UNIVERSITÀ
DI TRENTO
Department of
Information Engineering and Computer Science

Doctoral School in
Information and Communication Technology

Loss and Reward Functions for Generative Question Answering Systems

Matteo Gabburo

Advisor
Prof. Alessandro Moschitti
Amazon AGI

Feb 2025

Abstract

Recent advancements in AI, mainly through Large Language Models (LLMs), have transformed the field, driving both industrial and academic progress. These models are typically trained using causal language modeling tasks, which, while effective, face certain limitations compared to traditional machine learning frameworks. For example, they lack structured and easy-to-implement approaches to incorporate negative examples, making learning complex instructions or reasoning tasks challenging, such as distinguishing high-quality answers from suboptimal ones.

In this Thesis, we propose novel methods to enhance the training of generative models using classifiers trained on positive and negative examples. Our first contribution is detailed in "Knowledge Transfer from Answer Ranking to Answer Generation", which Addresses the challenge of training generative QA models with limited supervised data. We demonstrate how knowledge from Answer Sentence Selection (AS2) models can be transferred to generative QA (GenQA) models by leveraging top-ranked candidates as generation targets and lower-ranked candidates as contextual input. Our approach further incorporates AS2 prediction scores for loss weighting and input/output shaping, significantly improving GenQA performance across multiple academic and industrial datasets. Building on this foundation, we extended our approach to automatic evaluation systems for QA. In particular, we designed SQuAre (Sentence-level QUestion AnsweRing Evaluation), an innovative reference-based metric that utilizes multiple positive and negative references for evaluating QA systems. As described in our paper, SQuAre consistently outperforms previous metrics in correlating with human evaluations, making it a reliable tool for assessing both extractive (AS2) and generative (GenQA) QA systems. Additionally, we explored the use of scores from automatic evaluation systems to further enhance generative QA models. Our research introduced strategies to transfer knowledge from QA evaluation models to generative models, including augmenting training data with QA-evaluated answers and weighted generator loss using evaluation scores. This approach, validated across several datasets, achieves state-of-the-art accuracy in answer generation.

Finally, we contributed to retrieval-augmented generation (RAG) research by introducing Retrieval Complexity (RC), a novel metric that measures the difficulty of the question based on document completeness and retrieval performance. Our unsupervised RC estimation pipeline identifies complex question types, such as multi-hop, compositional, and temporal queries, enabling retrieval systems to better address high-difficulty questions and target areas for improvement.

Contents

1 Introduction and Contributions **1**

1.1 Research Gaps and Contributions of the Thesis 1

1.1.1 Generating Quality Assessment for QA models 2

1.1.2 Train Without Human-Annotated Data 2

1.1.3 Complexity of Questions 3

1.2 Publications 4

1.3 Thesis Structure 5

2 Background **7**

2.1 Natural Language Processing 8

2.1.1 Early Conceptualization and Rule-Based Approaches (1950–1990) 8

2.1.2 The Rise of Statistical Methods (1990–2010) 8

2.1.3 The Deep Learning Revolution and Word Embeddings (2010–Present) 9

2.1.4 Applications and Core NLP Components 9

2.1.5 Focus on Question Answering (QA) 11

2.2 The Transformer Architecture 12

2.2.1 Architectural variations 14

2.2.2 Improvements to the original architecture 14

2.3 Pretraining Techniques and Pretrained Models 17

2.3.1 Pretraining Techniques 17

2.3.2 Pre-training datasets 20

2.3.3 Existing Pretrained Models 21

2.4 Question Answering 25

2.4.1 Question Answering Pipeline 25

2.4.2 Retrieval: Document Retrieval & Semantic Similarity 27

2.4.3 Ranking: Passage Ranking & Answer Sentence Selection 29

2.4.4 Getting an Answer: Machine Reading & Generative QA 30

2.5 Datasets 31

2.5.1 Benchmark Datasets to Evaluate Language Models 32

2.5.2 QA datasets 35

2.5.3 Industrial QA Datasets 38

2.5.4 Complex QA datasets 39

3 Understanding Language Models **43**
3.1 Case study 1: The Transformer Architecture and the Code Domain . . 44
3.1.1 The Evolution of Code Completion and Transformer Applications in IDEs . . . 44
3.1.2 Our Approach: Efficient and Versatile Transformer-Based Code Models . . . 45
3.1.3 Tasks . . . 46
3.1.4 Datasets for code . . . 47
3.1.5 Baseline Models . . . 48
3.1.6 Our Disjoint Encoder-Decoder Model . . . 49
3.1.7 Experiments . . . 51
3.2 Case study 2: Efficient pretraining techniques for transformer models . 58
3.2.1 Efficient Pretraining for Transformer Encoders . . . 58
3.2.2 Reducing Pretraining Costs in Transformer-based Models . . . 58
3.2.3 Effective Pre-training Objectives . . . 59
3.2.4 Experimental Setting . . . 62
3.2.5 Results . . . 64
3.2.6 The role of clustering . . . 67
3.2.7 Negative results: Non impactful objectives . . . 70
3.2.8 Final discussion . . . 70
3.3 Case study 3: Loss manipulation and automatic translation for AS2 tasks 71
3.3.1 Introduction . . . 71
3.3.2 improving Multilingual QA with Translated AS2 Corpora . . . 72
3.3.3 AS2 Translated Datasets . . . 73
3.3.4 AS2 Translated Datasets . . . 73
3.3.5 AS2 Reward Loss . . . 76
3.3.6 Experiments . . . 78
3.3.7 Ablation: Passage Ranking . . . 80
3.3.8 Ablation: Cross-Lingual . . . 82
3.3.9 Ablation: Ranks correlation . . . 82

4 Automatic Evaluation for Generative Question Answering **85**
4.1 Automatic Evaluation Metrics for QA . . . 86
4.2 SQuARe: Sentence-level QA Evaluation . . . 88
4.3 Experiments and Results . . . 91
4.3.1 Datasets . . . 91
4.3.2 Models and Baselines . . . 91
4.3.3 Results . . . 94
4.4 Qualitative Error Analysis . . . 99
4.4.1 Metrics Comparison . . . 101
4.5 Conclusion and Discussion . . . 105

5 Knowledge Transfer From AS2 to Generative QA **107**
5.1 Generative Models for Question Answering . . . 109
5.1.1 Answer Generation for QA . . . 109
5.1.2 Transfer Learning . . . 109
5.2 Knowledge Transfer Techniques . . . 110
5.2.1 AS2 Loss Weighting . . . 110
5.2.2 AS2 Score Conditioned I/O Shaping . . . 111
5.3 Experiments . . . 112
5.3.1 Evaluation . . . 114
5.4 Quantitative Results . . . 116
5.4.1 Comparison with fully supervised approaches . . . 116
5.4.2 Scarce Data Setting . . . 117
5.4.3 Industrial Setting . . . 119

6 Reward Function for Generative QA Based on Automatic Evaluators 121
6.1 Generating and Automatically Evaluating Answers . . . 122
6.1.1 Answer Generation . . . 122
6.1.2 Evaluation of QA Systems . . . 123
6.2 Generative QA (GenQA) . . . 123
6.3 SQuARe for Training GenQA . . . 124
6.3.1 Static Data Augmentation (SQuArE-SDA) . . . 124
6.3.2 Dynamic Data Augmentation (SQuArE-DDA) . . . 125
6.3.3 Loss Weighting (SQuArE-LW) . . . 126
6.4 Experiments . . . 128
6.4.1 Datasets . . . 128
6.4.2 Models and Evaluation . . . 128
6.4.3 Main Results . . . 129
6.4.4 Analysis and Ablations . . . 130
6.4.5 Qualitative results . . . 133
6.5 Conclusion and Discussion . . . 134

7 Retrieval Complexity **137**
7.1 Retrieval Complexity Motivation . . . 138
7.2 Modeling Retrieval Complexity . . . 139
7.2.1 Answerability . . . 141
7.2.2 Retrieval Set Completeness . . . 142
7.2.3 Classifying Complex Questions . . . 142
7.3 Retrieval Complexity Pipeline (RRCP) . . . 142
7.4 Experiments . . . 143
7.4.1 Datasets . . . 143
7.4.2 RRCP setup . . . 144
7.4.3 Supervised Baseline . . . 144
7.4.4 Complex Question Identification . . . 145
7.4.5 LLM Performance on Complex Questions . . . 146

7.4.6 Search Engine Performance on Complex Questions 147
7.4.7 Critical Error Analysis of RC Estimation 148
7.5 Qualitative Analysis and Limitations 148
7.6 Conclusion and Discussion 150

8 Conclusion **153**
8.1 Future Work 155

Bibliography **157**

List of Tables

2.1 Pretraining Dataset Statistics for Wikipedia, BookCorpus, Open-WebText and CC-News . . . 21

2.2 The statistics for the different tasks that compose the GLUE benchmark. The table reports the number of examples for both the train and the test splits divided per task. . . . 33

2.3 Statistics for dataset splits of Question Answering Datasets. Specifically, we provide a number of examples for the training, validation, and test split for SQuAD, Natural Questions, ASNQ, WikiQA, TREC-QA, MS-MARCO, WQA, and QuoraQP-a. Notice that some datasets have never released a public testset. . . . 35

2.4 Statistics for dataset splits of Complex Question Answering Datasets. Specifically, we provide a number of examples for the train, validation, and test split for CWQ, HotPotQA, StrategyQA, and MuSiQue. . . . 40

3.1 Pretraining sets for different languages . . . 52

3.2 The perplexities of the trained models using different datasets and the same amount of training steps. These results describe the complexity of the different languages and compare the multilanguage and monolanguage settings. . . . 53

3.3 Dataset statistics for Py150 and JavaCorpus . . . 53

3.4 Results on NTP finetuning task on two public datasets (javaCorpus and py150). The CodeGPT model was distilled from a larger model (366M parameters). . . . 54

3.5 Dataset statistics for our finetuning corpora . . . 54

3.6 Results of the Next Word Prediction finetuning task divided in three sub-tasks (KW, OP, NAME) on our multilanguage-evaluation dataset. . 55

3.7 Results for the Line Completion task on two public datasets (javaCorpus and py150). . . . 56

3.8 Results for the Line Completion task on our evaluation corpora against two monolanguage versions of CodeGPT (Python and Java). The models marked with a **M** are trained on our multilanguage dataset. . . . 56

3.9 Memory usage on models of different sizes, considering a batch size of 1 and the most expensive time both during inference and training. Our models use less memory during the training while still having more parameters (e.g., 2.35B vs 1.56B for the xl model) 57

3.10 Results on the GLUE test set. We took the best models on the development set and submitted them to the GLUE leaderboard. Also, in this case, we don't do best model selection from a pool of candidates, and we do not use ensemble models. 64

3.11 Results on WikiQA and TREC-QA datasets with single-task fine-tuning and after the transfer step on ASNQ. We also report the results on the dev. set of ASNQ optimizing MAP. The NSP loss of the original BERT mainly improves the results without the transfer step. After the transfer on ASNQ, which trains especially the CLS token representation, the difference with our MLM-based BERT is much smaller. We show the standard deviation after 5 runs with different initialization seeds in rounded brackets. We underline results that are significantly different from the BERT-B / MLM baseline after a two-sided T-Test with a significance level equal to 95%. 65

3.12 FLOPS used to pre-train each *base* model, language modelling head complexity and real training time on the same machine. *bs* stands for batch size, L for the input sequence length and $|V|$ is the vocabulary size, equal to about 30K tokens in BERT models. Notice that BERT-RTS and BERT-C-RTS use less memory thanks to the smaller binary classification head (also potentially allowing for larger batch sizes.) 66

3.13 FLOPS used to pre-train each *small* model, language modelling head complexity and real training time on the same hardware. See the caption of Table 3.12 for more details. 66

3.14 Results of RTS, C-RTS and SLM compared with MLM applied to *small* architectures on AS2 benchmarks and GLUE test set. We underline the results that are significantly different from the BERT-S / MLM baseline after a two-sided T-Test with a significance level of 95%. We show the standard deviation after 5 runs with different initialization seeds in rounded brackets. 67

3.15 Large models comparison on the GLUE test set. We report the average accuracy over the different tasks of Table 3.10, while FLOPS refers to pre-training. 69

3.16 Performance comparison between RTS and C-RTS on five different benchmarks. The results reported for WikiQA and TREC-QA are on the test set, while for ASNQ, MRPC and QNLI, we report the highest score on the development set. We show the standard deviation after 5 runs with different initialization seeds in rounded brackets. 69

3.17 Dataset statistics for mASNQ, mWikiQA, and mTREC-QA for each language. The datasets have the same statistics in their original version, and considering all the languages, the corpora comprehend more than 100M examples. Notice that for mWikiQA, we also report the statistics of the clean and the no-all-negatives (++) splits. 75

3.18 Dataset statistics for mASNQ, mWikiQA, and mTREC-QA for each language. The datasets have the same statistics in their original version, and considering all the languages, the corpora comprehend more than 100M examples. Notice that for mWikiQA we report also the statistics of the clean and the no-all-negatives (++) splits. 75

3.19 Examples of translation artifacts. The artifacts are highlighted in red. Notice that (i) "Non lo so" is an Italian sentence which translated in English means "I don't know", (ii) "Ich bin ein guter Mensch" means "I am a good person" in German, and (iii) the maening of the "Sauter sur refuge" in French is "Jump on refuge". 76

3.20 Similarities between ASNQ and mASNQ. On the left of the arrow ($\rightarrow$) the similarity reached after the initial translation is reported; on the right side, there is the similarity score after the application of the heuristics. 77

3.21 Performance comparison of XLM-RoBERTa on mTREC-QA, mWikiQA, and Xtr-WikiQA (zero-shot from the model trained on mWikiQA). The transfer step is done on mASNQ, while the Adaptation is on mTREC-QA and mWikiQA. Results in terms of MAP and P@1, for various language and model configurations. The models trained with the reward loss prove improved performance compared with the other approaches (e.g., 0.853 vs 0.843 on mWikiQA in terms of average MAP). 79

3.22 Comparison of XLM-RoBERTa base transferred on mASNQ and ASNQ and tested on mWikiQA in a cross-lingual setting. 80

3.23 Performance comparison of XLM-RoBERTa base model in a zero-shot setting on the Xtr-WikiQA task. Models trained on mASNQ dataset, denoted by ✱, outperform those trained on other datasets like mMARCO and MSMARCO. Moreover, BERT-multilingual consistently performs better than XLM-RoBERTa in various languages (Italian, Portuguese, Spanish), indicating the robustness and competitiveness of the approach on AS2 datasets. 81

3.24 Comparison of BERT-multilingual performance on $\text{mMARCO}_{\text{ITA}}$ test set. We train the two baselines respectively on the English MSMARCO and the mMARCO Italian split. The models trained using the reward loss, consistently improve the two presented baselines showing that (i) the loss is beneficial also when used in the Passage Reranking task, and (ii) the transfer step on mASNQ is helpful also in this domain. 82

3.25 Kendall, Spearman and Pearson correlation computed between the ranks originated from model trained the original ASNQ and mASNQ. The reported values are computed using XLM-RoBERTa base models transferred on ASNQ and mASNQ and then finetuned on mWikiQA and mTREC-QA. 84

4.1 Results on WQA, IQAD, MS-MARCO measured using Accuracy, Area under the curve and Pearson Correlation with gold labels. Results on IQAD are relative to AVA-TR baseline (due to data being internal). # Refs refers to the total number of reference answers used for the metric. 92

4.2 Zero-shot evaluation using QA evaluation models trained on WQA. Same metrics used as Table 4.1. 93

4.3 Results on WQA, IQAD, MS-MARCO measured using Accuracy, Area under the curve, and Pearson Correlation with gold labels. Results on IQAD are relative to the AVA-TR baseline (due to data being internal). # Refs refers to the total number of reference answers used for the metric. 95

4.4 Zero-shot evaluation using QA evaluation models trained on WQA. Same metrics used as Table 4.3. 96

4.5 Comparing SQuArE against BEM and LLM baselines on the WikiQA, TrecQA, and AE datasets. The BEM baseline is trained on the AE dataset. We use the same metrics as Table 4.3. 97

4.6 Pearson Correlation of evaluation metrics with human annotations on GenQA-MTURK. 97

4.7 System-wise evaluation on MS-MARCO, WikiQA, and TREC-QA using six different systems based on generative question answering models trained on MS-MARCO. The accuracy column reflects manual annotations by professional annotators, while the remaining columns show results computed using SQuArE, BLEURT, and BERTScore. 98

4.8 Ablation studies evaluating the benefits of using negative references and the impact of a number of references on the performance of SQuArE. AVA-TQR(-) and SQuArE(+) refer to an AVA model that only uses negative references and a SQuArE model that only uses positive references. # Refs is the total number of references used for the metric. [1,5] refers to the number of references being randomly sampled $\in[1, 5]$. . . . 98

5.1 Datasets statistics. 113

5.2 Results using BLEU as the metric for measuring performance of answer generation. We train the GenQA models on MS-MARCO NLG for the supervised setting, and MS-MARCO QA for the knowledge transfer setting. 116

5.3 Results on the test split of MS MARCO-NLG for different training paradigms of GenQA models. The weak supervision is provided by a RoBERTa-Large AS2 model trained on ASNQ. We compare with a fully supervised GenQA baseline (Hsu et al., 2021) trained on the train split of MS MARCO-NLG. . 117

5.4 Results on the test split of WikiQA and TREC-QA. The weak supervision is provided by RoBERTa-Large AS2 models trained respectively on WikiQA and TREC-QA. We compare with a fully supervised GenQA baseline (Hsu et al., 2021) trained respectively on the train split of WikiQA and TREC-QA using ground truth correct answers as the target for generation. We use T5-Large for all GenQA models. 118

5.5 Results on AQAD-L for different training paradigms of GenQA models. All results are reported in absolute % changes w.r.t the AS2 baseline. 119

6.1 Answering accuracy (manual evaluation) and AVA-Score on WQA, MS-MARCO and IQAD datasets. Results on IQAD are presented relative to the baseline (due to the data being internal). For WQA and MS-MARCO, we use an AVA model trained on WQA, and for IQAD we use an AVA model trained on IQAD. Best results for each dataset are highlighted in bold. 129

6.2 Evaluation of different GenQA models using automatic evaluation metrics: BLEU, BLEURT, BERTScore in addition to SQuArE-Score on the MS-MARCO dataset. We present the correlation each metric achieves with human annotation. SQuArE achieves the best correlation with human evaluation of answer accuracy. 131

6.3 Variation of GenQA accuracy by changing θ for SQuArE-SDA approach, on the MS-MARCO dataset. We present human and automatic (SQuArE-Score) evaluation. | Augmented Set | indicates the number of data augmentation examples created using a particular value of θ (Lower $\theta \rightarrow$ more augmentation examples). 131

6.4 Evaluation of different GenQA models using automatic evaluation metrics: BLEU, BLEURT, BERTScore in addition to SQuArE-Score on the MS-MARCO dataset. We present the correlation each metric achieves with human annotation. SQuArE achieves the best correlation with human evaluation of answer accuracy. 133

6.5 Examples of correctly generated answers using SQuArE-DDA approach on the MS-MARCO dataset. Example (1) highlights that the model is correctly able to synthesize a correct answer using the reference answer candidates for the question. Example (2) highlights a case where the GenQA model uses information from a single reference answer, but reformulates it's style using the question to present as an answer. Example (3) highlights a case where the GenQA model is effectively functioning as an answer ranker, as it directly copies the best answer candidate among the references to produce the generated answer. 134

6.6 Examples of incorrectly generated answers using SQuArE-DDA approach on the MS-MARCO dataset. Example (1) highlights a case of hallucination during generation where the model introduces an incorrect year in the generated answer, even when it is not present in any of the input reference candidates. Example (2) highlights a failure case of GenQA where the model is unable to synthesize a good answer due to lacking evidence in the retrieved reference candidates. 135
6.7 Comparison between previous training strategies and our novel techniques. 136

7.1 Examples illustrating simple vs. complex queries. Our RC definition distinguishes them by measuring the degree of evidence fragmentation required to answer the query. 140
7.2 Categorization of datasets into compositional and multihop complexity classes based on their respective characteristics. 144
7.3 The supervised approach displays limited effectiveness for cross-dataset evaluation, indicating reduced performance when trained on one dataset and tested on another. Despite similar complexity classes between datasets (e.g., Multihop), the transferability of models across distinct datasets (e.g., training on CWQ and testing on HotPotQA) exhibits notable performance discrepancies. Notice that ALL refers to a model trained on all the datasets combined together. 145
7.4 Performance comparison of our pipeline against a complex question classifier in terms of Accuracy and F1. The pipeline has been tested ablating the two constraints (Answerability and Retrieval Diversity) and on different Complex QA datasets. The results show that our pipeline is able to identify complex questions with high accuracy and that combining the two constraints is beneficial. For these experiments, we selected the best pipeline thresholds based on the development set. 146
7.5 Average answerability of answers generated for complex and not-complex questions selected by our pipeline. The results show that the questions marked as complex are generally more difficult to answer. PCC denotes the correlation in terms of the Pearson Correlation Coefficient between the complex and not complex questions in terms of answerability. . . . 147
7.6 Probability for a question classified by RRCP of being answered by a state-of-the-art search engine. The answerability is computed using manual annotators. MCC stands for Matthew Correlation Coefficient and measures the correlation between the complexity classification and its answerability. 148
7.7 Critical error analysis: Cases where RC estimation works well versus challenges and their underlying causes. 149

List of Figures

2.1 Transformer architecture 12
2.2 Illustration of the Masked Language Modeling task 18
2.3 Illustration of the Causal Language Modeling task 18
2.4 Illustration of the Permutation Language Modeling task 19
2.5 Illustration of the Next Sentence Prediction task 19
2.6 Illustration of the Token Detection task in ELECTRA 20
2.7 General ODQA pipeline discussed in this Thesis 26

3.1 CLM Decoder architecture 49
3.2 MLM Encoder architecture 51
3.3 Architectures for MLM, RTS, and SLM (left to right). Notice that the classification head used by RTS is significantly smaller than those used by MLM and SLM. 60
3.4 Performance comparison of RTS and C-RTS on WikiQA, TREC-QA, MRPC and QNLI. 68

4.1 Multi-reference SQuArE employs multiple positive and negative reference answers to evaluate the correctness of a target answer for a particular question, producing a score $s \in [0, 1]$. 90

5.1 Overview of the Weak Supervision Technique 108
5.2 These plots show the differences between using the dev. loss and the average AS2 model score as the measure for model checkpoint selection. Notice that the vertical dashed red line is used to identify the best checkpoint (which is the one with the lowest loss in the first plot, and the one with the highest average AS2 score in the second). 115

6.1 Illustrative representation of the SQuArE-Static Data Augmentation (SQuArE-SDA) approach. 124
6.2 Illustrative representation of the SQuArE-Loss Weighting (SQuArE-LW) approach. 127
6.3 Comparison of baseline GenQA-WS and AVA-LW on WQA in terms of GAVA-Score (on validation split) varying across training epochs (GAVA-LW achieves higher GAVA-Score throughout training). 130

6.4 Comparison of three different GAVA-SDA models trained on MS-MARCO using different thresholds θ in terms of GAVA-Score (on validation split) varying across training epochs. The model trained with the largest value of θ achieves the highest GAVA-Score. 132

7.1 While both questions are complex using the definition in prior works, the "Low RC" question on the left is easy for RAG systems, while the "High RC" question on the right is far more difficult. Documents retrieved by commercial search engines illustrate this challenge. RC correlates with the probability that the retrieved documents contain sufficient information to generate a reference answer. 138
7.2 Overview of our RRCP for measuring the complexity of questions. A retrieval system retrieves documents for the input question. These are examined by a reference-based evaluation system, which provides the probability of correctness scores. These approximate the answerability and completeness of retrieved information, i.e., an estimation of RC. . . 141
7.3 Distribution of complexity classes from a sample of 200 high RC questions selected among the datasets described in Section 7.4.1. 150

Chapter 1

Introduction and Contributions

Natural Language Processing (NLP) is a branch of artificial intelligence that enables machines to understand and generate human language, facilitating applications like translation, chat systems, and question-answering (QA). QA systems in NLP connect the way people search for information with the vast knowledge stored in databases. The latest large language models (LLMs) have brought AI closer to more general intelligence. However, major challenges remain in assessing reliability, improving training efficiency, and handling complex information retrieval. Generative models like GPT (Radford and Narasimhan, 2018; Radford et al., 2019; Brown et al., 2020) have transformed QA by generating fluent and relevant responses. Still, they face significant limitations that affect their scalability and real-world use.

1.1 Research Gaps and Contributions of the Thesis

This Thesis examines these challenges and proposes concrete methods to enhance the performance, scalability, and trustworthiness of generative QA systems by addressing each research gap. In this regard, we highlight three major research gaps:

1. **(Research Gap 1)** Current metrics (e.g., BLEU and ROUGE) fail to capture semantic accuracy, coherence, and relevance. They also introduce biases that can misrepresent a model's true effectiveness (Papineni et al., 2002; Lin, 2004). Human evaluation is more accurate but is expensive, time-consuming, and impractical for large-scale assessments.

2. **(Research Gap 2)** Generative models require extensive labeled datasets for effective training. However, dataset creation is resource-intensive, and current approaches like Reinforcement Learning from Human Feedback (RLHF) (Christiano et al., 2023) remain costly and work-heavy.

3. **(Research Gap 3)** Challenges in Handling Complex Retrieval-based Queries: Retrieval-Augmented Generation (RAG) models (Lewis et al., 2020b) struggle with multi-hop and ambiguous queries that require integrating information from multiple sources. Existing definitions of question complexity mainly focus on

linguistic or structural difficulty, neglecting the broader challenges of retrieving relevant knowledge from diverse and scattered sources.

These challenges are interconnected: improving evaluation metrics (Gap 1) strengthens training methods (Gap 2), which in turn helps develop more effective models for handling complex queries (Gap 3). To systematically address these gaps, we: (i) Develop a new automatic evaluation system for QA that closely aligns with human assessments (Gap 1), (ii) introduce new training techniques for generative models using weak supervision and reward-based loss functions (Gap 2), and (iii) propose a new metric to estimate question complexity in RAG systems (Gap 3).

These methods are detailed in Chapters 4, 5, 6, and 7, with the theoretical foundations provided in Chapter 2.

1.1.1 Generating Quality Assessment for QA models

Evaluating generative QA models is challenging because their outputs are open-ended. While human evaluation is the most reliable way to assess answer accuracy and quality (Novikova et al., 2017), it is slow, expensive, and not scalable for large datasets. Most existing automatic metrics, such as BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004), were designed for machine translation and summarization, making them unsuitable for capturing the finer aspects of answer quality in QA systems (Reiter, 2018b; Chen et al., 2019). This creates a strong need for new evaluation frameworks that can accurately assess semantic correctness, relevance, and coherence while closely aligning with human judgment.

Our contribution: In this thesis, we introduce SQuArE (Sentence-level QUestion AnsweRing Evaluation), a novel, robust, and adaptable evaluation metric specifically designed for QA tasks (Chapter 4). Unlike traditional metrics, SQuArE considers multiple reference answers and incorporates both positive and negative examples, allowing for a more detailed and accurate assessment of answer quality. By addressing the limitations of manual evaluation (such as cost, time, and human errors) SQuArE significantly reduces the need for human evaluators while enabling faster and more efficient large-scale assessments of generative QA systems. Most importantly, SQuArE demonstrates a high correlation with human judgments, validating its effectiveness as a strong alternative to manual evaluation

1.1.2 Train Without Human-Annotated Data

A key challenge in generative QA is the difficulty of creating high-quality training data, which limits model improvements. Most existing QA datasets focus on simple factual questions and do not adequately prepare models for generating long-form, conversational responses (Rajpurkar et al., 2016a; Kwiatkowski et al., 2019). To address this, it is crucial to develop new methods that make better use of existing resources while reducing reliance on manual annotation.

Our contribution: In this Thesis, we introduce new training objectives and methodologies based on **weak supervision** to mitigate the need for manually annotated datasets. Specifically:

- We introduce a knowledge transfer framework that links Answer Sentence Selection (AS2) with generative QA models. This approach reduces reliance on large labeled datasets by leveraging the strong discriminative abilities of AS2 models.
- We propose a **feedback loop mechanism** to incorporate SQuArE into the training of generative QA models. Leveraging SQuArE scores as the only proxy during training, we synthesize high-fidelity data, which consistently improves performance on the model solutions and reduces the need for human-annotated large corpora.

These techniques enable the development of more robust and adaptable QA systems that can learn from noisy or partially labeled data. The weak supervision approaches discussed in Chapters 5 and 6 help bridge this gap, making it possible to build strong QA systems using unlabeled data.

1.1.3 Complexity of Questions

While Large Language Models (LLMs) excel at various NLP tasks (Radford and Narasimhan, 2018; Radford et al., 2019; Brown et al., 2020), some questions remain inherently challenging, requiring multi-hop reasoning or complex inference. For example, Chain-of-Thought prompting (Yang et al., 2018b) increases reasoning complexity, while Retrieval-Augmented Generation (RAG) introduces difficulties in retrieving relevant information. Traditional complexity analyses focus mainly on the model's reasoning ability but often overlook the retrieval process. However, the accuracy and relevance of retrieved information are critical for answering questions in RAG systems (Izacard and Grave, 2021).

Our contribution: we propose a definition of **retrieval complexity** to clarify and tackle some questions' challenges. Our contributions include:

- A formal definition of retrieval complexity, which measures how difficult it is to find the correct information within a document collection to answer a question.
- A framework for evaluating retrieval difficulty based on the distribution of relevant information across documents and the inherent ambiguity of the question.

This improved notion of complexity helps understand the relationship between retrieval and generation, enabling the development of more effective RAG pipelines and realistic QA datasets. Chapter 7 introduces a new definition and methodology for measuring retrieval complexity, offering new possibilities for improving RAG models and dataset design.

By addressing these three research gaps, this thesis enhances generative QA by improving evaluation reliability, training efficiency, and the handling of complex information retrieval challenges. The first research gap can be framed as a key research question: What factors influence the reliability of generative QA, and which of these can be measured? These contributions strengthen the scalability and robustness of QA systems, providing practical tools for advancing NLP research and applications.

1.2 Publications

In this section, we provide a list of the publications that led to the creation of this work. We sorted the entries in order of publication.

- ***Effective Pre-Training Objectives for Transformer-based Autoencoders***
  Di Liello Luca, Gabburo Matteo and Moschitti Alessandro
  Findings of the Association for Computational Linguistics: EMNLP 2022

- ***Knowledge Transfer from Answer Ranking to Answer Generation***
  Gabburo Matteo, Koncel-Kedziorski Rik, Garg Siddhant, Soldaini Luca and Moschitti Alessandro
  Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing

- ***Learning Answer Generation using Supervision from Automatic Question Answering Evaluators***
  Gabburo Matteo, Koncel-Kedziorski Rik, Garg Siddhant, Soldaini Luca and Moschitti Alessandro
  Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics

- ***SQUARE: Automatic Question Answering Evaluation using Multiple Positive and Negative References***
  Gabburo Matteo, Koncel-Kedziorski Rik, Garg Siddhant, Soldaini Luca and Moschitti Alessandro
  Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics

- ***Measuring Retrieval Complexity in Question Answering Systems***
  Gabburo Matteo, Jedema, Nicolaas Paul, Garg Siddhant, Ribeiro Leonardo F. R. and Moschitti Alessandro
  Findings of the Association for Computational Linguistics: ACL 2024

- ***Datasets for Multilingual Answer Sentence Selection***
  Gabburo Matteo, Campese Stefano, Agostini Federico, and Moschitti Alessandro
  Findings of the Association for Computational Linguistics: EMNLP 2024

1.3 Thesis Structure

This Thesis is structured into eight chapters. The first three chapters provide essential background information. Specifically, apart for this introduction chapter (Chapter 1), the following two chapters are meant to describe all the necessary background to understand the core chapters of this Thesis. In detail, Chapter 2, describes the essential background knowledge for the Thesis, covering key aspects of Natural Language Processing (NLP). It then presents and analyzes the transformer architecture, highlighting its superiority over previous models like RNNs and discussing its implementations and pretraining techniques.

In Chapter 3, we will start going in-depth on the main arguments of this Thesis, providing three different case studies that allow us to focus on the Transformer architecture, its training objectives, and its application on Question Answering tasks. In detail, this chapter is organized into three parts where each part is a case study based on a paper on which I worked during my PhD programme. The first part provides an overview of transformers, showcasing their versatility and effectiveness in solving various tasks, including novel domains like code generation and analysis. The second part explores efficient pretraining techniques for transformer encoders and it is based on (Di Liello et al., 2022a). In this paper, we propose new methods that balance efficiency and performance. The third part allows us to dive deep into the question-answering domain, examining fine-tuning techniques for transformers. This work, which is partially based on (Gabburo et al., 2024a), highlights the adaptability of transformer models to complex multilingual NLP tasks and improves performance through innovative methods like automatic dataset translation and loss manipulation.

Chapters 4,5, 6 and 7 represent the core of the main arguments of this Thesis and directly tackle the research gaps mentioned in Section 1.1. In detail, Chapter 4 discusses a key contribution of this Thesis, which is the development of SQuArE (Sentence-level QUestion AnsweRing Evaluation), a new evaluation metric for Generative QA systems. This chapter, which is based on a published paper (Gabburo et al., 2023b), addresses the limitations of existing automatic metrics and the costs of the manual evaluation by incorporating multiple reference answers and leveraging transformer-based models for semantic similarity assessment. Our experiments demonstrate that SQuArE achieves a higher correlation with human judgments compared to traditional metrics, providing a more reliable tool for evaluating both extractive and generative QA systems.

In Chapter 5, we propose an innovative approach to training generative QA models by transferring knowledge from discriminative Answer Sentence Selection models based on Gabburo et al. (2022). This method allows for effective training of generative models using unlabeled data, addressing the challenge of limited high-quality training data for generative QA. Our experiments show that this approach can outperform both the AS2 teacher models and supervised generative QA models on various benchmarks without the need for costly human-annotated corpora.

In the following chapter (Chapter 6), capitalizing on the success of SQuArE, we develop a novel training paradigm for generative QA models incorporating feedback from the SQuArE metric. This approach, which we described in the ACL paper Gabburo

et al. (2023a), uses SQuArE scores to guide the training process, leading to improved answer quality and relevance. Our results demonstrate significant improvements in generative QA performance across multiple datasets and languages.

Our final core contribution introduces the concept of retrieval complexity in Chapter 7, offering a new way to understand and measure the difficulty of questions in QA tasks. We propose a methodology for quantifying retrieval complexity and demonstrate its effectiveness in identifying challenging questions across various datasets. This published work (Gabburo et al., 2024b) provides valuable insights into the nature of question difficulty and offers a new tool for dataset creation and model evaluation in QA research.

This Thesis concludes with Chapter 8, which summarizes its major contributions and outlines potential directions for future research.

Chapter 2

Background

This chapter introduces all concepts, theories, and background knowledge to understand and read the further chapters of this Thesis. In this chapter, we decided to make groups of sections, which will focus in certain topics to explain features of Natural Language Processing, the domain of Question Answering and systems designing principles that can be applied to solve their task.

First, Section 2.1 will present an introduction to the Natural Language Processing domain with its significant tasks like Machine Translation (MT), Named Entity Recognition (NER), and Sentiment Analysis (SA).

Second, in Section 2.2, we provide a detailed description of the current state-of-the-art used in Natural Language Processing and other fields (e.g., computer vision): the transformer architecture (Vaswani et al., 2017). Specifically, this peculiar architecture, which mainly relies on the attention mechanism, outperforms previous deep learning solutions, such as Recurrent Neural Networks (e.g., LSTM, GRU), in every task, setting a new standard in NLP. In addition, we discuss the key innovations and why it has become the go-to choice for modern NLP applications (Devlin et al., 2019) (Radford et al., 2019).

Third, Section 2.3 describes in depth one of the major improvements of the transformer architecture over the previous models, which is its training paradigms, and specifically analyzes in depth the pretraining step, explaining why these trained models are often named as pretrained language models (PLM) (Devlin et al., 2019). In addition, we will describe in detail the most common techniques usually used to train these models, such as Masked Language Modeling and Causal Language Modeling (Radford and Narasimhan, 2018; Clark et al., 2020). These techniques characterize PLMs such as BERT, GPT, and ELECTRA (Devlin et al., 2019; Radford et al., 2019; Clark et al., 2020).

Fourth, in Section 2.4, we will focus on the central topic of this Thesis, which is the Question Answering (QA) Field (Ferrucci et al., 2010). QA can be summarized as a family of tasks involved in automatically answering natural language questions (Rajpurkar et al., 2016b). In this Thesis we touch some of these tasks, such as Extractive Question Answering (Rajpurkar et al., 2018a), Information Retrieval (Karpukhin et al., 2020) Passage Ranking (Nogueira and Cho, 2020)), Answer Sentence Selection (Severyn

and Moschitti, 2015), and Generative Question Answering (Raffel et al., 2020a). In addition, we will explore in depth the typical pipeline used in industrial and academic applications to solve Open Domain Question Answering (ODQA) (Chen et al., 2017a) and the methodologies to deal with questions from various fields without prior domain knowledge (Kwiatkowski et al., 2019).

Lastly, in Section 2.5, we discuss the different datasets and benchmarks considered in this Thesis for training, evaluating, and benchmarking NLP and QA systems.

2.1 Natural Language Processing

Natural Language Processing (NLP) is a subfield of artificial intelligence that is focused on software programs that can understand human languages as they are spoken naturally. Its multidisciplinarity implies crossovers with computational linguistics, information retrieval, knowledge representation, and machine learning (Manning and Schütze, 1999; Jurafsky and Martin, 2009). Different paradigms guiding the field can thus be tracked in the evolution of the sociological understanding of social change; and can help make sense of a developing line of sociology of social change as well.

2.1.1 Early Conceptualization and Rule-Based Approaches (1950–1990)

NLP has roots in the 1940s and 1950s, when its priorities were solving practical problems like translating foreign languages automatically. The early systems were rule-based symbolic systems, which were manually crafted linguistic rules (Chomsky, 1956; Winograd, 1972). These systems used grammars and ontologies, which require considerable human expertise, to encode linguistic knowledge. For example, transformational grammar approaches formalize syntactic structures.

Though pioneering, and a good starting point for computational linguistics, rule-based systems were not very scalable or flexible. They were limited to relying on handcrafted rules, which made them less flexible in conveying the complexities of natural language. Consequently, researchers started looking at different methodologies.

2.1.2 The Rise of Statistical Methods (1990–2010)

The most notable shift, which took place in the 1990s, involved moving away from rule-based methods towards statistical techniques driven by the proliferation of large datasets along with computation resources, greatly enhanced through the progress of hardware described by Moore's law (Moore, 1965). Statistical methods were a departure from the previous paradigms of rule-based processing. Groundbreaking successes in the 1990s consisted of the use of Hidden Markov Models (HMMs) (Rabiner, 1989) and n-gram language models (Jelinek, 1990). These systems have shown great success in POS tagging (Brill, 1995), NER (Tjong Kim Sang and De Meulder, 2003), and machine translation.

This is also the era where machine learning was incorporated into NLP. Statistical approaches offered a foundation to create algorithms capable of directly learning from data, leading to substantial performance gains. Researchers pioneered basic approaches to tasks, including parsing, word alignment for translation, and semantic similarity (Church and Hanks, 2002). By the early 2000s, these approaches had become the norm, achieving state-of-the-art performance in various branches of NLP.

2.1.3 The Deep Learning Revolution and Word Embeddings (2010–Present)

The deep learning techniques that emerged in the 2010s revolutionized NLP. Word embeddings, e.g. Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), were central to this revolution, and made possible dense vector representations of words. These embeddings retained semantic and syntactic relationships, enabling models to generalize across distinct linguistic settings. The ability to model sequential data took huge leaps with the advent of neural network architectures like Recurrent Neural Networks (RNNs) (Mikolov et al., 2010) and Long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997a), giving rise to applications such as Speech Recognition (Graves et al., 2013) and Machine Translation (Bahdanau et al., 2014a). These architectures enhanced performance by handling long-term dependency and contextual nuances in language.

Transformer architecture (Vaswani et al., 2017) was a breakthrough for NLP. In contrast to RNNs and LSTMs, Transformers utilized attention mechanisms to process sequences in parallel, effectively addressing the challenges of vanishing gradients and short-term dependencies. Transformers were the basis of strong models like BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) and GPT (Generative Pre-trained Transformer) (Radford and Narasimhan, 2018). These models have demonstrated state-of-the-art performance on tasks including Text Summarization (Liu and Lapata, 2019), Sentiment Analysis (Zhang et al., 2018), and Question Answering (QA) (Devlin et al., 2019).

2.1.4 Applications and Core NLP Components

NLP encompasses a wide range of tasks that can be categorized into core components, each contributing uniquely to the understanding and processing of human language:

- **Syntax:** Syntax focuses on analyzing the grammatical structure of sentences and the relationships between words. This includes tasks such as Part-of-Speech (POS) tagging, dependency parsing, and constituency parsing, which provide foundational insights into how sentences are structured. Applications include grammar-checking tools, like those used in word processors, and syntactic analysis in machine translation systems.

- **Semantics:** The semantics is for the meaning of the words, phrases, and sentences. Word number disambiguation, semantic role labeling, and entailment

detection None of these tasks are specifically linguistics, but they do all contribute to understanding the nuances of meaning. I mean, semantic analysis is critical in applications such as question-answering (QA) systems, where the meaning and intent of the words and surrounding context must be identified to return appropriate, valuable answers.

- **Pragmatics:** Pragmatics interprets language in its broader context, understanding not only what the speaker says but also what he means, state implications, social cues, etc. This component is vital in tasks such as dialogue systems, sentiment analysis, and conversational AI, where a single sentence can take on varied meanings based on the context. Pragmatics is especially relevant for the development of virtual assistants like Siri or Alexa, which must tailor individual responses to user intent and situational context.

- **Morphology :** Morphology refers to the internal structure of words and the rules governing how words are formed (not just inflection), derivation, compounding, and so on. In languages with rich inflectional systems, such as Arabic or Finnish, morphological analysis is crucial. It has been used in applications such as spell-checking systems, text normalization in speech recognition, and most importantly, in low-resource language processing; because of the morphology rules, it enables to solve the data sparseness problem.

These pillars are the fundamental building blocks for various NLP applications, such as:

- **Named Entity Recognition (NER):** This task distinguishes and classifies entities like people, organizations, and locations in a given text. NER has been widely used in the extraction of information systems and search engines.

- **Neural Machine Translation (NMT):** Translation of one language into another using neural networks. Modern NMT systems based on Transformers have achieved a major leap in translation fluency and quality.

- **Speech Recognition:** The task of translating spoken words into written text and voice commands facilitates usage in products like smart speakers, voice transcription, and automated systems.

- **Text Summarization:** Generating concise summaries from extensive text sources is key in news aggregation and document summarization applications. **Sentiment Analysis** — Expressed sentiments (for example, in tweets, analyzes customer feedback texts)

- **Question Answering (QA):** Extracting or generating correct answers to user questions from textual data. QA systems are foundational to search engines, chatbots, and customer support automation.

Until then, the components and applications of NLP served as the building blocks of advanced models to deal with complex linguistic problems. The significance of this research lies not only in its impact on language technology but also on its role as an intersection of linguistics, machine learning, and data-driven techniques to solve problems.

2.1.5 Focus on Question Answering (QA)

This paper introduces an overview of the role of Question Answering (QA) as a central area for modern NLP, one of the most challenging and influential uses of language understanding. QA systems need models that can retrieve, interpret, and produce correct answers to questions asked in natural language in a wide variety of domains, data sources, and question types. Such systems are required to have multiple components of NLP working together, such as information retrieval, semantic understanding, and context-aware generation.

Earlier QA systems were primarily extractive approaches like Answer Sentence Selection (AS2), whereby candidate sentences were ranked based on their relevance to the query (Garg et al., 2019). These approaches aim to identify and extract segments of text from a provided source, such as the aforementioned document. Although they work well for some use cases, like fact-based QA, extractive approaches cannot synthesize information across sources or generate fluent, relevant answers. However, the rise of Generative QA (GenQA) methods represented a meaningful step forward, as generative language models were used to create answers rather than simply extracting them. GenQA systems generate answers by synthesizing information extracted from the top candidates, usually within a retrieval-augmented generation (RAG) framework. This allows for clarity, mellifluousness, and relevance in the answers, regardless of fragmentation in the input, which may refer to multiple documents. The training of GenQA models enables flexibility and adaptability, which makes them suitable for open-domain QA and conversational systems.

The development of QA systems is part of a more significant trend in NLP that is moving toward models that can participate in complex, more human-like interactions. Recent QA methods go as far as employing sophisticated means, including pre-trained models based on Transformers (BERT, GPT-based class models) and reinforcement learning from human feedback (RLHF), coupled to enhance performance and answer quality. Such developments allow QA systems to tackle challenges that require more sophisticated reasoning patterns: multi-hop reasoning, temporal reasoning, answering ambiguous or underspecified questions, etc.

This Thesis contributes to the improvement of QA systems by proposing new methodologies for better answer generation and evaluation. This work seeks to raise the bar for QA systems by exploring novel methods of merging generative and evaluative methods, with the ultimate goal of a truly cohesive, human-like experience.

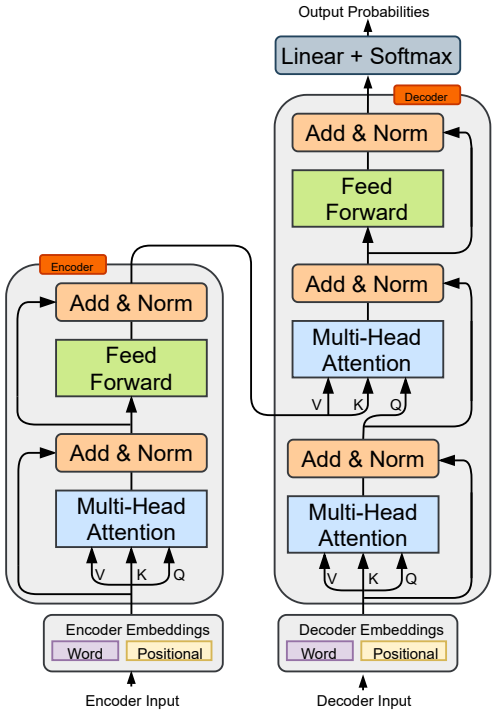

Figure 2.1: Transformer architecture

2.2 The Transformer Architecture

The Transformer architecture (Vaswani et al., 2017) represents the current state of the art in multiple tasks, outperforming all previous approaches based on Recurrent Neural Networks (RNN). Unlike deep architectures such as Long Short-Term Memory (LSTM) by Hochreiter and Schmidhuber (1997b) or Gated Recurrent Unit (GRU) by Cho et al. (2014), the Transformer architecture relies solely on the attention mechanism (Bahdanau et al., 2014b), which does not use recurrence and is typically based on complex gating systems.

