examples", *arXiv preprint arXiv:2209.11755*, 2022.

[162] S. Ruder *et al.*, "XTREME-R: Towards more challenging and nuanced multilingual evaluation", in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 10 215–10 245.

[163] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning", in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 3045–3059.

[164] Z. Yang *et al.*, "HotpotQA: A dataset for diverse, explainable multi-hop question answering", in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 2369–2380.

[165] P. Bajaj *et al.*, "Ms marco: A human generated machine reading comprehension dataset", *arXiv preprint arXiv:1611.09268*, 2016.

[166] S. Shakeri *et al.*, "End-to-end synthetic data generation for domain adaptation of question answering systems", in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 5445–5460.

[167] A. Kramchaninova and A. Defauw, "Synthetic data generation for multilingual domain-adaptable question answering systems", in *Proceedings of the 23rd Annual Conference of the European Association for Machine Translation*, Ghent, Belgium: European Association for Machine Translation, Jun. 2022, pp. 151–160.

[168] T. Vu, A. Barua, B. Lester, D. Cer, M. Iyyer, and N. Constant, *Overcoming catastrophic forgetting in zero-shot cross-lingual generation*, 2022.

[169] J. Wei *et al.*, "Chain of thought prompting elicits reasoning in large language models", *arXiv preprint arXiv:2201.11903*, 2022.

[170] G. Attardi, *Wikiextractor*, `https://github.com/attardi/wikiextractor`, 2015.

[171] M. Chen *et al.*, "Evaluating large language models trained on code", *arXiv preprint arXiv:2107.03374*, 2021.

[172] H. W. Chung *et al.*, "Scaling instruction-finetuned language models", *arXiv preprint arXiv:2210.11416*, 2022.

[173] Y. Wu *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation", *arXiv preprint arXiv:1609.08144*, 2016.

[174] C. Alberti, D. Andor, E. Pitler, J. Devlin, and M. Collins, "Synthetic QA corpora generation with roundtrip consistency", in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 6168–6173.

[175] N. Shazeer and M. Stern, "Adafactor: Adaptive learning rates with sublinear memory cost", in *International Conference on Machine Learning*, PMLR, 2018, pp. 4596–4604.

[176] L. Xue *et al.*, "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*, 2021, pp. 483–498.

[177] T. Sellam, D. Das, and A. Parikh, "Bleurt: Learning robust metrics for text generation", in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 7881–7892.

[178] Z. Chi *et al.*, "Xlm-e: Cross-lingual language model pre-training via electra", in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022, pp. 6170–6182.

[179] P. Lewis *et al.*, "PAQ: 65 million probably-asked questions and what you can do with them", *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 1098–1115, 2021.

[180] M. Bartolo, T. Thrush, R. Jia, S. Riedel, P. Stenetorp, and D. Kiela, "Improving question answering model robustness with synthetic adversarial data generation", in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 8830–8848.

[181] B. Yao *et al.*, "It is AI's turn to ask humans a question: Question-answer pair generation for children's story books", in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 731–744.

[182] L. Xue *et al.*, "Byt5: Towards a token-free future with pre-trained byte-to-byte models", *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 291–306, 2022.

[183] K. Lee, K. Yoon, S. Park, and S.-w. Hwang, "Semi-supervised training data generation for multilingual question answering", in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan: European Language Resources Association (ELRA), May 2018.

[184] V. Kumar, N. Joshi, A. Mukherjee, G. Ramakrishnan, and P. Jyothi, "Cross-lingual training for automatic question generation", in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 4863–4872.

[185] Z. Chi, L. Dong, F. Wei, W. Wang, X.-L. Mao, and H. Huang, "Cross-lingual natural language generation via pre-training", in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 7570–7577.