This architecture (Figure 2.1) is a stacked structure logically divided into two parts: the encoder and the decoder. Both the encoder and the decoder of a Transformer can be seen as stacks of different layers composed of attention mechanisms and fully connected layers.

By elaborating on the relations between the words, these models are able to "learn" the semantic representation of each word and the semantics of the entire sentence. Indeed, similar to earlier architectures such as the RNNs, the input of the Transformer is always a sequence, and in the case of NLP, this sequence consists of words, tokens, or characters. The tradition of splitting a sentence up into words or characters has been abandoned in favor of working with tokens over the past 10 years.

This approach allowed us to solve the two principal problems of using words and characters. First, the techniques based on word embeddings (Mikolov et al., 2013) need to start from a vocabulary of known atomic components of a sequence (e.g., words),

which could compose all the possible inputs. This vocabulary is complex to modify once the model is trained, and the vocabulary based on words leads to the Out-Of-Vocabulary (OOV) problem (for example, when the input word contains a typo or is a neologism). Second, the approach based on characters is not able to build a strong semantic representation of each character (Sennrich et al., 2016a). In contrast, a token can be seen as a smaller part of a larger word, which could be big enough to get a strong learned semantic representation but small enough to be composed with other tokens to build words to solve the OOV problem. In detail, tokens are the fundamental units that are used to compose words. For example, the word *"literature"*, can be seen as the composition of the letters $\{l, i, t, e, r, a, t, u, r, e\}$ or of the tokens $\{lit, era, tur, e\}$.

Formally, given a vocabulary V, a word W of size k can be seen as an ordered list of tokens:

$$W = \{t_0, t_1, \ldots, t_k\}, \tag{2.1}$$

where each $t_i \in V$ is a part of the original word. This is crucial for breaking down natural language into manageable pieces that the transformer architecture can process.

Starting from the input, the first part of a Transformer model is the embedding layer. The role of this layer is to project the textual inputs into a vector space to maximize its expressive power. This idea was initially inspired by the authors of the Word2Vec model (Mikolov et al., 2013), who noticed that previous encoding techniques, such as lexicons or one-hot encoding, were not optimal for representing important semantic properties. Following this intuition, they proposed a two-layer neural network that was able to learn this kind of relation, calling them word embeddings.

The Transformer architecture builds embeddings that can more accurately encode context using the semantics of each word in the input sequence and their positions. This second type of vector is called positional embeddings, and like for words, they are learned during training.

After the embedding layer of the Transformer, there is an encoder model, which is composed of many identical stacked layers. Each layer is defined as a dot-product attention mechanism (Equation 2.2) called self-attention and a fully connected layer. The attention layer is fed with three different vectors called queries (Q), keys (K), and values (V), which are matrices representing the input embeddings.

$$\text{Attention}(Q, K, V) = \text{softmax}\Big(\frac{QK^T}{\sqrt{d_k}}\Big)V \tag{2.2}$$

The attention mechanism in the encoder and the decoder are slightly different: Here in the encoder, these three inputs are different representations of the same context, while in the decoder, K and V are representations of what the encoder produced, where Q comes from the decoder inputs.

The Transformer implementation took the multi-head attention mechanism another step further: separate attention layers (heads) are computed in parallel in different parts of the input, to separate dimensions of the embeddings. Due to the joining of many different representation sub-spaces in the output, this extension improves robustness and performance. The second component of each layer is a fully connected

feed-forward network, which is applied to all symbols in a position-independent manner and independently to each position. It performs a linear transformation. The network defined by Equation 2.3 is shared between every consecutive layer and usually consists of a 2-layer network with ReLU (Rectified Linear Unit) activation in the middle.

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

After the encoder, there is the second part of the architecture that consists of the decoder. The decoder structure is similar to the encoder but implements some changes, such as the already explained modified self-attention that takes into consideration the queries (Q) coming from the encoder input.

2.2.1 Architectural variations

The original Transformer architecture proposed by Vaswani et al. (2017) consisted of an encoder-decoder architecture, which was specifically designed for text-to-text tasks, such as neural machine translation. Nevertheless, based on its potential performance, it has been adapted to meet various tasks and goals. The two additional variations of this architecture are, at a component level, two, the encoder-only and the decoder-only Transformer, which we describe below.

Encoder Only Transformer

The encoder-only architecture, first proposed by BERT (Devlin et al., 2019), consists of a transformer where the decoder part has been removed. This particular architecture does not conserve the mechanism to ensure the left-to-right knowledge of the input present in the decoder of Vaswani et al. (2017). This modification allows the Transformer to get a bidirectional understanding of the input, which could benefit specific sentence-level tasks such as text classification, sentiment analysis, NER, intent classification, passage ranking, and answer sentence selection.

Indeed, this architecture can be easily adapted to solve downstream tasks by simply attaching a classification head on top of the decoder or the encoder part and tuning it with fine-tuning on a target task.

Decoder Only Transformer

Despite the name, the decoder-only Transformer is not a decoder in terms of architecture. This modification of the Vaswani et al. (2017) Transformer, firstly presented by the authors of Radford and Narasimhan (2018), conserves the encoder part of the original Transformer but implements a mechanism to guarantee the left-to-right knowledge of the input as it happens for the RNNs.

2.2.2 Improvements to the original architecture

Despite the considerable improvement and contribution that the Transformer made to the NLP domain, this architecture suffered from some limitations.

Static Positional Embeddings: Unlike the RNNs, in the transformers architecture, the position of the tokens that compose the input sequences are not self-encoded. Therefore, in (Vaswani et al., 2017), they introduced a method of encoding the positions of tokens using static positional encodings that are added to the input sequence embeddings. This was a very inconvenient property of these architectures since it meant their flexibility was limited by the length of the training sequences they had seen, and they were less able to generalize to longer sequences. This problem was later quickly solved by learning the positions also using embedded representations instead of static encodings (Devlin et al., 2019).

Quadratic Computational Complexity: One of the major problems involving the attention architecture is its quadratic complexity towards the input. Unfortunately, this behavior impacts the architecture both in terms of time and required space. Indeed, the computational complexity of the attention mechanism is $O(n^2)$. This means the effort required for the attention mechanism to perform an operation grows exponentially with the length of the input. Since the transformer architecture mainly relies on the attention mechanism, it also suffers from this limitation. In addition to the time complexity issue, this system presents another important problem, which is the space needed to store the attention matrices. Unfortunately, similar to the computational complexity, the size of these matrices also grows with the length of the input sequences, making it difficult to use on machines with low memory.

Indeed, to train and use these models with a reasonable amount of time and resources, the first applications of these architectures, such as BERT, RoBERTa, Electra, or T5, were trained to have a maximum input sequence length capped at 512 tokens making it impossible for them to handle longer sequences.

Many researchers extended the attention mechanism to improve the pitfalls. For instance, the authors of (Child et al., 2019) and (Beltagy et al., 2020) introduced a sparse factorization of the attention mechanism. This operation reduces the number of calculations, as the attention simply needs to focus on the part of the tokens at the time by means of sliding windows or fixed intervals.

Besides, the authors of the Linformer (Wang et al., 2020a) proposed a solution to linearize the attention mechanism based on a linear projection of the attention matrixes to smaller matrixes, while the Performer (Choromanski et al., 2022) apply randomized kernels to approximate the attention. Another relevant approach is represented by (Kitaev et al., 2020), where the authors introduced a mechanism that groups similar words and uses hashing techniques to reduce the computational complexity, avoiding recomputing already seen combinations of tokens. Similarly, the linear and routing transformers (Katharopoulos et al., 2020; Roy et al., 2021) implemented efficient versions of the attention matrixes by reducing their dimensionality and using compact data structures. With FNet (Lee-Thorp et al., 2022), its authors tried to replace the self-attention mechanism in the transformer architecture with linear transformations based on unparameterized Fourier Transformations, showing significant improvements in terms of required computational effort at the cost of a slight drop in terms of model accuracy. Recently, the authors of (Dao et al., 2022) and (Dao, 2023a) proposed the

FlashAttention, which is an engineered version of the original full-attention. These optimizations allow multiple operations to be combined into a single GPU kernel using a technique named "kernel fusion," which combines the softmax and matrix multiplication into a single GPU operation and saves space and time.

Long dependencies: Although Transformers can theoretically capture long-range dependencies, in practice, especially in the early stages of the training, they mainly focus on local context. This non-optimal understanding of these relationships directly impacts the final performance of the transformer models on the downstream tasks requiring the understanding of long-range relationships. Several researchers addressed this issue, proposing different solutions based on the custom versions or modifications of the attention mechanisms in order to improve the trade-off between computational efficiency and the capacity to capture meaningful long-range relationships in the input data. For example, some approaches tried to combine the local attention windows with some global attention nodes, making it possible to capture both local and long-range context without having to deal with the quadratic complexity of full attention. The authors of the Longformer (Beltagy et al., 2020), uses a sliding window attention pattern with additional global attention on specific tokens, allowing it to process sequences of thousands of tokens in an efficient way. In addition, Big Bird (Zaheer et al., 2020) extends this idea by incorporating random attention and improving the ability of the model to deal with long-range dependencies. Another solution is represented by architectures that mix local and global contexts to enhance the capture of long-range dependencies, like the Routing Transformer (Roy et al., 2021). Indeed, this architecture employs a content-based sparse attention mechanism that is able to route the queries of the attention mechanism to the most relevant attention keys, allowing the balance of the processing of local and global information.

Model sizes and scalability: Differently from previous state-of-the-art architectures like RNNs, the transformer performance scales with its size. This feature led to rapid growth in the pretrained language models in terms of a number of parameters. For example, BERT (Devlin et al., 2019), which was one of the first pretrained models, has from 110 to 340 Million parameters, while modern pretrained language models such as LLama3 (AI@Meta, 2024) have up to 405 Billion parameters. However, this scalability has introduced new challenges related to the computational and memory resources required for training and making inferences on these systems, motivating several researchers to work on the idea of reducing these costs and then producing interesting techniques. For example, techniques such as "distillation" allow the transfer of knowledge from expert teacher models to smaller students, obtaining a reduction of the model's size but maintaining a good trade-off between the number of parameters and the model performance (Hinton et al., 2015; Sanh et al., 2019). Additionally, other techniques such as gradient checkpointing (Chen et al., 2016), model parallelism, and pipeline parallelism (Shoeybi et al., 2019) allowed scaling the architecture parallelizing and distributing training and inference across multiple machines, overcoming the memory limitations of individual devices.

2.3 Pretraining Techniques and Pretrained Models

One of the key ideas that makes transformer-based models the new standard for natural language processing (NLP) tasks is that they can be trained unsupervised on large amounts of unlabeled data. This specific training is known as "pretraining". Indeed, pretraining techniques allow us to learn rich language representations from vast text corpora before fine-tuning specific downstream tasks (Devlin et al., 2019). This section will provide an exhaustive background of the most common pretraining techniques and illustrate their details.

2.3.1 Pretraining Techniques

Transformers are usually trained in two steps, which are (i) pretraining and (ii) fine-tuning. During the pretraining stage, the model is trained on a generic task using massive unlabeled datasets and consuming a significant amount of computational resources that, in the worst case, can have a negative impact on the environment (Strubell et al., 2019; Bender et al., 2021). Subsequently, the pretrained model can be fine-tuned on downstream tasks using smaller annotated datasets and reduced training time (Devlin et al., 2019).

Researchers studied and developed different techniques to perform this training stage. The next sections will provide an insight of some of these techniques.

Masked Language Modeling (MLM)

The Masked Language Modeling task (MLM) is an unsupervised pretraining task originally presented by Devlin et al. (2019). The key idea behind this famous pretraining objective, which is generally used to train transformer encoder models only (Section 2.2.1), is to force the model to learn token representation by understanding the original meaning of randomly masked tokens as shown in Figure 2.2. In detail, MLM is a token-level pretraining objective where, during the training, a certain percentage of input tokens are randomly masked, and the model is trained to predict the masked tokens based on the surrounding context. This encourages the model to learn bidirectional representations of the input text. MLM has been a foundational approach for models like BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019), which have set new benchmarks across various NLP tasks by leveraging the deep contextual understanding facilitated by MLM. The percentage of masked tokens varies according to the model. For example, BERT has been trained to mask the 15% of the tokens.

Casual Language Modeling (CLM)

The Causal Language Modeling task (CLM) is a well-known autoregressive task used to train language models. Despite CLM is currently used to train GPT-like (Radford and Narasimhan, 2018) transformer models, this task has been commonly used before to train models like RNNs based models such as Long Short-Term Memory (LSTM) and Gate Recurrent Unit (GRU) networks. The key idea of this objective is to train

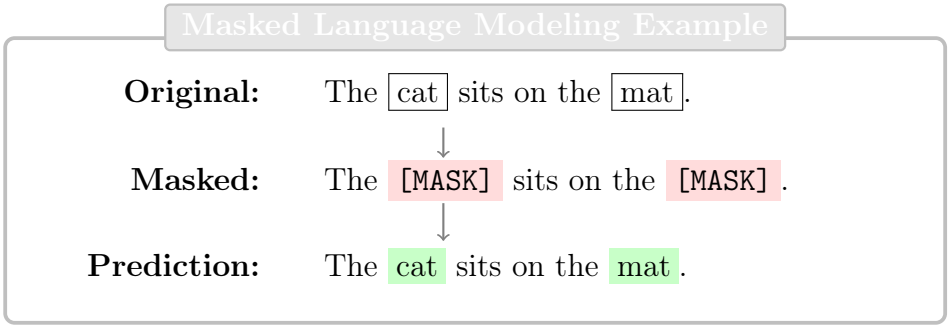

Figure 2.2: Illustration of the Masked Language Modeling task

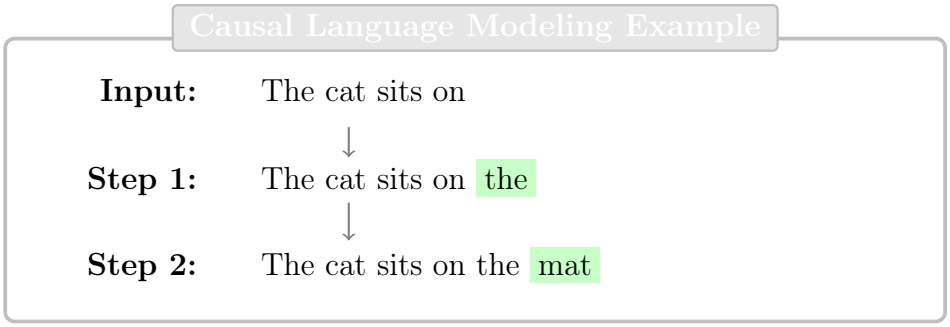

Figure 2.3: Illustration of the Causal Language Modeling task

the model to predict the next word in a sequence (as shown in Figure 2.3), given all the previous words. Although, differently from MLM (Section 2.3.1), the autoregressive behavior of this approach does not allow for a bidirectional understanding of the input, CLM aligns with natural language generation tasks, where the model needs to produce text sequentially. This straightforward objective achieved remarkable success and is currently one of the most used pretraining objectives to train large language models.

Permutation Language Modeling (PLM)

The Permutation Language Modeling task (PLM) is an unsupervised pretraining objective introduced by (Yang et al., 2019b) with the XLNet model. With this task, the authors leveraged the well-known Autoregressive Causal Language Modeling task but improved it, allowing a bidirectional understanding of the input. The key idea of this task is to let the model learn the token representation of the next token, providing different factorizations of the input and asking the model to predict the next token (as shown in Figure 2.4).

This mechanism allows the model to maintain autoregressive behavior while acquiring a better bidirectional understanding of the input.

Next Sentence Prediction (NSP)

The Next Sentence Prediction task (NSP) is a pretraining objective first presented in Devlin et al. (2019). This task has been used in parallel with MLM (Section 2.3.1) to

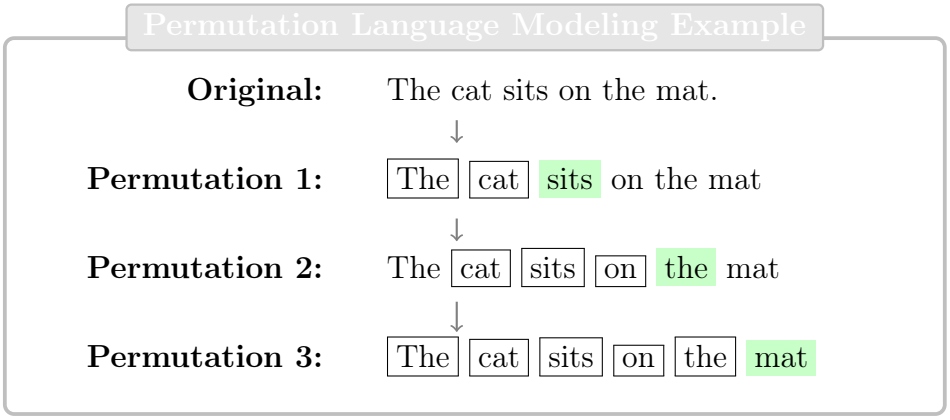

Figure 2.4: Illustration of the Permutation Language Modeling task

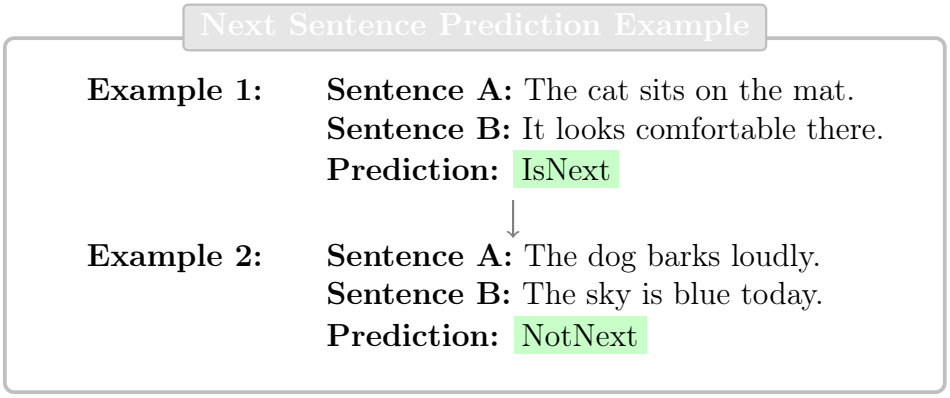

Figure 2.5: Illustration of the Next Sentence Prediction task

pretrain transformer encoder models. However, unlike MLM, which was a token-level pretraining task, NSP is a sentence-level objective. Indeed, during the training, the model sees pairs of sentences and has to predict whether the second sentence logically follows the first in the original text, as shown in Figure 2.5.

However, although this task has been initially used by several models such as BERT (Devlin et al., 2019), ALBERT (Lan et al., 2020), and DistilBERT (Sanh et al., 2019), NSP fallen out of favor in more recent models. Indeed, RoBERTa (Liu et al., 2019) showed that NSP was not making a significant contribution to the training.

Token Detection Pretraining

Token Detection (TD) was introduced by ELECTRA (Clark et al., 2020), which is an architecture composed of a discriminator and a smaller generator network. First, the generator is trained with MLM and finds suitable replacements for the masked tokens, as in BERT. Then, those candidates are inserted in the original sentence, and the resulting sequence is fed to the discriminator, which classifies whether a token is original or not. TD has the advantage of computing the loss on the whole discriminator output and having a minimal memory footprint. However, the whole system is inefficient because of the presence of the generator, which is still MLM-based.

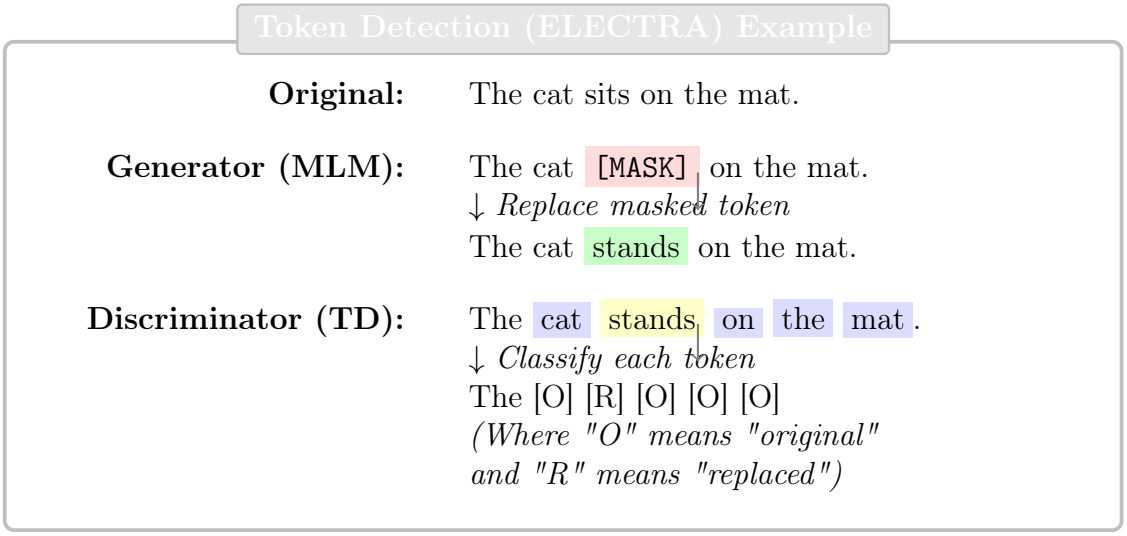

Figure 2.6: Illustration of the Token Detection task in ELECTRA

2.3.2 Pre-training datasets

This section presents some of the datasets and corpora used to pretrain transformer models. These datasets are usually extensive unlabeled collections of data commonly used together with unsupervised pretraining objectives described in Section 2.3.1.

Wikipedia

This dataset is a collection of textual data extracted from Wikipedia, an online encyclopedia[1], covering a wide range of topics such as science, history, geography, arts, and more. Specifically, this dataset, often referred to as the complete dump of Wikipedia articles, is a widely used resource for pretraining language models. Thanks to its nature as an encyclopedia, the data are structured and well-structured, containing paragraphs, explanations, citations, and references, which are comprehensive sources of valuable information for language modeling tasks. Indeed, it allows these models to learn rich representations of natural language through unsupervised learning.

BookCorpus

The BookCorpus dataset is a relatively small collection of textual data (when compared with other pretraining corpora) created by Google Research to provide a diverse and extensive source of textual information for Machine Learning purposes. The books included in the dataset were published between 2007 and 2020 and have been released under an open-source license. This dataset has been used to train language models such as BERT (Devlin et al., 2019). We report the statistics of this dataset in Table 3.1.

[1] https://www.wikipedia.org/

Dataset	# Documents	# Words	Size
Wikipedia	4,184,712	1,610,571,882	9.9GB
BookCorpus	17,868	1,149,401,064	6.2GB
Open-Web-Text	8,013,769	6,401,507,179	38.8GB
CC-News	149,954,415	62,676,498,100	385.4GB

Table 2.1: Pretraining Dataset Statistics for Wikipedia, BookCorpus, Open-Web-Text and CC-News

OpenWebText

OpenWebText is a large-scale dataset made up of public texts collected from various online sources, including Wikipedia. It aims to display a broad spectrum of language patterns, writing styles, and topics, offering more variety than datasets that only include Wikipedia. This dataset consists of content from public web pages, blogs, forums, and news articles. OpenAI developed the script for collecting this data, but it has not been released publicly. This dataset is highly beneficial for pretraining language models because it provides diverse content. By incorporating information from various parts of the internet, OpenWebText helps language models understand different contexts and writing styles, which is crucial for mastering natural language. We report the statistics of this dataset in Table 3.1.

CC-News

CC-News is a large corpora provided by the Common Crawl Foundation. It focuses on news articles gathered from the web and seeks to offer a wide-ranging collection of news articles covering many topics, languages, and sources. In fact, this vast data set includes content from various news sites, touching on topics such as politics, sports, technology, and entertainment. We report the statistics of this dataset in Table 3.1.

2.3.3 Existing Pretrained Models

In the last few years, the Natural Language Processing field has seen rapid improvements in terms of public recognition and advancement. One of the significant things that allowed this fast evolution is the development of the transformer architecture and its easy application to existing problems. Indeed, as we mentioned in Section 2.3, the transformer architecture exhibited exceptional performance across a wide range of NLP tasks, being able to outperform also human baselines. In addition, due to its pretraining capability, the architecture described in Vaswani et al. (2017) can be easily modified and trained in different ways on different tasks with different data, and with in sizes. In literature, we refer to these models as "pretrained models" (PM). Each PM represents a distinct approach to pretraining and architecture, contributing to the rich landscape of transformer-based NLP models and advancing state-of-the-art across various tasks. In this Section, we briefly describe the principal models discussed in this Thesis.

BERT (Bidirectional Encoder Representations from Transformers)

BERT (Devlin et al., 2019) is one of the first and most famous pretrained models, and when it was released, it was a breakthrough model setting the bar in many different NLP tasks. Differently from the other pretrained transformers already published at the time, BERT was trained in a particular way to have a bidirectional understanding of the input sequences. In detail, BERT is an only-encoder architecture trained using a multitask learning solution based on the Masked Language Modeling (Section 2.3.1) and the Next Sentence Prediction (Section 2.3.1) tasks. Indeed, it introduced a new language representation model designed to pre-train deep bidirectional representations from the unlabeled text by joint conditioning on both the left and right context in all layers. BERT was trained on large corpora such as the English Wikipedia (Sec. 2.3.2) and the Book Corpus (Sec. 2.3.2). At the time, it achieved state-of-the-art results on various natural language processing tasks with minimal task-specific architecture modifications. For instance, fine-tuning BERT with just one additional output layer can produce high-performance models for tasks like question answering and language inference. BERT's simplicity and effectiveness are evident in its performance on multiple benchmarks.

ELECTRA

ELECTRA (Clark et al., 2020) is a model that adopts the Generative Adversarial Network (GAN) paradigm, comprising a generator and a discriminator. Unlike traditional masked language modeling (MLM) pretraining methods such as BERT or RoBERTa, which corrupt input by replacing tokens with [MASK] and reconstructing them, ELECTRA proposes a more sample-efficient pretraining task called replaced token detection. In this approach, the input is corrupted by replacing some tokens with plausible alternatives sampled from a small generator network. Subsequently, instead of training a model to predict the original identities of the corrupted tokens, ELECTRA trains a discriminative model to distinguish whether each token in the corrupted input was replaced by a generator sample or not. Thorough experiments demonstrate the efficiency of this new pretraining task compared to MLM, as it operates over all input tokens rather than just the tiny subset that was masked out. Consequently, the contextual representations learned by ELECTRA substantially outperform those learned by BERT, even with similar model sizes, datasets, and computational resources. Particularly noteworthy is its performance with smaller models; for instance, a model trained on a single GPU for 4 days outperforms GPT (trained with significantly more computing) on the GLUE natural language understanding benchmark. Furthermore, ELECTRA demonstrates strong performance at scale, comparable to RoBERTa and XLNet, while utilizing less than 1/4 of their computational resources. ELECTRA outperforms both RoBERTa and XLNet in scenarios with the same computational resources.

GPT (Generative Pretrained Transformer)

GPT (Radford et al., 2019; Radford and Narasimhan, 2018; Brown et al., 2020): GPT is an autoregressive language model trained using the Closed Loop Memory (CLM) task. It operates by predicting the next token in a sequence based on the preceding tokens, enabling it to generate coherent text. GPT's significance extends beyond its autoregressive capabilities. By leveraging vast amounts of text data, such as the WebText dataset comprising millions of web pages, GPT demonstrates the ability to learn natural language processing tasks without explicit supervision. Conditioned on a document and questions, GPT-generated answers achieve an F1 score of 55 on the CoQA dataset, surpassing three out of four baseline systems without utilizing the 127,000+ training examples. The model's capacity plays a pivotal role in zero-shot task transfer, with performance improving logarithmically across tasks as capacity increases. Notably, GPT-2, the largest model with 1.5 billion parameters, achieves state-of-the-art results on seven out of eight language modeling datasets in a zero-shot setting, albeit still underfitting WebText. Sample outputs from the model reflect these advancements, displaying coherent text paragraphs and suggesting a promising avenue for developing language processing systems that learn tasks from naturally occurring demonstrations. Furthermore, recent advancements highlight the efficacy of pre-training large language models on extensive text corpora followed by fine-tuning on specific tasks. While this approach remains task-agnostic in architecture, it necessitates task-specific fine-tuning datasets comprising thousands or tens of thousands of examples. In contrast, humans can adeptly perform new language tasks with minimal examples or simple instructions, which is a challenge for current NLP systems. Scaling up language models, exemplified by GPT-3 with 175 billion parameters, significantly enhances task-agnostic, few-shot performance, sometimes even rivaling prior state-of-the-art fine-tuning approaches. GPT-3 achieves impressive results across numerous NLP datasets, including translation, question answering, cloze tasks, and tasks requiring on-the-fly reasoning or domain adaptation, such as unscrambling words or performing arithmetic. However, GPT-3 still encounters challenges in some few-shot learning scenarios and faces methodological issues related to training on large web corpora. Moreover, it generates indistinguishable news article samples from those written by humans, raising important societal considerations regarding its broader impact.

T5 (Text-to-Text Transfer Transformer)

T5 (Raffel et al., 2020b) stands as an encoder-decoder transformer model that employs a text-to-text approach for pretraining on diverse tasks. Its versatility allows fine-tuning for summarization, text classification, machine translation, and question-answering tasks. Transfer learning has emerged as a cornerstone technique in natural language processing (NLP), where models are initially pre-trained on data-rich tasks before fine-tuning on downstream tasks. This approach's efficacy has led to a proliferation of methodologies and practices in the field. In their paper, Raffel et al. explore the landscape of transfer learning techniques for NLP by introducing a unified framework that translates all text-based language problems into a text-to-text format. Their system-

atic study compares pretraining objectives, architectures, unlabeled datasets, transfer approaches, and other factors across dozens of language understanding tasks. By amalgamating insights gleaned from their exploration with scale and leveraging their new "Colossal Clean Crawled Corpus," the authors achieve state-of-the-art results on numerous benchmarks, spanning summarization, question answering, text classification, and beyond. To foster future research in transfer learning for NLP, they generously release their dataset, pre-trained models, and accompanying code.

Large Language Models

Large language models are considered the most recent breakthrough in the fields of natural language processing and artificial intelligence over the last few years. Such models are neural networks trained with massive text data, understanding and generating human-like text related to a wide variety of tasks. Rapid improvement has been made for LLMs in this field, and several models have contributed to a big jump in state-of-the-art fields. These genius LLMs have changed the face of AI research by giving much more natural, capable language interactions. They have found applications in everything from content creation and generation of code to question answering and language translation. Given continued research, we can foresee considerable performance, efficiency, and applicability improvements in various domains. The fast technological advancement driven by LLMs raises important questions about their impact on society, ethical usage, and the need for responsible development and deployment of these powerful tools. Some examples of Large Language Models are Mistral, LLaMA, and GPT-3.

Mistral: The Mistral model (Jiang et al., 2023), the flagship model of Mistral AI, was also highly discussed within the LLM community due to its excellent architecture that ensured excellent performance considering its relatively small size. Mistral showed that many innovations in the design of models could realize impressive capabilities without an enormous count of parameters.

LLaMA: Another very influential work in this respect is Meta's large language model, LLaMA (Touvron et al., 2023). LLaMA has shown that small models can be as good as very large ones if trained efficiently on diverse, high-quality data. This might entail huge implications for just how accessible and deployable LLMs really are.

GPT-3: The family that has probably had the most influence is that of the GPT, or Generative Pre-trained Transformer, from OpenAI. When the GPT-3 (Brown et al., 2020) model increased its capacity to 175 billion parameters, it made an enormous leap in LLM capacity, allowing few-shot learning with remarkable capabilities and versatility across many language tasks. Its successor, GPT-4, further increased performance and added multimodal capabilities for processing both text and images.

2.4 Question Answering

Question Answering (QA) is the task of automatically answering questions posed in natural language. A typical QA pipeline consists of multiple modules that handle two primary functions: (i) the retrieval of data containing relevant information and (ii) the extraction or generation of the correct answer span that is provided to the user. QA systems deal with many challenges, including how to organize data for retrieval, parse unstructured text, handle and clarify questions, and find the correct answers from documents. For the retrieval phase, data can be set up in different ways, like ontologies, graphs, or simple lists. Ontologies and graphs make it easier to access important information quickly but take time to build and keep up. Meanwhile, lists of documents are more straightforward to put together and update but store data in an unstructured format, making it harder to search through efficiently.

This Thesis primarily focuses on the question-answering field and, more specifically, on open-domain question Answering (ODQA). ODQA is a branch of QA where questions are not restricted to a single domain but can potentially cover all the possible domains covered by human knowledge. This task presents unique challenges as the system does not know the domain of the question beforehand. For example, a typical ODQA system could be a QA pipeline that, given a question, searches for the possible answer or relevant documents online and then uses complex machine learning models to extract or generate the final answer.

2.4.1 Question Answering Pipeline

In literature and in the real world, there are multiple approaches and pipelines to solve question-answering tasks. Generally, choosing the best approach for a given problem is a complex task. For example, specific applications have to deal with efficiency constraints but without particular constraints on the output naturalness. In contrast, others must guarantee accurate and natural answers at the expense of the computational effort. In this Thesis, we focus on a specific question-answering pipeline (shown in Figure 2.7), which consists of three main components: (i) a document retrieval component, (ii) a ranker, and (iii) a Generative/Extractive Model for QA.

Document Retrieval: The first step in a question-answering pipeline is retrieving documents. The most common techniques leveraged to reach this goal are generally information retrieval (IR) approaches. Indeed, given an input query, these methodologies allow to find relevant documents containing information related to the input query and often rely on search algorithms to (i) scan extensive collections of data, (ii) build efficient indexes, and (iii) employ sophisticated lexical and semantic similarity approaches. Intuitively, the principal role of the retrieval systems is to reduce the vast amount of available data to a smaller and more manageable set of data that is most likely to contain a correct answer for the input query (Manning et al., 2008; Robertson and Zaragoza, 2009). In detail, the methods used in this step range from traditional lexical IR techniques like BM25 (Robertson and Jones, 1976) and TF-IDF (Salton and

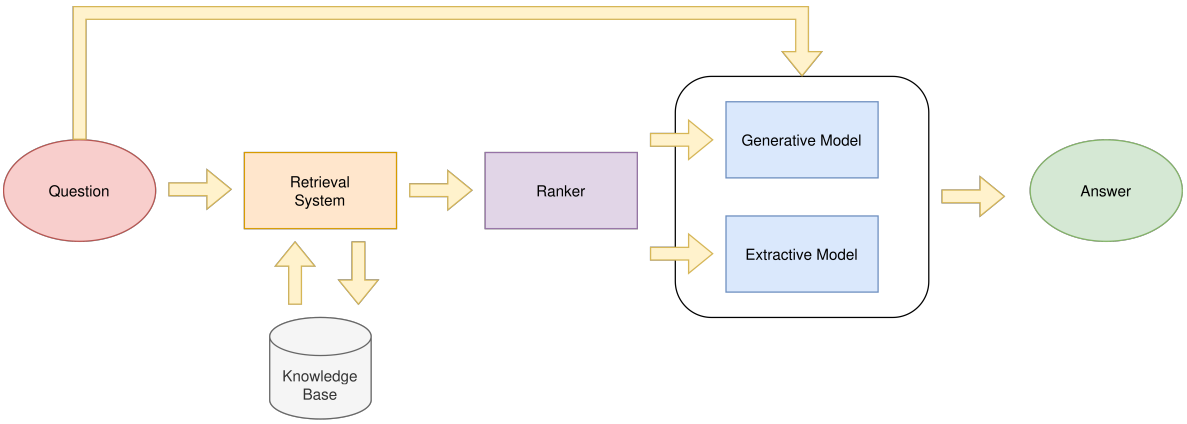

Figure 2.7: General ODQA pipeline discussed in this Thesis

Buckley, 1988) to advanced neural network approaches such as (Karpukhin et al., 2020; Campese et al., 2023) based on deep models like the transformer architecture.

Ranking: After the retrieval of the relevant documents, the following step process is the passage ranking phase. At this point in the pipeline, the IR approaches used in the retrieval step reduced the quantity of data to a small subset that could be processed by more sophisticated and powerful (but computationally expensive) algorithms. The main role of the ranking phase is to further reduce and clean the set of relevant documents in order to keep only the ones that contain the context needed to provide the final answer. To reason about the magnitude and the size of this set of documents, we can think that in real applications of this pipeline, the retrieval system may have selected several thousands of relevant documents: the ranker must provide to the next component of the pipeline only a tiny part of these (generally one or two magnitudes smaller). To do it, the ranker, which is usually a deep neural model, has to read, review, and score the documents, trying to maximize the relevance of the input query. These scores are then used to rank relevant documents (Nogueira and Cho, 2020; Yang et al., 2019a; Garg et al., 2019).

Generative/Extractive Models for QA: The final step in a question-answering (QA) pipeline involves using either Generative or Extractive Models to provide the answer.

- **Extractive Models:** These models recognize and extract the exact text containing the answer from the ranked passages. They perform it using specific architectures trained to highlight the most relevant text spans (Rajpurkar et al., 2016b). These approaches are highly effective when the answer is directly found in the retrieved passages but can suffer several weaknesses when the answer is not contained in single spans of text or questions requiring aggregation of concepts from different documents. The performance of these models is simpler to evaluate and control than the generative ones because the output is always a piece of the input. However, the output is not always natural, and generally, they are not

good at performing reasoning operations since they can not synthesize an answer generating it.

- **Generative Models:** Unlike extractive models, given an input query, generative models generate the final answers token by token. These models receive in input the relevant documents retrieved and ranked from the previous steps, analyze them, and then synthesize an answer by combining information from different contexts. Generally, these models present more natural and complex answers than the extractive models since they have better summarization and rephrasing capabilities (Radford et al., 2019). In contrast to the extractive methods, the performance of these models is more difficult to evaluate since their output is not directly extracted from the input, and for this reason, comparing it with a gold reference answer could be a not trivial operation to do manually. Another problem involving these models is "hallucination," which happens when the generated answer is wrong and not grounded but is likely to be true.

Each component of the pipeline is specialized to address specific QA tasks. In the following sections, we will provide a detailed description of these components and tasks, presenting baselines and existing approaches. Firstly, in Section 2.4.2, we discuss the retrieval component, covering tasks such as semantic similarity and existing retrieval systems. Secondly, in Section 2.4.3, we describe the ranker component and its tasks, including Passage Ranking (PR) and Answer Sentence Selection (AS2). Finally, in Section 2.4.4, we examine the last component of the pipeline, which may involve either extractive or generative QA techniques.

2.4.2 Retrieval: Document Retrieval & Semantic Similarity

The first component shown in Figure 2.7 represents the retrieval system. Document retrieval is a crucial stage of the pipeline since it involves the identification and retrieval of a set of documents that are likely to contain information relevant to the user's query. In most cases, this process is conducted by Information Retrieval systems, which are generally search algorithms that allow quick scanning of extensive text collections (Manning et al., 2008). The IR systems take advantage of advanced indexing methods and search techniques to find their way through documents and data sources in search of the query terms. Document retrieval aims to reduce the available data to a minimal number in a subset of relevant data likely to bear the answer. This step is foundational for the subsequent components of the QA pipeline because of the kind of essential context from which specific answers can eventually be extracted. It consists of searching in massive corpora for documents relevant to the user's query.

The retrieval step begins with indexing, which includes parsing the documents, looking for key terms, and storing them in a structure called an index for fast searching. Traditional retrieval systems often use methods like TF-IDF Term Frequency-Inverse Document Frequency or BM25. These models compute the importance of terms in documents and a collection to rank order documents by significance. All these techniques

are oriented to token-level matches and statistical properties of text; hence, they are very good at finding documents that contain query terms.

Token Similarity Task: This task of token similarity focuses on the individual words or tokens in the query and documents. The notion is to compute how much the tokens in the query are close to those in the documents. Conventional methodologies could be applied by matching keywords, but most of these conventional methods lack when considering language variations. More sophisticated approaches apply measures of token-based similarity, including cosine similarity, Jaccard similarity, or edit distance, to capture variations and misspellings. These methods allow to find documents that, though not matching the query terms, have very high relevance at the token level.

Recent advancements in IR have introduced dense retrieval methods, which rely on deep learning techniques for forming dense vector representations for queries and documents. ColBERT and DPR (Khattab and Zaharia, 2020; Karpukhin et al., 2020) are methods that, after using a pre-trained language model for query and document encoding into high-dimensional vectors, apply some similarity measure to find relevant documents. This process is where semantic information is captured in such vectors and subjected to similarity measures like cosine similarity for the identification of relevant documents. In fact, the dense approaches are compelling at handling the subtleties of natural language. That is, context and meaning are captured beyond simple token matches.

Semantic Similarity Task: Semantic similarity can go beyond token similarity, for it considers the meaning of words and expressions in a context. In this task, understanding is needed at a conceptual level concerning what a query is and what documents pertain to. This involves certain specific measures making use of word embeddings, such as Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), and, more recently, contextual embeddings like BERT and GPT, which model semantic relationships among words (Devlin et al., 2019; Brown et al., 2020). These are embeddings of words as high-dimensional vectors carrying semantic information, which allow the retrieval system to identify contextually relevant documents to a query, even if they do not share exact words. Because the semantic similarity allows the retrieval system to handle synonyms, polysemy, and different expressions very well, it would significantly improve the relevance of retrieved documents. It is, in particular, useful for systems such as QUADRo (QUestion Answer Database Retrieval) (Campese et al., 2023), which tries to semantically match the user's query against its database of question/answer pairs to find a proper match as efficiently as possible. This semantic similarity will help QUADRo retrieve and rank answers contextually most appropriate, eventually improving the whole accuracy and effectiveness of the QA pipeline.

2.4.3 Ranking: Passage Ranking & Answer Sentence Selection

Once an IR system, as described in Section 2.4.2, has retrieved a relevant set of documents, the next step of the QA pipeline described by Figure 2.7 is the Ranking phase. This crucial step performs the analysis of the retrieved documents, identifying and ranking the documents, or more in general, the data, most relevant to the input query (Manning et al., 2008). Ranking systems are part of current QA pipelines that digest the broad set of retrieved documents to be then refined into a more focused relevant subset, significantly increasing the chance of finding the correct answer (Robertson and Zaragoza, 2009). Such ranking systems can use traditional and modern techniques and ensure that only the most relevant information reaches the next pipeline stage.

Passage Ranking Task (PR): The passage Ranking step is the task of assessing and ranking the retrieved passages for their relevance to the input query. This constitutes an important module of QA pipelines since it decides which among retrieved passages bear information most helpful in answering the question (Nogueira and Cho, 2020). Effective ranking techniques not only increase the precision of the QA system but also put the most contextually appropriate and informative passages higher in the list, thus facilitating better answer extraction in further stages (Clark and Gardner, 2018). Traditional methods like BM25 and TF-IDF laid the foundation of passage ranking, while more recent methodologies either attain their backbone from advanced neural network models or implement state-of-the-art transformer models with advanced dense retrieval techniques (Karpukhin et al., 2020). These models leverage semantic understanding at both the question and passage levels and thus make ranking more subtle and accurate (MacAvaney et al., 2020). Essentially, cross-attention mechanisms and other late-interaction operators have been proposed to enhance the estimation of passage relevance such that it becomes possible for the QA system to narrow down efficiently the most relevant information in the generation of an answer (Qu et al., 2021).

Answer Sentence Selection (AS2) Task: The Answer Sentence Selection task is to select the correct sentence that answers a given question and pinpoint the exact response. Much research has been done into approaches to solving this task, each of which applies new techniques. Traditional methods include heuristic rules or simple classifiers; more recently, neural network-based models have been utilized, especially Transformers like BERT. These models learn the context of the question and the candidate sentences and adjust their accuracy accordingly by further fine-tuning with AS2-specific datasets. Severyn and Moschitti (2016) proposed separated embeddings for questions and answers using convolutional layers along with relational information to enrich the embeddings. Later Tayyar Madabushi et al. (2018) integrated deep learning with Question Classification, thereby improving the performance of AS2. Garg et al. (2019) implemented transformer-based models using intermediate fine-tuning for domain adaptation with ASNQ derived from Natural Questions (Kwiatkowski et al., 2019). Information extracted for context aids AS2 models (Tan et al., 2018), with hierarchical neural net-

works integrating context-dependent representations (Lauriola and Moschitti, 2021a). Vu and Moschitti (2021b) handled multilingual challenges in AS2 by translating the English data of the AS2 task and further proposed multilingual ranking models. Gupta et al. (2023) tackled low-resource languages using cross-lingual knowledge distillation (CLKD). The performance was improved without labeled data in target languages. Machine translation quality should be taken into consideration for further improvements since that is of prime importance to the most effective multilingual solutions for AS2 tasks.

2.4.4 Getting an Answer: Machine Reading & Generative QA

After the Document Retrieval phase, where the Retrieval System retrieves relevant documents for the question, and the Ranking phase cleans and ranks the retrieval set, the final component of the pipeline is responsible for extracting or generating the final answer to the given question. Traditionally, this involves an extraction model that processes the query and the documents containing potential answers to extract a concise response. However, in recent years, thanks to the capability of large language models, the extractive paradigm has been shifting towards the generative one.

Machine Reading (MR) Task: After the Document Retrieval phase retrieves the relevant documents, the last task in the question-answering pipeline is Machine Reading. It primarily seeks to extract information from the retrieved passages that are nuanced and contextually relevant, leading to a response directly answering the user's question. Extractive models are used with state-of-the-art transformer architectures like BERT (Devlin et al., 2019). These models are carefully trained to acquire high performance in finding and extracting specific text spans within the retrieved documents that contain precise answers or pertinent information related to the query (Rajpurkar et al., 2016a). The Extractive Models read through the context provided by the query and consider the surrounding text in a retrieved document. These kinds of models can efficiently capture and highlight precisely the most relevant and informative text segments because of their use of pre-trained embeddings and the attention mechanisms (Bahdanau et al., 2014a). Machine Reading does something beyond simple extraction; it acts as a crucial intermediate step between the initial document retrieval and the answer generation at the end. A task like this assures that the latter parts of the pipeline work with the most relevant and contextually accurate content possible through complete comprehension and extraction of key information in the retrieved passages. Extractive models have intrinsic advantages when used in machine reading. They extract high precision by selecting text spans directly from the documents retrieved through the model, which guarantees that the answers are always present in the source texts. Besides this, models demonstrate scalability by quickly processing large volumes of text and hosting variable data sets with complex queries. Their approach also encourages this because the extraction of text spans makes it transparent to a user how the reasoning process has taken place to trace back how an answer is derived from the original documents. However, the Extractive Models are not free from challenges. Complex queries or ambiguous

language could cause problems in exactly identifying the proper text spans. Moreover, not all the answers may always be present in the documents; hence, extraction alone can return an answer for only a subset of possible questions. This is another reason why hybrid models form the most promising way forward in integrating extractive and generative approaches as the field progresses toward more robust QA systems, such as those done by (Wang et al., 2019).

Answer Generation Task: Traditional Extractive approaches to Machine Reading have recently been complemented or superseded by generative methods. Contrary to the selection of existing text spans that characterize the extractive methods, generative models take as input the query and contextual documents and generate answers on the fly (Lewis et al., 2020a; Hsu et al., 2021). One of the most important advantages of the generative approaches over the extractive ones is that a generative model can tackle and potentially answer questions that do not have a correct answer present in the retrieved documents (and so generating an answer based on its parametric knowledge, or answering questions which require inductive and reasoning steps. Indeed, generative models can synthesize answers through source text aggregation, many of which often make coherent responses beyond what can explicitly be found in documents, helping alleviate many challenges in providing contextually accurate responses. By incorporating generative models into the answer generation task, informative and adaptable answers could be developed that would give users a better experience of QA systems. For this reason, much answer generation research has been done using methods such as retrieval-augmented generation, which has been explored by studies Izacard and Grave (2021); Lewis et al. (2020b). In addition, question-based summarization approaches have also been looked into to enhance answer quality, among others, by Iida et al. (2019); Goodwin et al. (2020); Deng et al. (2020). Asai et al. (2022a) incorporated evidentiality from passages into generator training for open-domain MR span-extraction tasks. Xu et al. (2021) innovatively obtained extractive answer spans from generative models using decoder cross-attention patterns, while Fajcik et al. (2021) proposed a hybrid approach combining generative and extractive readers for open-domain span-extraction. Efforts have also explored generating complete sentences as answers, akin to those by extractive Answer Span Selection (AS2) systems. Hsu et al. (2021) introduced GenQA, where an AS2 model selects optimal answer candidates for subsequent generation by GenQA models. This approach enables learning to produce natural responses using top-ranked candidates, extending to multilingual QA by Muller et al. (2022) to broaden answer generation across languages.

2.5 Datasets

Datasets are a fundamental part of every NLP and QA application. Indeed, these collections of data are not only necessary to train machine learning models but also to test and benchmark them. For QA, datasets generally contain pairs of questions and their relevant answers, sometimes with any extra context like supporting documents or

passages. The richness and quality of these datasets have a direct impact on how well QA models work and generalize. High-quality datasets containing diverse examples allow the model to grasp the language very well and tackle broad content related to questions and contexts. In this section, we will review some important datasets that have powered advances in QA and NLP. These datasets range from answer sentence selection to machine reading comprehension and open-domain question answering, each with a different set of challenges and gains in insight. These datasets will be important in understanding the following chapters. They are also fundamentally crucial to training and assessing the models and methods described in this Thesis. We give an overview of benchmark datasets designed to probe general language understanding and specific QA tasks. These benchmarks use standard evaluation metrics, making it easier to compare different models and approaches. Next, we will introduce sets of specialized datasets for tasks like answer sentence selection, wherein the model has to select the most relevant sentence from a list of candidates. Thereafter, we shall see datasets that are focused on the task of machine reading comprehension, those requiring the extraction of answers from the given passages. We will then describe datasets related to open-domain QA tasks, entailing finding and combining information from diverse sources. Knowing these datasets and their usage, we can realize the methods and advances that have shaped the current state of QA and NLP. Each one of these datasets contributes to the progress in this domain by offering valuable resources for training, testing, and improvement in QA systems.

2.5.1 Benchmark Datasets to Evaluate Language Models

GLUE

The General Language Understanding Evaluation (Wang et al., 2018) is a comprehensive benchmark suite designed to evaluate the performance of natural language understanding models across a range of tasks. This collection contains diverse datasets for evaluating models on various natural language understanding tasks, including Question Answering, Natural Language Inference, Question Pairs Detection, Entailment Recognition, and Language Acceptability. By covering a spectrum of tasks, GLUE allows to get an assessment of a model's ability to comprehend and process natural language. In detail, GLUE contains nine sub-tasks logically.

In this Thesis, we used the GLUE benchmark to evaluate new pretraining approaches in Chapter 3. Indeed, the only reasonable way to understand the capabilities of new unsupervised pretraining tasks is to test them on some fine-tuning tasks. Under these circumstances, the GLUE evaluation suite provides some interesting and valuable tools to evaluate these loss functions.

Table 2.2 describes the statistics of the GLUE benchmark tasks. In detail, the tasks covered by GLUE are:

CoLA: The Corpus of Linguistic Acceptability (Warstadt et al., 2019) consists of English acceptability judgments sourced from linguistic theory books and journal articles. Each example is a sequence of words annotated to indicate whether it forms a

	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE	WNLI
Train	8.5k	67k	3.7k	7k	364k	393k	105k	2.5k	634
Test	1k	1.8k	1.7k	1.4k	391k	20k	5.4k	3k	146

Table 2.2: The statistics for the different tasks that compose the GLUE benchmark. The table reports the number of examples for both the train and the test splits divided per task.

grammatical English sentence. The evaluation metric used is the Matthews correlation coefficient (MCC), which ranges from -1 to 1, with 0 representing the performance of uninformed guessing. Performance is reported based on the combination of in-domain and out-of-domain sections of the standard test set.

SST-2 : The Stanford Sentiment Treebank (Socher et al., 2013) consists of sentences extracted from movie reviews, each annotated by humans for sentiment. The task associated with this dataset is to predict whether a given sentence expresses a positive or negative sentiment. This binary classification challenge focuses solely on sentence-level sentiment labels, providing a clear and focused framework for evaluating the sentiment analysis capabilities of various models.

MRPC: The Microsoft Research Paraphrase Corpus (Dolan and Brockett, 2005) is a collection of sentence pairs automatically extracted from online news sources, with each pair annotated by humans for semantic equivalence. This dataset presents a challenge due to its imbalanced class distribution, with 68% of the pairs labeled as positive (semantically equivalent). As a result, both accuracy and F1 score are reported to provide a comprehensive evaluation of model performance on this paraphrase detection task.

QQP: The Quora Question Pairs (Chen et al., 2017c) dataset comprises pairs of questions from the community question-answering website Quora, with the task being to determine whether the questions in each pair are semantically equivalent. Similar to MRPC, QQP has an imbalanced class distribution, with 63% of the pairs labeled as negative (not semantically equivalent). To address this, both accuracy and F1 score are used as evaluation metrics, ensuring a robust assessment of model performance.

STS-B: The Semantic Textual Similarity Benchmark (Cer et al., 2017) contains sentence pairs drawn from various sources, including news headlines, image captions, and inference data. Each pair is human-annotated with a similarity score ranging from 1 to 5. The task is to predict these similarity scores, with performance evaluated using Pearson and Spearman correlation coefficients. This benchmark challenges models to accurately capture the degree of semantic similarity between sentences, providing a nuanced assessment of their understanding capabilities.

MNLI: The Multi-Genre Natural Language Inference Corpus (Williams et al., 2018) is a comprehensive dataset consisting of sentence pairs with textual entailment annotations. The task is to evaluate whether the premise entails, contradicts, or is neutral to the hypothesis. This dataset is notable for its diversity, drawing from ten different sources, including transcribed speech, fiction, and government reports. It includes both matched (in-domain) and mismatched (cross-domain) sections for evaluation, allowing models to be tested on their ability to generalize across different genres and domains.

QNLI: The Question Natural Language Inference (White et al., 2017) dataset is derived from the Stanford Question Answering Dataset (SQuAD). It converts the question-answering task into a sentence pair classification task, where the objective is to determine whether a given context sentence contains the answer to a corresponding question. This transformation shifts the focus from finding the exact answer to evaluating the presence of the answer within the context, thus challenging models to understand and match the relevant information in sentences accurately.

RTE: The Recognizing Textual Entailment (RTE) dataset combines data from several RTE challenges (Dagan et al., 2005; Bar-Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009), which focus on textual entailment from news and Wikipedia texts. The task involves determining whether one sentence logically follows from another, simplified into a two-class classification problem: entailment or not entailment. This dataset provides a rich resource for training and evaluating models on their ability to perform entailment recognition, which is crucial for many natural language understanding applications.

WNLI: The Winograd Schema Challenge (Levesque et al., 2012) is a reading comprehension task designed to resolve pronoun reference ambiguities within sentences. This dataset constructs sentence pairs by replacing ambiguous pronouns with possible referents, with the task being to predict if the sentence with the substituted pronoun is entailed by the original sentence. The WNLI is specifically crafted to be challenging for simple statistical methods, requiring a deep understanding of contextual information to resolve ambiguities accurately.

CodeXGLUE

CodeXGLUE (Lu et al., 2021) is a benchmark dataset and open challenge platform for code intelligence developed by Microsoft Research Asia, Developer Division, and Bing. It includes 14 datasets for 10 code intelligence tasks, such as code completion, code-to-code translation, and code summarization. The platform supports model evaluation and comparison, featuring baseline models like CodeBERT and CodeGPT. CodeXGLUE aims to advance AI-driven code understanding and generation, enhancing developer productivity by leveraging large pre-trained models similar to those used in natural language processing. We used this dataset to evaluate our pretraining experiments on our disjoint training for the Transformer on code (Chapter 3).

Dataset	Train	Validation	Test
SQuAD	87,599	10,570	-
Natural Questions	307,373	7,830	7,842
ASNQ	19,446,120	870,404	-
WikiQA	2,118	296	633
TREC-QA	1,229	65	68
MS-MARCO	808,731	101,093	6,980
WQA	87,361	1,269	2,096
AE	5,219	299	409
QuoraQP-a	?	?	?

Table 2.3: Statistics for dataset splits of Question Answering Datasets. Specifically, we provide a number of examples for the training, validation, and test split for SQuAD, Natural Questions, ASNQ, WikiQA, TREC-QA, MS-MARCO, WQA, and QuoraQP-a. Notice that some datasets have never released a public testset.

2.5.2 QA datasets

In this section, we will explore the diverse Question Answering (QA) datasets that we consider in the following chapter of this Thesis. These datasets were carefully chosen to address a wide range of QA challenges, offering thorough benchmarks for assessing and improving QA model performance. Each dataset brings unique perspectives due to its varied complexity, sources, and annotation methods, providing valuable insights into the strengths and weaknesses of modern QA systems. Table 2.3 summarizes the statistics of the datasets considered in this section.

SQuAD

The Stanford Question Answering Dataset (SQuAD) comprises two versions designed to evaluate and enhance machine reading comprehension. SQuAD 1.1 (Rajpurkar et al., 2016a) features over 100,000 questions created by crowd workers based on Wikipedia articles, with answers being specific spans within the text. This dataset focuses on the model's ability to locate precise answers within a paragraph. SQuAD 2.0 (Rajpurkar et al., 2018b) builds on this by introducing more than 50,000 unanswerable questions that resemble answerable ones, requiring models to determine the correct answers and when no answer is present in the provided text. This addition significantly increases the dataset's complexity and realism, better simulating real-world scenarios where information might be incomplete or misleading. Both datasets serve as critical benchmarks for advancing the natural language understanding and reading comprehension capabilities of AI systems.

Natural Questions

Natural Questions, as outlined by (Kwiatkowski et al., 2019), is a large-scale dataset meticulously curated by considering authentic Google queries. Each query within the dataset is meticulously paired with a corresponding Wikipedia page and the pertinent passage containing the answer. Notably, this dataset primarily comprises questions that can be answered by a single passage, aligning with our predefined criteria for question complexity. Despite this characteristic, studying the Natural Questions dataset offers valuable insights into the limitations of the retrieval systems employed by the pipeline, as well as the inherent correlations between what constitutes complexity for human users versus question-answering systems. By analyzing this dataset, researchers gain a deeper understanding of the nuanced interplay between human comprehension and machine-based question answering, paving the way for improved system performance and robustness.

ASNQ

ASNQ (Answer-Sentence Natural Questions) is a meticulously curated dataset tailored for the task of Answer Sentence Selection (AS2) (Garg et al., 2019). Derived from the acclaimed Natural Questions (NQ) corpus (Kwiatkowski et al., 2019), originally intended for Machine Reading, ASNQ serves as a specialized resource for evaluating models on the nuanced task of selecting pertinent sentences that directly address natural language queries. Comprising thousands of questions sourced from the Google search engine, along with candidate sentences extracted from the top-ranked Wikipedia pages, ASNQ offers a rich and diverse pool of data for training and assessing models' ability to discern relevant information from a vast corpus of textual sources. Further details of the dataset are presented in Table 2.3.

WikiQA

WikiQA is a compact yet valuable dataset designed for Answer Sentence Selection, constructed from inquiries submitted to the Microsoft Bing search engine (Yang et al., 2015). Each query in the dataset has been meticulously paired with candidate answers sourced from Wikipedia articles labeled as either relevant or irrelevant. While training is conducted on the entire training split, validation and testing are carried out in a meticulously controlled environment termed "clean," wherein questions with exclusively positive or negative candidates are filtered out. This meticulous curation ensures a rigorous evaluation of the models' ability to discern pertinent information, free from the influence of imbalanced data instances. Further elucidation on the dataset is available in Table 2.3.

TREC-QA

TREC-QA stands as another prominent benchmark extensively utilized for the task of Answer Sentence Selection (Wang et al., 2007). Constructed from the TREC-8 to TREC-13 tracks of Question Answering, this dataset offers a diverse array of queries and

corresponding answers, meticulously curated to facilitate comprehensive evaluations of model performance. Similar to the approach outlined in (Garg et al., 2019), our training regimen leverages the expansive *train-all* split, optimizing for model robustness and adaptability. However, during the validation and testing phases, we exclusively consider questions featuring positive and negative answer instances, ensuring that the model can effectively identify important information even in complex situations. Notably, *train-all* encompasses a broader spectrum of data, including instances with noise and questions lacking positive answers. However, its larger size affords a more robust foundation for model fine-tuning. This standard practice of training on *train-all* and evaluating curated clean data exemplifies the prevailing setting within the TREC-QA framework. For additional insights, refer to Table 2.3.

MS-MARCO

MS-MARCO is a widely recognized machine reading dataset initially introduced by (Nguyen et al., 2016). The training split of MS-MARCO comprises approximately 800,000 unique user queries sourced from the Bing search engine, each accompanied by around 10 passages retrieved as potential answers. The dataset, available as MS-MARCO v2.1, offers a substantial resource for training and evaluating models in various natural language understanding tasks.

WQA

WQA Web Question Answers (WQA) is a publicly available dataset meticulously introduced and defined in (Zhang et al., 2021). This dataset is a treasure trove comprising 149,513 questions, each meticulously paired with approximately 15 answer candidates. The questions and their corresponding answer candidates are sourced from a vast, large-scale web index, ensuring a diverse and comprehensive coverage of real-world queries and responses. What sets WQA apart is its meticulous curation process, with each question-answer pair meticulously annotated for answer correctness by a team of professional annotators. This meticulous annotation process ensures the high quality and reliability of the dataset, making it an invaluable resource for training and evaluating models across various natural language understanding tasks.

Answer Equivalence (AE)

Answer Equivalence (AE) is a meticulously curated question-answering dataset introduced by Bulian et al. (2022). Each sample within this dataset comprises a question, a candidate answer (typically short answers), and a positive reference, meticulously chosen to circumvent exact matches between the candidate answer and the reference (known as an exact match or EM). This deliberate selection process ensures the dataset's richness and complexity, fostering the development and evaluation of models capable of discerning semantic equivalence between candidate answers and positive references beyond surface-level matching. AE is a valuable resource for advancing research in

question-answering systems, particularly in scenarios where understanding the nuanced semantic equivalence between responses is crucial.

QuoraQP-a

QuoraQP-a, introduced by (Wang et al., 2020b), is a question-answering dataset meticulously crafted by pairing existing questions from the Quora Question Pairs (QQP) dataset with their corresponding original answers. Leveraging the wealth of question pairs available in QQP, QuoraQP-a offers a valuable resource for training and evaluating question-answering models in a real-world context. By providing a diverse array of question-answer pairs sourced from the Quora platform, QuoraQP-a enables researchers to develop and assess models' ability to comprehend and respond to natural language inquiries, driving advancements in question-answering and natural language understanding.

2.5.3 Industrial QA Datasets

In the realm of industrial natural language processing, particularly within the context of virtual assistants and personal assistant technologies, the development and evaluation of question-answering (QA) models hinge on the availability and utilization of extensive and diverse datasets. This section introduces three significant industrial QA datasets that play a crucial role in advancing this field: AQAD-U, AQAD-L, and IQAD. These datasets, sourced from commercial virtual assistant platforms, offer a wealth of real-world user queries and candidate answers, providing a robust foundation for training, fine-tuning, and evaluating QA models.

AQAD-U

AQAD-U (Gabburo et al., 2022) is a substantial internal industrial question-answering dataset comprising *non-representative de-identified* user questions sourced from the Alexa virtual assistant platform. This unlabeled Alexa QA Dataset (AQAD-U) is characterized by its vast scale, containing approximately 50 million questions. For each question, the dataset includes around 400 answer candidates retrieved from a large-scale web index, encompassing over 100 million web documents. This comprehensive dataset serves as valuable transfer learning data for experiments conducted in the industrial setting, offering a rich and diverse source of real-world user queries and candidate answers. Leveraging AQAD-U facilitates developing and evaluating question-answering models in realistic scenarios, driving advancements in industrial natural language processing and virtual assistant technologies.

AQAD-L

AQAD-L (Gabburo et al., 2022) represents the labeled counterpart of the industrial dataset AQAD-U. In addition to the non-representative de-identified user questions and the associated answer candidates retrieved from a large-scale web index, AQAD-L

includes human annotations indicating the correctness or incorrectness of each answer candidate. This labeled version of the dataset offers a curated set of approximately 5,000 questions, meticulously annotated with ground truth labels, serving as valuable evaluation data for experiments conducted in the industrial setting. Results obtained on AQAD-L are typically presented relative to baseline AS2 model performance, as the dataset is internal and tailored to specific industrial requirements. Leveraging AQAD-L enables rigorous evaluation of question-answering models in real-world scenarios, contributing to the continual refinement and improvement of industrial natural language processing systems.

IQAD

IQAD (Gabburo et al., 2023b) is a substantial Industrial QA Dataset comprising non-representative de-identified user questions sourced from a commercial personal assistant platform. This internal dataset, as detailed by (Garg and Moschitti, 2021; Di Liello et al., 2022c), encompasses approximately 10,000 questions, with each question accompanied by around 200 answer candidates retrieved from a large-scale web index containing over 100 million web documents. Each question in IQAD is associated with approximately 15 answer candidates, each annotated by humans to denote correctness or incorrectness. To comply with company policies, results obtained on IQAD are typically presented relative to a baseline model. Leveraging IQAD enables rigorous evaluation and benchmarking of question-answering models in industrial settings, contributing to industrial natural language processing advancements and personal assistant technologies.

2.5.4 Complex QA datasets

Recently, with the general improvement of the question-answering methodologies, current pipelines are able to correctly answer the majority of the questions. To push the boundaries of these approaches further, several researchers have proposed QA datasets containing complex questions in the last few years. Indeed, these datasets are characterized by questions that necessitate reasoning processes, often beyond the reach of traditional QA pipelines. These questions are often used as benchmarks for developing and evaluating models that can handle the complexities of real-world queries involving multiple steps of reasoning, cross-referencing various sources of information, and interpreting implicit contextual clues. In this section, we present several complex QA datasets that will be utilized in the following chapters of this Thesis. Table 2.4 reports the main statistics of these complex QA datasets.

ComplexWebQuestions

ComplexWebQuestions (CWQ), introduced by (Talmor and Berant, 2018), stands as a pivotal dataset tailored for addressing intricate questions necessitating reasoning over multiple web snippets. The dataset comprises 34686 examples, partitioned into 27368, 3518, and 3530 instances for the train, development, and test splits, respectively. Each

Dataset	Train	Validation	Test
CWQ	27,639	3,518	3,531
HotPotQA	90,447	7,405	7,405
StrategyQA	2,290	229	490
MuSiQue	25,000	3,000	3,000

Table 2.4: Statistics for dataset splits of Complex Question Answering Datasets. Specifically, we provide a number of examples for the train, validation, and test split for CWQ, HotPotQA, StrategyQA, and MuSiQue.

example within the dataset consists of a question, an answer, and a SPARQL query meticulously crafted to retrieve the relevant web snippets necessary for constructing the context. ComplexWebQuestions is an invaluable resource for advancing research in natural language understanding and complex question-answering systems by encapsulating the complexities inherent in multi-faceted questions requiring cross-snippet reasoning.

HOTPOTQA

HOTPOTQA, introduced by (Yang et al., 2018b), stands as a comprehensive QA dataset meticulously designed to encompass complex questions that necessitate reasoning and additional context for accurate answers. To facilitate answering each question, the dataset includes diverse paragraphs that could serve as possible contexts. By providing this multi-faceted approach to question answering, HOTPOTQA facilitates the evaluation and development of models capable of understanding and reasoning over intricate queries, thereby advancing research in natural language understanding and complex question-answering systems.

StrategyQA

StrategyQA, as introduced by (Geva et al., 2021), represents a pioneering question-answering benchmark meticulously crafted to address open-domain queries where the underlying reasoning steps are implicit and need to be inferred through a discernible strategy. Comprising 2,780 examples, each instance within StrategyQA encapsulates a strategy question alongside its decomposition and supporting evidence paragraphs. Notably, the "train" split emerges as the primary utility source within the dataset, as it provides both the decompositions and the factual evidence necessary for training robust and effective question-answering models. By focusing on questions that demand strategic reasoning and implicit inference, StrategyQA offers a unique and challenging testbed for evaluating the capabilities of natural language understanding systems, driving advancements in complex question answering.

MuSiQue

MuSiQue, introduced by (Trivedi et al., 2022), represents a unique question-answering dataset created through a bottom-up approach, amalgamating multi-hop questions with single-hop questions from various datasets. Unlike previous methodologies, MuSiQue adopts a meticulous process to ensure the naturalness and coherence of the decomposed

questions, resulting in a more seamless integration of the components. By leveraging this approach, MuSiQue provides a rich and diverse set of questions that challenge models to perform both single-hop and multi-hop reasoning, thereby fostering advancements in the field of question answering and natural language understanding.

Chapter 3

Understanding Language Models

In recent years, language models have become important tools used by millions of humans on a daily basis. Indeed, under the hood of applications such as ChatGPT, Amazon Alexa, Apple Siri, and many other products, there are mechanisms involving this technology. As we mentioned in Chapter 2, in the last decade, the Natural Processing Field exploded, becoming one of the most funded research areas in computer science. In this chapter, and in the scope of this Thesis, we explore transformer architecture deeply, focusing on their training techniques. Specifically, we perform this analysis by organizing the chapter into three parts and proposing different case studies. Each case study considered in this chapter is based on research projects done during my PhD program.

In the first part (Section 3.1, we provide a comprehensive overview of the transformer architecture, discussing how it can be used to solve various tasks across different domains. Specifically, we implemented a transformer architecture from scratch, retrieved the data, and designed the pretraining and fine-tuning objectives to solve tasks related to the code domain. For each of these parts, we will analyze the critical aspects, providing insights and highlighting the versatility of the transformer architecture compared to older architectures (e.g., RNNs). In addition, in this part, we demonstrate that these language models can be applied not only to traditional language tasks but also to novel domains such as code generation and analysis. We performed this research in 2019. At that time, the pretrained language models were not as common as they are today, and when we wrote the first draft of this paper (we tried to publish at IJCAI 2020), there were no significant approaches or research papers using the transformer architecture for the code domain.

In the second part (Section 3.2), we explore in depth the pretraining of transformer encoders (Sec. 2.2.1). In detail, we analyze existing pretraining objectives (e.g., Masking Language Modeling), analyzing their tradeoff between efficiency and performance. After this analysis, we propose new pretraining methods with a special focus on efficiency without compromising model performance. In addition, we perform an extensive analysis of these approaches, evaluating them on existing downstream tasks and benchmarks. This case study is based on research at the beginning of my PhD program. It has resulted in a published paper at EMNLP 2022 (Di Liello et al., 2022a).

In the third part (Section 3.3), we explore the fine-tuning of transformer models within the context of the question-answering domain. In particular, with this section, we show the adaptability of transformer models in facing complex, real-world problems and the increasingly important domain of multilingual NLP. Specifically, we present the Answer Sentence Selection (AS2) task, proposing innovative methods to enhance its performance in a multilingual setting. Leveraging state-of-the-art approaches in machine translation, we demonstrate how to adapt and improve QA systems to handle multiple languages effectively in two ways: (i) the automatic translation of existing datasets and (ii) a loss manipulation technique that can be used to reduce the biases associated with automatic translation. Like the previous case studies, this work is partially based on a research paper (Gabburo et al., 2024a).

3.1 Case study 1: The Transformer Architecture and the Code Domain

In this case study, we explore the transformative impact of the Transformer architecture on solving downstream tasks, mainly focusing on the automatic completion of programming code. We detail the stages of model development, starting from data collection, progressing through pretraining and finetuning, and culminating in the final evaluation on downstream tasks.

This case study is inspired by research we conducted in 2019, which aimed to pioneer state-of-the-art approaches for code completion using Transformer architectures. At the time, our proposed methodology was novel; no published work had employed Transformers for automatic code completion, with existing baselines relying primarily on Long Short-Term Memory (LSTM) networks and decision trees. However, despite its innovative nature, the paper was ultimately unpublished, and its relevance diminished following the release of the IntelliCode paper by Microsoft in 2020 (Svyatkovskiy et al., 2020), which introduced a similar approach. This setback highlights the rapid pace of advancement in the field and underscores the need for continuous innovation.

3.1.1 The Evolution of Code Completion and Transformer Applications in IDEs

Integrated Development Environments (IDEs) are essential tools for modern programming, significantly enhancing developer productivity and efficiency. Among their most valuable features is code completion, which helps programmers by predicting and autocompleting code as they write. Over the years, researchers have explored various methodologies to improve this capability (Svyatkovskiy et al., 2020; Li et al., 2018; Wang et al., 2021a).

Before the advent of Transformer architectures, code completion systems predominantly relied on Recurrent Neural Networks (RNNs) and probabilistic models (Raychev et al., 2016; Hellendoorn and Devanbu, 2017). These methods treated source code similarly to natural language. However, they struggled to capture long-term dependencies

between instructions due to the limitations of RNNs, such as the vanishing gradient problem (Bengio et al., 1994). In programming, this issue is particularly pronounced; for example, a variable declared at the start of a Python file may not be used until many lines later, requiring models to maintain context over extended sequences.

Transformer architectures have addressed these shortcomings by leveraging attention mechanisms to capture global dependencies across sequences. This has revolutionized tasks involving source code. For instance, Kim et al. (2020) employed a Transformer-based model for next-token prediction, while Svyatkovskiy et al. (2020) implemented GPT-based models for IntelliCode in Visual Studio Code. Additionally, CodeT5 (Wang et al., 2021b) introduced an encoder-decoder model pretrained on programming languages, offering versatility across multiple languages. Despite their effectiveness, these models are computationally expensive, raising concerns about resource consumption and environmental impact (Strubell et al., 2019).

3.1.2 Our Approach: Efficient and Versatile Transformer-Based Code Models

Building on these advancements, our study proposes a cost-effective and versatile training pipeline for Transformer-based language models tailored to programming languages. This approach is designed to overcome challenges such as high training costs and the need for task adaptability. Specifically, we introduce a two-stage training process:

- **Pretraining :** The model is pretrained on programming languages in two distinct settings:
  - *Mono-language:* Models are trained on separate languages independently.
  - *Multilanguage :* The model is simultaneously trained on multiple programming languages. This setting leverages the shared constructs and syntax across languages, enhancing generalization and performance.
- **Finetuning :** Pretrained models are specialized for specific tasks, such as next-token prediction or line completion, using transfer learning.

Our pipeline demonstrates that relatively small architectures can generalize effectively across multiple programming languages, benefiting from shared structures and constructs. This is in contrast to large-scale models like GPT-3 (Brown et al., 2020), which require significantly more resources to achieve similar results. Our results reveal two key findings:

- **Efficiency and Performance:** Our two-stage pretraining approach performs comparably to existing models, such as GPTCode (Svyatkovskiy et al., 2020), while requiring fewer computational resources.
- **Cross-Language Generalization:** By capturing commonalities across programming languages, our multilanguage models outperform monolingual systems in challenging inference tasks.

This case study underscores the potential of Transformer architectures in programming tasks. Beyond code completion, the methods and findings presented here establish a foundation for broader applications in programming language processing. They also provide a framework for resource-efficient model development, addressing critical concerns about scalability and environmental sustainability.

3.1.3 Tasks

The primary goal of this work is to develop a model capable of auto-completing programming code. When given a partially written file represented as a list of tokens, the model suggests a sequence of tokens to continue the input. Below, we explain in detail the three specific tasks targeted in this section: next-token prediction, next-word prediction, and line completion.

Next Token Prediction (NTP): In the next-token prediction task, the model is given a sequence of tokens:

$$S = \{t_0, t_1, \ldots, t_n\}, \tag{3.1}$$

where n is the current length of the input sequence. The model's goal is to predict the next token t_{n+1} that should follow this sequence. This task is fundamental as it forms the basis for predicting larger structures.

Next Word Prediction (NWP): Building on the next-token prediction, the next-word prediction task involves predicting the next word as a sequence of tokens. Given the same sequence S, the model predicts the next word W_k:

$$W_k = \{t_{n+1}, t_{n+2}, \ldots, t_m\}, \tag{3.2}$$

where the sequence ends when a new word starts or when the maximum input length N is reached. This task helps the model understand and generate meaningful code segments.

Line Completion (LC): The line completion task is more complex. Given the sequence S and a set of separator tokens Q (such as punctuation marks or line breaks), the model predicts the continuation of the current line:

$$L = \{t_{n+1}, t_{n+2}, \ldots, t_m\}, \tag{3.3}$$

where the sequence stops when a separator token $t_m \in Q$ is encountered or the maximum length N is reached. This task is crucial for completing statements and ensuring the syntactical correctness of the code.

By focusing on these tasks, our model can progressively learn from predicting individual tokens to completing entire lines of code. We designed these tasks in order to have the latter built on top of the previous one, allowing the model to develop a nuanced understanding of programming syntax and structure. Next-token prediction

forms the base, next-word prediction adds context and meaning, and line completion ensures the code's structural and syntactical integrity. These tasks together enable the model to provide accurate and contextually appropriate code completions, thereby assisting developers in writing code more efficiently.

3.1.4 Datasets for code

To train and evaluate our models, we retrieved and processed many code repositories from GitHub, which are composed of programs written in five different languages: Python, C, C++, Java, and JavaScript. From this collection, we sampled files to build two disjoint sets consisting of (i) six different large datasets devoted to pretraining (Section 3.1.4), and (ii) several smaller datasets to be pre-processed and used for fine-tuning and evaluation (Section 3.1.4). In particular, we sampled the training, validation, and test sets. We built the dataset, preserving the order of the input files (so that contiguous lines are part of the same original project).

Pretraining Corpora Each dataset is composed of a collection of different documents with the same programming language in common. We prepared each document by (i) applying tokenization, and (ii) replacing formatting tokens (e.g., tabs, multiple spaces, carriage return) and comments with unique tokens (e.g., *DCTAB*, *DCLN*, *DCCOM*). Then, we removed all the tokenized documents that (i) contain syntactic errors and (ii) segments that do not contain symbols of the language (for example, documentation files). Finally, we produced a multilanguage corpus joining the different mono-language corpora.

Evaluation Corpora To analyze the behavior and the performance of the models, we produced different evaluation corpora divided by tasks and languages. For each task, we prepared a finetuning dataset and a testset. Every example of these datasets is a pair (S_x, S_y), where S_x is left-context to be used as input for the model, while S_y is the expected sequence generated by the model.

To have a fine-grained understanding of the model's ability to generate new grammar constructs, we identified three sub-tasks for Next-Word Prediction:

- **Next Keyword Prediction (NKP):** S_y is a reserved keyword of the language (for example "if" or "while").
- **Next Name Prediction (NNP):** S_y is a name defined by the user (for example variable names).
- **Next Operator Prediction (NOP):** S_y is always a single token representing an operator (for example, "+").

With these sub-tasks, we can have a clear perspective on how our model behaves when it has to predict reserved language keywords (KW and OP) or new words introduced by the user (NAME). For the Line-Completion task, S_y is a sequence of variable size terminated by a line delimiter.

Vocabulary The creation of the vocabulary is a critical aspect as the model must address the vocabulary word problem (OOV) (Goldberg, 2017). Specifically, since the Transformer vocabulary contains only a limited number of words, we use a vocabulary composed of words and sub-words (Sennrich et al., 2016b). Thus, OOV words are decomposed into a list of tokens that are part of our vocabulary. This approach increases the learning step's complexity since the input's granularity is higher than using a standard vocabulary not explicitly built for the code domain (e.g., vocabularies for natural language) (Luong et al., 2015b).

We merge two different vocabularies: the first is small and contains all the keywords of the specific programming language, the reserved tokens used by the model, and the special tokens inserted during the dataset pre-processing (Hellendoorn and Devanbu, 2017). The second contains the list of the sub-token strings that we obtained by (i) scraping source code repositories from GitHub and (ii) applying the unsupervised approach described in Kudo (2018), to build the minimal vocabulary that covers all the possible words of our input files. Since the vocabulary size linearly depends on the model's memory during the training and inference phases, a good practice is to (i) keep it as small as possible and (ii) cover all input items from our target domain. Following this principle, we generated a vocabulary of 30k tokens.

To optimize the vocabulary for a multilanguage setting, we decided to merge the keyword dictionaries of the different languages. This is because many constructs are shared among different languages (e.g., "if" and "while").

3.1.5 Baseline Models

In this section, we describe the main baseline model, which we used to compare against our disjoint models.

LSTM

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) capable of learning long-term dependencies (Hochreiter and Schmidhuber, 1997a). They are particularly effective for sequential data, making them suitable for tasks like language modeling (Sundermeyer et al., 2012) and code completion (Hindle et al., 2012). LSTMs consist of cells keeping an internal state over time, with gates that control the flow of information (Gers et al., 2000). This architecture helps LSTMs overcome the vanishing gradient problem, which is common in traditional RNNs (Bengio et al., 1994). In the context of code completion, LSTM models can capture the syntax and structure of programming languages, providing predictions based on the sequence of tokens (Allamanis and Sutton, 2013).

GPT-2

Generative Pre-trained Transformer 2 (GPT-2) (Radford et al., 2019) is a large-scale transformer model that builds on the architecture introduced in GPT (Radford and Narasimhan, 2018) and discussed in Section 2.2.1. Indeed, different from models like

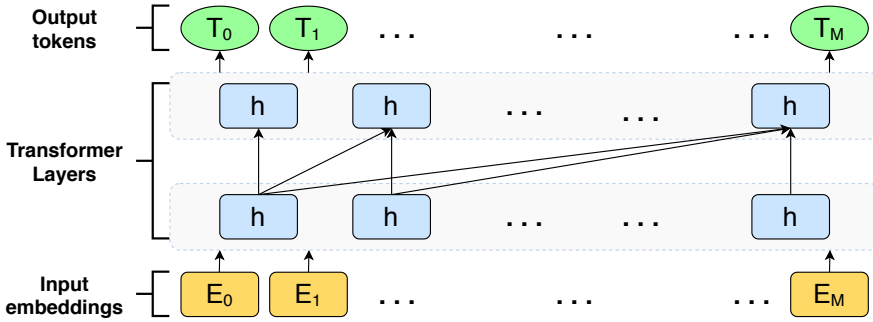

Figure 3.1: CLM Decoder architecture

BERT (Devlin et al., 2019), which uses the encoder part of the transformer, GPT-2 uses the decoder and is trained with a Causal Language Model (CLM) objective. This means it predicts the next token in a sequence, given the previous tokens, making it well-suited for autoregressive tasks such as text generation and code completion. GPT-2 was pre-trained on a vast corpus of natural language text, enabling it to generate coherent and contextually relevant text. When applied to the code domain, GPT-2's ability to understand and predict sequences can be leveraged to suggest code completions.

CodeGPT

CodeGPT (Svyatkovskiy et al., 2020) is an adaptation of the GPT architecture specifically tailored for code completion tasks. This model incorporates several modifications in order to better handle the unique characteristics of source code. In particular, their approach differs from the original GPT in two key points: First, they generated a new vocabulary using BPE (Sennrich et al., 2016a), and second, they pre-process their code-dataset, replacing comments and strings with special tokens. Additionally, they limited self-attention to enable autoregressive training from left to right.

3.1.6 Our Disjoint Encoder-Decoder Model

Our model is composed of two different transformer models in an encoder-decoder fashion. We consider these two models as separate entities, pre-training them separately on different tasks on the same dataset. The separate training of the two blocks of a sequence-to-sequence model was also studied by Rothe et al. (2020). After the separate pretraining, we join these two models into a single encoder-decoder architecture. Then, this joint architecture can be fine-tuned on specific tasks (see Sec. 3.1.7). Our training paradigm shows three advantages: First, the two models can be trained using different settings and early stopping criteria, optimizing the internal representations of the final model. Second, the bidirectional understanding of the input during the encoder pre-training improves the quality of the encoder. Third, since the two blocks are trained separately during the pretraining and then combined, we use less memory. This is very important as it also enables training on smaller machines (see embedded systems).

Encoder: Our encoder block (Fig. 3.2) is a standard Transformer encoder trained on two different tasks, which are (i) an adjusted version of the Masked Language Modeling (MLM) for programming languages, and (ii) Language Prediction (LP). Our MLM is different from (Svyatkovskiy et al., 2020), where strings and comments in the dataset are replaced with special tokens and masked as atomic constructs during training. Our MLM also differs from the original BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) approaches in two key aspects. First, our MLM uses Whole Word Masking (WWM), which consists of masking entire words and not separate tokens. This has been shown to improve the final model significantly. Second, and more importantly, we keep the string literals or comments in the input, but we do not allow MLM to mask and then predict them. We only enable MLM on their starting and ending tokens. This variation has a deep impact on the entire learning process since (i) the model, differently from (Svyatkovskiy et al., 2020), is used to see strings and comments values directly during the pretraining reducing the presence of special tokens, and (ii) these values can improve the semantic understanding of the model. This way, the latter can be easily finetuned on different tasks related to programming language, just using a few data (an example can be the code2comment task presented in (Miceli Barone and Sennrich, 2017)). The second task, LP, is useful when used in our multilanguage setting. Indeed, we added a second classification head to our model, which is used to predict the language of the current example. The resulting loss is computed as a summation of the losses originating from these two different tasks.

Decoder: The internal architecture and the learning task of the decoder block are the same as CodeGPT. The only difference is due to the self-attention component, which is substituted with cross-attention. The difference between these two blocks justifies our decision to combine them. Indeed, the encoder's role is to build a robust bidirectional representation of the input, while the decoder trained on CLM is only delegated to generate the output from left to right, starting from the knowledge of the input processed by the encoder. Differently from (Kim et al., 2020), we do not extract the positional information from the AST, but we use relative positional embeddings based on the CST (Concrete Syntax Tree). This choice is motivated firstly because a transformer model learns an optimal representation of the syntax after seeing a large number of examples. Secondly, in a multilanguage setting, extracting the AST from segments of programs is not always possible or practical since it requires a grammar parser. We did some preliminary experiments in this direction using a mono-language dataset without obtaining better results than our current approach, but we obtained faster convergence. We left a better exploration of these topics for future research. We designed the structure of our network to have two blocks of different sizes (number of parameters). We empirically derived that there is no increase in the model's final performance, using the same number of parameters for both blocks. For this reason, we decided to build a smaller model scaling the number of decoder layers to $\frac{1}{4}L_e$, where L_e is the number of layers used for the encoder.

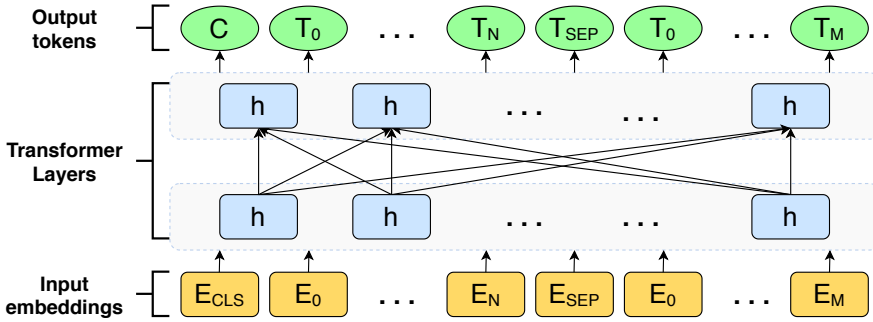

Figure 3.2: MLM Encoder architecture

3.1.7 Experiments

We logically divided the experiments into two separate settings: the first set of experiments verified two hypotheses: (i) the complexity of a language can impact the final training performance, and (ii) multilanguage models benefit from sharing common semantic structures from different languages. We prove these hypotheses by training different versions of our model. Specifically, we trained our disjoint architecture six times: five mono-language models (Python, C, C++, Java, and Javascript) and a multilanguage model considering the same number of training epochs. Then, we evaluate and compare the resulting pretrained models' with the Perplexity defined below.

Definition 1 *$Perplexity(D) = 2^{\frac{1}{N}\sum_{\forall x \in D}^{D} log_b q(x)}$, where D is the input sequence, and N is the number of labels.*

We carried out a second set of experiments on larger models using the multilanguage pretraining set. The latter is used to produce a robust model ready for finetuning models on the NWP and LC tasks. We compare our model with a CodeGPT, measuring the performance on the completion of tasks using the Perplexity (Def. 1), which is a well-known metric used to measure the performance of a language model and the normalized Levenshtein Similarity (Def. 2), which is a metric used to measure how far a generated string is from the expected one in terms of substitutions, additions and deletions of characters.

Definition 2 ***Normalized Levenshtein Similarity**: The Levenshtein similarity (LS) is defined as:*

$$LS = \frac{N - LD(S_1, S_2)}{N}, \tag{3.4}$$

where S_1 and S_2 are the two input sentences, N is the length of the longest of the two sequences S_1 and S_2, and LD is the Levenshtein Distance.

Pre-training

The pretraining part involves the encoder and the decoder blocks of Section 3.1.6. Specifically, we used the training techniques described in section 3.1.6. We trained the

Language	# Train examples	# Train instances
Python	9.6e+6	3.8e+4
C	9.2e+6	3.6e+4
C++	9.1e+6	3.6e+4
Javascript	9.4e+6	3.6e+4
Java	9.2e+6	3.6e+4
Multilanguage	4.3e+7	1.7e+5

Table 3.1: Pretraining sets for different languages

two disjointed models with the same datasets (described in Sec. 3.1.4) and for the same number of steps. In particular, we used a batch size of 256 with a starting learning rate of 1e-04 using Adam and a linear warmup of 10k steps, and then a weight decay of 0.01.

To fairly compare the MLM and CLM objectives, we considered the same architecture, which is a transformer composed of 12 layers, 768 of hidden size, 12 attention heads, and 3072 for the internal size (the same settings of the original BERT (Devlin et al., 2019)). During the pretraining, we trained our models for 1M steps with a sequence length of 128 and then for other 100k steps using a sequence length of 512. We selected the best checkpoints, minimizing the loss on the development set.

Datasets We used six different datasets to pre-train the transformer models: five languages (Python, C, C++, Java, and Javascript) and one multilanguage (all the languages together). To build these datasets, we follow the pre-processing steps described in Sec. 3.1.4.

Results The pretraining results summarized in Table 3.2 show that there is an evident difference in performance between MLM and CLM. We also note that mono-language models have different accuracy levels. Indeed, high-level languages such as Python, JS, and Java perform better than more verbose languages such as C and C++. We believe that the verbosity of the language itself can cause this disparity. The multilanguage approach shows that training with a congruous number of steps can improve the mono-language models in terms of Perplexity.

Regarding the comparison between mono-language and multilanguage models, we note that the latter reaches better performance with both MLM and CLM pretraining objectives. This shows that the quality of the internal model representation can effectively benefit from shared constructs and structures of different languages.

Next Token Prediction

For the Next Token Prediction task (NTP), we compare our models with existing models trained on the same datasets. For this comparison, we consider three different baselines: (i) an LSTM model that works on token level (using a BPE vocabulary), (ii) a GPT-2 pretrained on natural language and then finetuned on programming languages, and (iii) CodeGPT. For the latter, we considered the 12-layer version (124M parameters), which

Language	**MLM** (PPL)	**CLM** (PPL)
Python	1.21	2.26
C	1.26	3.02
C++	1.26	3.14
Java	1.18	2.10
JS	1.22	2.28
Multi	1.02	1.91

Table 3.2: The perplexities of the trained models using different datasets and the same amount of training steps. These results describe the complexity of the different languages and compare the multilanguage and monolanguage settings.

	Number of tokens	
Split	**Py150**	**JavaCorpus**
Train	72.1M	15.7M
Dev	4.4M	3.8M
Test	37.3M	5.3M

Table 3.3: Dataset statistics for Py150 and JavaCorpus

was distilled from a larger teacher (366M parameters). The results from these three models were computed using the code from the CodeXGlue framework (Lu et al., 2021). We trained our seq2seq disjoint model using a batch size of 64, and a learning rate of $1e-05$ for 5 epochs. We compared our results with existing and benchmarked models, i.e., LSTM and CodeGPT (Svyatkovskiy et al., 2020).

Datasets We considered two existing and well-known datasets for code-completion tasks: Py150 (Bielik et al., 2016) and JavaCorpus (Allamanis and Sutton, 2013), described in Tab. 3.3.

Results Table 3.4 reports the accuracy of the different models for the NTP task. The results confirm that the transformer models have improved significantly over previous RNN-based models. In particular, our disjoint seq2seq model is comparable with CodeGPT on the two considered datasets and better than GPT2 pretrained on natural language and then finetuned on programming languages. The fact that CodeGPT was distilled using a teacher of 366M parameters should be stressed. This cannot be done with scarce resources. In contrast, our model requires a memory of only 109M parameters and 1.1M steps of pretraining, using a machine of only 166M param. for refining the entire encoder-decoder architecture.

Model	Accuracy (%) Py150	JavaCorpus
LSTM + BPE	61.94	58.92
GPT-2 (124M)	75.90	75.40
CodeGPT (124M)	76.58	76.79
Disjoint Model (M) (166M)	76.82	75.94

Table 3.4: Results on NTP finetuning task on two public datasets (javaCorpus and py150). The CodeGPT model was distilled from a larger model (366M parameters).

	Number of words				
Split	**py**	**C**	**C++**	**Java**	**JS**
Train	59.1M	59.8M	63.3M	53.6M	59.1M
Dev	7.3M	7.6M	7.0M	6.5M	7.9M
Test	7.4M	7.7M	7.0M	6.5M	8.0M

Table 3.5: Dataset statistics for our finetuning corpora

Next Word Prediction

We finetuned the two models (the baseline and our disjoint seq2seq model) on the NWP corpora. In particular, we started from an existing checkpoint for the two versions of GPTCode (one for Python and one for Java) and from our best-pretrained model (see Table 3.2) for our disjoint model. Specifically, regarding the considered parameters, we used a batch size of 32 and $1e-05$ as the starting learning rate using Adam as Optimizer, and for 5 epochs, all the models. In this case (and differently from the LC task of Sec. 3.1.7), the models are trained to generate the next word. We divided the task into three different subtasks to understand and gain a fine-grained perspective on how the models learn the different programming languages' constructs. In particular, these subtasks are (i) keyword prediction (KW), (ii) operator prediction (OP), and (iii) the prediction of a name defined by the user (NAME). We used a decoding beam search of 5.

Datasets The datasets used for finetuning and evaluating the models are described in Sec. 3.1.4. We prepared a dataset for each task on every language, separating finetuning, dev. and test sets with a ratio of 80%-10%-10%, for a total of 100k training examples and 12.5k examples for both validation and test examples for each sub-task. We summarize these datasets in Table 3.5.

Results The results in terms of the Edit Similarity (Def. 2) for NWP are summarized in Table 3.6 and presented by sub-tasks. We note that our disjoint model shows comparable performance to the two versions of CodeGPT, both for Python and Java. However, our model shows to be easily adapted to different programming lan-

Model	Language	Edit Similarity (%) KW	OP	NAME
CodeGPT (py)	Python	77.2	72.2	49.3
CodeGPT (java)	Java	77.9	73.1	45.1
Disjoint Model (M)	Python	77.4	73.1	47.5
	C	74.7	65.8	39.2
	C++	73.1	69.2	37.4
	Java	75.9	73.1	44.1
	JS	77.4	71.0	40.1

Table 3.6: Results of the Next Word Prediction finetuning task divided in three subtasks (KW, OP, NAME) on our multilanguage-evaluation dataset.

guages without incurring to an evident drop in performance: we only need to perform an "easy" finetuning step. For example, analyzing the class of the keywords (KW) for GPTCode-py and our disjoint model, we observe a slight improvement in terms of Edit Similarity, i.e., 77.4 vs. 77.2. Another finding is related to the reduced difference in performance for the different languages. Indeed, we notice that thanks to the pretraining on different languages, the model acquired a strong capacity for handling different languages, reinforcing the preliminary analysis on the perplexities done in Section 3.1.7. Furthermore, these results show how the model behaves on different language constructs, and they verify the hypothesis that large language models quickly learn the syntax of programming languages.

Line Completion

Similarly to Sec. 3.1.7, we finetuned the baseline and our disjoint seq2seq model on the LC finetuning. This task is directly comparable with the task solved in (Svyatkovskiy et al., 2020). Indeed as explained in Section 3.1.3, in this task, the model is finetuned to predict a line until a terminal token is found (for example, a semicolon in C). For this task, we consider both the evaluation corpora described in the previous sections.

The setup for this experiment on the two models (CodeGPT and our model) is slightly different from the NWP setup (Sec. 3.1.7). We trained the models for 5 epochs on the entire finetuning dataset, with a learning rate of $5e-5$ and 64 as the batch size on a single A100.

Datasets For this task, we considered two different corpora: the two well-known datasets (Py150 and JavaCorpus) and our evaluation dataset described in sections 3.1.7 and 3.1.7.

Results The results on Py150 and JavaCorpus, described in Table 3.7, show that the joint seq2seq model obtains accuracy consistently better than the LSTM baseline, demonstrating the superiority of the transformer models over RNN-base models. In

Model	Edit similarity (%) Py150	JavaCorpus
LSTM + BPE	61.94	58.92
GPT-2 (124M)	70.60	60.36
CodeGPT (124M)	71.23	61.81
Disjoint Model (M) (166M)	71.26	61.02

Table 3.7: Results for the Line Completion task on two public datasets (javaCorpus and py150).

Model	Edit Similarity (%) py	Java
CodeGPT (py)	70.2	–
CodeGPT (java)	–	64.5
Disjoint Model (M)	70.8	63.3

Table 3.8: Results for the Line Completion task on our evaluation corpora against two monolanguage versions of CodeGPT (Python and Java). The models marked with a **M** are trained on our multilanguage dataset.

addition, we have achieved considerable improvement on the GPT -2 model, which is pretrained on natural language. This last result is interesting because it confirms the hypothesis that a dedicated vocabulary and code-domain pretraining help the model reach better accuracy. Lastly, we notice that our model is on par with CodeGPT (71.26 vs 71.23 on Python and 61.02 vs 61.81 on Java).

The results on 3.8 show a similar trend. Interestingly, our multilanguage model (Disjoint Model M) can match the performance of two existing transformer models trained in a monolanguage setting (CodeGPT). This confirms the hypothesis that some programming languages are more challenging to learn than others, even with the same experimental setting.

Results on Memory Usage

As mentioned in the previous sections, one of the main advantages of our disjoint training is the usage of the smallest amount of memory during the training. To show this point, we considered different models and measured the space used by each of them during inference and training. Specifically, we analyzed (i) the only decoder baseline and (ii) our disjoint encoder-decoder, taking into consideration different configurations of parameters. For each model, we monitored the VRAM usage, and we reported the maximum consumption in Table 3.9.

As expected, our model uses less memory during training than the only-decoder baseline (based on the gpt2 architecture), allowing bigger models to be trained on less powerful, non-specialized machines. In particular, we note that our largest model shows

Model	#Parameters	Memory	
		Inference	Training
gpt2 base	124M	1.32GB	3.49GB
gpt2 medium	354M	2.16GB	7.04GB
gpt2 large	774M	4.44GB	14.14GB
gpt2 xl	1.56B	10.05GB	26.32GB
Our base	166M	1.44GB	3.06GB
Our medium	516M	3.15GB	6.74GB
Our large	1.16B	6.63GB	13.87GB
Our xl	2.35B	15.08GB	26.07GB

Table 3.9: Memory usage on models of different sizes, considering a batch size of 1 and the most expensive time both during inference and training. Our models use less memory during the training while still having more parameters (e.g., 2.35B vs 1.56B for the xl model)

a significantly higher number of parameters than the most significant baseline (1.56B vs 2.35B) but uses a comparable amount of memory (26.07GB vs 26.32GB).

Final discussion

In this study, we provided a comprehensive overview of the transformer architecture, discussing its application across diverse tasks and domains. Indeed, starting from scratch, we covered all the necessary steps to cover all the aspects of training a transformer model, from the data collection to the final training on a downstream task. Specifically, we examined (i) methods for training transformer-based sequence-to-sequence (seq2seq) models for programming code autocompletion tasks and (ii) the impact of cross-language pretraining on relatively small architectures. We developed datasets for pretraining, finetuning, and testing our models. Additionally, we compared our models against other transformer-based models from previous work on two well-known datasets for code completions. Our results indicate that our disjointed seq2seq models perform similarly to the original CodeGPT model, offering enhanced flexibility due to their multilingual pretraining setting. These models require less memory for pretraining since our encoder and decoder are trained separately before being combined. The encoder is trained using Masked Language Modeling (MLM), which does not require a decoding strategy, thereby reducing computational power for each step. This training paradigm enables the training of larger architectures on non-specialized and cost-effective hardware, utilizing less memory. Moreover, we demonstrated that it is feasible to train cross-lingual models on relatively small architectures, which benefit from shared syntax and constructs across different languages. The comparative analysis of different languages revealed varying degrees of learning difficulty, with Python achieving significantly better performance during pretraining and finetuning tasks. At the same time, models showed the lowest performance with C++. We hypothesize that the verbosity

of the language may contribute to this performance disparity. Our cross-lingual models exhibit notable modularity. They require a single training session across all languages, which can then be applied to multiple programming languages for various tasks.

3.2 Case study 2: Efficient pretraining techniques for transformer models

3.2.1 Efficient Pretraining for Transformer Encoders

In this second case study, we investigate pretraining strategies for Transformer-based encoders, with a focus on reducing their time and resource consumption while maintaining competitive accuracy. The objective is to enhance the efficiency of pretraining, a crucial step in developing scalable and accessible NLP models. This exploration delves into the broader field of unsupervised training objectives, which represents one of the main pillars of this Thesis.

This case study is based on our paper, *"Effective Pretraining Objectives for Transformer-based Autoencoders"*, published at EMNLP 2022 (Di Liello et al., 2022a). Through this work, we propose novel alternatives to existing pretraining methodologies, aiming to strike an optimal balance between efficiency and performance. Our contributions advance the understanding of how unsupervised objectives impact the cost-effectiveness and accuracy of pretrained models.

3.2.2 Reducing Pretraining Costs in Transformer-based Models

Transformer-based models (Vaswani et al., 2017) have demonstrated exceptional capabilities in numerous NLP tasks but require substantial computational resources for pretraining (Strubell et al., 2019; Brown et al., 2020). This has motivated extensive research into reducing pretraining costs (Lan et al., 2020; Sanh et al., 2019; Turc et al., 2019). For example, ELECTRA introduced a novel approach by training BERT as a discriminator instead of a generator, replacing the Masked Language Modeling (MLM) objective (Devlin et al., 2019) with Token Detection (TD) (Clark et al., 2020). While this method improved efficiency, it required pretraining a secondary generator, increasing computational cost and complexity.

To address these limitations, we propose two new pretraining methods: Random Token Substitution (RTS) and Cluster-based Random Token Substitution (C-RTS). These approaches aim to balance the trade-off between efficiency and accuracy more effectively than existing alternatives.

Proposed Approaches RTS and C-RTS offer efficient alternatives to ELECTRA's generator by introducing simplified token substitution strategies:

- **RTS:** This method involves detecting tokens randomly replaced with others, resulting in a low-cost pretraining objective. RTS significantly reduces computational overhead by employing a lightweight binary classification head.

- **C-RTS:** A more challenging variation of RTS, this method utilizes predictions from previous iterations to select more informative token replacements. While slightly more expensive than RTS, C-RTS achieves better performance on complex tasks when trained for extended durations.

Both RTS and C-RTS achieve efficiency gains of 20%–45% by employing smaller classification heads compared to MLM. Despite their efficiency, these methods maintain comparable accuracy to MLM across various tasks, demonstrating their suitability for resource-constrained environments.

Additional Contributions We also introduce Swapped Language Modeling (SLM), a variant of BERT's MLM objective that eliminates the need for the MASK token, addressing the pretraining and fine-tuning mismatch highlighted by Clark et al. (2020). While SLM incurs higher costs than RTS and C-RTS, it consistently outperforms MLM under the same computational budget, making it a viable alternative for high-performance applications.

Empirical Validation To evaluate these approaches, we pretrain standard BERT models using RTS, C-RTS, and SLM and compare them against state-of-the-art architectures. Our analysis spans multiple natural language understanding benchmarks, including GLUE, ASNQ, WikiQA, and TREC-QA. We report metrics such as accuracy, efficiency, and computational cost, highlighting the trade-offs inherent in each approach. Additionally, we examine the impact of our objectives on smaller architectures (e.g., BERT-*small*), demonstrating their broader applicability.

Key Results

- RTS and C-RTS reduce computational cost while maintaining accuracy comparable to MLM. C-RTS shows superior performance with extended training, leveraging more challenging substitutions.
- SLM addresses the pretraining-fine-tuning discrepancy and achieves higher accuracy than MLM across most tasks, given an equivalent budget.
- Our methods provide significant efficiency gains (20%–45%) and are particularly impactful on smaller-scale architectures, broadening their applicability to resource-constrained settings.

By proposing and empirically validating these methods, we contribute to the development of efficient pretraining strategies that reduce the computational barriers associated with Transformer-based models while maintaining state-of-the-art performance.

3.2.3 Effective Pre-training Objectives

This section presents our alternative pre-training objectives, which can potentially be applied to a wide range of Transformer-based models. We focus on efficient techniques to replace expensive solutions like ELECTRA's generator or MLM.

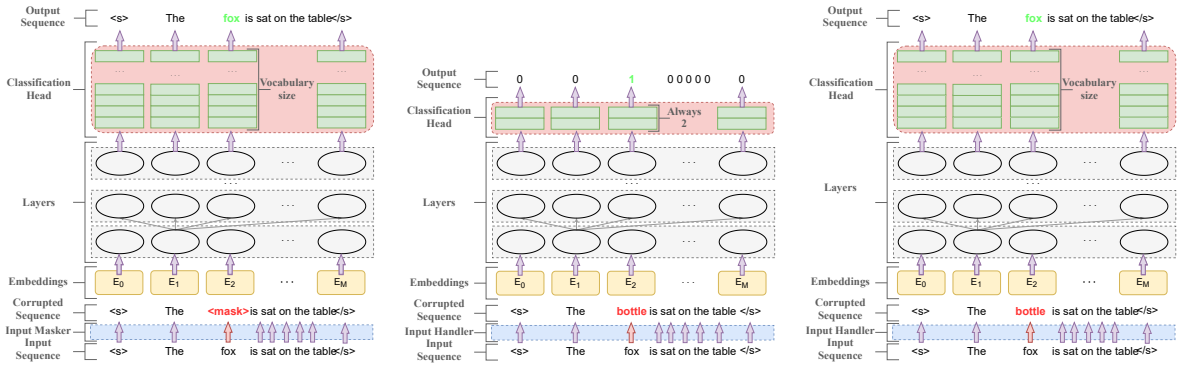

Figure 3.3: Architectures for MLM, RTS, and SLM (left to right). Notice that the classification head used by RTS is significantly smaller than those used by MLM and SLM.

Random Token Substitution (RTS) Like ELECTRA, RTS trains a model that discriminates between original and substituted tokens. The main difference is that RTS replaces 15% of the tokens with random alternatives, thus avoiding using computational resources to train a separate and expensive network. Besides, unlike MLM, this approach relies on a smaller classification head that is *not* proportional to the vocabulary size, see Figure 3.3. Indeed, to pretrain a transformer model with MLM, the model needs a large a large classification head. Since, its dimension is proportional to the vocabulary size, and (especially for *small* models) this constitutes a significant fraction of the entire architecture parameters. For example, in the *base* architectures, the MLM head constitutes about 20% of the model parameters while for *small* models, the fraction increases to 30%[1]. The memory footprint of the LM head while training is about 47% for *base* and 64% for *small* models[2].

Aggregated probabilities of token misclassification (C-RTS) The random selection applied by RTS may provide too many *easy cases* to perform effective pretraining. Thus, our idea is to use a token probability distribution, inversely proportional to the classification simplicity. More formally, let $(w_0, w_1, \ldots, w_n)$ be an input sequence of tokens. A transformer model m predicts $y = \{0, 1\}^n$, where $y_i = 0$ indicates w_i is original, and $y_i = 1$ that w_i was replaced with some w'_i ($w_i \rightarrow w'_i$). We aim at maximising $P(y_i = 0 \mid w_i \rightarrow w'_i)$, since we want to create substitutions $w_i \rightarrow w'_i$ that are difficult to be detected by m.

$P(y_i = 0 \mid w_i \rightarrow w'_i)$ can be estimated by counting the number of failures/successes in detecting $w_i \rightarrow w'_i$ in previous iterations. While ELECTRA exploits the whole input context to create challenging replacements, our algorithm uses only the prediction history over single tokens. Storing a counter for every pair or token would generate a vast and sparse matrix given the average transformer's vocabulary size. For this reason,

[1]Very often the language modelling head parameters are shared with the embedding layer

[2]Gradients have to be computed for every token in the vocabulary because of the final softmax layer

we partition tokens into n clusters by measuring the L2 norm between the relative embedding vectors. We train the word embeddings with a *word2vec* model (Mikolov et al., 2013) on the same data used for pre-training; First, token embeddings have been obtained by training a word2vec (Mikolov et al., 2013) model over the same data used in pre-training. We also tried to use embeddings from an already pre-trained BERT model, but we saw no significant difference in TD accuracy. Therefore, we decided to use word2vec to provide a complete pre-training pipeline and not rely on an already pre-trained checkpoint. We used a context of 2 words on either side of the target tokens and an embedding size of 300. The training algorithm written took less than 10 minutes on the same GPU used for pre-training. Thus this process is not significant concerning the whole pre-training time. We perform clustering of tokens using the K-means algorithm (Lloyd, 1982). We used the CPU implementation of K-means provided by the *scikit-learn* library, doing 20 random starts. The clustering took approximately 12 minutes on the Intel Xeon Platinum 8275CL in our machine. After that, we use *K-Means* (Lloyd, 1982) to group the tokens into the clusters $C_1, \ldots, C_n$.

We use a matrix $\boldsymbol{F} \in \mathbb{Z}^{n \times n}$, initialized with zeros, to count the difference between the failures and successes of the discriminator. While training, for each pair of tokens $(w_i \in C_a, w'_i \in C_b)$ such that $w_i \to w'_i$, we decrease $\boldsymbol{F}_{a,b}$ by 1 if $y_i = 0$, otherwise we increment it by 1.

To maximise $P(y_i = 0 \mid w_i \to w'_i)$ in our approach, we select the pairs (w_i, w'_i) with the highest mistake probability estimated with $\boldsymbol{F}$. We compute the probability of selecting $w_i \in C_a$ and replacing it with $w'_i \in C_b$ as follows:

$$P(w_i \in C_a \to w'_i \in C_b) = P(w_i)\ P(w'_i|C_b)\ P(C_b|C_a) \tag{3.5}$$

where we set (i) the probability of selecting a token in the input sequence $P(w_i)$ to 15%; (ii) $P(w'_i|C_b)$ equal to $\frac{1}{|C_b|}$ because it is computed assuming uniform probability of choosing w'_i in C_b; and (iii) we set $P(C_b|C_a)$ as the probability of detecting tokens from cluster C_a when replaced with tokens from C_b. The latter is computed from $\boldsymbol{F}$ as follows: given the candidate token $w_i \in C_a$, we define a multinomial distribution over the target clusters by indexing the a-th row of $\boldsymbol{F}$. To transform the counts in $\boldsymbol{F}_a$ into values interpretable as probabilities, we first apply the *min-max* normalization and then a γ-softmax. γ controls how the probability mass is concentrated or relaxed around the most probable cluster. For our token substitutions, we sample C_b from this multinomial distribution.

We searched for the best n among a reasonable set $\{30, 100, 300, 1000\}$ and the best γ in $\{1, 2, 5, 10\}$. After preliminary experiments, we found that the best combination is $n = 100, \gamma = 2$.

Swapped Language Modeling (SLM) is similar to BERT's MLM, but in this case, tokens are only randomly replaced with others and never with the special MASK. Then, unlike RTS, the model is trained to predict the original value and not discriminate between fakes and originals. SLM uses the same pre-training head as MLM; thus, it is computationally equivalent to it.

3.2.4 Experimental Setting

In these experiments, we compare the cost and accuracy of our models with the state-of-the-art methods on GLUE and several AS2 benchmarks. Finally, we summarize results derived from previous work. Then, we provide a description of the training set as well as our pre-training and fine-tuning experiments and results.

To perform these experiments, we applied every objective to the same BERT (Devlin et al., 2019) architecture to make a fair comparison. Furthermore, since pre-training time is not proportional to the number of parameters but to the number of computations, we also measure the floating point operations required to do pre-training, as in (Clark et al., 2020). FLOPS indicates the number of mathematical operations performed on the underlying hardware and are independent of the used accelerator (CPU, GPU or TPU) and the model size.

In detail, we considered two models in two different magnitudes of a number of parameters that we will call **base** and **small** models. In detail:

Base models: The Base architecture is the same as BERT_{Base}, featuring a vocabulary with 30,522 WordPiece tokens, a hidden size of 768, an intermediate size of 3072, 512 positional embeddings, 12 attention heads, and 12 layers.

Small models: The Small architecture we use is similar to ELECTRA_{Small} (Clark et al., 2020), which previous experiments have shown to outperform BERT_{Small} (Turc et al., 2019). ELECTRA highlighted that reducing the Transformer's width (hidden size) is more effective than reducing its height (number of layers). Our Small architecture employs a vocabulary of 30,522 WordPiece tokens, a hidden size of 256, an intermediate size of 1024, 512 positional embeddings, 12 attention layers, and 4 attention heads.

To test our techniques, we applied the training objectives to the same BERT (Devlin et al., 2019) architecture. To make a fair comparison, we group the experiments according to the number of parameters of the architecture. Furthermore, since pre-training time is not proportional to the number of parameters but to the number of computations, we also measure the floating point operations required to do pre-training, as in (Clark et al., 2020). FLOPS indicates the number of mathematical operations performed on the underlying hardware and are independent of the used accelerator (CPU, GPU or TPU) and the model size.

Pretraining

We pre-trained a model for every alternative objective in the same setting as BERT-*base* (Devlin et al., 2019) to perform a fair comparison. More precisely, we pre-trained models on the English Wikipedia and the BookCorpus dataset for 900K steps with a maximum sequence length of 128 tokens and another 100K steps at 512 with an uncased vocabulary. This saves much pre-training time because the computational complexity of the attention is quadratic in the sequence length. We use Adam and a triangular learning rate for optimization with a peak value of 10^{-4} and 10K warm-up steps. We

use a batch size of 256 examples. In detail, we pre-trained on the cleaned versions of the BookCorpus and the English Wikipedia. For the optimization of both the *small* and the *base* models, we use Adam with a learning rate equal to 10^{-4}, $\epsilon = 10^{-8}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate scheduler is designed to warm up for 10K steps and then decrease linearly. We use a batch size of 256 examples for the *base* models and 1024 for the smalls. Finally, we apply a constant weight decay rate of 0.01, and the dropout probability is set to 0.1. Since ELECTRA models require more FLOPS because of the generator, we reduce the number of steps proportionally to the presence of the additional generator, as in (Clark et al., 2020). Thus, we pre-trained ELECTRA for a total of 766K steps, of which 689K with a maximum sequence length of 128 tokens and the remaining 77K at 512.

For the smaller models, we pre-trained 4 models using the *small* architecture defined by (Clark et al., 2020) and MLM, SLM, RTS and C-RTS as objectives. We pre-train small models with the same data and hyper-parameters for the *base* models but using a larger batch size of 1024 for 500K steps and always using a reduced maximum sequence length of 128 tokens.

Finetuning

To prove the quality of our new objectives, we finetune the pretrained models trained with our objectives presented in Section 3.2.3 (RTS, C-RTS, SLM) and the baseline trained on MLM and TD (Section 2.3) on several benchmarks. In detail, we finetune those models on GLUE (Wang et al., 2018), SQUAD (Rajpurkar et al., 2016a), ASNQ (using the setting described in (Soldaini and Moschitti, 2020)) and ASNQ $\rightarrow$ WikiQA as in (Garg et al., 2019). The arrow means that the model was transferred on ASNQ and then fine-tuned and tested on WikiQA.

GLUE We use the same hyper-parameters used in (Liu et al., 2019). Specifically, we used a batch size of 16 with $lr = 1 \times 10^{-5}$ for CoLA and MRPC, a batch size of 16 with a learning rate of 2×10^{-5} for RTE and STS-B, and a batch size of 32 and a learning rate of 1×10^{-5} for MNLI, QNLI, QQP and SST-2. We set the maximum sequence length to 128 for every task. All the GLUE experiments use a triangular learning rate scheduler with 10% of warmup. We train models using half-precision and optimize them on the development set. We measure Spearman and Matthews's correlation coefficients for STS-B and CoLA respectively, and accuracy for all the other tasks. For every model, we take the best checkpoint on the development set and evaluate it on the GLUE Leaderboard.

ASNQ We train our models on ASNQ using a batch size of 2048 with a learning rate of 1×10^{-5} for 12 epochs with early stopping on the development set. Since most question-answer pairs are shorter than 128 tokens, we use this as the maximum sequence length. We measure the performance using Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR).

WikiQA & ASNQ → WikiQA Following the approach of TANDA (Garg et al., 2019), we fine-tune our models directly on WikiQA and again on WikiQA but after a transfer step on ASNQ. In particular, for each model, we run a hyperparameter search to obtain the best results. We search for the best batch-sizes in $\{32, 64, 128\}$, the best learning rates in $\{2 \times 10^{-6}, 5 \times 10^{-6}, 1 \times 10^{-5}, 2 \times 10^{-5}\}$, and we always use a maximum sequence length of 128 for up to 40 epochs. We repeat each experiment 10 times with different seeds to also report the results' standard deviation. We evaluate the model's performance on the test set using MAP and MRR.

TREC-QA & ASNQ → TREC-QA For these 2 tasks, we adopted the same strategy used for WikiQA and ASNQ → WikiQA. We repeat each experiment 10 times with different seeds; in this case, we evaluate with the same metrics above.

3.2.5 Results

We first report the cost of our models in comparison with the state-of-the-art methods in Tables 3.12 and 3.13 for *base* and *small* models, respectively. Notice also that RTS and C-RTS use about half the GPU memory of MLM with *base* and $\frac{1}{3}$ with *small* models. Thus, the training time of the former's to complete all the training steps could have been even shorter by increasing the batch size. However, we keep a constant batch size to make a fair comparison. After that, we carry out a comparison in terms of accuracy on GLUE datasets as well as question-answering benchmarks. Additionally, we compare our models with state-of-the-art results published in previous works, pointing out the cost of training them.

Model	CoLA matt. corr.	MNLI acc	MRPC acc	QNLI acc	QQP acc	RTE acc	SST-2 acc	STS-B spear	AVG %	Time
BERT-B / MLM+NSP ♣	53.0	82.7	82.8	89.6	88.2	62.8	91.2	80.6	78.9	× 1.00
BERT-B / MLM	53.7	83.6	82.6	89.9	89.1	63.1	92.3	83.6	79.7	× 1.00
BERT-B / RTS	57.3	82.6	81.3	89.3	88.9	66.2	91.7	82.2	79.9	× 0.81
BERT-B / C-RTS	57.3	82.8	81.9	89.4	88.6	60.9	91.0	82.2	79.3	× 0.82
BERT-B / SLM	57.0	83.3	83.1	89.7	88.9	65.0	92.3	83.8	80.4	× 1.00
ELECTRA-B / TD	60.6	83.6	84.1	90.2	89.0	69.1	92.4	85.5	81.8	× 1.05

Table 3.10: Results on the GLUE test set. We took the best models on the development set and submitted them to the GLUE leaderboard. Also, in this case, we don't do best model selection from a pool of candidates, and we do not use ensemble models.

Base models

GLUE Table 3.10 shows that all the considered approaches, except for ELECTRA vanilla (MLM+TD), obtain comparable performance on GLUE. In particular, despite a small performance decrease compared with our MLM BERT, the RTS and C-RTS models require about 20% less time to be pre-trained on the same machine. This suggests that using a smaller classification head greatly impacts the training time. The difference could be even broader when considering models using larger vocabularies such

Model	WikiQA		TREC-QA		ASNQ → WikiQA		ASNQ → TREC-QA		ASNQ		Time
	MAP	MRR	MAP	MRR	MAP	MRR	MAP	MRR	MAP	MRR	
BERT-B / MLM+NSP ♣	82.8 (0.9)	84.2 (1.0)	87.2 (0.9)	92.9 (1.0)	87.9 (0.4)	89.3 (0.4)	89.1 (0.4)	93.4 (0.5)	66.8 (0.1)	73.2 (0.2)	× 1.00
BERT-B / MLM	79.9 (1.1)	81.2 (1.2)	86.4 (0.7)	91.5 (0.7)	88.9 (1.0)	90.2 (1.0)	87.6 (0.9)	90.6 (1.1)	65.5 (0.2)	72.2 (0.3)	× 1.00
BERT-B / RTS	78.5 (2.5)	80.1 (2.5)	86.6 (1.5)	91.8 (1.5)	87.9 (0.5)	89.3 (0.5)	88.5 (0.5)	93.4 (0.4)	64.6 (0.1)	71.1 (0.2)	× 0.81
BERT-B / C-RTS	79.0 (1.9)	80.6 (1.6)	87.0 (0.8)	91.8 (1.1)	87.1 (0.6)	88.3 (0.6)	88.7 (0.4)	92.9 (0.7)	64.7 (0.1)	71.5 (0.1)	× 0.82
BERT-B / SLM	80.2 (1.5)	81.7 (1.6)	87.3 (1.3)	92.1 (1.5)	87.7 (0.8)	89.3 (0.7)	87.9 (0.6)	91.0 (0.6)	65.8 (0.3)	72.7 (0.3)	× 1.00
ELECTRA-B / TD	81.8 (1.6)	83.2 (1.6)	86.8 (1.4)	92.2 (1.5)	88.4 (0.4)	89.8 (0.4)	88.9 (0.3)	92.0 (0.5)	64.9 (0.3)	71.7 (0.4)	× 1.05

Table 3.11: Results on WikiQA and TREC-QA datasets with single-task fine-tuning and after the transfer step on ASNQ. We also report the results on the dev. set of ASNQ optimizing MAP. The NSP loss of the original BERT mainly improves the results without the transfer step. After the transfer on ASNQ, which trains especially the CLS token representation, the difference with our MLM-based BERT is much smaller. We show the standard deviation after 5 runs with different initialization seeds in rounded brackets. We underline results that are significantly different from the BERT-B / MLM baseline after a two-sided T-Test with a significance level equal to 95%.

as RoBERTa (Liu et al., 2019) or models with a smaller number of Transformer layers, such as DistilBERT (Sanh et al., 2019) or tiny-BERT (Turc et al., 2019).

It is also worthwhile mentioning that: (i) our BERT-SLM model achieves better performance than MLM BERT (+0.7%), and the original BERT (+1.5%) on average, confirming that removing the MASK token during pre-training is important; (ii) ELECTRA provides superior performance when fine-tuned on small datasets such as CoLA, while in other tasks it shows similar accuracy to SLM and MLM.

QA benchmarks We report the performance obtained by the models on a wide range of QA tasks in Table 3.11. The results on the dev. set of ASNQ are obtained by fine-tuning all models in the same setting. SLM provides the highest results among all models using token-level objectives, scoring even higher than ELECTRA.

On WikiQA and TREC-QA, RTS and C-RTS have mixed results compared to MLM. In some QA tasks, RTS performs slightly better than MLM, while, in others, slightly lower. To show that there is no statistical significant difference in performance between the two models, we do the following test: we split the test set of TREC-QA and WikiQA in 10 parts, and we test the performance of both models on all mini-batches. Finally, we take the difference in performance between the two models on every mini-batch, and we report the mean and standard deviation. We discover that MLM is better than RTS on ASNQ → WikiQA by 1.4 (±3.8) points while, similarly, RTS outperforms MLM on ASNQ → TREC-QA by 1.1 (±2.9). The high std. dev. makes the small difference between models insignificant. In contrast, RTS requires 20% less time to be fully pre-trained.

C-RTS shows minor but consistent improvements over RTS in most QA tasks, especially when trained longer, also featuring a much smaller std. dev. when fine-tuned without the transfer step on ASNQ, which generally reduces the variability of results (Garg et al., 2019). Moreover, similarly to GLUE, the model trained with SLM obtains small but consistent advantages over MLM in MAP and MRR on three benchmarks out of four.

We stress the fact that the only other differences between our BERT models and the original by Devlin et al. (2019) are the BookCorpus pre-training dataset and the additional NSP loss. We use only MLM to provide a fair comparison between token-level objectives. NSP may improve results slightly because it improves the sentence-level representation already while pre-training. This is especially true when models are not trained as long as RoBERTa or ELECTRA, which dropped NSP because it became insignificant for them, and may even hurt performance after many training steps.

Models	BERT-B / MLM	BERT-B / RTS	BERT-B / C-RTS	BERT-B / SLM	ELECTRA-B / TD
LM head complexity	$O(bs \times L \times \lvert V \rvert)$	$O(bs \times L)$	$O(bs \times L)$	$O(bs \times L \times \lvert V \rvert)$	$O(bs \times L \times \lvert V \rvert)$
FLOPS	1.61×10^{19}	1.54×10^{19}	1.54×10^{19}	1.61×10^{19}	1.98×10^{19}
Training time	3d 14h	2d 22h	2d 23h	3d 14h	3d 18h

Table 3.12: FLOPS used to pre-train each *base* model, language modelling head complexity and real training time on the same machine. *bs* stands for batch size, L for the input sequence length and $|V|$ is the vocabulary size, equal to about 30K tokens in BERT models. Notice that BERT-RTS and BERT-C-RTS use less memory thanks to the smaller binary classification head (also potentially allowing for larger batch sizes.)

Models	BERT-S / MLM	BERT-S / RTS	BERT-S / C-RTS	BERT-S / SLM
LM head complexity	$O(bs \times L \times \lvert V \rvert)$	$O(bs \times L)$	$O(bs \times L)$	$O(bs \times L \times \lvert V \rvert)$
FLOPS	1.83×10^{18}	1.64×10^{18}	1.64×10^{18}	1.83×10^{18}
Training time	1d 7h	17h	17h	1d 7h

Table 3.13: FLOPS used to pre-train each *small* model, language modelling head complexity and real training time on the same hardware. See the caption of Table 3.12 for more details.

Small models

Table 3.14 shows that RTS and C-RTS on *small* models outperform MLM in most tasks. In addition, RTS also improves over SLM by a wide margin in two of the three considered benchmarks (WikiQA and ASNQ). Besides, thanks to the smaller architecture, RTS greatly impacts the pre-training time. In particular, Table 3.13 shows that *small* models exploiting RTS or C-RTS for pre-training consume half the resources used by MLM. We controlled the training of RTS and C-RTS on a small fraction of the pre-training set used for validation. We discovered that the last-epoch F1 scores of RTS and C-RTS in detecting replaced tokens were 96.3 and 94.7, respectively. This confirms that C-RTS is a more challenging task and may be well suited for longer pre-training. RTS, instead, would be more easily solved, thus providing weaker loss signals to the model.

Models cost

Tables 3.12 and 3.13 show the training compute and time required by several models. FLOPS are significant indicators but may reflect different practical performances if the

Model	WikiQA		TREC-QA		ASNQ		GLUE	Time
	MAP	MRR	MAP	MRR	MAP	MRR	AVG	
BERT-S / MLM	66.9 (1.7)	68.2 (1.6)	80.9 (2.2)	85.8 (2.6)	58.6 (0.3)	66.0 (0.4)	74.1	× 1.00
BERT-S / RTS	71.8 (1.3)	73.5 (1.5)	81.4 (0.5)	87.3 (1.7)	59.8 (0.1)	66.9 (0.2)	75.4	× 0.54
BERT-S / C-RTS	69.4 (1.3)	70.8 (1.0)	81.3 (1.4)	86.5 (0.7)	59.6 (0.2)	67.1 (0.2)	75.7	× 0.54
BERT-S / SLM	69.5 (1.0)	70.8 (1.0)	81.9 (1.1)	87.3 (1.5)	59.2 (0.2)	66.5 (0.3)	75.7	× 1.00

Table 3.14: Results of RTS, C-RTS and SLM compared with MLM applied to *small* architectures on AS2 benchmarks and GLUE test set. We underline the results that are significantly different from the BERT-S / MLM baseline after a two-sided T-Test with a significance level of 95%. We show the standard deviation after 5 runs with different initialization seeds in rounded brackets.

underlying hardware implements special acceleration for some operations. In fact, the training times on our NVIDIA A100 GPU are not perfectly proportional to the model's FLOPS. For example, RTS and C-RTS are much faster to be pre-trained in practice in comparison to MLM and even more than the theoretical FLOPS difference thanks to the smaller memory footprint. Additionally, even by reducing the number of training steps of ELECTRA proportionally to the additional weight of the generator network, as suggested by the authors, ELECTRA still uses more computing than BERT for pre-training.

The tradeoff between good modelling and computing

We aim to produce models that require less computing budget to achieve performance similar to MLM-based models. Our results show that the performance improvement is logarithmic in the size of the pre-training dataset. At the same time, there is no statistically significant difference between our computationally lighter objectives and more expensive models such as ELECTRA. Indeed, Table 3.15 shows that top-performing architectures outperform MLM-based models only when trained on much more data. For example, ELECTRA-*Base* uses 21 times more resources than BERT-*Base*, while RoBERTa uses 53 times more. It is impressive the fact that to reach a score of 90.0 on GLUE, a model has to be trained for 2000 times the original BERT.

3.2.6 The role of clustering

We compared our two efficient approaches (RTS and C-RTS) to understand whether selecting more challenging replacement tokens could really improve the model performance on the downstream tasks on a long run. In order to perform this analysis, we pre-trained two *small* models with both objectives using a different setting. In particular, we increased the sequence length to 512 and trained for 200K steps with a batch size of 1024, thus letting the models (i) see much more tokens and (ii) compute attention scores over long sequences. Then, we evaluated them on five different benchmarks (WikiQA, TREC-QA, ASNQ, MRPC and QNLI). We chose these datasets because they cover a wide range of tasks (AS2, Paraphrasing and NLI) and a wide range of sizes, from the 3.6K examples of MRPC to the 20M of examples in ASNQ. The results

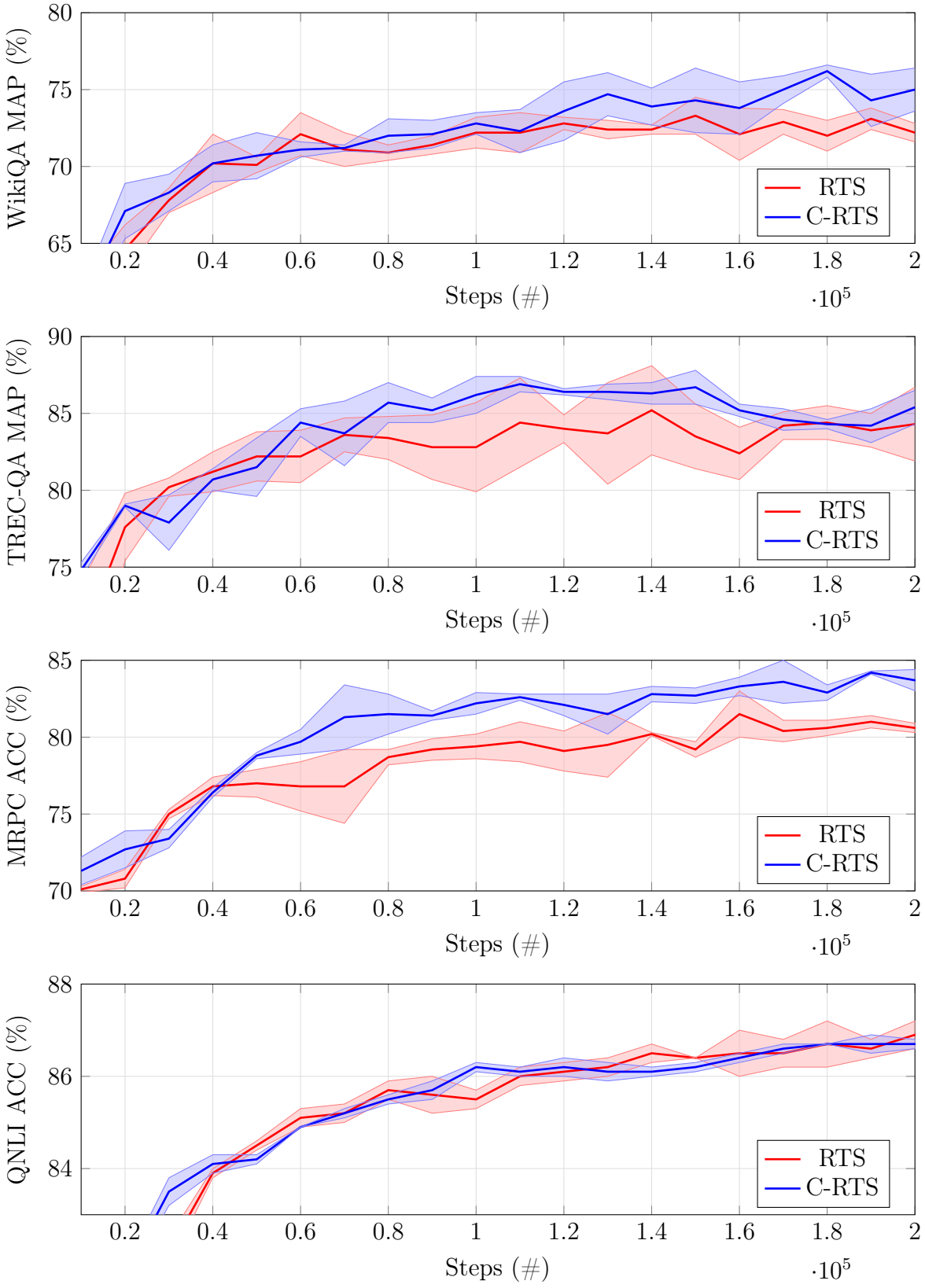

Figure 3.4: Performance comparison of RTS and C-RTS on WikiQA, TREC-QA, MRPC and QNLI.

Model	GLUE	FLOPS
BERT-S / RTS	75.4	× 0.10
BERT-S / MLM	74.1	× 0.12
BERT-S / SLM	75.7	× 0.12
BERT-B / RTS	79.9	× 0.81
BERT-B / MLM	79.7	× 1
BERT-B / SLM	80.4	× 1
BERT-L / MLM+NSP (Devlin et al., 2019)	83.3	× 12
ELECTRA-B / TD (Clark et al., 2020)	85.7	× 21
RoBERTa-B / MLM (Liu et al., 2019)	86.3	× 53
ELECTRA-L / TD (Clark et al., 2020)	88.6	× 194
RoBERTa-L / MLM (Liu et al., 2019)	88.8	× 200
ALBERT-L / MLM+SOP (Lan et al., 2020)	90.0	× 1937

Table 3.15: Large models comparison on the GLUE test set. We report the average accuracy over the different tasks of Table 3.10, while FLOPS refers to pre-training.

Dataset	Metric	RTS	C-RTS
WikiQA	MAP	72.2 (0.6)	75.0 (2.4)
TREC-QA	MAP	84.3 (2.4)	85.4 (1.1)
ASNQ	MAP	59.9 (0.2)	60.7 (0.1)
MRPC	Accuracy	81.5 (1.5)	84.2 (0.1)
QNLI	Accuracy	86.9 (0.3)	86.9 (0.1)

Table 3.16: Performance comparison between RTS and C-RTS on five different benchmarks. The results reported for WikiQA and TREC-QA are on the test set, while for ASNQ, MRPC and QNLI, we report the highest score on the development set. We show the standard deviation after 5 runs with different initialization seeds in rounded brackets.

shown in Table 3.16 clearly underline that in every experiment, C-RTS achieves better performance than RTS. For example, it scores between 1 and 3% points over RTS in MAP on the three AS2 datasets. Moreover, it provides more stable results, as it can be seen from the standard deviation across all experiments. We claim that the advantages of C-RTS over RTS derive from a more difficult pre-training objective, which is slower to converge and provides better loss signals in the last training epochs.

We demonstrate the superiority of C-RTS over RTS by doing a longer pre-training on a *small* model. We increased the maximum sequence length from 128 to 512 and kept the same batch size of 1024. We train until both models converge and no longer improve, exploiting the same pre-training data used for the other experiments. After every epoch, we evaluate the checkpoints on various tasks: WikiQA, TREC-QA and MRPC. We avoid very large datasets such as ASNQ because, on *small* models, they underline more the architectural differences than the pre-training technique as shown

in Figure 3.4.

3.2.7 Negative results: Non impactful objectives

C-RTS sampling within the same cluster We experimented with a simplification of C-RTS where tokens are always replaced with tokens within the same cluster. We found that it is important to maintain the possibility to sample also from other clusters because the model was able to learn how tokens are clustered after an enough large number of steps.

Position-based techniques We experimented with changing the position of tokens, i.e., we masked some positional encoding in an MLM-like approach. The objective was to retrieve the position of the masked tokens in the original sentence. The classification head of this approach is smaller than MLM (slightly larger than RTS, because of the 512 possible positions in BERT), but the results were worse by 3-4% points on GLUE.

We also defined another objective consisting of (i) shuffling input positions of some tokens, and then (ii) predicting their original position. We selected 15% of the tokens, and we permuted them. Although the results were below MLM by 1-2% on the GLUE average, the approach was as much fast as RTS. A combination of position-based objectives with the token-level ones is a possible future direction.

LM head-on ELECTRA's discriminator Capitalizing on the good performance reached from SLM, we implemented and evaluated a version of SLM for ELECTRA. Since SLM cannot be directly applied to the ELECTRA discriminator (the predictions are only performed on the output positions corresponding to tampered tokens nullifying the task of detecting fake tokens), we propose an alternative objective called *SLM-all*. In this case, the discriminator has to predict the whole input sentence, estimating which tokens were changed and predicting their original values. At the same time, it should only reproduce the input in output for unchanged inputs. As for the other objectives, we evaluated this approach on GLUE, WikiQA, ASNQ and TREC-QA but we obtained generally worst results than some of the other approaches, also with a more expensive training (1.42 times the time required by MLM with *base* models). This gap in the efficiency between SLM-all and the other models is so large because the latter predicts MLM-like token for every input token. Notice that even by reducing the number of training steps of the ELECTRA model by about 25% (to balance the presence of the additional generator with size 1/3), it uses slightly more FLOPS than BERT-MLM. Specifically, it scores 80.02 on the GLUE average score, 78.7, 86.7, 86.7 of MAP in WikiQA, TREC-QA and ASNQ.

3.2.8 Final discussion

In this work, we studied several alternative methods to pre-train Transformer models. Our approaches aim at designing pre-training objectives that (i) match the results

of well-known methods using fewer resources or (ii) outperform previous techniques with the same computing budget. This translates into shorter training, lower memory usage, and the possibility of increasing the batch size. Among other benefits, more efficient models can reduce carbon footprint and infrastructure costs. Notice that this advantage is even higher for smaller models since the MLM classification head is not shrunk proportionally to the architecture size (Turc et al., 2019), thus it increases its weight on the computational complexity for smaller models. Moreover, we demonstrate that the MASK token is useless and similar or even better results can be achieved using only token substitution. We reiterate that our objectives could be easily applied to many different transformer models; we chose the BERT setting due to computational resource constraints and easy reproducibility. Finally, we show that recent models' performance improvements are mostly driven by longer training phases rather than by more refined architectures.

We evaluated our approaches on several datasets, such as GLUE, WikiQA, TREC-QA and ASNQ. The results show that RTS and C-RTS match the accuracy of MLM in most tasks, requiring a lower amount of computational effort (20% less), while SLM outperforms MLM in most tasks. C-RTS also shows a lower accuracy in detecting replaced tokens, meaning that the task is harder and better suited for longer pre-training sessions.

In addition, we tested our pre-training objectives on smaller transformer architectures. In this scenario, RTS and C-RTS obtained better performances than MLM by requiring half of its time to be pre-trained. For example, RTS outperforms MLM by almost 5 MAP points on WikiQA. This last finding is potentially helpful for training transformer models from scratch with limited resources.

3.3 Case study 3: Loss manipulation and automatic translation for AS2 tasks

In this case study, we deep dive into the Question Answering field. In particular, we face, for the first time in this Thesis, two specific problems: the scarcity of data (in this case motivated by the multilingual setting) and the manipulation of the training loss.

This case study is based on a paper named *"Datasets for Multilingual Answer Sentence Selection"* that, at the time of the writing of this Thesis, is currently under review. However, the paper has already been released in the form of a preprint (Gabburo et al., 2024a).

3.3.1 Introduction

In this case study, we delve into the field of Question Answering (QA) to address two significant challenges: the scarcity of data, particularly in multilingual settings, and the effective manipulation of the training loss to enhance model performance. These challenges are critical in advancing QA systems for low-resource languages and optimizing their training processes to achieve robust performance.

This case study is based on our paper, *"Datasets for Multilingual Answer Sentence Selection"*, which is currently under review. Despite this, the work has been made publicly available as a preprint (Gabburo et al., 2024a). The study focuses on creating large-scale multilingual datasets and exploring innovative methods for training Answer Sentence Selection (AS2) models using these resources. By targeting the dual problems of data scarcity and training optimization, this work contributes significantly to bridging the gap in multilingual QA research and lays the groundwork for further advancements in low-resource language processing.

3.3.2 improving Multilingual QA with Translated AS2 Corpora

Answer Sentence Selection (AS2) is a core component of many Question Answering (QA) systems, playing a critical role in identifying the most relevant answer for a given question from a pool of candidate sentences. AS2 systems have seen significant advancements in both academic and industrial contexts, with substantial improvements in models and datasets over recent years (Wang et al., 2007; Yang et al., 2015; Garg et al., 2019; Zhang et al., 2022a; Di Liello et al., 2022b; Gupta et al., 2023). However, much of this progress has been concentrated on English, leaving a noticeable gap for other widely spoken languages, such as French, German, Italian, Portuguese, and Spanish. These medium-resource languages face persistent challenges due to the lack of high-quality datasets for training AS2 models.

Recent research has explored Machine Translation (MT) as a viable solution to overcome data scarcity in low-resource languages, with promising results (Kumar et al., 2021; Ranathunga et al., 2023; Gupta et al., 2023). In the AS2 domain, widely used English datasets such as ASNQ (Garg et al., 2019), WikiQA (Yang et al., 2015), and TREC-QA (Wang et al., 2007) provide substantial benchmarks, but their absence in other languages has hindered the development of multilingual AS2 systems.

In our work, we address this gap by creating three large multilingual AS2 datasets: mASNQ, mWikiQA, and mTREC-QA. These datasets span five European languages: French, German, Italian, Portuguese, and Spanish, and collectively comprise over 100 million question-answer pairs. To construct these resources, we translated the original English datasets (ASNQ, WikiQA, and TREC-QA) using the state-of-the-art NLLB model (Team et al., 2022). Beyond translation, we developed an optimized loss function designed to (i) incorporate knowledge about translation quality and (ii) leverage supervision from an expert AS2 model.

To validate our approach, we trained multiple AS2 models on the mASNQ, mWikiQA, and mTREC-QA datasets and evaluated their performance across several benchmarks. Our results demonstrate that the combination of these multilingual datasets and the optimized loss function significantly outperforms existing approaches, producing robust rankers for medium-resource languages. These findings not only reduce the language barrier but also provide valuable assets for researchers working in low-resource NLP settings.

Additionally, we conducted extensive ablation studies to explore the broader impact of our methods. These studies revealed that:

- The proposed loss function generalizes effectively to other tasks, such as passage reranking.
- Dedicated multilingual datasets are essential to achieving optimal performance across multiple languages.
- The correlation between ranks produced by expert models and student models is critical for improving AS2 model robustness.

Our contributions advance the state of multilingual AS2 research, addressing long-standing challenges in data scarcity and language inclusivity while offering practical tools for future studies in this domain.

3.3.3 AS2 Translated Datasets

For the dataset translation process, we utilized the largest variant of the NLLB model (NLLB-200-3.3B), which comprises 3.3 billion parameters. Our translation efforts focused on three English datasets: TREC-QA, WikiQA, and ASNQ, with both the questions and answers translated into five target languages French, German, Italian, Portuguese, and Spanish.

The translation process was conducted in two steps. First, we applied the NLLB model to translate the source datasets. Second, we implemented quality assurance measures to refine the translated content. Specifically:

- We employed a cross-language semantic similarity model (Reimers and Gurevych, 2020) to assess the semantic alignment between the original English sentences and their translated counterparts. This model, available as a pre-trained Hugging Face model[3], identified translations with significant semantic deviations.
- We applied heuristic-based corrections to address translation errors. These heuristics targeted issues such as mistranslations, non-original text artifacts, and clarity inconsistencies, ensuring the overall quality of the datasets.

The resulting multilingual datasets provide high-quality resources for training robust AS2 models, enabling advancements in medium-resource languages and facilitating more inclusive NLP research. These datasets and methodologies represent a significant step toward overcoming linguistic barriers in QA systems.

3.3.4 AS2 Translated Datasets

For dataset translation, we use the largest variant of the NLLB model (NLLB-200-3.3B), which has 3.3 billion parameters. We consider three datasets: TREC-QA, WikiQA, and ASNQ. For each one, we translate both the questions and the answers. Our translation process can be described as a *two-step* procedure. In the first step, we utilize the NLLB model to translate the source datasets into five languages: French, German, Italian,

[3] https://huggingface.co/sentence-transformers/paraphrase-multilingual-mpnet-base-v2

Portuguese, and Spanish. In the second step, we employ two techniques to evaluate the quality of the translations: identifying poor translations and correcting any misleading sentences.

Firstly, we use a cross-language semantic similarity model released by (Reimers and Gurevych, 2020) [4] to assess the quality of the translated sentences. This model compares the semantic similarity between the original sentences in English and their translated versions in the target language. By measuring the similarity, we can identify bad translations that deviate significantly from the original sentences in English due to the presence of errors. Secondly, we target these errors by applying a set of heuristics to correct misleading sentences. These heuristics are designed to correct errors, improve clarity, remove non-original text, and enhance the overall quality of the translated datasets.

Datasets

In this work, we considered and translated three datasets for answer sentence selection (AS2) in 5 different locales:

- **mTREC-QA**, originates from TREC-QA (Wang et al., 2007), which is created from the TREC 8 to TREC 13 QA tracks. TREC 8-12 constitutes the training set, while TREC 13 questions are set aside for development and testing. We used the *Clean* setting, meaning that questions without an answer or with only correct or incorrect answer-sentence candidates are removed.

- **mWikiQA** is the translated version of WikiQA (Yang et al., 2015). It contains 3047 questions sampled from Bing query logs; candidate answer sentences are extracted from Wikipedia and then manually labeled to assess whether it is a correct answer. Some sentences do not have a correct answer (*all -*) or have only correct answers (*all +*). We trained using *no all -* mode and tested in the *clean* setting (without both *all +* and *all -*).

- **mASNQ** comes from ASNQ (Garg et al., 2019) which is an AS2 dataset created by adapting the Natural Question (Kwiatkowski et al., 2019) corpus from Machine Reading (MR) to the AS2 task. We replicated this passage using the scripts provided by Lauriola and Moschitti (2021b).

We summarize the statistics of these datasets in Table 3.17.

Removing Translation Artifacts

Despite the good quality of the translation, the dataset still presents some inconsistencies and artifacts. We identified four major classes of translation artifacts: (i) the Meaning mismatch between the original and the translated sentences, (ii) The addition of unnecessary suffixes and prefixes, (iii) The difficulty in interpreting and translating numerical strings, (iv) Out-of-topic translations of partial contexts. We provide some examples of these translation artifacts in Table 3.19.

[4] https://huggingface.co/sentence-transformers/paraphrase-multilingual-mpnet-base-v2

Dataset	**Split**	**#Question**	**#QA Pairs**
mASNQ	Train	57240	20377168
	Validation	2672	930062
mWikiQA	Train	2118	20356
	Validation	296	2731
	Validation (++)	126	1130
	Validation (clean)	122	1126
	Test	633	6160
	Test (++)	243	2350
	Test (clean)	237	2340
mTREC-QA	Train	1227	53282
	Validation	65	1117
	Test	68	1441

Table 3.17: Dataset statistics for mASNQ, mWikiQA, and mTREC-QA for each language. The datasets have the same statistics in their original version, and considering all the languages, the corpora comprehend more than 100M examples. Notice that for mWikiQA, we also report the statistics of the clean and the no-all-negatives (++) splits.

Dataset	**Split**	**#Question**	**#QA Pairs**
mASNQ	Train	57240	20377168
	Validation	2672	930062
mWikiQA	Train	2118	20356
	Validation	296	2731
	Validation (++)	126	1130
	Validation (clean)	122	1126
	Test	633	6160
	Test (++)	243	2350
	Test (clean)	237	2340
mTREC-QA	Train	1227	53282
	Validation	65	1117
	Test	68	1441

Table 3.18: Dataset statistics for mASNQ, mWikiQA, and mTREC-QA for each language. The datasets have the same statistics in their original version, and considering all the languages, the corpora comprehend more than 100M examples. Notice that for mWikiQA we report also the statistics of the clean and the no-all-negatives (++) splits.

Split	Language	Examples
Original	ITA	*ISSN 0362 - 4331*
Translated		ISSN 0362 - 4331 - Non lo so.
Original	DEU	*Kusumanjali Prakashan .*
Translated		Ich bin ein guter Mensch.
Original	FRA	*Jump up to : " Safe Haven (2013)*
Translated		Sauter sur refuge

Table 3.19: Examples of translation artifacts. The artifacts are highlighted in red. Notice that (i) "Non lo so" is an Italian sentence which translated in English means "I don't know", (ii) "Ich bin ein guter Mensch" means "I am a good person" in German, and (iii) the maening of the "Sauter sur refuge" in French is "Jump on refuge".

To tackle these issues, we apply some heuristics to improve the dataset quality by designing a simple human-centered pipeline to mitigate these artifacts.

In our approach, we first compute the similarity score between every translated example in the dataset and the corresponding original test. Then we filter out translated examples below a similarity threshold of 0.8 and, on the remaining set, we compute the most common $1k$ n-grams with n ranging from 4 to 9. Second, we manually inspect these extracted n-grams, identifying and removing artifact patterns that could distort the data. Subsequently, we systematically remove occurrences of those problematic artifacts from the translated dataset. To further improve this operation, we also identify the examples where the original sentence and the 75% of the not-blank characters are numbers, and if the similarity score is low (under 0.8), we replace the translated sentences with the original one.

Semantic Similarities

To assess the quality of the translations and to quantify the benefit given by our heuristics, we evaluate the semantic similarity between the original sentences and their translated versions. For each question-answer pair of each dataset, we compare the original sentences in English with their translated version in the target language. The overall similarity measure between the originals and the translated sentences in each dataset is computed considering the mean of the semantic similarity scores across all the question-answer pairs. This average score indicates how closely the translated sentences align with their original counterparts in terms of semantic meaning. Table 3.20 reports the comparison between the similarity scores of the translated dataset and the original one.

3.3.5 AS2 Reward Loss

High-resource languages such as English allow us to train compelling ranker models due to the huge quantity of available data. Unfortunately, the scarcity of data for

	Language	Similarity
Dev	DEU	$0.869 \pm 0.178 \rightarrow 0.991 \pm 0.003$
	FRA	$0.838 \pm 0.223 \rightarrow 0.990 \pm 0.003$
	ITA	$0.913 \pm 0.115 \rightarrow 0.990 \pm 0.001$
	POR	$0.915 \pm 0.117 \rightarrow 0.992 \pm 0.003$
	SPA	$0.857 \pm 0.211 \rightarrow 0.990 \pm 0.001$
Train	DEU	$0.871 \pm 0.175 \rightarrow 0.923 \pm 0.110$
	FRA	$0.841 \pm 0.218 \rightarrow 0.923 \pm 0.101$
	ITA	$0.915 \pm 0.112 \rightarrow 0.940 \pm 0.081$
	POR	$0.915 \pm 0.115 \rightarrow 0.941 \pm 0.082$
	SPA	$0.860 \pm 0.206 \rightarrow 0.932 \pm 0.090$

Table 3.20: Similarities between ASNQ and mASNQ. On the left of the arrow ($\rightarrow$) the similarity reached after the initial translation is reported; on the right side, there is the similarity score after the application of the heuristics.

low-resource languages has a noticeable impact on the quality of the trained models. This work aims to mitigate this problem by proposing new datasets (Sec. 3.3.4) and designing a new loss function. The usage of automatically translated datasets presents challenges and biases that can affect the final model performance. We have identified two main problems in this regard. Firstly, despite the improvement in the quality of recent translation pipelines, they are not entirely error-proof and may introduce errors during the translation process. Secondly, specific idiomatic sentences pose difficulties for translation and cannot be accurately captured by semantic-similarity models.

Indeed, to bridge this performance gap, we designed a specific loss function that, combined with the use of translated datasets, allows us to overcome these limitations. Specifically, we modeled the loss as

$$l = \alpha \, \mathrm{CE}(Y, \hat{Y}) \, W_{\text{trans}} + \beta \, \mathrm{MSE}(Y, \hat{S}_{\text{AS2}}) \tag{3.6}$$

Where α and β are normalization parameters used to balance the relative importance of the two loss terms. $\mathrm{CE}(Y, \hat{Y})$ represents the cross entropy loss between the model prediction scores $\hat{Y}$ and the dataset labels Y. W_{trans} is a vector of translation weights $w_i \in [0, 1]$ obtained doing the semantic similarity between the original and the translated texts. $\mathrm{MSE}(Y, \hat{S}_{\text{AS2}})$ is the Mean Square Error used to perform the knowledge distillation from an expert teacher model trained on the original untranslated dataset.

Intuitively, the first term of Equation 3.6 allows us to weigh in different ways the examples characterized by good translation quality and bad examples. The key idea is similar to the one presented by the loss weighting approach in Gabburo et al. (2022, 2023a), in which they forced the model to learn more from good instances and less from poor ones. Meanwhile, the second term introduces a bias to the model by incorporating knowledge from an expert model, similar to what has been recently proposed in Gupta et al. (2023).

3.3.6 Experiments

In this section, we measure the benefits of our datasets applied to existing multilingual models. With these experiments, we aim to verify and prove the effectiveness of our contributions. Specifically, we want to show that the translated data, combined with our reward loss, could be used to train state-of-the-art AS2 models in multiple languages. For each considered language, we finetuned existing multilingual transformer models on both the original and our translated datasets. We measure the performance by using information retrieval metrics like Mean Average Precision (MAP) and the Precision at 1 (P@1). These metrics allow us to compare the results with the English baselines trained on the original datasets and to measure the performance improvement given by the ASNQ transfer step on different languages.

To verify these hypotheses, we consider an existing multilanguage pre-trained cross-encoder transformer model, which is XLM-RoBERTa base[5], and BERT-multilingual[6]. Following the TANDA (Garg et al., 2019) approach, we perform a two-stage training for each model. Precisely, this technique consists of a two-stage training paradigm, where the first training stage, named *transfer step*, consists of training the models on ASNQ to teach them to recognize and solve the AS2 tasks. In the second step, named *adaptation step*, the transferred models are finetuned on the final target AS2 datasets. In our setting, we apply this paradigm by first training and doing a separate transfer step on each language of mASNQ. Secondly, we finetunefinetune the obtained models on mWikiQA and mTREC-QA (e.g., $\text{mASNQ}_{\text{ITA}} \rightarrow \text{WikiQA}_{\text{ITA}}$).

To have a comprehensive perspective of how our approach behaves, we measure the performance on three test sets from (i) mWikiQA, (ii) mTREC-QA, and (iii) Xtr-WikiQA[7].

With the first two datasets, we aim to show the performance of our models in a controlled environment where the translation pipeline and the heuristics are the same as those used on mASNQ. The third dataset, instead, allows us to prove that (i) our approach is robust in a zero-shot setting and (ii) can be extended to datasets translated using different pipelines.

To perform our experiments, we test XLM-RoBERTa on ASNQ and mASNQ datasets with specific parameters: batch size of 1024, Adam optimizer with a learning rate of $5e-6$, precision set to 32, and 10 training epochs. For mWikiQA and mTREC-QA datasets, the batch size was 32, Adam optimizer with a learning rate of $5e-6$, precision set to 16-mixed, and 40 training epochs with early stopping. We select the best model, maximizing the mean average precision (MAP) on the development set. The same parameters were used to train the Multilingual BERT architecture. All experiments utilized 8 NVIDIA V100 32 GB GPUs.

Language	Transfer	Adapt	Reward Loss	mWikiQA MAP	mWikiQA P@1	mTrecQA MAP	mTrecQA P@1	Xtr-WikiQA MAP	Xtr-WikiQA P@1
ENG		✓		0.796 (± 0.011)	0.691 (± 0.015)	0.866 (± 0.015)	0.853 (± 0.031)	0.795 (± 0.011)	0.689 (± 0.015)
	✓	✓		0.874 (± 0.007)	0.813 (± 0.017)	0.892 (± 0.007)	0.871 (± 0.019)	0.872 (± 0.007)	0.809 (± 0.015)
DEU		✓		0.770 (± 0.024)	0.657 (± 0.031)	0.870 (± 0.011)	0.871 (± 0.016)	0.779 (± 0.012)	0.669 (± 0.015)
	✓	✓		0.853 (± 0.005)	0.767 (± 0.006)	0.904 (± 0.006)	0.903 (± 0.017)	0.868 (± 0.005)	0.800 (± 0.011)
	✓	✓	✓	0.847 (± 0.005)	0.763 (± 0.010)	0.906 (± 0.001)	0.897 (± 0.010)	0.868 (± 0.010)	0.806 (± 0.018)
FRA		✓		0.769 (± 0.012)	0.662 (± 0.018)	0.872 (± 0.007)	0.859 (± 0.013)	0.760 (± 0.009)	0.646 (± 0.012)
	✓	✓		0.836 (± 0.006)	0.752 (± 0.012)	0.891 (± 0.010)	0.874 (± 0.029)	0.844 (± 0.005)	0.778 (± 0.014)
	✓	✓	✓	0.830 (± 0.010)	0.743 (± 0.017)	0.905 (± 0.004)	0.912 (± 0.010)	0.830 (± 0.010)	0.743 (± 0.017)
ITA		✓		0.768 (± 0.019)	0.660 (± 0.026)	0.855 (± 0.024)	0.844 (± 0.049)	0.761 (± 0.021)	0.657 (± 0.027)
	✓	✓		0.828 (± 0.004)	0.749 (± 0.008)	0.870 (± 0.006)	0.871 (± 0.016)	0.820 (± 0.012)	0.742 (± 0.027)
	✓	✓	✓	0.865 (± 0.003)	0.800 (± 0.008)	0.866 (± 0.012)	0.868 (± 0.028)	0.867 (± 0.007)	0.804 (± 0.012)
POR		✓		0.798 (± 0.011)	0.704 (± 0.019)	0.855 (± 0.021)	0.704 (± 0.019)	0.780 (± 0.011)	0.684 (± 0.020)
	✓	✓		0.853 (± 0.018)	0.781 (± 0.029)	0.874 (± 0.009)	0.781 (± 0.029)	0.849 (± 0.023)	0.775 (± 0.042)
	✓	✓	✓	0.868 (± 0.005)	0.808 (± 0.008)	0.885 (± 0.003)	0.808 (± 0.008)	0.868 (± 0.005)	0.808 (± 0.008)
SPA		✓		0.795 (± 0.009)	0.691 (± 0.016)	0.882 (± 0.013)	0.900 (± 0.016)	0.786 (± 0.015)	0.691 (± 0.024)
	✓	✓		0.847 (± 0.013)	0.768 (± 0.020)	0.898 (± 0.004)	0.929 (± 0.019)	0.859 (± 0.010)	0.790 (± 0.020)
	✓	✓	✓	0.854 (± 0.004)	0.787 (± 0.007)	0.902 (± 0.011)	0.929 (± 0.019)	0.852 (± 0.011)	0.775 (± 0.015)
Average	✓	✓		0.843 (± 0.009)	0.763 (± 0.015)	0.887 (± 0.007)	0.877 (± 0.022)	0.848 (± 0.011)	0.777 (± 0.023)
	✓	✓	✓	0.853 (± 0.005)	0.780 (± 0.010)	0.893 (± 0.006)	0.883 (± 0.015)	0.857 (± 0.009)	0.787 (± 0.014)

Table 3.21: Performance comparison of XLM-RoBERTa on mTREC-QA, mWikiQA, and Xtr-WikiQA (zero-shot from the model trained on mWikiQA). The transfer step is done on mASNQ, while the Adaptation is on mTREC-QA and mWikiQA. Results in terms of MAP and P@1, for various language and model configurations. The models trained with the reward loss prove improved performance compared with the other approaches (e.g., 0.853 vs 0.843 on mWikiQA in terms of average MAP).

AS2 Results

In this section, we present the experimental results of our approaches on three different AS2 datasets: mWikiQA, mTREC-QA, and the existing Xtr-WikiQA dataset (Gupta et al., 2023). Table 3.21 provides an overview of the performance achieved by our models. First, we observe that our models trained with the loss function achieve performance levels comparable to those of English models. This finding is particularly noticeable when considering the Portuguese language across all datasets. When evaluating the models on Xtr-WikiQA, which can be considered as a zero-shot scenario, as the models are trained on mWikiQA and tested on Xtr-WikiQA, we find that our approaches demonstrate robustness even when dealing with datasets translated using a different translation pipeline. Specifically, Xtr-WikiQA is translated using Amazon Translate. The results obtained on Xtr-WikiQA validate the effectiveness of our procedures in handling such translation variations. However, we also observe some negative results.

Specifically, our experiments highlight challenges specific to the French language. The XLM-RoBERTa model performance in this context is notably subpar, which aligns with earlier findings documented in the relevant literature. Furthermore, we highlight the impact of employing the reward loss on the average results of the multilingual datasets. Across mWikiQA, mTREC-QA, and Xtr-WikiQA, we consistently observe

[5] https://huggingface.co/xlm-roberta-base
[6] https://huggingface.co/bert-base-multilingual-cased
[7] https://huggingface.co/datasets/AmazonScience/xtr-wiki_qa

Language	ASNQ		mASNQ	
	MAP	P@1	MAP	P@1
DEU	0.814	0.705	0.839	0.755
FRA	0.819	0.717	0.793	0.671
ITA	0.819	0.715	0.839	0.726
POR	0.822	0.722	0.842	0.755
SPA	0.830	0.738	0.835	0.751

Table 3.22: Comparison of XLM-RoBERTa base transferred on mASNQ and ASNQ and tested on mWikiQA in a cross-lingual setting.

improved performance when incorporating the reward loss. For instance, the average results with the reward loss for mWikiQA are 0.853, compared to 0.843 without it. Similarly, for mTREC-QA, the respective values are 0.893 and 0.887, and for Xtr-WikiQA, the values are 0.857 and 0.848. These results highlight the importance of utilizing the reward loss to enhance performance. In addition, we present a comparison of various models on Xtr-WikiQA in a zero-shot setting in Table 3.23. These models have been trained on mASNQ using our reward loss and are evaluated against existing models trained on well-known and extensive passage reranking datasets. We find that models trained on mASNQ for the Xtr-WikiQA task outperform models trained on other datasets, such as mMARCO and MSMARCO. This observation suggests that mASNQ is a more suitable dataset for AS2 than mMARCO and MSMARCO. Moreover, when comparing the performance of BERT-multilingual and XLM-RoBERTa, we find that, on average, BERT-multilingual performs better. This finding is evident when analyzing the results across different languages, including Italian, Portuguese, and Spanish. Overall, our results demonstrate the effectiveness and robustness of our approaches on AS2 datasets, showcasing competitive performance and the superiority of specific training configurations and models over others.

3.3.7 Ablation: Passage Ranking

To further demonstrate the effects of our new loss function, we also perform several experiments on a different task: Passage RerankingReranking (PR). Passage RerankingReranking is an Information Retrieval (IR) task that consists of reordering a set of retrieved passages for a given query. For this reason, we consider a well-known dataset named mMARCO (Bonifacio et al., 2021), well known in the multilanguage IR community. Specifically, we select a random language among the ones considered in the previous experiments, and we train several multilanguage models using our loss. In detail, we split the original Italian dataset in train, validation, and test splits (Tab. 3.17). We compare the results obtained by our approaches with two models: the first is a multilanguage BERT trained on the English MSMARCO, while the second model is trained on our train split but without our loss. In Table 3.24, we present the results of this comparison. They clearly show that our models based on the reward loss outperform the model trained on mMARCO and MSMARCO (e.g., 0.687 vs 0.682 in terms of MAP).

Language	Model	MAP	P@1
ENG	✱ mmarco-mMiniLMv2	0.812	0.722
	✱ bert-multilingual-msmarco	0.798	0.714
	xlm-roberta-base	0.855	0.794
	bert-multilingual	0.814	0.724
DEU	✱ mmarco-mMiniLMv2	0.797	0.700
	✱ bert-multilingual-msmarco	0.759	0.663
	xlm-roberta-base	0.844	0.770
	bert-multilingual	0.834	0.753
FRA	✱ mmarco-mMiniLMv2	0.782	0.675
	✱ bert-multilingual-msmarco	0.734	0.621
	xlm-roberta-base	0.813	0.720
	bert-multilingual	0.863	0.807
ITA	✱ mmarco-mMiniLMv2	0.778	0.671
	✱ bert-multilingual-msmarco	0.735	0.634
	xlm-roberta-base	0.830	0.741
	bert-multilingual	0.846	0.765
POR	✱ mmarco-mMiniLMv2	0.809	0.724
	✱ bert-multilingual-msmarco	0.755	0.646
	xlm-roberta-base	0.840	0.761
	bert-multilingual	0.841	0.761
SPA	✱ mmarco-mMiniLMv2	0.791	0.691
	✱ bert-multilingual-msmarco	0.753	0.650
	xlm-roberta-base	0.832	0.737
	bert-multilingual	0.853	0.774

Table 3.23: Performance comparison of XLM-RoBERTa base model in a zero-shot setting on the Xtr-WikiQA task. Models trained on mASNQ dataset, denoted by ✱, outperform those trained on other datasets like mMARCO and MSMARCO. Moreover, BERT-multilingual consistently performs better than XLM-RoBERTa in various languages (Italian, Portuguese, Spanish), indicating the robustness and competitiveness of the approach on AS2 datasets.

Dataset	Transfer	Adapt	Reward Loss	mMARCO MAP	mMARCO P@1
MSMARCO	✓			0.631	0.502
mMARCO		✓		0.682	0.553
mMARCO		✓	✓	0.686	0.558
mASNQ→mMARCO	✓	✓	✓	<u>0.687</u>	<u>0.559</u>

Table 3.24: Comparison of BERT-multilingual performance on $\text{mMARCO}_{\text{ITA}}$ test set. We train the two baselines respectively on the English MSMARCO and the mMARCO Italian split. The models trained using the reward loss, consistently improve the two presented baselines showing that (i) the loss is beneficial also when used in the Passage Reranking task, and (ii) the transfer step on mASNQ is helpful also in this domain.

Although there has been a modest improvement, it is noticeable that this improvement becomes noticeable because of the large size of the mMarco test set. In addition, these findings highlight the advantages our approach offers for tasks beyond AS2. Even with a marginal improvement, it is evident that adapting a model transferred on mASNQ can yield further performance improvements.

3.3.8 Ablation: Cross-Lingual

This ablation aims to determine the advantages of using the mASNQ dataset to train state-of-the-art answer ranking models on languages different from English. To achieve this, we compare the performance of cross-lingual models trained on ASNQ and WikiQA, with models that were firstly trained on mASNQ and mWikiQA, on the different languages that compose mWikiQA.

Table 3.22 compares the performance of models trained only on the original versions of ASNQ and WikiQA with the performance of the same architecture (XLM-RoBERTa base) but trained on our multilanguage datasets. To achieve this goal, we measure the performance of each model across all the different test sets of mWikiQA and across their languages. To perform the evaluation, we considered two proxy measures to understand the quality of the models: Mean Average Precision (MAP) and Precision at 1 (P@1). The results show that the models achieve higher MAP and P@1 scores when trained on mASNQ compared to ASNQ, indicating that training on the mASNQ dataset improves the performance of multilingual models in cross-lingual tasks. Across all languages, the models trained on mASNQ consistently outperform the models trained on ASNQ. This suggests that the mASNQ dataset can guarantee a performance boost for non-English target datasets, confirming our hypotheses.

3.3.9 Ablation: Ranks correlation

This study compares the ranking outputs of two sets of models, analyzing the correlation between their rankings. The first set comprehends models trained on mASNQ and

mWikiQA and then tested on the mWikiQA test set, while the second set contains models trained on ASNQ and WikiQA and evaluated on the original English WikiQA test set.

We design this experiment in order to compare the rank provided for each question q^i_{Eng} of the original English dataset (WikiQA) with the semantically equivalent question q^i_T and its rank for each language T in mWikiQA. To measure the performance, we compute three correlation metrics to properly evaluate the correlation between the rankings of each pair of questions $\{q^i_{Eng}, q^i_T\}$; in this way, we allow to determine the level of agreement between the two models ranking outputs, providing insights into the potential differences between them. Specifically, we consider the XLM-RoBERTa base, and we compute the Kendall, Spearman, and Pearson correlation metrics on mWikiQA and mTREC-QA. The results in Table 3.25 show a strong positive correlation between the performance of machine translation models trained in English and tested in English and the models trained in other languages (using mASNQ, mWikiQA, and mTREC-QA). This correlation is evident across all evaluation metrics, with Kendall correlations ranging from 0.694 to 0.720, Spearman correlations ranging from 0.802 to 0.824, and Pearson correlations ranging from 0.872 to 0.908 for the mASNQ→mWikiQA task. The high correlation values, ranging from 0.547 to 0.733, across all languages for the mASNQ→mTREC-QA task further support this notion. The Kendall, Spearman, and Pearson correlations show consistently high values, indicating that the translation quality and model performance are consistently strong. The results of the analysis demonstrate (i) the effectiveness of the translation process for mASNQ and (ii) the strong performance of the models.

mASNQ→mWikiQA			
Language	**Kendall**	**Spearman**	**Pearson**
DEU	0.720	0.820	0.908
FRA	0.694	0.802	0.872
ITA	0.713	0.817	0.903
POR	0.710	0.824	0.908
SPA	0.698	0.807	0.902
mASNQ→mTREC-QA			
Language	**Kendall**	**Spearman**	**Pearson**
DEU	0.547	0.666	0.713
FRA	0.566	0.663	0.709
ITA	0.513	0.629	0.695
POR	0.567	0.669	0.733
SPA	0.587	0.688	0.728

Table 3.25: Kendall, Spearman and Pearson correlation computed between the ranks originated from model trained the original ASNQ and mASNQ. The reported values are computed using XLM-RoBERTa base models transferred on ASNQ and mASNQ and then finetuned on mWikiQA and mTREC-QA.

Chapter 4

Automatic Evaluation for Generative Question Answering

Natural Language Processing (NLP) has advanced significantly, particularly in the development of Question Answering (QA) systems. These systems aim to provide accurate answers to user queries by leveraging large datasets and advanced algorithms. However, evaluating the performance of QA systems remains a significant challenge. Traditional evaluation methods, such as human annotations, are accurate but resource-intensive, requiring substantial time and effort. This has driven the development of automated evaluation metrics capable of providing reliable assessments of QA performance.

Despite the progress in automated metrics, most existing approaches, such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), BERTScore (Zhang* et al., 2020), and BLEURT (Sellam et al., 2020), show weak correlations with human judgment. These metrics often rely on limited references and lack the ability to adequately penalize irrelevant or incorrect answers, making them less effective for evaluating modern QA systems. To overcome these limitations, we introduce SQuARe (Sentence-level QUestion AnsweRing Evaluation), a novel reference-based evaluation metric. SQuARe is a state-of-the-art approach based on transformer architectures that enables accurate evaluation of both extractive and generative QA models. This technique was specifically designed to improve upon existing methodologies by addressing their shortcomings. The innovations of SQuARe include:

- **Multiple References:** Unlike traditional metrics, SQuARe evaluates answers against multiple positive reference answers, capturing the diversity of valid responses and reflecting real-world variability in human answers.
- **Incorporation of Negative References:** SQuARe leverages incorrect reference answers to penalize irrelevant or erroneous outputs, enhancing its ability to differentiate between high- and low-quality answers.
- **Maximizing Correlation with Human Judgment:** The model is specifically trained to align with human evaluations by leveraging the combined use of positive and negative references, ensuring a more reliable assessment of QA system performance.

This work is based on our paper, "SQuARe: Automatic Question Answering Evaluation using Multiple Positive and Negative References," published at IJCNLP-AACL 2023 (Gabburo et al., 2023b). In the paper, we demonstrate that SQuARe significantly outperforms existing metrics, achieving stronger correlations with human judgment across various datasets. By relying on the robust architecture of transformers and a novel training paradigm, SQuARe provides an accurate and scalable solution for evaluating both extractive and generative QA models. Evaluating generative QA models requires an understanding of both syntactic fluency and semantic correctness. Traditional token-based metrics (e.g., BLEU, ROUGE) primarily capture lexical overlap rather than meaning. This leads to poor evaluations in scenarios involving paraphrasing or varied yet valid responses. The introduction of SQuARe represents a substantial advancement in the field of automated QA evaluation. By addressing the limitations of traditional metrics and incorporating a more nuanced approach, SQuARe enables researchers and practitioners to assess QA systems more effectively, fostering the development of more robust and reliable models. The results presented in this chapter underscore its potential as a pivotal tool for advancing QA evaluation methodologies. SQuARe aligns more closely with linguistic evaluation principles by integrating:

- **Semantic Similarity:** By leveraging transformer-based contextual embeddings, SQuARe evaluates meaning rather than just word overlap.
- **Discourse Coherence:** By considering entire answer structures rather than isolated words, SQuARe ensures logical flow and contextual relevance.
- **Error Detection through Contrastive Learning:** Negative references allow for recognizing factually incorrect but lexically similar outputs, improving factual correctness in evaluation.
- **Pragmatic Relevance and Entailment:** SQuARe captures implicit entailment and pragmatic cues that ensure the generated answer is contextually appropriate.

Moreover, by modeling the broader semantic space and variability in human responses, SQuARe addresses challenges that traditional metrics undercount. This approach not only ensures better alignment with human judgments but also provides a more robust and interpretable framework for generative QA.

4.1 Automatic Evaluation Metrics for QA

Traditional token/n-gram level similarity metrics like BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004) are commonly used in text generation tasks. However, these metrics have been shown to achieve poor correlation with human judgments in the context of QA (Reiter, 2018a; Gabburo et al., 2022). The primary issue is that QA often requires understanding and generating contextually relevant and semantically accurate answers, which n-gram overlap metrics fail to capture effectively.

To address these limitations, Kusner et al. (2015) propose using distance functions between word embeddings for text similarity, specifically leveraging the Wasserstein

distance. This approach captures semantic similarity more effectively than token-based methods. Building on this, other works (Clark et al., 2019) have explored embedding-based evaluation metrics. Recent advancements have introduced metrics for Neural Machine Translation (NMT) and summarization tasks that utilize contextual embeddings from transformer models. Notable examples include BERTScore (Zhang* et al., 2020), which uses BERT embeddings to evaluate similarity based on context and meaning, BLEURT (Sellam et al., 2020), and COMET (Rei et al., 2020), which integrates both linguistic and semantic aspects for evaluation. These approaches have been extended to other domains like text style (Wegmann and Nguyen, 2021) and summarization (Cao et al., 2020; Zeng et al., 2021).

In the domain of QA, particularly for entity-level span-extraction machine reading (MR) tasks, adaptations of BLEU and ROUGE have been proposed for answer comparison (Yang et al., 2018a), focusing on "yes-no" and "entity" questions. These adaptations, however, often fall short in generative QA contexts where answers can be more diverse and less predictable. To improve evaluation for generative QA, Si et al. (2021) suggest mining entities from knowledge bases (KBs) to use as additional gold standards, recognizing the need for multiple diverse reference answers. Similarly, Chen et al. (2019) propose a modification of BERTScore for QA by incorporating the question, paragraph context, and answer, though they note that traditional metrics like F1 are still reasonable for extractive MR tasks but not for generative QA.

For generative QA, where the goal is to generate novel and contextually appropriate answers, human annotations often serve as a critical component of evaluation (Min et al., 2021). Human evaluations can capture nuances that automated metrics might miss, such as the relevance and informativeness of answers. Recent advancements include learned metrics for sentence-level extractive QA, such as AVA (Vu and Moschitti, 2021a) and BEM (Bulian et al., 2022), which are trained to predict human judgment scores and offer a more nuanced evaluation compared to traditional metrics.

Additionally, recent studies have proposed reference-free metrics to evaluate QA. Ji et al. (2022) introduced Qascore, an unsupervised and unreferenced metric for question generation evaluation. Mohammadshahi et al. (2022) proposed RQUGE, a reference-free metric for evaluating question generation by answering the question. These metrics aim to address the limitations of reference-based evaluations and better align with human judgment.

Another emerging approach involves using large language models (LLMs) as evaluators. Studies by Chang et al. (2023), Guo et al. (2023), Zhang et al. (2024), and Hada et al. (2024) have shown that LLMs can effectively evaluate the quality of generated texts, including QA, by providing scores that correlate well with human judgments. This research direction highlights the potential of LLMs to complement or even replace traditional evaluation metrics in specific contexts.

The following sections of this chapter are structured as follows: we examine related studies on QA evaluation metrics, describe the SQuARe methodology, provide experimental results along with error analysis, and conclude with a discussion on linguistic insights and potential future directions.

4.2 SQuARe: Sentence-level QA Evaluation

Automatic evaluation of Question Answering (QA) systems, being a knowledge-intensive task, necessitates leveraging external knowledge sources to assess the correctness of answers (e.g., Knowledge Bases, Gold Standard reference answers). The process of automatic QA evaluation can be formalized using the notation $f(q, a, c) \rightarrow p$, where f represents the evaluation function applied to the question q, the target answer a, and the reference context c, yielding a correctness score $p \in [0, 1]$.

Prior studies (AVA, BEM) have demonstrated that employing a single Gold Standard (GS) reference answer as the context c achieves a higher correlation with human annotations compared to using only q and a. We introduce a supervised metric, SQuARe, which enhances the reference context c by incorporating (i) multiple gold standard references and (ii) negatively annotated answers as negative references.

Multiple Reference Answers The limitation of using a single correct reference in AVA and BEM restricts the evaluation scope of QA system predictions.

Example 4.2.1: Multiple Reference Evaluation

Question: What is the capital of Australia?

Single Reference:

Canberra.

Multiple References:

1) Canberra is the capital city of Australia.
2) Canberra is located in the Australian Capital Territory.

System Output:

The capital of Australia is Canberra.

SQuARe Score: High (correctly accounts for paraphrasing).

- Diverse questions may have multiple correct answers: for example, the question *"What is a band?"* can be correctly answered by both *"A flat, thin strip or loop of material, used as a fastener"* and *"A band is a group of people who perform instrumental and/or vocal music"*.

- Knowledge-seeking questions may require information spread across several references: for instance, *"Who is Barack Obama?"* can be answered by combining information from multiple sources such as *"He served as the 44th president of the U.S. from 2009-2017"* and *"He was a member of the Democratic Party, and served as a U.S. senator from 2005-2008"*.

- Ambiguous or under-specified questions that lack a single correct answer, or opinion-seeking questions, can benefit from multiple references. For example, the question *"When is the next world cup?"* could be correctly answered by *"The next FIFA football world cup is in 2026"* and *"The next ICC cricket world cup is in 2023 in India"* since the question does not specify the sport.

One of the primary innovations of SQuARe is the incorporation of multiple reference answers to improve evaluation accuracy. Example 4.2 shows how multiple references capture diverse yet correct phrasings:

Negative Reference Answers Utilizing the information and semantics from incorrect answers can refine the accuracy of automatic QA evaluation. For example, given the question *"Which movies of Dwayne Johnson were released in 2017?"* and the positive reference *"Dwayne 'The Rock' Johnson starrer Baywatch premiered in 2017"*, both answers *"Baywatch and Jungle Cruise"* and *"The Fate of the Furious and Baywatch"* seem plausible. However, incorporating the incorrect reference *"Jungle Cruise is a movie starring The Rock and Emily Blunt that was released in 2021"* enables the evaluation system to discern that the latter answer is less accurate. Many QA datasets (e.g., ASNQ, WikiQA, TREC-QA) contain numerous negatively labeled answer candidates that can be leveraged in evaluation.

Example 4.2 illustrates how SQuARe leverages negative references to reduce overestimation:

Example 4.2.2: Negative Reference Evaluation

Question: When was the Eiffel Tower built?

Correct Reference:

The Eiffel Tower was completed in 1889.

Incorrect Reference:

The Eiffel Tower was built in 1920.

System Output:

The Eiffel Tower was built in 1921.

Score w/ Only Correct Reference: Moderately high (due to lexical overlap).
Score w/ Negative References (SQuARe): Significantly lower (penalizes factual errors).

By explicitly incorporating negative references, SQuARe can penalize misleading responses that share vocabulary with the correct reference but diverge factually. This contrastive approach aligns more closely with human evaluation standards.

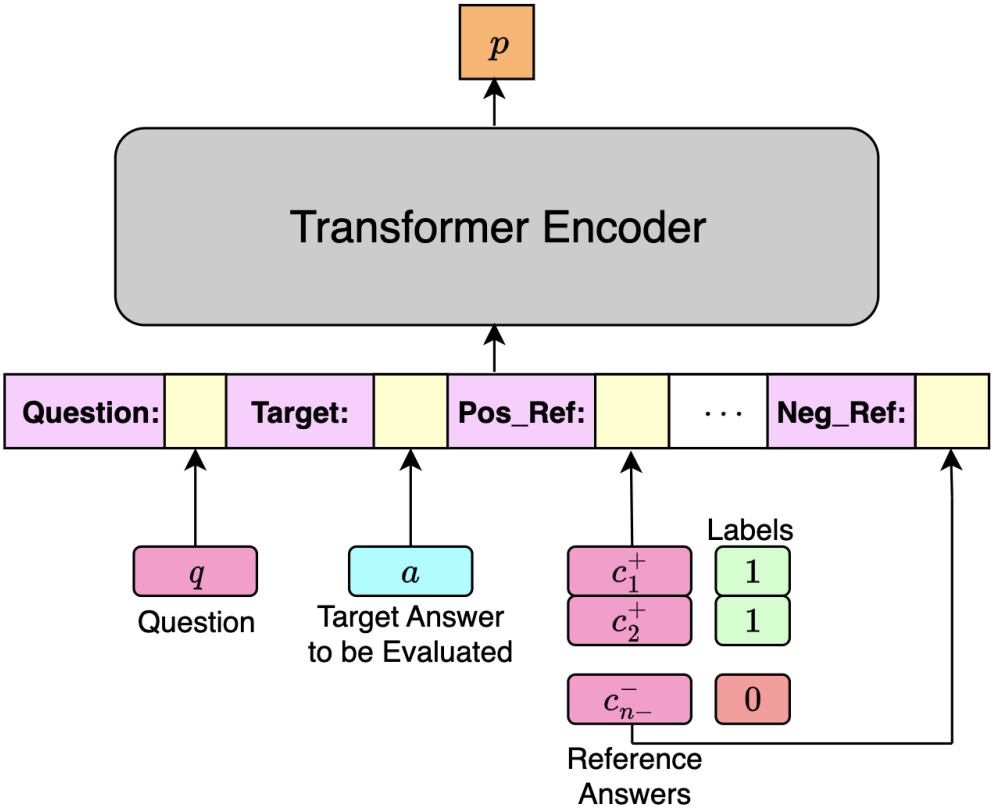

Figure 4.1: Multi-reference SQuArE employs multiple positive and negative reference answers to evaluate the correctness of a target answer for a particular question, producing a score $s \in [0, 1]$.

SQuArE To address the aforementioned limitations, we enhance the context c in the automatic evaluation function $f(q, a, c) \rightarrow p$ by including a combination of n_+ correct and n_- incorrect reference answers, denoted as $c : c^+ = \{c_1^+, ..., c_{n_+}^+\} \cup c^- = \{c_1^-, ..., c_{n_-}^-\}$. During supervised learning, SQuArE minimizes the semantic distance between a correct target answer and the set of correct references c^+, while maximizing the semantic distance from the set of incorrect references c^-. We prefix a prompt (Pos_Ref / Neg_Ref) to each reference to indicate its correctness or incorrectness. Specifically, a (q, a, c^+, c^-) input for SQuArE is encoded as "Question: q Target: a Pos_Ref: $c_1^+ \cdots$ Pos_Ref: $c_{n_+}^+$ Neg_Ref: $c_1^- \cdots$ Neg_Ref: $c_{n_-}^-$ " as illustrated in Figure 4.1.

Selecting reference answers can introduce biases in automatic QA evaluation. Collecting a diverse set of reference answers that comprehensively cover all necessary concepts is challenging and costly. Therefore, we utilize existing annotated answer candidates (both positive and negative) in high-quality labeled datasets as references. Extending automatic QA evaluation to unseen questions (without any references) remains a challenging open problem in NLP QA.

4.3 Experiments and Results

In this section, we detail the experimental setup including datasets, models, baselines, and performance results on both extractive and generative QA tasks.

4.3.1 Datasets

GenQA-MTURK This dataset contains 3,000 questions from three datasets (1,000 each): MS-MARCO (Nguyen et al., 2016), WikiQA, and TREC-QA, which we evaluate using GenQA models as described in (Hsu et al., 2021; Gabburo et al., 2022). For each question, we generate an answer using eight different GenQA models based on T5-Large. We derive this dataset from a reduced sample of the MS-MARCO NLG development set, the WikiQA test set, and the TREC-QA test set, following the approach in (Zhang et al., 2022b).

We train eight different T5-Large models on MS-MARCO using training techniques from (Hsu et al., 2021) and (Gabburo et al., 2022): (i) Supervised GenQA, (ii) Weak Supervision (WS), (iii) WS+Loss Weighting (LW), (iv) WS+Score-conditioned Input (SCI), (v) WS+Score-conditioned Output (SCO), (vi) WS+SCI+SCO, (vii) WS+LW+SCI+SCO, and (viii) Supervised GenQA+WS+LW+SCI+SCO.

We annotate all answers in this dataset for correctness using MTurk, obtaining five independent annotations for each QA pair and applying majority voting to determine the final annotation.

4.3.2 Models and Baselines

We use DeBERTaV3-Large (He et al., 2021) for SQuARe and compare it with three baselines (proposed in AVA/BEM):

- **Question-Target (QT):** takes a question and the target answer;
- **Target-Reference (TR):** takes a reference GS answer and the target answer;
- **Target-Question-Reference (TQR):** takes a question, the target answer, and a reference GS answer.

In our experiments, we set the total number of references per question to $(n_+) + (n_-) = 5$.

We train SQuARe starting from the DeBERTaV3-Large model (He et al., 2021), following the approach in (Gabburo et al., 2023a). We test various parameters and find the best combination: training the model for 20 epochs on each dataset using a batch size of 32, fp32, and Adam (Kingma and Ba, 2015) as the optimizer with a learning rate of 1e-06. At the beginning of each epoch, we shuffle the training set and select the best checkpoint based on the highest AUROC (Area Under the Curve) on the validation set. We train QT, TQR, and TR using the parameters described in (Vu and Moschitti, 2021a).

Dataset	Technique	# Refs	Accuracy	AUROC	Correlation
Answer Sentence Selection (AS2)					
WQA	AVA-TR	1	0.734	0.809	0.716
	AVA-QT	0	0.790	0.851	0.750
	AVA-TQR	1	0.809	0.873	0.771
	SQuArE	5	**0.833**	**0.896**	**0.793**
IQAD	AVA-TR	1	Baseline	Baseline	Baseline
	AVA-QT	0	+1.94%	-0.393%	+0.682%
	AVA-TQR	1	+8.02%	+5.7%	+6.178%
	SQuArE	5	**+22.24%**	**+14.01%**	**+16.062%**
Answer Generation (GenQA)					
MS-MARCO	AVA-TR	1	0.882	0.768	0.610
	AVA-QT	0	0.882	0.777	0.623
	AVA-TQR	1	0.878	0.790	**0.636**
	SQuArE	5	**0.895**	**0.832**	0.629

Table 4.1: Results on WQA, IQAD, MS-MARCO measured using Accuracy, Area under the curve and Pearson Correlation with gold labels. Results on IQAD are relative to AVA-TR baseline (due to data being internal). # Refs refers to the total number of reference answers used for the metric.

Dataset	Technique	Accuracy	AUROC	Correlation
Answer Sentence Selection (AS2)				
WikiQA	AVA-TR	0.701	0.633	0.532
	AVA-QT	0.900	0.804	0.637
	AVA-TQR	0.903	0.805	0.632
	SQuArE	**0.919**	**0.851**	**0.676**
TrecQA	AVA-TR	0.911	0.913	0.816
	AVA-QT	0.885	0.927	0.737
	AVA-TQR	0.906	**0.972**	0.797
	SQuArE	**0.924**	0.969	**0.842**
Answer Generation (GenQA)				
MS-MARCO	AVA-TR	0.843	0.683	0.587
	AVA-QT	0.772	0.693	0.580
	AVA-TQR	0.839	0.738	0.601
	SQuArE	**0.845**	**0.773**	**0.620**
WikiQA	AVA-TR	0.692	0.670	0.602
	AVA-QT	0.627	0.798	0.667
	AVA-TQR	0.671	0.811	0.678
	SQuArE	**0.694**	**0.819**	**0.690**
TrecQA	AVA-TR	0.847	0.784	0.615
	AVA-QT	0.709	0.816	0.612
	AVA-TQR	0.779	**0.857**	0.647
	SQuArE	**0.890**	0.818	**0.671**

Table 4.2: Zero-shot evaluation using QA evaluation models trained on WQA. Same metrics used as Table 4.1.

We also compare SQuARe with two additional baselines: (i) **BEM** (Bulian et al., 2022), a recently released reference-based automatic evaluation metric (trained on the AE dataset), and (ii) a large language model (**LLM**) approach using two versions of the Falcon (Almazrouei et al., 2023) model. To ensure a fair comparison, we evaluate the WikiQA and TREC-QA datasets in the zero-shot setting and, after fine-tuning the AE dataset,

For BEM, we use the evaluation script and checkpoints from the original paper[1], and for the LLM-based Falcon experiments, we use the initially released checkpoints[2][3]. Both approaches (BEM and LLM) employ the *AVA-TQR* data set, where we provide the question and a positive reference and ask the model to evaluate the correctness of the target answer. We fine-tune SQuARe on the AE dataset using the same parameters as for WQA.

4.3.3 Results

We present results comparing SQuArE with the baselines on large datasets (from both extractive QA: AS2 and generative QA: GenQA) in Table 4.3. Using human-annotated gold standards, we compute accuracy, AUROC, and Pearson correlation for each QA metric. On all datasets, SQuArE significantly outperforms the baselines and achieves the highest accuracy, as does AUROC with human annotations.

Results for Answer Sentence Selection (AS2) and Answer Generation (GenQA) : Table 4.3 showcases the results of various evaluation metrics across multiple datasets, including WQA, IQAD, and MS-MARCO. For Answer Sentence Selection (AS2) tasks, SQuArE outperforms the AVA-TR, AVA-QT, and AVA-TQR techniques across all metrics (Accuracy, AUROC, and Correlation). Specifically, for the WQA dataset, SQuArE achieves an accuracy of 0.833, an AUROC of 0.896, and a correlation of 0.793, all of which are higher than the other methods.

In the case of the IQAD dataset, SQuArE shows a significant improvement over the baseline, with an increase of +22.24% in accuracy, +14.01% in AUROC, and +16.062% in correlation. This indicates that SQuArE is particularly effective in leveraging multiple references to enhance evaluation accuracy.

For Answer Generation (GenQA) tasks on the MS-MARCO dataset, SQuArE again demonstrates superior performance with an accuracy of 0.895, an AUROC of 0.832, and a correlation of 0.629. These results highlight the robustness of SQuArE in both extractive and generative QA evaluations.

Zero-shot Evaluation Performance Table 4.4 presents the zero-shot evaluation performance of various techniques across different datasets. The results indicate that SQuArE consistently outperforms the AVA-based techniques in both Answer Sentence Selection (AS2) and Answer Generation (GenQA) tasks.

[1] https://github.com/google-research-datasets/answer-equivalence-dataset
[2] https://huggingface.co/tiiuae/falcon-7b-instruct
[3] https://huggingface.co/tiiuae/falcon-40b-instruct

Dataset	Technique	# Refs	Accuracy	AUROC	Correlation
Answer Sentence Selection (AS2)					
WQA	AVA-TR	1	0.734	0.809	0.716
	AVA-QT	0	0.790	0.851	0.750
	AVA-TQR	1	0.809	0.873	0.771
	SQuArE	5	**0.833**	**0.896**	**0.793**
IQAD	AVA-TR	1	Baseline	Baseline	Baseline
	AVA-QT	0	+1.94%	-0.393%	+0.682%
	AVA-TQR	1	+8.02%	+5.7%	+6.178%
	SQuArE	5	**+22.24%**	**+14.01%**	**+16.062%**
Answer Generation (GenQA)					
MS-MARCO	AVA-TR	1	0.882	0.768	0.610
	AVA-QT	0	0.882	0.777	0.623
	AVA-TQR	1	0.878	0.790	**0.636**
	SQuArE	5	**0.895**	**0.832**	0.629

Table 4.3: Results on WQA, IQAD, MS-MARCO measured using Accuracy, Area under the curve, and Pearson Correlation with gold labels. Results on IQAD are relative to the AVA-TR baseline (due to data being internal). # Refs refers to the total number of reference answers used for the metric.

For instance, in the WikiQA dataset, SQuArE achieves the highest accuracy (0.919) and AUROC (0.851), significantly surpassing AVA-TR, AVA-QT, and AVA-TQR. Similarly, for TrecQA, SQuArE demonstrates superior performance with an accuracy of 0.924 and an AUROC of 0.969, highlighting its robustness in zero-shot settings.

In the context of Answer Generation, SQuArE shows marked improvements in both MS-MARCO and WikiQA datasets. For MS-MARCO, SQuArE achieves the highest accuracy (0.845) and AUROC (0.773), indicating its effectiveness in evaluating generative QA models.

Comparison with BEM and LLM Baselines Table 4.5 compares SQuArE with the BEM and Falcon (LLM) baselines on the WikiQA, TrecQA, and AE datasets. In the WikiQA dataset, while Falcon-40B achieves the highest accuracy (0.963), SQuArE provides a more balanced performance with a competitive accuracy (0.919) and the highest AUROC (0.851). This demonstrates SQuArE's ability to maintain high evaluation quality while being robust across different metrics.

For the TrecQA dataset, SQuArE outperforms both BEM and Falcon-40B, with an accuracy of 0.924 and an AUROC of 0.969. This highlights the effectiveness of incorporating multiple references in SQuArE for more accurate QA evaluation.

On the AE dataset, SQuArE (fine-tuned on AE) surpasses BEM in both accuracy (0.908) and AUROC (0.966), further validating the advantage of using a diverse set of

Dataset	Technique	Accuracy	AUROC	Correlation
Answer Sentence Selection (AS2)				
WikiQA	AVA-TR	0.701	0.633	0.532
	AVA-QT	0.900	0.804	0.637
	AVA-TQR	0.903	0.805	0.632
	SQuArE	**0.919**	**0.851**	**0.676**
TrecQA	AVA-TR	0.911	0.913	0.816
	AVA-QT	0.885	0.927	0.737
	AVA-TQR	0.906	**0.972**	0.797
	SQuArE	**0.924**	0.969	**0.842**
Answer Generation (GenQA)				
MS-MARCO	AVA-TR	0.843	0.683	0.587
	AVA-QT	0.772	0.693	0.580
	AVA-TQR	0.839	0.738	0.601
	SQuArE	**0.845**	**0.773**	**0.620**
WikiQA	AVA-TR	0.692	0.670	0.602
	AVA-QT	0.627	0.798	0.667
	AVA-TQR	0.671	0.811	0.678
	SQuArE	**0.694**	**0.819**	**0.690**
TrecQA	AVA-TR	0.847	0.784	0.615
	AVA-QT	0.709	0.816	0.612
	AVA-TQR	0.779	**0.857**	0.647
	SQuArE	**0.890**	0.818	**0.671**

Table 4.4: Zero-shot evaluation using QA evaluation models trained on WQA. Same metrics used as Table 4.3.

reference answers.

Comparison with Text Similarity Metrics Table 4.6 provides the Pearson correlation of SQuArE, BLEURT, and BERTScore with human annotations on the GenQA-MTURK dataset. SQuArE demonstrates higher correlation with human evaluations compared to BLEURT and BERTScore across all datasets (MS-MARCO, WikiQA, and TrecQA). This underscores the effectiveness of SQuArE in providing a more human-aligned evaluation metric for generative QA models.

System-wise Evaluation Table 4.7 presents the system-wise evaluation on three datasets (MS-MARCO, WikiQA, TREC-QA) using six different systems based on generative question-answering models trained on MS-MARCO. The table compares the performance of SQuArE with BLEURT and BERTScore, demonstrating that SQuArE

Dataset	Approach	# Refs	Accuracy	AUROC
WikiQA	BEM	1	0.863	0.553
	Falcon-7B	1	0.081	0.448
	Falcon-40B	1	**0.963**	0.499
	SQuArE	5	0.919	**0.851**
TrecQA	BEM	1	0.866	0.819
	Falcon-7B	1	0.601	0.529
	Falcon-40B	1	0.848	0.509
	SQuArE	5	**0.924**	**0.969**
AE	BEM	1	0.897	0.959
	SQuArE	1	0.572	0.718
	SQuArE(AE)	1	**0.908**	**0.966**

Table 4.5: Comparing SQuArE against BEM and LLM baselines on the WikiQA, TrecQA, and AE datasets. The BEM baseline is trained on the AE dataset. We use the same metrics as Table 4.3.

Dataset	SQuArE	BLEURT	BERTScore
MS-MARCO	**0.238**	0.142	0.168
WikiQA	**0.425**	0.219	0.233
TrecQA	**0.862**	0.341	0.646

Table 4.6: Pearson Correlation of evaluation metrics with human annotations on GenQA-MTURK.

achieves the highest accuracy and correlation with human annotations across most systems and datasets.

Ablation Studies Table 4.8 provides the results of ablation studies that evaluate the benefits of using negative references and the impact of the number of references on the performance of SQuArE. The table includes comparisons of various configurations, such as AVA-TQR(-) using only negative references and SQuArE(+) using only positive references. The results indicate that the combination of both positive and negative references in SQuArE yields the highest performance across all metrics (accuracy, AUROC, and correlation).

The study also explores the impact of varying the number of references, showing that using five references results in the best performance, with an accuracy of 0.833, AUROC of 0.896, and a correlation of 0.793. This demonstrates that a higher number of diverse references enhances the evaluation quality of QA systems.

Dataset	Sys	Accuracy	SQuArE	BLEURT	BERTScore
MS-MARCO	1	0.946	0.881	0.772	0.732
	2	0.901	0.913	0.578	0.486
	3	0.958	0.779	0.569	0.485
	4	0.878	0.781	0.578	0.496
	5	0.880	0.761	0.689	0.620
	6	0.906	0.768	0.714	0.658
WikiQA	1	0.804	0.513	0.514	0.406
	2	0.803	0.316	0.447	0.331
	3	0.826	0.323	0.491	0.360
	4	0.791	0.306	0.480	0.350
	5	0.804	0.557	0.669	0.623
	6	0.837	0.609	0.579	0.510
TrecQA	1	0.874	0.769	0.563	0.324
	2	0.976	0.830	0.557	0.409
	3	0.968	0.932	0.482	0.498
	4	0.871	0.643	0.491	0.420
	5	0.968	0.868	0.628	0.442
	6	0.976	0.917	0.580	0.595

Table 4.7: System-wise evaluation on MS-MARCO, WikiQA, and TREC-QA using six different systems based on generative question answering models trained on MS-MARCO. The accuracy column reflects manual annotations by professional annotators, while the remaining columns show results computed using SQuArE, BLEURT, and BERTScore.

Technique	# Refs	Accuracy	AUROC	Correlation
AVA-TQR(-)	1	0.800	0.864	0.763
SQuArE(+)	5	0.815	0.885	0.783
SQuArE	3	0.821	0.889	0.787
SQuArE	[1,5]	0.820	0.889	0.786
SQuArE	5	**0.833**	**0.896**	**0.793**

Table 4.8: Ablation studies evaluating the benefits of using negative references and the impact of a number of references on the performance of SQuArE. AVA-TQR(-) and SQuArE(+) refer to an AVA model that only uses negative references and a SQuArE model that only uses positive references. # Refs is the total number of references used for the metric. [1,5] refers to the number of references being randomly sampled $\in[1, 5]$.

4.4 Qualitative Error Analysis

While SQuARe significantly improves evaluation accuracy, it is instructive to examine common error types in traditional metrics and how SQuARe mitigates them. In the following, we break the analysis into smaller examples, each followed by a brief commentary, to illustrate the limitations of traditional metrics and the corresponding improvements offered by SQuARe.

Example 1: Lexical Overlap Bias. Traditional metrics like BLEU and ROUGE rely on exact n-gram matching and often fail to recognize valid paraphrases. Consider the following example:

Example 4.4.1: Lexical Overlap Bias Example

Question: What is a band?

Reference:

A band is a group of people who perform music.

System Output:

A band is a group of musicians.

BLEU Score: Low (due to n-gram mismatch).
SQuARe Score: High (correctly captures semantic equivalence).

This example shows that while traditional metrics penalize acceptable lexical variations, SQuARe overcomes this limitation by using contextual embeddings to capture semantic similarity.

Example 2: Factual Hallucination. Another error occurs when outputs that are lexically similar to the reference still contain factual inaccuracies.

Example 4.4.2: Factual Hallucination Example

Question: When was the Eiffel Tower built?

Reference:

The Eiffel Tower was completed in 1889.

System Output:

The Eiffel Tower was built in 1921.

BLEU Score: Moderately high (due to lexical overlap).
SQuARe Score: Low (penalizes factual inaccuracy via negative references).

Here, although the system output shares lexical similarity with the reference, it contains a factual error. SQuARe's use of negative references ensures such errors are appropriately penalized.

Example 3: Reference Variability. Single-reference setups may undercount valid alternative answers. This example illustrates the benefit of incorporating multiple references.

Example 4.4.3: Reference Variability Example

Question: Who is Barack Obama?

Single Reference:

Barack Obama served as the 44th president of the United States.

Multiple References:

1) Barack Obama was the 44th president of the United States.
2) Barack Obama held office from 2009 to 2017.

System Output:

Barack Obama served as the 44th president of the United States.

BLEU Score: Low (if only one reference is used and phrasing differs).
SQuARe Score: High (by integrating multiple valid references).

Using multiple references, SQuARe better captures the range of acceptable answers, addressing the variability that single-reference evaluations miss.

Example 4: Context Ignorance. Traditional metrics often ignore the discourse-level context and pragmatic nuances of a response. The following example demonstrates this issue:

Although the system output shares some key terms with the reference, it omits important contextual details. SQuARe evaluates full answer structures to ensure that coherence and pragmatic relevance are rewarded. Overall, these examples illustrate how SQuARe addresses the limitations of traditional metrics, mitigating lexical overlap bias, factual hallucination, reference variability, and context ignorance, thereby achieving evaluations that more closely mirror human judgment.

Example 4.4.4: Context Ignorance Example

Question: Describe the process of photosynthesis.

Reference:

> Photosynthesis is the process by which green plants and some other organisms use sunlight to synthesize nutrients from carbon dioxide and water.

System Output:

> Photosynthesis converts sunlight into energy.

BLEU Score: High (due to shared keywords).
SQuARe Score: Low (penalizes the lack of contextual detail).

4.4.1 Metrics Comparison

To further illustrate the superiority of SQuARe over traditional metrics, we provide a series of explicit examples using the `examplebox` environment. Each example is followed by a brief comment that connects it to the next, highlighting how BLEU/ROUGE can misjudge answer quality while SQuARe succeeds.

Example 4.4.5: Capital of Australia Evaluation

Question: What is the capital of Australia?

Short Reference:

> Canberra.

Long Reference:

> Canberra is the capital city of Australia, located in the Australian Capital Territory and home to many national institutions.

System Output:

> The capital of Australia is Canberra.

BLEU Score with Short Reference: Low (due to insufficient n-gram overlap).
BLEU Score with Long Reference: Moderate (improved overlap with more lexical content).
SQuARe Score: High (robust to variations in reference length).

This example demonstrates that BLEU is sensitive to the length and detail of the reference; a short reference may yield a low BLEU score despite the output being correct, whereas SQuARe remains robust regardless of reference length.

Example 4.4.6: Eiffel Tower Construction Evaluation

Question: When was the Eiffel Tower built?

Positive Reference:

> The Eiffel Tower was completed in 1889.

Negative Reference:

> The Eiffel Tower was built in 1920.

System Output:

> The Eiffel Tower was built in 1921.

BLEU Score: High (due to lexical overlap with the positive reference).
SQuARe Score: Low (penalizes the factual inaccuracy by incorporating the negative reference).

In this case, BLEU fails to capture the factual error because of lexical similarity with the positive reference, while SQuARe correctly lowers the score by leveraging the negative reference, thus highlighting its robustness.

Example 4.4.7: Barack Obama Evaluation

Question: Who is Barack Obama?

Reference:

> Barack Obama was the 44th president of the United States from 2009 to 2017.

System Output:

> Barack Obama served as America's 44th president between 2009 and 2017.

BLEU Score: Low (due to differences in phrasing and word choices).
SQuARe Score: High (recognizes the semantic equivalence despite surface differences).

This example shows that even though the system output conveys the same meaning as the reference, BLEU may penalize it for differing wording. In contrast, SQuARe accurately evaluates the response based on its semantics.

Example 4.4.8: Band Definition Evaluation

Question: What is a band?

Reference:

A band is a group of people who perform music.

System Output:

A band is a group of musicians.

BLEU Score: Moderate (due to missing specific lexical cues).
SQuARe Score: High (correctly recognizes the semantic equivalence).

This final example, similar to the previous one, underscores that SQuARe is robust to paraphrasing variations, while BLEU may underperform due to its sensitivity to the exact wording.

These explicit examples underscore how traditional metrics, limited by their reliance on n-gram overlap and single-reference comparisons, can misjudge answer quality. This sets the stage for a detailed analysis of the standard evaluation errors that motivate the design of SQuARe.

Error Analysis

Traditional evaluation metrics such as BLEU and ROUGE rely on exact n-gram matching, which can lead to several shortcomings when assessing generated text. For example, as shown in Example 4.4.1, a focus on lexical overlap may cause valid paraphrases to be overlooked. In that instance, while the metric misses the semantic similarity, SQuARe overcomes this limitation by using transformer-based embeddings that capture deeper meaning.

Example 4.4.9: Lexical Overlap Bias Analysis

Issue: Metrics like BLEU/ROUGE focus on exact n-gram matching and fail to recognize valid paraphrases.

SQuARe Mitigation: Uses transformer-based embeddings to capture semantic similarity, accounting for paraphrases.

Another issue arises when outputs contain factual inaccuracies yet still score highly because of their lexical resemblance to the reference. This is evident in Example 4.4.1, where even with factual errors the traditional metric gives a high score. SQuARe mitigates this problem by incorporating negative references that penalize outputs with incorrect facts.

Example 4.4.10: Factual Hallucination Analysis

Issue: System outputs may include factual errors but still score high if lexically similar.

SQuARe Mitigation: Incorporates negative references to penalize outputs that are factually incorrect despite the lexical overlap.

The challenge of reference variability is also significant. As illustrated in Example 4.4.1, relying on a single reference can fail to recognize multiple valid responses, often resulting in unduly low scores for correct answers. By using multiple references, SQuARe enables a more robust evaluation that acknowledges the diversity of acceptable answers.

Example 4.4.11: Reference Variability Analysis

Issue: Single-reference evaluation cannot capture multiple correct answers, leading to low scores for valid outputs.

SQuARe Mitigation: Multiple references enable robust evaluation across diverse yet correct responses.

Lastly, traditional metrics often neglect the importance of context and pragmatic relevance. Example 4.4.1 demonstrates that evaluating responses solely based on isolated sentence structures can ignore discourse-level coherence. In contrast, SQuARe evaluates complete sentence structures to ensure that contextual and pragmatic aspects are properly rewarded.

Example 4.4.12: Context Ignorance Analysis

Issue: Traditional metrics ignore discourse-level context and pragmatic relevance.

SQuARe Mitigation: Evaluates full sentence structures to ensure coherence and contextual relevance are rewarded.

Together, these examples reveal that the limitations of conventional metrics ranging from an overemphasis on surface-level lexical features to a failure in capturing context and factual accuracy directly motivate the design choices behind SQuARe. By emphasizing semantic coherence, pragmatic entailment, and discourse-level relevance, SQuARe aligns more closely with human judgment and proves particularly effective for generative QA tasks. This analysis highlights specific limitations of traditional metrics, such as their inability to account for paraphrasing, factual inconsistencies, and context variability. These shortcomings directly motivate SQuARe's design choices, which are further supported by key linguistic principles:

- **Semantic Coherence:** SQuARe captures the intended meaning and logical progression by analyzing complete sentence structures.

- **Pragmatics and Entailment:** This model evaluates whether the generated response logically follows the question, noticing details that n-gram metrics may overlook.
- **Discourse-Level Relevance:** SQuARe capitalizes on multiple references to model the variability of human language, ensuring that diverse yet accurate responses are properly recognized.

These principles enable SQuARe to evaluate answer quality in a manner that closely mirrors human judgment, making it particularly effective for generative QA tasks.

4.5 Conclusion and Discussion

In this chapter, we presented SQuARe, a novel automatic evaluation metric for question-answering (QA) systems that leverages multiple reference answers, both positive and negative, to provide a comprehensive assessment of QA performance. Through extensive experiments and comparisons with existing techniques, including BLEU, ROUGE, AVA, BEM, and Falcon-based LLM approaches, SQuARe demonstrated superior performance across various datasets and evaluation settings.

Our explicit examples (see Examples 4.2 and 4.2) illustrate why SQuARe outperforms traditional metrics, particularly in capturing valid paraphrases and penalizing factual errors. The expanded error analysis (Sec. 4.4.1) categorizes common issues with traditional metrics and shows how SQuARe's integration of multiple and negative references mitigates these shortcomings. By capturing semantic similarity, discourse coherence, pragmatic relevance, and entailment, SQuARe provides an evaluation that closely aligns with human judgment, addressing challenges that surface-level n-gram metrics cannot overcome.

By addressing the limitations of existing evaluation metrics through explicit examples, detailed error analysis, and a strong linguistic foundation, SQuARe represents a significant advancement in the automatic evaluation of QA systems. These contributions pave the way for more accurate and reliable QA model assessments in the field of Natural Language Processing.

Chapter 5

Knowledge Transfer From AS2 to Generative QA

Generative Question Answering (GenQA) models, which provide conversational, complete, and natural-sounding answers, have become a key focus in advanced Natural Language Processing (NLP). Unlike extractive QA systems that pull a section of text from a document, generative models create full, well-structured answers that fit the context smoothly, without awkward grammar issues. These advanced models are highly valuable for chatbots, search engines, and similar applications where clarity, flow, and user engagement are top priorities.

A major challenge in building GenQA models is the lack of large, high-quality training data created by humans. Gathering these datasets is expensive because human reviewers need to carefully analyze questions, check context, and write proper responses. This process is both costly and time-consuming, making high-quality data scarce and slowing the spread of GenQA technology.

To address this problem, weak supervision methods have gained popularity. These techniques allow generative models to learn from data created by other models instead of relying entirely on human-labeled examples. One such method uses Answer Sentence Selection (AS2) systems, which rank answers based on their quality. Predictions from AS2 models can act as a substitute for human-labeled data, helping GenQA models learn effectively while reducing the need for extensive manual work.

Our approach further distinguishes generative QA from extractive methods by emphasizing the generation of syntactically well-formed, contextually complete responses with enhanced discourse coherence. By leveraging AS2 for data efficiency and aligning with linguistic principles, our method ensures fluency and logical structure.

Moreover, our methodology diverges from traditional fine-tuning approaches such as Reinforcement Learning from Human Feedback (RLHF) (Christiano et al., 2023), Proximal Policy Optimization (PPO) (Schulman et al., 2017), and Direct Preference Optimization (DPO) (Rafailov et al., 2023) by reducing direct reliance on human-in-the-loop processes. Instead, we propose a supervised learning paradigm driven by knowledge transfer across model classes. This paradigm simplifies training and enables scalable improvements by using AS2 models to generate pseudo-labeled data for generative

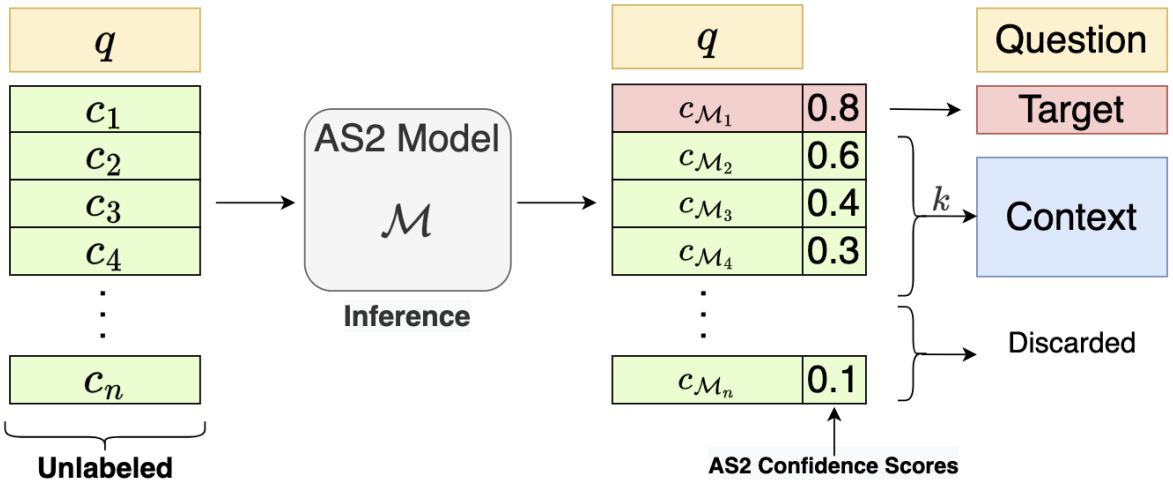

Figure 5.1: Overview of the Weak Supervision Technique

training.

To illustrate the potential of weak supervision, we propose two novel techniques:

1. **AS2 Loss Weighting:** Assigns weights to training examples based on AS2 scores, emphasizing higher-quality candidates.

2. **Structured Conditioned Input & Output (SCI/O):** Discretizes AS2 confidence scores into structured buckets, enriching both input and output representations and enhancing the model's ability to produce coherent answers.

By integrating these techniques, our framework enhances generative QA models while reducing dependence on manually annotated data. It addresses the key limitations of both AS2 and machine reading (MR) approaches. AS2 efficiently ranks candidates but may lack contextual completeness, while MR provides high precision at a high computational cost. Our approach synthesizes comprehensive answers from top-ranked candidates.

In addition, the SCI/O technique encodes linguistic features such as sentence cohesion and discourse organization. Higher-ranked AS2 candidates tend to be more semantically aligned with the question, reducing the risk of generating misleading or verbose answers. By focusing on high-confidence outputs, our approach captures essential linguistic elements (e.g., subject-verb-object relationships) and uses SCI/O labels as discourse markers, leading to more natural and coherent outputs.

Overall, our contributions, detailed in the paper *"Knowledge Transfer from Answer Ranking to Answer Generation"* (Gabburo et al., 2022), highlight the transformative potential of weak supervision in generative QA. By reducing dependency on manual annotations and exploiting large-scale unlabeled datasets, this work sets the stage for scalable, resource-efficient advancements in generative QA research and offers a practical pathway toward developing conversational, human-like QA systems capable of addressing real-world challenges.

5.1 Generative Models for Question Answering

In this section, we provide an overview of recent advances in generative QA. We first review techniques for answer generation, which focus on producing complete and natural responses, and then discuss transfer learning methods that enable efficient training of generative QA models by leveraging discriminative approaches.

5.1.1 Answer Generation for QA

Answer generation for MR has been studied by Izacard and Grave (2021); Lewis et al. (2020b), while Iida et al. (2019); Goodwin et al. (2020); Deng et al. (2020) have studied question-based summarization (QS). Asai et al. (2022b) incorporate the evidentiality of retrieved passages for training a generator, evaluated for the QA task of open-domain MR span-extraction. Xu et al. (2021) obtain extractive answer spans from a generative model by leveraging the decoder cross-attention patterns. Fajcik et al. (2021) combine a generative reader with an extractive reader to aggregate evidence from multiple passages for open-domain span-extraction.

All the previously described approaches focus on identifying short answer spans for answering questions. Research on generating complete sentences as answers (similar to answer sentences produced by extractive AS2 systems) is rarer, but includes Hsu et al. (2021), that propose a QA pipeline for GenQA (refer Fig. 5.1). This pipeline starts with an AS2 model that selects 'good' answer candidates that are then used to generate the answer. Hsu et al. learn to generate natural responses to questions using the top-ranked candidates from the AS2 model as an input context to the GenQA model. GenQA has also been explored for multilingual QA (Muller et al., 2022) by extending the answer generation approach to a multilingual setting, where the answer candidates (that are used as input to the GenQA model) can be from a mix of different languages.

In all these works, a major challenge is finding training data for effectively training GenQA models, which requires annotator-authored natural responses. In this work, we alleviate this problem by showing that it is possible to use AS2-ranked candidates to create the input context and output target for training GenQA, achieving state-of-the-art results.

5.1.2 Transfer Learning

Transfer learning is well studied in NLP, including pre-training (Devlin et al., 2019; Liu et al., 2019), multi-task learning (Luong et al., 2015a), cross-lingual transfer (Schuster et al., 2019) and domain adaptation (Gururangan et al., 2020). Our work is squarely located in this space: our underlying language models are based on pre-training for text generation (Radford and Narasimhan, 2018; Raffel et al., 2020b); our main contribution is to show that knowledge can be transferred sequentially from a ranking (discriminative) task to a generation task. Recently, Wang et al. (2022) proposed a new domain adaptation method leveraging large unlabeled datasets and a query generator model.

Izacard and Grave used retrieved text passages containing evidence to train a generative model for open-domain QA.

5.2 Knowledge Transfer Techniques

In this section, we present two methods to effectively transfer knowledge from Answer Sentence Selection (AS2) to Generative Question-Answering (GenQA). These methods, AS2 Loss Weighting and AS2 Score Conditioned I/O Shaping (SCI/O Shaping) enhance the quality and accuracy of GenQA models using weak supervision from an AS2 model.

5.2.1 AS2 Loss Weighting

Our first approach, AS2 Loss Weighting, employs the binary cross-entropy loss used for training discriminative AS2 models to improve the GenQA model's weakly supervised training loss. By modifying the loss function, we can emphasize high-quality target answers from AS2 based on their prediction scores. The AS2 prediction score, $\mathcal{M}(q, c_{M_1})$, provided by the AS2 model for the top-ranked answer candidate c_{M_1} serves as the weight for the GenQA loss term. This approach encourages the GenQA model to focus more on high-scoring, accurate target answers during training and be less influenced by lower-scoring, less precise answers.

$$\mathcal{L}_{\mathcal{M}G}(q, c_{M_1}) = \frac{1}{Z}\mathcal{M}(q, c_{M_1}) \times \mathcal{L}_G(y_r, c_{M_1}), \tag{5.1}$$

Where $\mathcal{L}_G(y_r, c_{M_1})$ is the standard loss function for generating each word in the output sequence, and Z represents the normalizing constant for AS2 scores computed on the training dataset.

The GenQA model's training setup in Equation 5.1 resembles knowledge distillation (KD), with the predictions of the AS2 model guiding the learning of the GenQA model. The main distinctions lie in the fact that the teacher (AS2) and student (GenQA) belong to different training paradigms, each using a discriminative classifier and generative models, respectively. Furthermore, standard KD techniques incorporate both supervision from the teacher (KL divergence) and supervision from the labeled data (Cross-Entropy) when teaching the student model. In contrast, our approach employs only the supervision signal from the teacher model.

To demonstrate the effect of AS2 Loss Weighting in a more structured manner, we present two simplified QA pairs in Example 5.2.1:

Example 5.2.1: Loss Weighting and AS2 Confidence

Question 1: Who directed *Inception*? **(AS2 Score: 0.98)**

Christopher Nolan directed *Inception*.

Question 2: What is the capital of France? **(AS2 Score: 0.45)**

France's main city is Paris, but many people visit Marseille.

Without AS2 Loss Weighting, the model treats both QA pairs equivalently. However, with our weighting strategy, the high-scoring pair (Question 1) contributes more to the training objective, directing the GenQA model to prioritize learning from reliable data. This reduces noise and improves generalization, especially when large unlabeled corpora are used.

5.2.2 AS2 Score Conditioned I/O Shaping

Our previous discussion on AS2 Loss Weighting dealt with using the top-ranked answer candidate's AS2 prediction score to weigh the GenQA loss term. To account for the untapped potential of other answer candidates' AS2 scores, we introduce AS2 Score Conditioned I/O Shaping.

We define a bucket function, $\mathcal{F}$, that maps the AS2 prediction score of a QA pair, $\mathcal{M}(q, a)$, to a confidence bucket label $b_i \in [b_1, \ldots, b_l]$. The function assigns a label to an answer candidate according to its score interval within the normalized interval $[0, 1]$. Now, we can prepend each input answer candidate with its respective bucket label b_j in the sequence:

$$\texttt{Input:} \quad q[SEP]b_2\ c_{M_2}[SEP]\ldots[SEP]b_{k+1}\ c_{M_{k+1}}.$$

Similarly, we prepend the target answer candidate with its bucket label b_1:

$$\texttt{Target:} \quad b_1\ c_{M_1}.$$

The prepending of bucket labels to input/output candidates enables more effective knowledge transfer, keeping the model focused on high-quality answer candidates during training. This can be solely used or combined with the AS2 Loss Weighting method. At decoding time, the generated bucket label token could also serve as a simple confidence score for answers from the GenQA model. We can also force generation to start from any of the bucket tokens and examine the effects on the GenQA model's output.

By incorporating these techniques, we enhance the GenQA model's ability to generate accurate and high-quality answers, leveraging the strengths of AS2 models through weak supervision.

In Example 5.2.2, we illustrate how confidence buckets guide the model to differentiate between high-confidence and low-confidence inputs:

By leveraging discrete confidence labels, the GenQA model more accurately handles

uncertain candidates. This structured input method can also be extended to the output side, allowing for explicit confidence tagging in the generated responses.

Example 5.2.2: Score-Conditioned Responses

Bucket Label: `Highly Confident`

Question: Who won the 2022 FIFA World Cup?

AS2 Selected Answer:

Argentina won the 2022 FIFA World Cup.

GenQA Output:

Argentina won the 2022 FIFA World Cup under coach Lionel Scaloni.

Bucket Label: `Uncertain`

Question: Which team was most popular in the 2022 World Cup?

AS2 Selected Answer:

Brazil, Argentina, and France had significant support.

GenQA Output:

Many teams had supporters, but it's unclear which was most popular.

5.3 Experiments

This section describes the experiments conducted to evaluate our proposed weak supervision techniques. We detail the datasets used, the hyperparameters selected, the computational setup, the checkpoint selection process, and the evaluation methodology.

Data and Datasets

We conducted experiments on three primary datasets: MS-MARCO, WikiQA, and TrecQA, as well as an industrial dataset for evaluation. These datasets were chosen for their diversity and relevance to QA tasks.

MS-MARCO Datasets The MS-MARCO datasets contain question-answering (QA) and natural language generation (NLG) data. They are widely used benchmarks for evaluating QA systems. We generated weakly supervised training data using a RoBERTa-Large AS2 model on ASNQ and employed the human-annotated split for evaluation. MS-MARCO QA includes 655,006 questions and 19,543,964 question-answer pairs, while MS-MARCO NLG consists of 153,725 questions and 4,694,170

Dataset	Split	# of Questions	QA Pairs
MS-MARCO QA	Train	655006	19543964
MS-MARCO QA	Dev	1000	29309
MS-MARCO NLG	Train	153725	4694170
MS-MARCO NLG	Dev	1000	28353
MS-MARCO NLG	Test	1000	28529
WikiQA	Train (clean)	857	8651
WikiQA	Dev (clean)	121	1126
WikiQA	Test	633	6165
TrecQA	Train	804	32964
TrecQA	Dev	216	9590
TrecQA	Test	340	13416

Table 5.1: Datasets statistics.

question-answer pairs. The comprehensive statistics of these datasets are reported in Table 5.1.

WikiQA and TrecQA Datasets WikiQA and TrecQA are smaller datasets used to evaluate information-seeking and passage retrieval tasks, respectively. WikiQA contains 857 questions and 8,651 question-answer pairs, whereas TrecQA consists of 804 questions and 32,964 question-answer pairs. For these datasets, we utilized the original test splits during evaluation and compared our weakly-supervised approaches, denoted as WS, **LW**, and **SC**, against an AS2 baseline and several supervised GenQA baselines using ground-truth labels.

Industrial Dataset Due to confidentiality reasons, detailed statistics of the industrial dataset, including the number of questions and question-answer pairs, are not disclosed. However, this dataset provided a real-world benchmark to assess the effectiveness of our techniques.

Hyperparameter Selection

For our experiments, we considered various combinations of hyperparameters, including the learning rate $\in \{1e^{-6}, 5e^{-5}, 1e^{-4}\}$, the number of answer candidates considered $k \in \{5, \max\}$, batch size $\in \{64, 128, 256\}$, and precision $\in \{16, 32\}$, along with both Adam (Kingma and Ba, 2015) and Adafactor (Shazeer and Stern, 2018) optimizers. We identified the best hyperparameter combination by manually evaluating model outputs on a reduced version of the MS-MARCO NLG dataset. For decoder configurations, we performed a qualitative evaluation using different parameters to select the optimal settings: beam search of 5 and ensuring the generated answer contains between 6 and 100 tokens.

Computational Setup

Our experiments were performed on 8 NVIDIA A100 GPUs (each with 40GB RAM) using Distributed Data-Parallel (DDP)[1] as our distributed training strategy. The extensive computational resources allowed us to train models efficiently, with training on MS-MARCO datasets taking approximately 6 days per model and TREC-QA and WikiQA experiments taking 4 hours each.

Checkpoint Selection using AS2 Model

We discovered that minimizing loss on the development set does not strictly correlate with high-quality answer generation according to human annotations. To address this, we experimented with an alternative measure for best model checkpoint selection. Specifically, we generated outputs for validation examples using each checkpoint and scored them with an AS2 model. We used the AS2 model's average scores as the metric for selecting the optimal model checkpoint. The differences between using the loss on the development split and the average AS2 score can be observed in Figure 5.2. Manual evaluations indicated that AS2 scores correlated better with our judgments than development set losses. Future work will explore this technique further.

5.3.1 Evaluation

We implemented manual evaluation performed by expert annotators to evaluate the model's performance. Instead of using an automated metric measure, like BLEU or ROUGE, manual annotation was more desirable since it would provide fine-grained assessments; in contrast, these metrics just provide lexical similarity assessments. The accuracy measures for this evaluation are detailed in Chapter 4.

Utilizing the Amazon Mechanical Turk framework, we facilitated the manual evaluation process for our model's output. Annotation tasks were formulated for our skilled pool of annotators (Turkers), who were required to evaluate a query, the answer produced by our model, and, when applicable (e.g., MS MARCO-NLG), a well-constructed reference answer. Annotators determined if the generated answer correctly answered the presented query for each query-answer pair (hit). They received a fee of $0.10 for each hit, and five distinct annotators were engaged. The selected annotators had approval rates exceeding 95% and verified histories of over 500 approved hits.

Comparing human evaluations to BLEU scores from GenQA model outputs, BLEU proved to be an unreliable performance metric for our task and evaluation conditions. For the BLEU evaluation, two references were used for the generated target answer: (i) the manually crafted response from MS-MARCO NLG and (ii) the top-ranked response derived from the AS2 model acting as the knowledge transfer model. In Table 5.2, we provide a comparison of the human evaluation rankings with those obtained from the BLEU metrics, revealing a significant discrepancy between the two rankings.

[1] https://pytorch.org/docs/master/generated/torch.nn.parallel.DistributedDataParallel.html

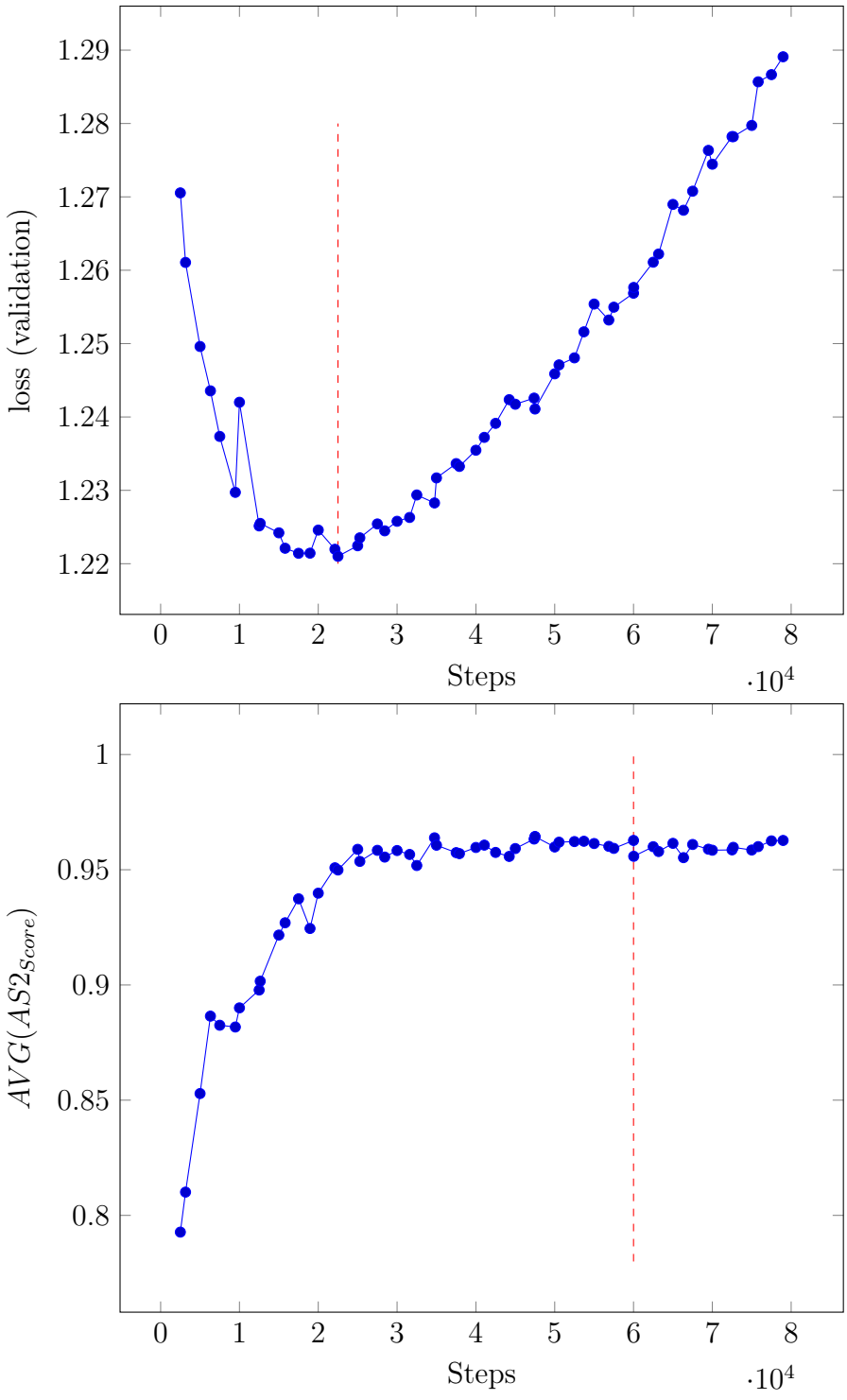

Figure 5.2: These plots show the differences between using the dev. loss and the average AS2 model score as the measure for model checkpoint selection. Notice that the vertical dashed red line is used to identify the best checkpoint (which is the one with the lowest loss in the first plot, and the one with the highest average AS2 score in the second).

Model	Transfer	Supervised	BLEU NLG answer	BLEU AS2 answer
RoBERTa	-	-	34.76	100.0
T5	**WS**	-	46.67	32.69
T5	**WS**+**LW**	-	38.47	65.55
T5	**WS**+**LW**+**SC**	-	38.85	63.76
T5	-	(Hsu et al., 2021)	36.94	61.22
T5	**WS**+**LW**+**SC**	(Hsu et al., 2021)	49.33	37.62

Table 5.2: Results using BLEU as the metric for measuring performance of answer generation. We train the GenQA models on MS-MARCO NLG for the supervised setting, and MS-MARCO QA for the knowledge transfer setting.

Additionally, we performed a qualitative evaluation of answers generated by our models, focusing on (i) the variations between the generated answer and answer candidates used as input for GenQA and (ii) the manipulation of generated answers by initiating decoding from diverse **SCO**bucket label tokens.

5.4 Quantitative Results

We perform experiments in three data settings to evaluate different features of our method. On the MS-MARCO datasets, we show that weak supervision can augment strong models trained on in-domain data. On WikiQA and TREC, we show that weak supervision of large data improves over direct supervision of small data for this QA task. We also present an experiment on a very large industrial dataset to measure the contribution of each of our proposed techniques for using unlabeled training data at scale.

5.4.1 Comparison with fully supervised approaches

These experiments aim at (i) understanding the effectiveness of our weakly supervised methods (**WS**, **LW** and **SC**) and (ii) comparing them with a GenQA state-of-the-art model, which is trained using fully supervised training data. We create weakly supervised data with a RoBERTa-Large AS2 model trained on ASNQ by applying it to the MS-MARCO QA dataset. It is important to note that, in this setting, during the training, both the student and the teacher models do not have any knowledge of the original labels of MS-MARCO QA as we consider this dataset to be unlabeled. We compare our approach against a supervised GenQA model baseline (Hsu et al., 2021), which is trained on MS-MARCO NLG. The MS-MARCO NLG dataset is much smaller than MS-MARCO QA but has higher quality answers as the targets since they are manually written. We also investigate a two-stage training strategy, by first applying

Approach	Model	Unlabeled: MS-MARCO QA		Labeled: MS-MARCO NLG		Accuracy (%)
		Used	Training Strategy (Ours)	Used	Training Strategy	
AS2	RoBERTa-Large	✗	-	✗	-	79.3
GenQA	T5 - Large	✓	**WS**	✗	-	79.9
	T5 - Large	✓	**WS + LW**	✗	-	81.5
	T5 - Large	✓	**WS + SCI**	✗	-	82.0
	T5 - Large	✓	**WS + SCO**	✗	-	82.5
	T5 - Large	✓	**WS + LW + SC**	✗	-	83.7
	T5 - Large	✗	-	✓	(Hsu et al., 2021)	82.6
	T5 - Large	✓	**WS + LW + SC**	✓	(Hsu et al., 2021)	**85.3**

Table 5.3: Results on the test split of MS MARCO-NLG for different training paradigms of GenQA models. The weak supervision is provided by a RoBERTa-Large AS2 model trained on ASNQ. We compare with a fully supervised GenQA baseline (Hsu et al., 2021) trained on the train split of MS MARCO-NLG.

our knowledge transfer approach, followed by the supervised training to understand if the two approaches are complementary or essentially capture similar information. Manual annotators evaluate all models on the same MS-MARCO NLG test set.

We present the results on the MS-MARCO NLG test set in Table 5.3. The baseline zero-shot accuracy of the AS2 model on this data is 79.3, and the baseline accuracy of the fully supervised GenQA model (Hsu et al., 2021) is 82.6. Our weak supervision (**WS**) transfer technique, which does not use any answers written by annotators as targets, shows improvements over the AS2 baseline (0.6%). This shows that our approach can transfer information learned by an AS2 model from ASNQ (a large labeled dataset) into a GenQA model.

Ablating each of the approaches (**LW**,**SCI**,**SCO**) individually in addition to **WS**, we observe consistent improvements (+1.6, +2.1 and +2.6% respectively) over the performance of **WS**, indicating that the AS2 scores can help in the knowledge transfer. Additionally, combining all the approaches with **WS** significantly improves the performance and surprisingly can even outperform the supervised GenQA baseline (by 1.1% = 83.7-82.6). This shows that the knowledge transferred by our approach from ASNQ exceeds what can be learned from MS-MARCO NLG.[2]

Finally, when we combine our weak supervised training techniques with the supervised training in a two-stage pipeline, we observe very significant performance gains, e.g., 2.7% over the supervised approach. This shows that (i) the information in MS-MARCO NLG is complementary to the knowledge transferred from ASNQ, and (ii) our approach is effective in transferring knowledge from a discriminative ranker to a downstream GenQA model.

5.4.2 Scarce Data Setting

In this experiment, we measure the quality of our weak supervision approaches by evaluating their performance on two popular AS2 benchmark datasets: WikiQA and

[2]MS-MARCO NLG is larger than ASNQ, but the latter is a much higher quality dataset in terms of diversity and complexity of questions and answer annotations

Approach	Unlabeled (MS-MARCO QA)		Labeled (WikiQA Train)		Accuracy (%)
	Used	Training Strategy	Used	Training Strategy	
AS2	✗	-	✓	Fine-tuning	78.3
GenQA	✓	**WS + LW + SC**	✗	-	78.7
	✗	-	✓	(Hsu et al., 2021)	72.9
	✓	**WS + LW + SC**	✓	(Hsu et al., 2021)	**79.8**

(a) WikiQA

Approach	Unlabeled (MS-MARCO QA)		Labeled (TREC-QA)		Accuracy (%)
	Used	Training Strategy	Used	Training Strategy	
AS2	✗	-	✓	Fine-tuning	85.9
GenQA	✓	**WS + LW + SC**	✗	-	**90.5**
	✗	-	✓	(Hsu et al., 2021)	80.7
	✓	**WS + LW + SC**	✓	(Hsu et al., 2021)	89.8

(b) TREC-QA

Table 5.4: Results on the test split of WikiQA and TREC-QA. The weak supervision is provided by RoBERTa-Large AS2 models trained respectively on WikiQA and TREC-QA. We compare with a fully supervised GenQA baseline (Hsu et al., 2021) trained respectively on the train split of WikiQA and TREC-QA using ground truth correct answers as the target for generation. We use T5-Large for all GenQA models.

TREC-QA. We train the AS2 teacher model on this data and still use the unlabeled data from MS-MARCO QA for performing the knowledge transfer. This way, we test if our approach is applicable in real scenarios where data can be scarce, and no large labeled data dataset is available (such as ASNQ or MS-MARCO NLG). Additionally, we verify if our approach works for other domains and if fine-tuning GenQA on the target domain data can help knowledge transfer in that domain, even in case of data scarcity.

We compare our weakly-supervised approaches with an AS2 baseline and a GenQA model trained on the target datasets using their ground-truth labels. We used the original test splits of the datasets. Note that for these experiments, (i) we use the best-performing strategy for our transfer learning, i.e., **WS** along with **LW** and **SC**, and (ii) the AS2 baseline is the same model that we use to transfer knowledge on the MS-MARCO QA.

From our results in Table 5.4, we make the following observations:

1. AS2 accuracy evaluated with our human annotations is around 10% lower than results from previous works, e.g., (Zhang et al., 2021). As we use the same model,[3], the difference is due to the fact that we use the 'raw' test setting, which includes questions with no correct answer candidates.

2. Our transfer learning techniques have better performance than both the AS2

[3]Starting from the https://huggingface.co/roberta-large checkpoint

Approach	Model	Unlabeled: AQAD-U		Accuracy (%)
		Used	Training Strategy	
AS2	ELECTRA-Base	✗	-	Baseline
GenQA	T5 - Base	✓	**WS**	+1.34%
	T5 - Base	✓	**WS** + **LW**	+4.08%
	T5 - Base	✓	**WS** + **LW** + **SC**	**+7.35%**

Table 5.5: Results on AQAD-L for different training paradigms of GenQA models. All results are reported in absolute % changes w.r.t the AS2 baseline.

model and the supervised GenQA baselines. For WikiQA, our knowledge transfer approach, which has only seen unlabeled MS-MARCO data and no labeled training data from WikiQA, gets higher accuracy than both the AS2 baseline (+0.4%) and a fully supervised GenQA baseline (+5.8%), which uses the ground truth labels from the target datasets.

3. We observe the same trend for TREC-QA: our weakly supervised models improve over both AS2 (4.6%) and supervised GenQA (9.8%) baselines.

4. In contrast to our observations from Table 5.3, the supervised GenQA baseline for WikiQA and TREC-QA is less accurate than the AS2 baseline. We explain this for two reasons: (a) the small size of these datasets (only a few thousand training questions) might be insufficient to train a large T5 model for GenQA, and (b) the usage of extracted answers as the target for generation instead of a human-written and natural sounding answer affects the quality of answer generation.

5. Finally, supervised fine-tuning applied after our transfer learning only improves performance on WikiQA. The WikiQA dataset has several questions without correct answers (∼40%). Fine-tuning on the supervised dataset reinforces the generator's training on questions that have actual positive labels, thereby helping to reduce noise and improving the final accuracy of the entire test set.

5.4.3 Industrial Setting

In this experiment, we aim to show that our experimental findings extend to very large-scale and real-world data, i.e., *non-representative de-identified* customer questions from Alexa virtual assistant. We use the 50M question AQAD-U corpus as the unlabeled QA corpus for training the GenQA model, transferring the knowledge of an AS2 teacher model (no human-authored answer is used for training fully-supervised GenQA models on this data). We compare our methods for weak supervision against the AS2 teacher model on the labeled test split: AQAD-L.

We present the results in Table 5.5 relative to the AS2 baseline due to the data being internal, which is used as the 'teacher' for transferring knowledge to train the GenQA model. For these experiments, we use T5-Base as the GenQA model (due to the large

size of AQAD-U), and our results show that knowledge transfer from the AS2 model using the unlabeled data surprisingly improves the accuracy of the baseline by 1.34%. This indicates that the weak supervision provided by the AS2 model is able to train a GenQA model that performs better than the AS2 teacher itself. Furthermore, using loss weighting (**LW**) and input/output shaping (**SC**) significantly improves our weak supervision approach. The T5-Base model trained using a combination of **LW** and **SC** on the unlabeled AQAD-U corpus achieves an impressive 7.35% gain in accuracy over the baseline AS2 model (which has been trained on labeled data with annotations for answer correctness). used in Natural Language Processing and other fie

Chapter 6

Reward Function for Generative QA Based on Automatic Evaluators

Question Answering (QA) is a cornerstone task in Natural Language Processing (NLP), requiring systems to retrieve, interpret, and generate accurate and contextually appropriate answers to user queries. Traditional QA systems have primarily focused on two retrieval-based approaches: (i) *Answer Sentence Selection (AS2)*, where candidate answers are ranked based on their relevance to a given question (Garg et al., 2019), and (ii) *Machine Reading (MR)*, which identifies a specific text span within a document that answers the question (Chen et al., 2017b). While these methods have shown effectiveness, they face significant limitations: the reference text may lack critical information, include distracting or irrelevant content, or express the answer in convoluted or indirect ways. Moreover, stylistic or contextual dependencies may render extracted answers unsuitable as standalone responses.

To address these challenges, researchers have shifted toward generative Question Answering (GenQA), which leverages generative models to synthesize complete, coherent, and concise answers. Unlike MR approaches, which typically target short, entity-focused answers, GenQA systems aim to generate full-sentence or paragraph-level responses, making them better suited for open-domain abstractive QA. Recent advancements in GenQA have demonstrated its potential to improve both answering accuracy and style suitability by synthesizing information from multiple sources (Hsu et al., 2021; Muller et al., 2022; Gabburo et al., 2022). However, training effective GenQA models pose a significant challenge due to the high cost and difficulty of creating large-scale, high-quality training datasets, which typically require annotators to rewrite answers into concise, natural-sounding responses.

This work investigates how automatic QA evaluation systems can be leveraged as supervision signals for training and fine-tuning GenQA models. Recent research (Vu and Moschitti, 2021a; Bulian et al., 2022) has shown that transformer-based QA evaluators can effectively assess the correctness of sentence-form answers using relatively inexpensive annotations of correctness/incorrectness for question-answer pairs. Building upon this, we propose SQuArE, an advanced QA evaluation framework that extends prior approaches by incorporating multiple reference answers and evaluating generative

model outputs. SQuArE enables a robust evaluation of QA performance, providing a foundation for innovative methods to improve GenQA training.

This Thesis introduces three novel techniques leveraging SQuArE to enhance GenQA models:

- **SQuArE-SDA (Static Data Augmentation):** Augments the training dataset by generating multiple answers for each question and retaining only high-scoring, high-quality answers as determined by SQuArE.
- **SQuArE-DDA (Dynamic Data Augmentation):** Dynamically performs data augmentation during each training epoch, adapting the augmented dataset to the evolving state of the GenQA model.
- **SQuArE-LW (Loss Weighting):** Introduces a loss weighting mechanism that prioritizes training samples associated with higher SQuArE scores, enabling the model to focus on learning from high-quality answers.

Our empirical evaluation of two academic and one industrial dataset demonstrates that these techniques significantly improve GenQA performance. Models trained with SQuArE consistently achieve higher answering accuracy and generate answers with enhanced quality, as reflected by superior SQuArE scores compared to baseline approaches.

By improving training methods for generative QA models, this work helps bridge the gap between retrieval-based and generative approaches, leading to more effective QA systems. These contributions marks a significant step toward enabling QA systems that can generate high-quality, human-like answers efficiently and effectively.

6.1 Generating and Automatically Evaluating Answers

6.1.1 Answer Generation

Several research works (Izacard and Grave, 2021; Lewis et al., 2020c) have studied the problem of generating short answer spans (typically entity level) for MR systems. The most relevant of these works for GenQA is the work of Asai et al. (2022b), which trains an answer generation model using the evidentiality of retrieved passages. Xu et al. (2021) uses decoder cross-attention patterns to generate extractive answer spans. Fajcik et al. (2021) generate answer spans by using a combination of a generative and extractive reader (aggregating information over multiple passages). An independent but related line of research is question-based summarization, and there have been several research works in this field: (Iida et al., 2019; Deng et al., 2020).

Hsu et al. (2021) proposed the first formulation for generating complete answer sentences using evidence retrieved by an answer sentence selection (AS2) model. This model was termed GenQA, and it uses the top most relevant answer sentence candidates for a question as an input context to an encoder-decoder model to generate a natural-sounding complete answer sentence for this question. Muller et al. (2022) extend GenQA for multiple languages by using answer sentence candidates from multiple

languages as input context for the GenQA model. Recently, Gabburo et al. (2022) proposed training of GenQA models using unlabeled data by leveraging weak supervision from trained AS2 ranking models. This approach was shown to combine well with the supervised GenQA approach (Hsu et al., 2021) to improve the answering accuracy. Note that all of these approaches are different from the ones described in the previous paragraph as they aim to generate complete answer sentences and not just short answer spans.

6.1.2 Evaluation of QA Systems

Token level similarity metrics like BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), METEOR (Banerjee and Lavie, 2005), etc have been shown (Reiter, 2018a) to not extend to sentence-form QA evaluation. For MR tasks, Yang et al. (2018a) adapt BLEU and ROUGE metrics but limit their evaluation to only yes-no and entity questions. Si et al. (2021) uses multiple gold reference answers (extracted from Knowledge Bases) to be used as references for evaluating answer span extraction.

There have been several learnable automatic metrics: BERTScore (Zhang* et al., 2020), COMET (Rei et al., 2020), BLEURT (Sellam et al., 2020), etc., that have been proposed for some tasks in NLP such as MT and Summarization. These are based on transformer encoder models. Chen et al. (2019) proposed extending BERTScore for MR tasks using the question and paragraph context in addition to the answer. In a similar line of work, Vu and Moschitti (2021a) propose AVA, an automatic QA evaluation metric that learns a transformer to encode the question, a reference gold answer, and the target answer to be evaluated. Very recently, Bulian et al. (2022) presented similar findings as AVA by proposing BEM, which can be used for evaluating sentence-level extractive QA (AS2). AVA and BEM have not been evaluated for GenQA systems before. Hsu et al. (2021) and Gabburo et al. (2022) show that automatic metrics like BLEU, BLEURT, and BERTScore do not correlate well with human judgments for evaluating the accuracy of GenQA systems. We extend AVA for our experiments as the automatic QA evaluation system.

6.2 Generative QA (GenQA)

Answer generation-based QA (GenQA) refers to a text generation model for generating an answer to a question. Specifically, a generation model $\mathcal{M}$ is provided a question q and some context as input and generates an answer g. Hsu et al. (2021) proposed GenQA for generating natural-sounding complete answer sentences by leveraging labeled datasets containing high-quality human-authored answers as the targets for generation.

Specifically, a dataset, $\mathcal{D}$, for training a GenQA model, $\mathcal{M}$, contains examples of the format: $\Big(q, \{a_1, a_2, \ldots, a_k\}, t\Big)$ where q is the question, $\{a_1, \ldots, a_k\}$ are the k answer candidates used as input context to $\mathcal{M}$, and t is the target output answer (GS human-authored answer).

Gabburo et al. (2022) extended this line of work by proposing a novel approach

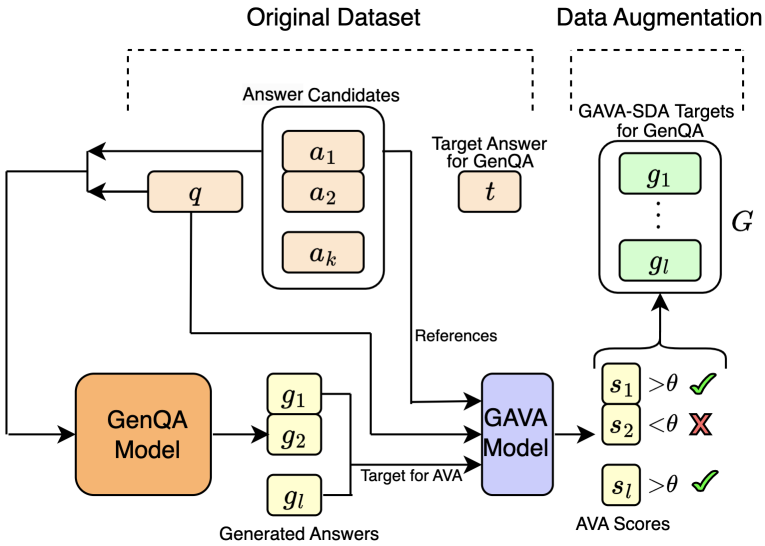

Figure 6.1: Illustrative representation of the SQuArE-Static Data Augmentation (SQuArE-SDA) approach.

to train GenQA models using unlabeled data by transferring knowledge from an AS2 model (which is used to produce silver labels). Specifically, for each question q, the AS2 model is used to rank a set of answer candidates without any labels of correctness/incorrectness. The top-ranked answer is used as the generation target for the GenQA model, while the question, along with the following k top-ranked answers, is used as the input for the GenQA model.

6.3 SQuARe for Training GenQA

In this section, we propose three approaches for training GenQA models using SQuArE. Example 6.3.1 provides an insight into how SDA improves training data quality by filtering out low-quality candidates. Although the AS2 system initially provides five candidate answers, SQuArE discards two of them, one completely off-target and one with unnecessary details, ensuring that only the best three answers are used. In this way, all the noise produced by wrong, incomplete, or verbose candidates is filtered out during the training. This targeted filtering aims to improve the quality of GenQA-generated answers by reducing distracting or misleading content.

6.3.1 Static Data Augmentation (SQuArE-SDA)

We create new training examples starting from $\mathcal{D}$, using a SQuArE model, $\mathcal{E}$, and a base GenQA model $\mathcal{M}_0$ (trained only on $\mathcal{D}$). The new examples are added to $\mathcal{D}$ to create an improved training corpus for learning an enhanced GenQA model, $\mathcal{M}$.

For every question $q \in \mathcal{D}$, along with its answer candidates $\{a_1, \ldots, a_k\}$ as input context, we apply $\mathcal{M}_0$ to generate multiple possible answers $\{g_1, g_2, \ldots, g_l\}$ using a probabilistic decoding approach (Zarrieß et al., 2021). Then, we apply the SQuArE

model to each generated answer g_i to obtain SQuArE scores s_i, i.e., $\mathcal{E}(q, g_i, \{a_1, \ldots, a_k\})$. Using a predefined threshold θ, we filter and retain only those answers $G = \{g_i : s_i \geq \theta\}$. We then use these filtered answers as alternate targets to produce new training examples for GenQA: $\Big(q, \{a_1, \ldots, a_k\}, g\Big)$, where $(q, \{a_1, \ldots, a_k\}, t) \in \mathcal{D}$ and $g \in G$. Figure 6.1 illustrates this approach.

Example 6.3.1: Example 1 – Training Data Enhancement via SDA

Question: *How do I get from Washington, DC to Alexandria, VA?*

Rank	Candidate	SQuArE Score
1	"Take the Metro."	0.96
2	"Board the Blue Line at Farragut North."	0.93
3	"Transfer at Rosslyn."	0.91
4	"Drive a taxi downtown."	0.42
5	"There are many ways; taking a bus might be an option if you're not in a hurry."	0.58

SQuArE Filtering Process: Among these five candidates, SQuArE selects only the high-scoring answers (Candidates 1, 2, and 3) for use in training. Candidate 4 is clearly irrelevant, and Candidate 5, while not completely wrong, includes extra content that does not directly answer the question.
Training Data for GenQA (after SDA): The final training instance is built using only the three positive candidates. Consequently, during training, the model learns from this refined, high-quality input set.

Generated without SDA	Generated with SDA
"There are many ways to get there you could take the Metro, drive a taxi, or even take a bus."	"Take the Metro: board the Blue Line at Farragut North and transfer at Rosslyn."
Vague	Correct

6.3.2 Dynamic Data Augmentation (SQuArE-DDA)

We extend the SQuArE-SDA approach by producing new examples at regular intervals during training, e.g., at the beginning of every epoch. This dynamic data augmentation makes the training process adaptive, as the GenQA model $\mathcal{M}$ continuously improves and generates higher quality answers, which are then filtered by SQuArE to augment subsequent training iterations.

Example 6.3.2: Example 2 – Improved Specificity via DDA

Question: *What is the recommended lifespan of a central air conditioner?*

Epoch 1 Candidate Answers:

Rank	Candidate	SQuArE Score
1	"Usually, a central air conditioner lasts 10–15 years, but proper maintenance can extend its life."	0.93
2	"About 10–15 years."	0.75
3	"A central air conditioner typically lasts around 12 years."	0.70

Dynamic Data Augmentation (DDA) Process:
At epoch 1, the generative model is trained using these three high-quality candidate answers. The model synthesizes these inputs to generate a new answer. If the generated answer achieves a high SQuArE score (e.g., 0.90), it is incorporated into the candidate set for epoch 2. This continuous distillation process cleans and enriches the training data, which is especially beneficial when the dataset is small.

Generated at epoch 1	SQuARe Score
"A central air conditioner typically lasts between 10 and 15 years, depending on usage and maintenance."	0.88

Epoch 2 Updated Candidate Answers:

Rank	Candidate	SQuArE Score
1	"Usually, a central air conditioner lasts 10–15 years, but proper maintenance can extend its life."	0.93
2	"A central air conditioner typically lasts between 10 and 15 years, depending on usage and maintenance."	0.88
3	"About 10–15 years."	0.75
4	"A central air conditioner typically lasts around 12 years."	0.70

6.3.3 Loss Weighting (SQuArE-LW)

While SQuArE-SDA and SQuArE-DDA augment training data, SQuArE-LW directly uses the SQuArE score to modify the GenQA training loss.

For training $\mathcal{M}$ with an example $(q, \{a_1, \ldots, a_k\}, t) \in \mathcal{D}$, we:

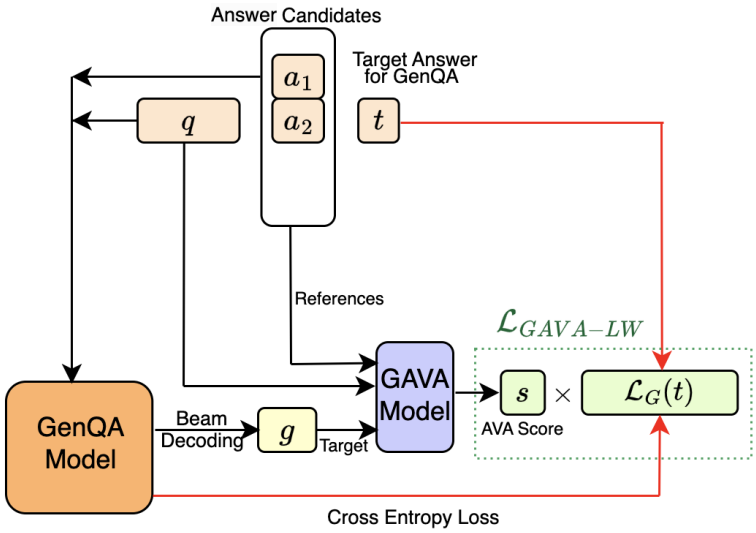

Figure 6.2: Illustrative representation of the SQuArE-Loss Weighting (SQuArE-LW) approach.

1. Compute the standard cross-entropy loss $\mathcal{L}_G(t)$ on input $(q, \{a_1, \ldots, a_k\})$ with target t.

2. Run inference on $(q, \{a_1, \ldots, a_k\})$ to obtain a generated answer g.

3. Compute the SQuArE score $\mathcal{E}(q, g, \{a_1, \ldots, a_k\})$, and weight the loss:

$$\mathcal{L}_{SQuArE-LW} = \Big(1 - \mathcal{E}(q, g, \{a_1, \ldots, a_k\})\Big) \times \mathcal{L}_G(t).$$

Additionally, we refine the input context: after obtaining the filtered set of generated answers G, we combine them with the original answers from D, forming $\mathbb{A} = \{a_1, \ldots, a_k, t\} \cup G$. We then use $\mathcal{E}$ to score $\mathbb{A}$ and select the top-ranked answer as the target for GenQA, while the following k top-ranked answers are used as context. This further improves both input quality and target answer quality. We call this combined approach Dynamic Data Augmentation (SQuArE-DDA).

This example demonstrates the effectiveness of DDA as a continuous distillation process. In epoch 1, three candidate answers with varying detail levels are used to train the model. The model then generates a refined answer that, upon receiving a high SQuArE score, is added to the candidate set in epoch 2. This dynamic refinement helps clean the dataset and improves the overall quality of the model.

By emphasizing examples with low SQuArE scores, SQuArE-LW forces the model to learn from challenging cases, thus improving generalization.

Example 6.3.3: Example 3 – Conciseness and Accuracy via LW

Question: *Who won the 2022 FIFA World Cup?*

Candidate Answer (AS2 input):

Rank	Candidate	SQuArE Score
1	"Argentina won the tournament."	0.88

Training with Loss Weighting (LW):
Although during training, the model usually generates a single answer, for this example, we obtain two outputs:

Bad Answer	Good Answer
"Argentina won the World Cup." (SQuArE score: 0.65)	"Argentina emerged as the champion of the 2022 FIFA World Cup, defeating France in the final." (SQuArE score: 0.92)

The loss is weighted as $(1 - \text{SQuArE score})$. Thus, the bad answer incurs a higher loss weight of $1 - 0.65 = 0.35$, while the good answer incurs a lower loss weight of $1 - 0.92 = 0.08$. This higher penalty for the bad answer forces the model to learn from its error.

This example illustrates the loss weighting mechanism in action. During training, the model generates an answer which is evaluated by SQuARe. A low SQuArE score (0.65), results in a high loss weight (0.35) and is thus heavily penalized. In contrast, the good answer, scoring 0.92, incurs a much lower loss weight (0.08) pushing the model toward generating more accurate and informative answers over time.

6.4 Experiments

6.4.1 Datasets

For training and evaluating our models, we consider two academic and one industrial dataset representing real-world customer questions. We used WQA, MS-MARCO, and IQAD.

6.4.2 Models and Evaluation

For our experiments, we consider two types of models: (i) SQuArE evaluation models, as described in Chapter 4, and (ii) GenQA models, using techniques described in Section 6.3. For SQuArE $\mathcal{E}$, we use a DebertaV3-Large (He et al., 2021) pre-trained model using up to $n = 5$ reference answers per question. We train two SQuArE models: one

Approach	WQA		MS-MARCO		IQAD	
	Accuracy	SQuArE-Score	Accuracy	SQuArE-Score	Accuracy	SQuArE-Score
GenQA-WS(Gabburo et al., 2022)	0.655	0.409	0.775	0.770	Baseline	Baseline
(Ours) SQuArE-SDA	**0.868**	0.498	**0.877**	**0.869**	+8.55%	-1.48%
(Ours) SQuArE-DDA	0.769	0.439	0.843	0.855	**+9.85%**	**+0.54%**
(Ours) SQuArE-LW	0.796	**0.527**	0.794	0.784	+8.81%	+0.51%

Table 6.1: Answering accuracy (manual evaluation) and AVA-Score on WQA, MS-MARCO and IQAD datasets. Results on IQAD are presented relative to the baseline (due to the data being internal). For WQA and MS-MARCO, we use an AVA model trained on WQA, and for IQAD we use an AVA model trained on IQAD. Best results for each dataset are highlighted in bold.

on WQA and one on IQAD, using the former for both public datasets. [1] For GenQA, we consider a baseline model from Gabburo et al. (2022), which is a T5-Large (Raffel et al., 2020b) encoder-decoder transformer trained using weak supervision on MS-MARCO. We use this as the baseline GenQA model $\mathcal{M}_0$, and apply all our techniques: SQuArE-SDA, SQuArE-DDA, and SQuArE-LW. Unless otherwise stated, we use $\theta = 0.9$ for both SQuArE-SDA and SQuArE-DDA. For the GenQA models, we take $k = 5$ answer candidates as inputs and select the best checkpoint based on the highest AVA-Score on the development set.

We perform human evaluation of generated answers using Amazon MTurk (5 annotations per QA pair, averaged). Turkers with an approval rate higher than 95% and more than 500 approved hits were selected. Annotators evaluated each question, the generated answer, and a correct reference answer. For each hit, turkers received \$0.10. We compute answering accuracy as the percentage of correct answers, and we also evaluate using the automatic SQuArE metric.

6.4.3 Main Results

We evaluate GenQA models trained with our three proposed techniques in Table 6.1 using human evaluation of accuracy and SQuArE-Score (automatic evaluation). Across all datasets, our approaches outperform the baseline and improve GenQA training.

Specifically for WQA, the SQuArE-SDA approach achieves the highest answering accuracy (improving by +21.3% points over the baseline). This ideal case benefits from having a SQuArE model trained on the same dataset. We also observe improvements in the SQuArE score for our approaches, although the relative ordering does not perfectly correlate with human rankings.

On MS-MARCO, SQuArE-SDA again achieves the highest answering accuracy (+10.2% over baseline), with a perfect correlation between human evaluation and SQuArE. This demonstrates the transferability of using SQuArE for training GenQA across distributions (the SQuArE model here is trained on WQA since MS-MARCO lacks human

[1] WQA contains human annotations of answer correctness, which can be used as references for training a strong SQuArE model. The sentence version of MS-MARCO does not have human annotations of answer correctness.

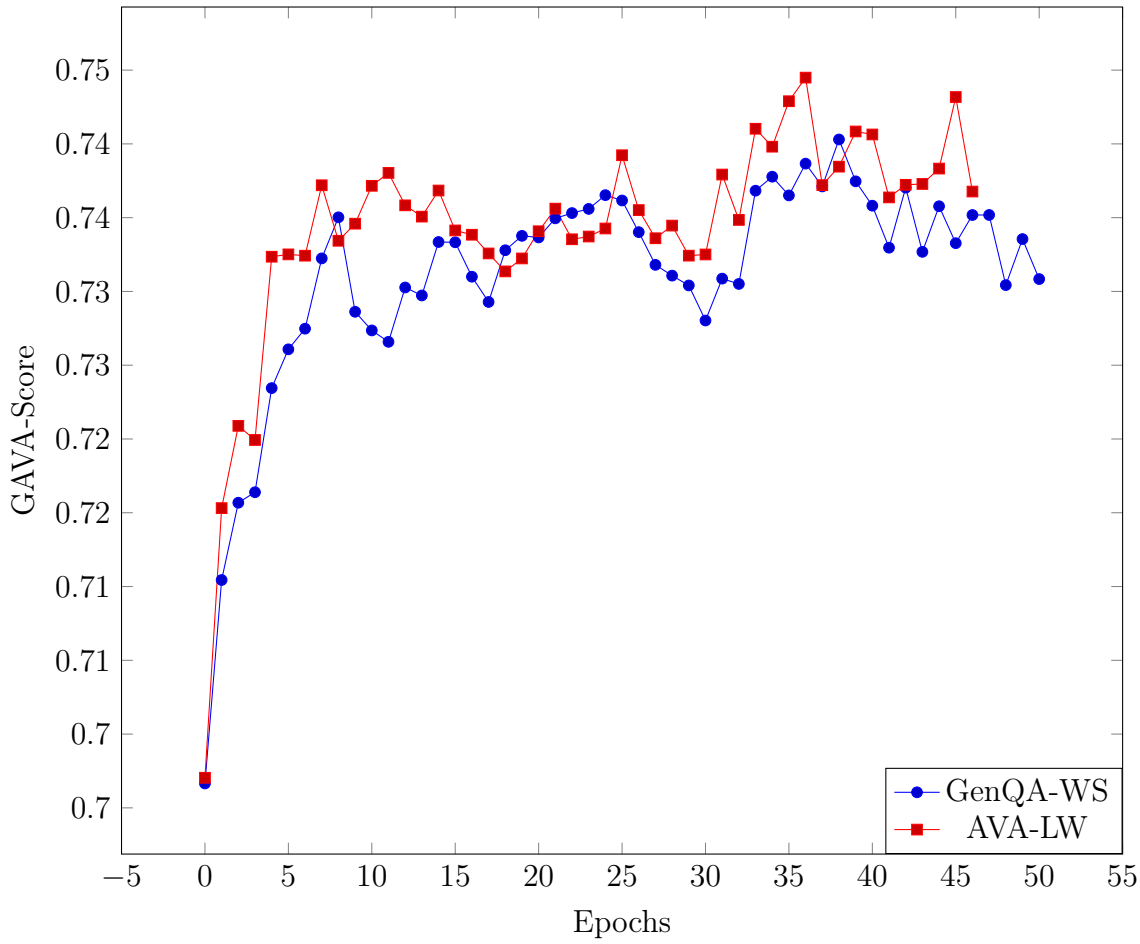

Figure 6.3: Comparison of baseline GenQA-WS and AVA-LW on WQA in terms of GAVA-Score (on validation split) varying across training epochs (GAVA-LW achieves higher GAVA-Score throughout training).

annotations).

On the industrial dataset, IQAD, the SQuArE-LW loss weighting approach achieves the highest accuracy (+9.85% relative improvement over baseline). These results support our approach in real-world customer question scenarios.

6.4.4 Analysis and Ablations

Variation of SQuArE Score over Training: To understand how the SQuArE score of our proposed techniques improves over the baseline, we plot its variation over training epochs on the MS-MARCO dataset for SQuArE-LW. As shown in Fig. 6.3, SQuArE-LW consistently achieves a higher SQuArE score than the GenQA baseline, demonstrating effective knowledge transfer.

Variation of Threshold θ for SQuArE-SDA: We study the impact of the parameter θ using values $\{0.5, 0.7, 0.9\}$ on the MS-MARCO dataset. Results in Table 6.3 and Fig. 6.4 indicate that a higher θ yields better GenQA accuracy and higher SQuArE

Approach	Manual	SQuArE-Score	BLEURT	BERTScore	BLEU
GenQA-WS	0.775	0.770	0.587	0.492	30.8
SQuArE-SDA(θ=0.5)	0.846	0.857	0.578	0.496	35.2
SQuArE-SDA(θ=0.7)	0.861	0.864	0.576	0.491	34.9
SQuArE-SDA(θ=0.9)	0.877	0.869	0.573	0.486	34.1
SQuArE-DDA	0.843	0.855	0.627	0.541	38.1
SQuArE-LW	0.794	0.784	0.627	0.502	32.2
Correlation (Pearson)		0.979	-0.429	-0.035	0.679
Correlation (Spearman's)		1	-0.812	-0.6	0.429

Table 6.2: Evaluation of different GenQA models using automatic evaluation metrics: BLEU, BLEURT, BERTScore in addition to SQuArE-Score on the MS-MARCO dataset. We present the correlation each metric achieves with human annotation. SQuArE achieves the best correlation with human evaluation of answer accuracy.

θ	| Augmented Set |	Accuracy	SQuArE-Score
0.5	91.348	0.846	0.857
0.7	84.272	0.861	0.864
0.9	69.557	**0.877**	**0.869**

Table 6.3: Variation of GenQA accuracy by changing θ for SQuArE-SDA approach, on the MS-MARCO dataset. We present human and automatic (SQuArE-Score) evaluation. | Augmented Set | indicates the number of data augmentation examples created using a particular value of θ (Lower $\theta \rightarrow$ more augmentation examples).

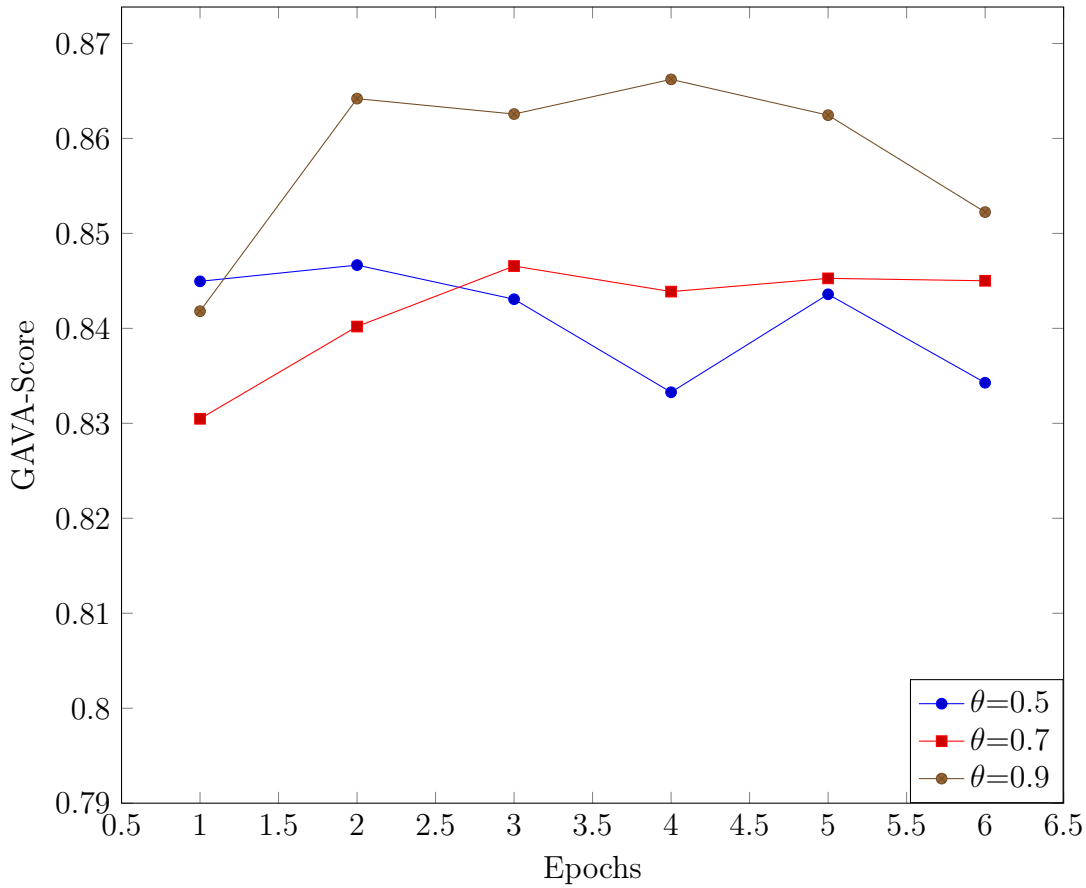

Figure 6.4: Comparison of three different GAVA-SDA models trained on MS-MARCO using different thresholds θ in terms of GAVA-Score (on validation split) varying across training epochs. The model trained with the largest value of θ achieves the highest GAVA-Score.

scores throughout training.

Correlation with other Automatic Metrics We perform a study to analyze how well can other automatic evaluation metrics perform for the task of evaluating answer generation. Specifically, we consider BLEU, BLEURT, and BERTScore. We use the MS-MARCO dataset and evaluate several different GenQA models trained using our approaches. We evaluate performance using each of these automatic metrics and SQuArE and present the Pearson and Spearman's rank correlation between these metrics and the manual evaluation in Table 6.4. From the table, we observe that SQuArE achieves the strongest correlation with human evaluation, highlighting its efficacy to be used as an automatic QA evaluation metric. Other text similarity matching metrics achieve poor correlation with the human annotation of answer correctness.

Approach	Manual	SQuArE-Score	BLEURT	BERTScore	BLEU
GenQA-WS	0.775	0.770	0.587	0.492	30.8
SQuArE-SDA(θ=0.5)	0.846	0.857	0.578	0.496	35.2
SQuArE-SDA(θ=0.7)	0.861	0.864	0.576	0.491	34.9
SQuArE-SDA(θ=0.9)	0.877	0.869	0.573	0.486	34.1
SQuArE-DDA	0.843	0.855	0.627	0.541	38.1
SQuArE-LW	0.794	0.784	0.627	0.502	32.2
Correlation (Pearson)		0.979	-0.429	-0.035	0.679
Correlation (Spearman's)		1	-0.812	-0.6	0.429

Table 6.4: Evaluation of different GenQA models using automatic evaluation metrics: BLEU, BLEURT, BERTScore in addition to SQuArE-Score on the MS-MARCO dataset. We present the correlation each metric achieves with human annotation. SQuArE achieves the best correlation with human evaluation of answer accuracy.

6.4.5 Qualitative results

We perform a qualitative analysis highlighting anecdotal examples to study success and failure cases of our answer-generation approach. Specifically, we pick the MS-MARCO dataset and the SQuArE-DDA approach, and present both success and failure cases of answer generation to gain insights into the strengths and limitations of our approach.

Table 6.5 shows instances where SQuArE-DDA successfully generates accurate answers. These examples highlight various sub-tasks that the model implicitly performs. Firstly, the model demonstrates its ability to synthesize information from multiple answer candidates. For example, for the question *"How do I get from DC to Alexandria VA?"*, the model correctly synthesized information from each of the reference answer candidates into the generated answer about the *Metrorail service* connecting the two locations. Second, the model exhibits reasoning capabilities highlighting identification of the correct reference answer candidate, along with improving it's style suitability for answering the input question. This is observed in the second example with the question *"How long should a central air conditioner last?"*, where the model identified the first reference *"10 to 20 years - sometimes longer"* to contain the most relevant information for answering the question. At times, the model acts as an answer sentence selection (AS2) model that simply re-ranks (without any modification) and generates one of the reference answer candidates.

Table 6.6 presents some examples where the model hallucinates and produces incorrect answers. This is highlighted in the question about Albany Minnesota's Population, where the model hallucinates and introduces an incorrect year in the generated answer, even when it is not present in any of the input reference candidates. Additionally, at times, the model may be unable to synthesize a good answer due to lacking evidence in the retrieved reference candidates. This is highlighted in the question about the Earth's magnetic field. This limitation emphasizes the importance of reliable and accurate answer candidates for grounding the answer generation from the model.

While our reward functions significantly improve GenQA performance, a closer look reveals that previous training strategies suffered from several key issues. Table 6.7

1)	Question	How do I get from DC to Alexandria VA?
	Answer #1	Check out the Metrorail system map and WMATA's official trip planner to plot the best route to your destination.
	Answer #2	To get to downtown DC from Alexandria, take the Yellow Line toward Greenbelt or Fort Totten, or take the Blue Line toward Largo Town Center or Stadium-Armory.Check out the Metrorail system map and
	Answer #3	WMATA's official trip planner to plot the best route to your destination.o get to downtown DC from Alexandria, take the Yellow Line toward Greenbelt or Fort Totten, or take the Blue Line toward Largo Town Center or Stadium-Armory.
	Answer #4	In addition to the King St-Old Town Metrorail station, Alexandria is serviced by three other stations: Braddock Road, Eisenhower Ave, and Van Dorn Street.
	GenQA Answer	The best way to get to Alexandria from Washington, DC is by Metrorail.
2)	Question	How long should a central air conditioner last?
	Answer #1	10 to 20 years - sometimes longer.
	Answer #2	You wouldn't run your car without regular maintenance and tune ups and you shouldn't run your a/c that way either - if you want it to last as long as possible. You should have a service tech come out once a year for a tune up.
	Answer #3	Source(s): 20 years working for a major manufacturer of central heating and air conditioning.
	GenQA Answer	The average life span of a central air conditioner is 10 to 20 years.
3)	Question	How long before a bearded dragon starts to grow
	Answer #1	When grown, males are larger than females, with notably bigger heads.
	Answer #2	Determining the gender of your bearded dragon before maturity isn't easy.
	Answer #3	Bearded dragons reach sexual maturity between the ages of 8 months and 18 months.
	GenQA Answer	Bearded dragons reach sexual maturity between the ages of 8 months and 18 months.

Table 6.5: Examples of correctly generated answers using SQuArE-DDA approach on the MS-MARCO dataset. Example (1) highlights that the model is correctly able to synthesize a correct answer using the reference answer candidates for the question. Example (2) highlights a case where the GenQA model uses information from a single reference answer, but reformulates it's style using the question to present as an answer. Example (3) highlights a case where the GenQA model is effectively functioning as an answer ranker, as it directly copies the best answer candidate among the references to produce the generated answer.

summarizes common error types such as oversimplification, factual inaccuracy, stylistic inconsistency, and ambiguity and contrasts them with the improvements achieved using our proposed reward functions. In particular, our methods enable the model to incorporate richer context, correct inaccuracies through targeted penalization, and produce more coherent and precise responses.

6.5 Conclusion and Discussion

In this Chapter, we propose a novel training paradigm for learning answer generation systems (GenQA) using supervision from automatic QA evaluation metrics based on transformer encoders. We propose three strategies: augmenting the training corpus with high SQuARe-scoring generated answers for GenQA (either statically once before training or dynamically during each training epoch) and using the SQuARe score to weigh the loss during training. Our empirical evaluation of two academic and one industrial dataset demonstrates that our approaches outperform the baseline, as evidenced by both human annotations and automatic QA evaluation metrics (SQuARe score).

Illustrative Examples: Our detailed examples (Examples 1–5) demonstrate that the reward functions (SDA, DDA, and LW) lead to generate answers that are clearer, more specific, and linguistically refined compared to baseline models. For instance, answers

1)	Question	Albany MN population ?
	Answer #1	Albany, Minnesota, as per 2017 US Census estimate, has a community population of 2,662 people.
	Answer #2	Albany is located in Stearns County, 20 miles west of St. Cloud and 80 miles northwest of Minneapolis/St. Paul on Interstate 94 (I-94). Albany has direct access to State Highway 238, which originates in Albany.
	GenQA Answer	The population was 2,662 at the 2010 census.
2)	Question	Which layer is responsible for the earth's magnetic field?
	Answer #1	Best Answer: The Earth's magnetic field is produced by convective currents in the outer core.
	Answer #2	The (presumably) molten iron core.
	Answer #3	The outer core is liquid iron.
	GenQA Answer	The outer core is liquid iron.

Table 6.6: Examples of incorrectly generated answers using SQuArE-DDA approach on the MS-MARCO dataset. Example (1) highlights a case of hallucination during generation where the model introduces an incorrect year in the generated answer, even when it is not present in any of the input reference candidates. Example (2) highlights a failure case of GenQA where the model is unable to synthesize a good answer due to lacking evidence in the retrieved reference candidates.

generated with SDA and DDA often include additional contextual details (e.g., specific transit lines for travel queries) and improved phrasing over the terse AS2 outputs.

Critical Error Analysis and Limitations: Our deep error analysis (Table 6.7) shows that previous reward models struggled with oversimplification, factual inaccuracies, stylistic inconsistencies, and ambiguity. In contrast, our proposed reward functions mitigate these issues by emphasizing high-confidence examples and integrating multiple candidate information. Nonetheless, challenges remain in borderline cases, and ambiguous inputs suggest avenues for future work.

Linguistic Insights: The reward-based approach not only enhances factual correctness but also improves linguistic qualities such as coherence, conciseness, and naturalness. By dynamically refining training examples based on SQuARe scores, the model learns to generate responses that better align with human preferences.

In conclusion, the techniques presented in this Chapter significantly advance the training of generative QA models. By incorporating reward signals derived from automatic evaluators like SQuARe, our methods yield higher-quality, human-like answers that outperform traditional approaches. Future work will explore further integration of these rewards in reinforcement learning settings and refinement of candidate re-ranking strategies to address remaining ambiguities and improve overall robustness.

Error Type	**Mistakes in Previous Training Strategies**	**Improvements with Our Reward Functions**
Over-Simplification	Tendency to generate overly concise answers lacking context	SDA and DDA encourage the synthesis of additional, relevant context, improving informativeness.
Factual Inaccuracy	Inadequate penalization of hallucinated or imprecise details	LW weights the loss to focus on high-confidence examples, thereby reducing factual errors.
Stylistic Inconsistency	Outputs often lack coherence and natural phrasing	Our methods enhance linguistic coherence by leveraging multiple candidate answers and confidence cues.
Ambiguity	Generic or vague responses due to ambiguous AS2 signals	Enhanced reward feedback helps the model resolve ambiguity by favoring clear, specific answers.

Table 6.7: Comparison between previous training strategies and our novel techniques.

Chapter 7

Retrieval Complexity

Question Answering (QA) is a fundamental task in Natural Language Processing (NLP), requiring systems to retrieve, interpret, and generate accurate answers to user queries. Among modern approaches, **Retrieval Augmented Generation (RAG)** has emerged as a dominant paradigm for tackling challenging QA tasks by combining information retrieval (IR) techniques with the generative capabilities of Large Language Models (LLMs) (Lewis et al., 2020b; Hsu et al., 2021). RAG systems enable QA models to access external knowledge sources, allowing them to handle diverse queries, including those targeting long-tail distributions or requiring fresh knowledge unavailable in static LLM training corpora (Asai et al., 2022b). For instance, systems like Bing-GPT illustrate the real-world potential of RAG by retrieving real-time information to answer questions about events occurring after the LLM training cutoff (Dao, 2023b).

Despite their advantages, RAG systems vary in effectiveness across different question types, particularly those requiring synthesis from multiple sources. The effectiveness of a RAG system often depends on the quality of the retrieved evidence. Specific questions, particularly those requiring evidence synthesis from multiple documents or targeting sparse or fragmented information, pose unique challenges. Identifying and characterizing these challenging questions is critical for improving the performance of RAG-based QA systems.

For example, to answer the question "Are lions bigger than tigers?" (Fig. 7.1) requires a comparison between two related entities. While this question could be considered complex, the close relation of the lions and tigers increases the probability that a single snippet talks about the size of both animals. However, The probability above is an order of magnitude lower for a similar question, such as "Are lions bigger than freezers?".

In this work, we introduce **Retrieval Complexity (RC)**, a novel metric designed to quantify the difficulty of answering a question based solely on the retrieval stage of RAG pipelines. Inspired by the observation that the difficulty of a question often correlates more with retrieval quality than with the reasoning capability of generative models, RC quantifies how dispersed or fragmented the necessary information is across retrieved documents, indicating the difficulty of constructing a complete answer. Questions with highly fragmented evidence require synthesis across multiple documents, making them

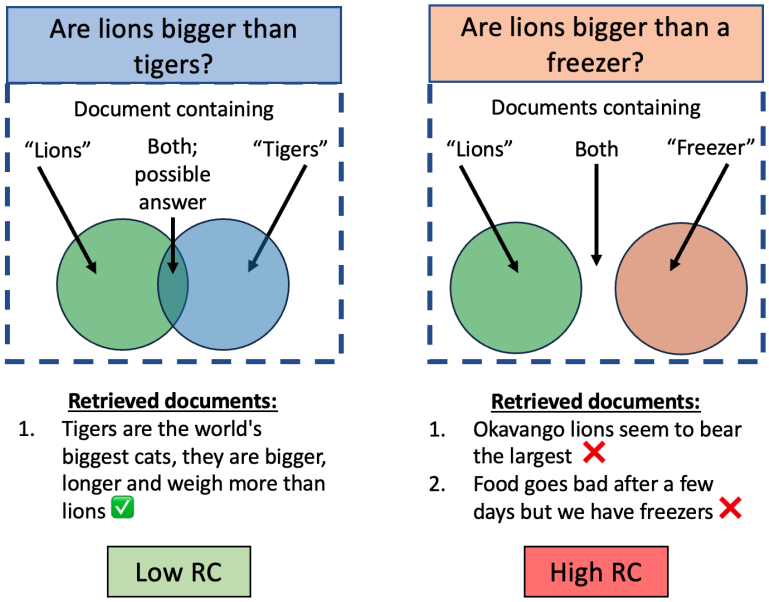

Figure 7.1: While both questions are complex using the definition in prior works, the "Low RC" question on the left is easy for RAG systems, while the "High RC" question on the right is far more difficult. Documents retrieved by commercial search engines illustrate this challenge. RC correlates with the probability that the retrieved documents contain sufficient information to generate a reference answer.

inherently more challenging for RAG systems, even those with advanced generative capabilities (AI@Meta, 2024; Jiang et al., 2023; Achiam et al., 2023).

7.1 Retrieval Complexity Motivation

Prior literature has broadly defined question complexity in terms of the number of reasoning hops or the compositional nature of the query (Mavi et al., 2022; Min et al., 2019). In contrast, our definition of Retrieval Complexity (RC) focuses on the retrievability and coherence of the evidence itself. Rather than solely counting reasoning steps, RC quantifies how fragmented or distributed the necessary information is across multiple documents. This approach offers several practical benefits:

- **Novelty:** Unlike traditional metrics that measure reasoning complexity (e.g., chain-of-thought lengths), RC quantifies the difficulty of retrieving the necessary evidence.
- **Practicality:** By evaluating the quality and completeness of retrieved evidence, RC provides actionable insights for improving RAG pipelines.
- **Versatility:** The unsupervised nature of our Retrieval Complexity Pipeline (RRCP) allows it to be applied across diverse QA benchmarks without needing manual complexity annotations.

A popular notion of question complexity in QA literature considers the number of reasoning steps (hops) required to answer a query. Multi-hop questions (Mavi et al., 2022), those expected to require at least two reasoning steps, are one source of complex questions. Popular benchmarks for multi-hop QA include HotPotQA (Yang et al., 2018b) and MuSiQue (Trivedi et al., 2022), which are constructed by aggregating multiple one-hop questions into a multi-hop query. Additionally, compositional complexity, as discussed by Talmor and Berant (2018), assesses the degree to which the answer must be composed from multiple pieces of evidence.

Linguistic Analysis: Our RC definition further integrates linguistic insights:

- **Multi-hop Complexity:** Linguistically, multi-hop questions require bridging semantic gaps across distinct sentences or documents, challenging the retrieval system to capture dispersed discourse elements.
- **Compositional Complexity:** Such queries require combining multiple semantic units (e.g., entities and relations) into a coherent answer, reflecting a high degree of syntactic and semantic integration.
- **Temporal Complexity:** Temporal questions often involve implicit cues (e.g., "Who was the president in 1945?" versus "Who is the last president?") that demand an understanding of temporal markers and context.
- **Comparative and Aggregate Complexity:** These queries often involve subtle comparative structures or the need to synthesize lists. The linguistic challenge lies in mapping comparative adjectives or aggregative phrases across documents.

Because these linguistic phenomena directly impact retrieval performance, our RC metric effectively quantifies the resulting fragmentation. When necessary evidence is scattered or expressed in varied linguistic forms, the RC metric captures the resulting fragmentation, providing a robust measure of the challenge posed to RAG systems.

To provide a better understanding of the differences between complex (High RC) and not complex (Low RC), in Table 7.1 we compare several queries classified as simple or complex based on our retrieval-centric criteria.

These examples demonstrate that while a query like "Who was president in 1945?" can be answered from a single snippet, queries requiring synthesis from multiple sources tend to have higher RC scores.

7.2 Modeling Retrieval Complexity

We propose an unsupervised pipeline (RRCP) to measure RC, described in Fig. 7.2: given a question Q, the pipeline retrieves k documents, $D = \{d_0, d_1, ..., d_k\}$, and analyzes them to predict the complexity of Q. We model the complexity problem along two dimensions: **question answerability** and **retrieval set completeness**. The first aims to verify if, for a given question Q, there exists a document $d_i \in D$ that could contain a correct answer. The second approximates the completeness of retrieved information relevant to the question.

Query	Simple (Low RC)	Complex (High RC)
Comparative Query	`"Are lions bigger than tigers?"`	`"Are lions bigger than freezers?"`
Multi-hop Query	`"Who was the 44th president of the US?"`	`"Who was the president during the transition from the 20th to the 21st century, and what major policy did he introduce?"`
Temporal Query	`"Who was president in 1945?"`	`"Who was leading the country during the post-World War II reconstruction, and how did his policies evolve over time?"`
Aggregate Query	`"List the Nobel Prize winners in literature in 2020."`	`"Which authors have won the Nobel Prize in literature in the 21st century, and what common themes can be identified in their works?"`

Table 7.1: Examples illustrating simple vs. complex queries. Our RC definition distinguishes them by measuring the degree of evidence fragmentation required to answer the query.

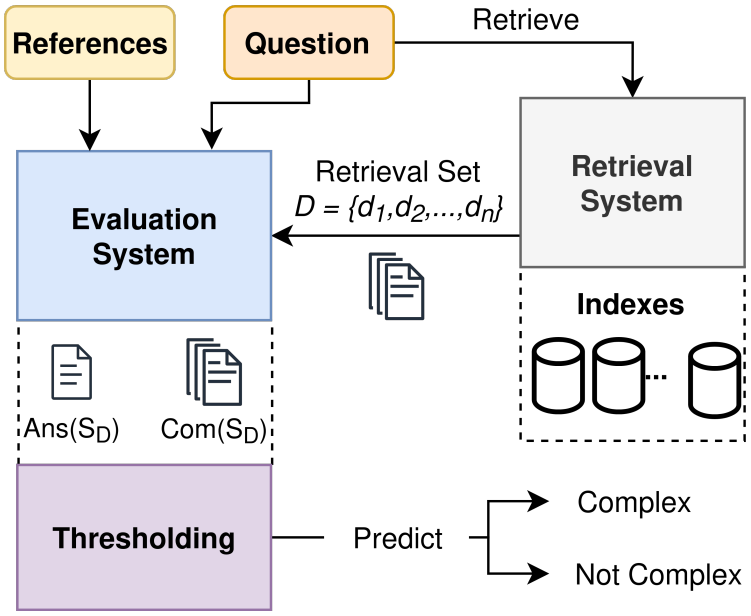

Figure 7.2: Overview of our RRCP for measuring the complexity of questions. A retrieval system retrieves documents for the input question. These are examined by a reference-based evaluation system, which provides the probability of correctness scores. These approximate the answerability and completeness of retrieved information, i.e., an estimation of RC.

7.2.1 Answerability

The answerability constraint is satisfied if at least one document $d_i \in D$ contains the correct answer for the question Q. Prior works approximate answerability by checking if any retrieved documents contain a string Exact Match (EM) with a reference answer (Rajpurkar et al., 2016a). However, this is a poor estimation for the generative QA domain (Deutsch and Roth, 2022) or when the original question allows multiple correct answers.

To mitigate this limitation, we use SQuARe (Gabburo et al., 2023b), an automatic evaluator that uses an LLM to estimate the answerability of a generated answer given one or more reference answers. Specifically, given a question Q, a document $d_i \in D$, and a set of gold references $R = \{r_0, r_1, ..., r_k\}$, it assesses the correctness of d_i for answering Q.

We define a threshold $T_{ans} \in [0, 1]$ over the SQuARe probability of a document containing the correct answer s_i: if the score exceeds T_{ans}, the question is *answerable*. Intuitively, if a single document d_i can answer the question, s_i will approach 1. Thus, we model the answerability function $Ans(S_D)$ as follows:

$$Ans(S_D) = \begin{cases} 0 \text{ if } \max(s_i \in S_D) < T_{ans} \\ 1 \text{ if } \exists\, s_i \in S_D \geq T_{ans}, \end{cases} \tag{7.1}$$

where T_{ans} distinguishes between not answerable and answerable questions.

7.2.2 Retrieval Set Completeness

Retrieval set completeness determines how the information required to answer is spread among different documents. For example, a document might be partially relevant yet lack sufficient detail to yield a correct answer (Fan et al., 2018; Baumgärtner et al., 2022). To estimate this, we compute the entropy of the relevance of each document in D with each token in Q. By assessing whether all parts of Q are adequately covered by the retrieved evidence, we approximate if the answer requires synthesizing multiple snippets.

We leverage the attention distribution extracted from SQuARe to create a matrix M of size $|D| \times |Q|$, where each row corresponds to a token t_j from Q, and each column indicates $Rel(d_i, t_j)$, the relevance of document d_i to token t_j. We then normalize the entropies and compute the average to obtain the completeness score S_D:

$$S_D = \frac{\sum_{i=1}^{|D|} \left\| \sum_{j=1}^{|Q|} Rel(d_i, t_j) \right\|}{|D|}. \tag{7.2}$$

We apply a threshold T_{com} to S_D:

$$Com(S_D) = \begin{cases} 1 \text{ if } S_D \geq T_{com} \\ 0 \text{ if } S_D < T_{com}. \end{cases} \tag{7.3}$$

7.2.3 Classifying Complex Questions

While answerability and retrieval set completeness are standalone criteria, RRCP leverages both to approximate retrieval complexity (RC). Specifically, RRCP classifies a question as complex if it is not answerable by a single document and the retrieved evidence is incomplete (i.e., $Ans(S_D) = 0$ and $Com(S_D) = 1$). In addition to classification, the answerability and completeness scores serve as diagnostic metrics.

7.3 Retrieval Complexity Pipeline (RRCP)

To make RC measurable in practical applications, we introduce the Retrieval Complexity Pipeline (RRCP), an unsupervised framework for estimating RC. The pipeline integrates:

- A retriever to obtain relevant documents for a given question.
- A reference-based evaluator (Gabburo et al., 2023b) to assess the quality and completeness of the retrieved evidence.

RRCP quantifies RC by evaluating both the retrievability (answerability) and the coherence (completeness) of retrieved evidence. This methodology enables RC to be

measured across arbitrary QA benchmarks, making it a versatile tool for the QA community.

7.4 Experiments

In this section, we focus on studying Retrieval Complexity (RC) and our RRCP framework for its detection. We validate them by conducting both quantitative and qualitative analyses. Specifically, we first compare RRCP against a supervised baseline (Section 7.4.3) and a prompted LLM directly on the inner complexity classes of a set of 4 selected datasets in Section 7.4.4. Secondly, in Section 7.4.5, we evaluate the correlation between the RC classification (complex/not complex) with LLM answer capacity. This allows us to understand some possible limitations of our notion of complexity. In the third quantitative experiment (Section 7.4.6), we measure the answerability of the questions identified as complex by our pipeline using a state-of-the-art search engine, such as Bing. Finally, we perform a qualitative analysis to inspect the limitations and the generalizability of our approach (Section 7.5).

7.4.1 Datasets

We evaluated RRCP on the following academic benchmarks: ComplexWebQuestions (CWQ) (Talmor and Berant, 2018), HotPotQA (Yang et al., 2018b), StrategyQA (Geva et al., 2021), and MuSiQue (Trivedi et al., 2022). We used the question complexity information provided in each dataset to define "complex" or "not complex" labels. For CWQ, we labeled their simple questions as not complex and the more challenging ones composed of them as complex. For HotPotQA, we used the "difficulty level" associated with each question in the dataset. For MuSiQue and StrategyQA, we considered one-hop questions as not complex and multi-hop ones as complex.

We also used Natural Questions (Kwiatkowski et al., 2019) and QuoraQP-a (Wang et al., 2020b) to evaluate our pipeline on more natural user-generated questions.

Manual annotation: To perform the manual annotation described in Section 7.4.6, we employed a set of 5 voluntary international experts (from the US, Brazil, India, and Italy) in the QA domain to annotate the data. We instruct each of them on the task, providing the notion of retrieval complexity 7.1 and a set of examples to reference during the annotation. Specifically, for each question present in our evaluation set, the annotation has been done by presenting the question to the annotator, the $top-5$ web pages retrieved by the search engine, and the top document retrieved using our retrieval system. Then, for each question, the annotators determine whether the question is answerable by one or more Bing search results (applying or inferring reasoning if needed). During the annotation process, the annotator was unaware of the RC assigned by RRCP.

Table 7.2 reports the complexity categories we found in some of the datasets above. Additional details regarding the distribution, size, and splits of these datasets are defined in Section 2.5.

Dataset	Complexity Class	
	Compositional	Multihop
CWQ (Talmor and Berant, 2018)	✓	✓
HotPotQA (Yang et al., 2018b)	✗	✓
StrategyQA (Geva et al., 2021)	✓	✓
MuSiQUe (Trivedi et al., 2022)	✗	✓
Natural Questions (Kwiatkowski et al., 2019)	✗	✗
QuoraQP-a (Wang et al., 2020b)	✗	✗

Table 7.2: Categorization of datasets into compositional and multihop complexity classes based on their respective characteristics.

7.4.2 RRCP setup

We implemented RRCP (described in Section 7.1), with a state-of-the-art hybrid retrieval system based on BM25 (Crestani et al., 1998) and ColBERT (Khattab and Zaharia, 2020) with an index of documents containing (Wikipedia (Petroni et al., 2021), and MS MARCO (Nguyen et al., 2016)). For the automatic evaluation system, we extended SQuARe (Gabburo et al., 2023b) with an additional head to provide the answerability and completeness score of retrieved documents. We set thresholds $T_{ans} = 0.15$ and $T_{com} = 0.80$ based on a small set of 50 manually written test questions.

7.4.3 Supervised Baseline

We compare RRCP (unsupervised approach) with a supervised baseline, which is therefore an upper-bound model. Note that, as the experiment shows, these upper bounds are not realistic as they do not generalize across different domains. It means that they only capture some specific properties of the data and specific complexity definitions.

We build the upper-bound models using a cross-encoder on the datasets discussed in Section 7.4.1. Our approach involved training this model in two distinct settings; a "merged" setting, in which the model is trained on all four complex datasets (CWQ, HotPotQA, MuSiQue, and StrategyQA), and an "independent" setting, i.e., one model is trained for each dataset. All 5 resulting models are tested on all 4 datasets, with results shown in Table 7.3. Notice that Natural Questions (Kwiatkowski et al., 2019) and QuoraQP-a (Wang et al., 2020b) do not have an inner definition of complexity (Table 7.2), making the automatic evaluation and the training impossible.

Implementation details: The supervised model was trained on Amazon AWS P3dn.24xlarge hosts using specific hyperparameters selected after a parameter search. The model architecture utilized the roberta-base (Liu et al., 2019) configuration, with a batch size (bs) of 256 instances. We used an Adam optimizer considering a learning rate (lr) of $1e-05$ during training, carried out over 10 epochs. The model selection criterion was based on achieving the highest F1 score on the development set (devset), ensuring the

	Datasets	Evaluated			
		CWQ	HotPotQA	MuSiQue	StrategyQA
Trained	ALL	0.974	0.834	**0.956**	0.921
	CWQ	**1.000**	0.448	0.609	0.266
	HotPotQA	0.901	**0.842**	0.714	0.656
	MuSiQue	0.851	0.491	0.861	0.618
	StrategyQA	0.075	0.253	0.238	**0.978**

Table 7.3: The supervised approach displays limited effectiveness for cross-dataset evaluation, indicating reduced performance when trained on one dataset and tested on another. Despite similar complexity classes between datasets (e.g., Multihop), the transferability of models across distinct datasets (e.g., training on CWQ and testing on HotPotQA) exhibits notable performance discrepancies. Notice that ALL refers to a model trained on all the datasets combined together.

selection of the most effective model variant. Additionally, the training process incorporated mixed-precision arithmetic (fp16) to enhance computational efficiency and speed up the training procedure. We estimate that GPU hours used for baseline training and pipeline inference are no more than 462 GPU hours.

While the model trained on the merged dataset exhibited commendable performance, independent models struggle to transfer this strong performance across datasets. This disparity suggests that fine-tuning induces the models to overfit the distribution of the questions rather than grasp the intricacies of question complexity. This outcome underscores the poor suitability of using standard fine-tuning to develop models to detect question complexity in a dataset-agnostic manner.

It should be noted that the reason of why ALL model performs well is simply because it can recognize which dataset the test question belongs to (thanks to topics and typical question shape characterizing each dataset) and activate the parameters specific to that dataset.

7.4.4 Complex Question Identification

To compare the ability of RRCP to detect complex questions with alternative approaches (e.g., LLMs, supervised models), we use the answerability and completeness scores it generates to classify questions as described in Section 7.4.4. We compare the performance in terms of accuracy and f1-measure against two strong baselines: (i) the supervised model described in Section 7.4.3 that has learned the dataset distribution and (ii) prompting an unsupervised, state-of-the-art LLM for this task. We report the results of these experiments in Table 7.4.

Furthermore, we ablated different configurations of the pipeline to assess the benefits given by the two constraints. Combining the answerability and the retrieval set completeness constraints proved to be beneficial, enabling a more accurate classification than alternatives, showing to be second only to the upper-bound supervised approach

Approach	Answerability	Diversity	CWQ		HotPotQA		StrategyQA		MuSiQue		Average
			ACC	F1	ACC	F1	ACC	F1	ACC	F1	F1
Supervised	✗	✗	0.994	0.974	0.799	0.834	0.941	0.956	0.944	0.956	0.930
LLM	✗	✗	0.649	0.780	0.699	0.815	0.548	0.436	0.582	0.709	0.685
RRCP	✓	✗	0.905	0.872	0.611	0.731	0.754	0.737	0.799	0.834	0.794
	✗	✓	0.818	0.189	0.560	0.674	0.554	0.493	0.679	0.651	0.502
	✓	✓	0.912	0.882	0.718	0.829	0.623	0.742	0.780	0.838	0.823

Table 7.4: Performance comparison of our pipeline against a complex question classifier in terms of Accuracy and F1. The pipeline has been tested ablating the two constraints (Answerability and Retrieval Diversity) and on different Complex QA datasets. The results show that our pipeline is able to identify complex questions with high accuracy and that combining the two constraints is beneficial. For these experiments, we selected the best pipeline thresholds based on the development set.

(0.823 vs 0.930). RRCP, based only on the answerability constraint, has generally higher results than the LLM baseline, showing that also, without considering the completeness constraint, the pipeline provides accurate predictions. On the other hand, the model-based only on retrieval completeness achieves lower performance than the other approaches, highlighting the fact that it can not be used alone.

7.4.5 LLM Performance on Complex Questions

In this section, we study if predictions made by RRCP (complex vs not complex) correlate with the notion of complexity of LLM-based systems (only using parametric knowledge). Specifically, we evaluate the correlation between its RC classification and LLM answer capacity (0/1 label), where the latter is computed by (i) generating an answer with LLM and (ii) manually annotating its correctness.

For each dataset in Section 7.4.1, we apply RRCP, employing our defined criteria to filter complex from non-complex questions. We use an end-to-end QA system composed of a prompted Mistral 7B (Jiang et al., 2023) (a large language model) designed to receive input questions and generate accurate answers. Then, we evaluated each generated answer, comparing it with the original gold answers in the datasets. Finally, we measured the agreement between the detected RC and answer correctness in terms of Pearson Correlation.

We present the results of this analysis in terms of accuracy in Table 7.5, where we show the accuracy scores for both complex and non-complex questions across the different datasets and their Pearson Correlation. Each entry in the table represents the percentage of questions correctly answered by this LLM-based QA system.

The results confirm the effectiveness of our approach in identifying complex questions also in a generative setting. Questions identified as complex by the pipeline generally exhibit higher accuracy in terms of answerability, indicating that these questions are more straightforward for the model to answer. Specifically, questions categorized as complex in CWQ, HotPotQA, MuSiQue, and Natural Questions display higher accuracy than those classified as not complex. Notably, the complexity assessment in Quora and StrategyQA does not significantly impact the question answerability, suggesting

Dataset	Complex	Not Complex	PCC
CWQ	0.328	0.641	0.449
HotPotQA	0.266	0.328	
StrategyQA	0.000	0.078	
MuSiQue	0.031	0.563	
Natural Questions	0.023	0.547	
Quora	0.016	0.016	

Table 7.5: Average answerability of answers generated for complex and not-complex questions selected by our pipeline. The results show that the questions marked as complex are generally more difficult to answer. PCC denotes the correlation in terms of the Pearson Correlation Coefficient between the complex and not complex questions in terms of answerability.

a different nature of complexity in this dataset or some bias. Indeed, upon closer examination, we discover unique challenges in StrategyQA and Quora. In StrategyQA, questions labeled as complex often have reference answers limited to "true" or "false," making them de facto complex due to uninformative references. The evaluation metric, which considers exact matches in strings, leads to an artifact where complex questions have a higher rate of "true/false" answers, contributing to the observed performance gap. Addressing this issue by modifying the prompt eliminates the artifact, emphasizing the importance of precise evaluation metrics in question complexity assessment. In the case of Quora, the complexity designation is influenced by the poor quality of reference answers rather than retrieval results. The noisy and unreliable nature of Quora references highlights a limitation in the evaluation process, emphasizing the need for caution when interpreting complexity labels in datasets with sub-optimal reference quality. Despite these challenges, the study underscores the significance of accurate complexity assessment methods.

7.4.6 Search Engine Performance on Complex Questions

We further explore the quality of our pipeline by sampling a set of 500 questions from the "complex" questions of each dataset and analyzing how many of them can be answered by a state-of-the-art search engine such as Bing[1]. This dataset is constructed by selecting 256 questions from datasets with their own notion of complexity, including CWQ, HotPotQA, StrategyQA, and MuSiQue (64 questions per dataset), and another 256 from natural datasets not explicitly focused on complexity (128 each from Natural Questions and Quora). To conduct the evaluation, we select a pool of expert annotators. Their task is to determine whether a given question can be answered by examining the top 5 search engine results. If the answer is found within these results or can be inferred through reasoning based on them, the question is considered answerable. Furthermore, the annotators are tasked with evaluating whether the question aligns

[1] https://www.bing.com/

RRCP Classification	Answerable	Not Answerable	MCC
Not Complex	0.600	0.113	0.508
Complex	0.400	0.887	

Table 7.6: Probability for a question classified by RRCP of being answered by a state-of-the-art search engine. The answerability is computed using manual annotators. MCC stands for Matthew Correlation Coefficient and measures the correlation between the complexity classification and its answerability.

with our predefined notion of retrieval complexity as described in Section 7.1.

The results of our analysis, detailed in Table 7.6, reveal significant insights. Indeed, the probability of a question marked as complex by RRCP not being answered by a state-of-the-art search engine is high (0.887). Similarly, a question marked as not complex has a high chance to be answered (0.6). In addition, the results show a good correlation between question-predicted complexity and their answerability (manually annotated), with a Matthew correlation of 0.508, indicating a moderate to strong positive correlation between the predicted classifications and the actual classifications.

7.4.7 Critical Error Analysis of RC Estimation

Although RRCP shows robust overall performance, our critical error analysis reveals two main challenges:

- **False Positives Due to Incomplete Retrieval:** Occasionally, a query may be labeled as complex because the retrieval system fails to capture a complete document even if the question itself is simple. This may occur in cases of low-index coverage or misranking.
- **Overestimation in Ambiguous Queries:** Queries with ambiguous phrasing can be misclassified as complex if the retrieved evidence is scattered across multiple documents, even when a clear answer exists.

Table 7.7 summarizes these issues alongside the underlying reasons.

These error cases suggest that while our RC metric effectively identifies evidence fragmentation, its accuracy can be affected by retrieval system coverage and query ambiguity. Future work may involve incorporating broader retrieval strategies or improved disambiguation techniques.

7.5 Qualitative Analysis and Limitations

We further examine the behavior of our pipeline through a rigorous qualitative evaluation. We notice that RRCP, initially designed to recognize retrieval complexity, can address other complexity classes beyond its primary scope. These complexity classes

Error Type	Observed Issue	Underlying Cause / Discussion
False Positives	Simple queries are misclassified as complex.	Incomplete retrieval due to limited index coverage or misranking of highly relevant documents.
Ambiguity Overestimation	Ambiguous queries flagged as complex.	Ambiguity in query phrasing leads to dispersed retrieval, even when a definitive answer exists.

Table 7.7: Critical error analysis: Cases where RC estimation works well versus challenges and their underlying causes.

are part of a wide range of question types, such as comparative questions (e.g., "Is an elephant bigger than a cat?"), multi-hop questions (e.g., "How old is the wife of the tallest NBA player?"), questions needing the "aggregation" of multiple answers in the form of disconnected entities (e.g., "List every football player who played in the last World Championship?"), time-based questions, both implicit and explicit (e.g., "Who is the current president of the US?" and "Which movie won the Oscar for the best movie in 1992?") and superlative questions (e.g., "What are the best countries to travel to in March?") following the distribution shown in Fig. 7.3.

However, our analysis also highlights notable limitations within our pipeline. We identified two primary challenges. Firstly, the reference-based nature of our approach is susceptible to the quality of the references used. Consequently, the predictive accuracy of the pipeline is significantly impacted by the quality of these references. This issue becomes evident during the examination of results from Quora, where we observed a discernible drop in performance. Despite employing a robust retrieval system, the precision of our pipeline is closely linked to the quality of the references.

This finding underscores the critical role of reference quality in shaping the effectiveness of our methodology. Addressing this challenge necessitates a comprehensive evaluation of the reference sources employed and potential enhancements in the retrieval system to mitigate the adverse effects on prediction accuracy. By acknowledging and addressing these limitations, our research aims to refine the qualitative analysis pipeline, ensuring its applicability across various complexity classes and enhancing its overall robustness in handling diverse question types.

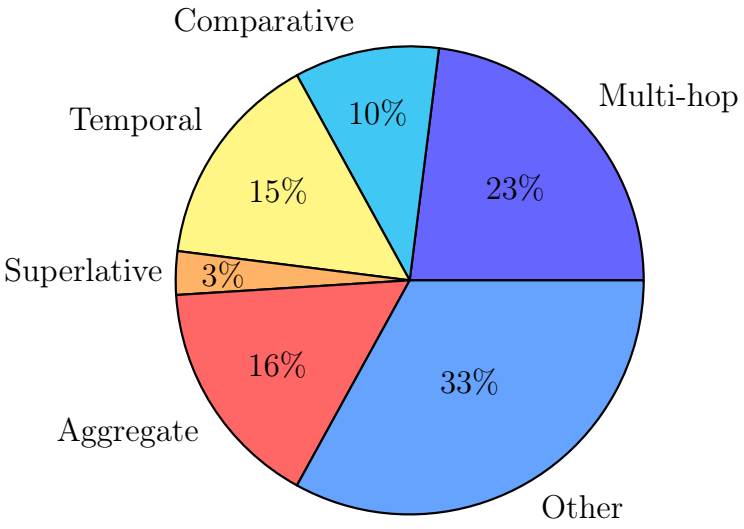

Figure 7.3: Distribution of complexity classes from a sample of 200 high RC questions selected among the datasets described in Section 7.4.1.

7.6 Conclusion and Discussion

In this paper, we introduced a novel concept of question complexity, measuring the difficulty of finding accurate answers from multiple sources. By integrating a hybrid BM25-ColBERT retrieval system with an advanced evaluation model, our approach (RRCP) effectively handled complex questions that extend beyond single-source responses. We conducted extensive quantitative and qualitative analyses on various academic and industrial datasets, proving the robustness of our pipeline in managing complex question types including multi-hop, temporal, comparative, and aggregate queries.

Theoretical and Practical Justification: Our definition of Retrieval Complexity (RC) contrasts with prior literature by focusing on evidence fragmentation rather than solely counting reasoning hops. This perspective highlights specific retrieval challenges in RAG systems, guiding improvements in search and indexing techniques.

Illustrative Examples: Detailed examples (Table 7.1) explicitly illustrate the differences between simple and complex queries based on our RC definition, demonstrating that complex queries require synthesis from multiple disparate sources.

Critical Error Analysis: Our critical error analysis (Table 7.7) highlights cases where our RC estimation performs well and identifies challenges such as false positives due to incomplete retrieval and overestimation in ambiguous queries. These insights inform future improvements.

Linguistic Analysis: Linguistically, our approach addresses the challenges posed by multi-hop, compositional, and temporal queries by capturing the nuances of discourse

cohesion, semantic integration, and temporal inference. These linguistic phenomena critically impact retrieval, and our RC metric quantifies the resulting fragmentation, providing a more comprehensive measure of query difficulty.

In conclusion, our work fills a critical gap in QA research by providing a robust, unsupervised measure of retrieval complexity that is both theoretically motivated and practically beneficial. Future work will explore integrating advanced retrieval techniques and linguistic disambiguation methods to further enhance the robustness and generalizability of our RC metric.

Chapter 8

Conclusion

In this PhD Thesis, I have explored and advanced several key research areas within Natural Language Processing (NLP), focusing on generative approaches and robust evaluation methodologies for Question Answering (QA) systems. My work has addressed significant challenges in the field by proposing novel solutions that contribute to the evolution of QA technologies through a series of interconnected studies. In doing so, I have played an active role in conceiving, developing, and experimentally validating new methods that go beyond the work reported in my published papers.

The Thesis is organized around a coherent research agenda. I began by providing an in-depth analysis of the research background, then moved from automatic evaluation of answer correctness to training generative models for QA using reward functions and weak supervision, and finally introduced a new definition of retrieval complexity for difficult questions. This structure situates my contributions within the broader literature while demonstrating the progressive development of my ideas.

Overall Contributions and New Insights

My contributions can be summarized as follows:

- **Development of SQuARe:** I led the design and implementation of SQuARe, a novel transformer-based automatic evaluation metric for QA that integrates multiple positive and negative references to achieve higher correlation with human judgment than traditional metrics such as BLEU, ROUGE, and BERTScore. In addition to the work published at IJCNLP-AACL 2023, I introduced new error analyses and linguistic insights that explain why SQuARe outperforms prior approaches.

- **Knowledge Transfer Framework from AS2 to Generative QA:** I developed a framework that utilizes discriminative Answer Sentence Selection (AS2) models to generate silver labels, significantly improving generative QA training efficiency. This framework significantly mitigates the scarcity of high-quality training data and reduces reliance on manual annotation. My work extends previous publications by providing detailed qualitative examples and a critical error analysis that

clarifies the linguistic improvements and limitations of the knowledge transfer approach.

- **Reward Function for GenQA using Automatic Evaluators:** Building upon SQuARe, I introduced innovative reward-based training strategies, including Static Data Augmentation (SDA), Dynamic Data Augmentation (DDA), and Loss Weighting (LW) to directly integrate automatic evaluation feedback into the training of generative QA models. This approach enhances answer quality regarding factual correctness and linguistic fluency and offers a new paradigm for using automatic rewards during model training. In this Thesis, I extend these ideas with new illustrative examples and deep critical error analysis, providing practical insights into the improvements over baseline reward functions.

- **Retrieval Complexity for QA:** I proposed a novel metric for quantifying question difficulty Retrieval Complexity (RC), which is based on evidence fragmentation in retrieval outputs. My work introduces the Retrieval Complexity Pipeline (RRCP), an unsupervised framework that quantifies evidence dispersion across documents. This new perspective not only contrasts with prior definitions that focus on reasoning steps (e.g., multi-hop complexity) but also brings practical benefits for improving retrieval-augmented generation systems. I have provided detailed illustrative examples and linguistic analysis that explain how RC captures multi-hop, compositional, and temporal challenges in QA.

In addition to my already published work, in this Tesis, we expanded our findings with novel insight, including:

- **Theoretical Clarification:** We provide a theoretical justification for retrieval complexity, distinguishing it from classical complexity measures that focus entirely on reasoning hops and outlining its advantages in practice for RAG systems.

- **Qualitative Analysis and examples:** Each proposed technique achieves a tangible gain in the quality of answers noticeably seen in detailed qualitative examples, including a richer set of query-specific error analyses with explicit linguistic commentary on coherence, conciseness, and informativeness.

- **Impact:** Training generative QA models with automatic evaluators and retrieval complexity measures may enable applications in reinforcement learning (RL), multilingual QA models, as well as the ethical impact of large-scale QA systems.

Automatic Evaluation for Generative QA: In Chapter 4, I introduced SQuARe, a new sentence-level evaluation metric that uses multiple reference answers and transformer-based contextual embeddings. SQuARe's design fills a critical gap by providing a more accurate and reliable evaluation for both extractive and generative QA systems, accelerating progress by offering better benchmarking tools.

Knowledge Transfer from AS2 to Generative QA: In Chapter 5, I presented a novel framework that transfers knowledge from discriminative AS2 models to generative QA models, enabling efficient training with large volumes of unlabeled data. This approach alleviates the dependency on expensive human-annotated datasets and demonstrates significant qualitative and quantitative improvements in answer generation quality.

Reward Function for Generative QA: In Chapter 6, I designed new reward functions that incorporate SQuARe feedback into the training of GenQA models. My work shows that using reward signals (through data augmentation and loss weighting) can guide generative models toward producing more accurate and contextually relevant answers, as evidenced by improved performance across multiple datasets.

Retrieval Complexity in QA Systems: In Chapter 7, I introduced Retrieval Complexity (RC) as a new metric for quantifying question difficulty based on evidence fragmentation. I provided theoretical and practical justification for RC, along with detailed illustrative examples and linguistic analysis demonstrating how RC captures multi-hop, compositional, and temporal challenges in QA.

Overall, this Thesis pushes the boundaries of state-of-the-art generative QA by proposing novel evaluation metrics, training paradigms, and theoretical approaches. The work I presented in my publications from my individual contributions not only have paved the way for novel methodology as well as the realization of the linguistic and practical challenges associated with QA systems These advances lay a groundwork for future work and have broad implications for both the academic and industrial worlds, to build the next generation of QA systems.

8.1 Future Work

In this section, we discuss potential advancements based on the work presented in this Thesis:

- **Extending SQuARe:** Further adapt SQuARe to other NLP tasks beyond QA, such as summarization or dialogue systems, to explore its broader applicability.
- **Multilingual and Cross-lingual QA:** Building on our work with multilingual AS2, develop more robust multilingual and cross-lingual generative QA methods.
- **Integrating Retrieval Complexity:** Investigate how to incorporate RC measures into the training and evaluation phases of QA systems to build models capable of handling diverse query types.
- **Large Language Models and QA:** Adapt our methods to large language models emerging in the field, exploring their potential to improve performance on more challenging QA tasks.

- **Ethical Considerations:** Address bias, fairness, and transparency in QA systems as these technologies advance, ensuring ethical deployment in real-world applications.

Bibliography

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

Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Pondé de Oliveira Pinto, Michael Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario D. Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin D. Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas A. Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cer'on Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll L. Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qim ing Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2023. Gpt-4 technical report.

AI@Meta. 2024. Llama 3 model card.

Miltiadis Allamanis and Charles Sutton. 2013. Mining source code repositories at massive scale using language modeling. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, page 207–216. IEEE Press.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. Falcon-40B: an open large language model with state-of-the-art performance.

Akari Asai, Matt Gardner, and Hannaneh Hajishirzi. 2022a. Evidentiality-guided generation for knowledge-intensive nlp tasks.

Akari Asai, Matt Gardner, and Hannaneh Hajishirzi. 2022b. Evidentiality-guided generation for knowledge-intensive NLP tasks. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2226–2243, Seattle, United States. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014a. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

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

Satanjeev Banerjee and Alon Lavie. 2005. 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*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second PASCAL recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Tim Baumgärtner, Leonardo F. R. Ribeiro, Nils Reimers, and Iryna Gurevych. 2022. Incorporating relevance feedback for information-seeking retrieval using few-shot document re-ranking. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8988–9005, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

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

Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5 2:157–66.

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. 2009. The fifth PASCAL recognizing textual entailment challenge. In *Proceedings of the Second Text Analysis Conference, TAC 2009, Gaithersburg, Maryland, USA, November 16-17, 2009*. NIST.

Pavol Bielik, Veselin Raychev, and Martin T. Vechev. 2016. Phog: Probabilistic model for code. In *International Conference on Machine Learning*.

Luiz Henrique Bonifacio, Vitor Jeronymo, Hugo Queiroz Abonizio, Israel Campiotti, Marzieh Fadaee, , Roberto Lotufo, and Rodrigo Nogueira. 2021. mmarco: A multilingual version of ms marco passage ranking dataset.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Jannis Bulian, Christian Buck, Wojciech Gajewski, Benjamin Börschinger, and Tal Schuster. 2022. Tomayto, tomahto. beyond token-level answer equivalence for question answering evaluation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 291–305, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Stefano Campese, Ivano Lauriola, and Alessandro Moschitti. 2023. QUADRo: Dataset and models for QUestion-answer database retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15573–15587, Singapore. Association for Computational Linguistics.

Meng Cao, Yue Dong, Jiapeng Wu, and Jackie Chi Kit Cheung. 2020. Factual error correction for abstractive summarization models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6251–6258, Online. Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2023. A survey on evaluation of large language models.

Anthony Chen, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Evaluating question answering evaluation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 119–124, Hong Kong, China. Association for Computational Linguistics.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017a. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017b. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. 2016. Training deep nets with sublinear memory cost.

Zihang Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao. 2017c. Quora question pairs.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *ArXiv*, abs/1904.10509.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

Noam Chomsky. 1956. Three models for the description of language. *IRE Transactions on Information Theory*, 2:113–124.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. 2022. Rethinking attention with performers.

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2023. Deep reinforcement learning from human preferences.

Kenneth Church and Patrick Hanks. 2002. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 845–855, Melbourne, Australia. Association for Computational Linguistics.

Elizabeth Clark, Asli Celikyilmaz, and Noah A. Smith. 2019. Sentence mover's similarity: Automatic evaluation for multi-sentence texts. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2748–2760, Florence, Italy. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Fabio Crestani, Mounia Lalmas, Cornelis J. Van Rijsbergen, and Iain Campbell. 1998. "is this document relevant?. . . probably": A survey of probabilistic models in information retrieval. *ACM Comput. Surv.*, 30(4):528–552.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

Tri Dao. 2023a. Flashattention-2: Faster attention with better parallelism and work partitioning.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness.

Xuan-Quy Dao. 2023b. Performance comparison of large language models on vnhsge english dataset: Openai chatgpt, microsoft bing chat, and google bard.

Yang Deng, Wai Lam, Yuexiang Xie, Daoyuan Chen, Yaliang Li, Min Yang, and Ying Shen. 2020. Joint learning of answer selection and answer summary generation in community question answering. In *AAAI*.

Daniel Deutsch and Dan Roth. 2022. Benchmarking answer verification methods for question answering-based summarization evaluation metrics. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3759–3765, Dublin, Ireland. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Luca Di Liello, Matteo Gabburo, and Alessandro Moschitti. 2022a. Effective pretraining objectives for transformer-based autoencoders. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5533–5547, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Luca Di Liello, Siddhant Garg, Luca Soldaini, and Alessandro Moschitti. 2022b. Paragraph-based transformer pre-training for multi-sentence inference. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2521–2531, Seattle, United States. Association for Computational Linguistics.

Luca Di Liello, Siddhant Garg, Luca Soldaini, and Alessandro Moschitti. 2022c. Pretraining transformer models with sentence-level objectives for answer sentence selection. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11806–11816, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Martin Fajcik, Martin Docekal, Karel Ondrej, and Pavel Smrz. 2021. R2-D2: A modular baseline for open-domain question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 854–870, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yixing Fan, Jiafeng Guo, Yanyan Lan, Jun Xu, Chengxiang Zhai, and Xueqi Cheng. 2018. Modeling diverse relevance patterns in ad-hoc retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, page 375–384, New York, NY, USA. Association for Computing Machinery.

David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager, Nico Schlaefer, and Christopher A. Welty. 2010. Building watson: An overview of the deepqa project. *AI Magazine*, 31(3):59–79.

Matteo Gabburo, Stefano Campese, Federico Agostini, and Alessandro Moschitti. 2024a. Datasets for multilingual answer sentence selection.

Matteo Gabburo, Siddhant Garg, Rik Koncel-Kedziorski, and Alessandro Moschitti. 2023a. Learning answer generation using supervision from automatic question answering evaluators. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8389–8403, Toronto, Canada. Association for Computational Linguistics.

Matteo Gabburo, Siddhant Garg, Rik Koncel-Kedziorski, and Alessandro Moschitti. 2023b. SQUARE: Automatic question answering evaluation using multiple positive and negative references. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 20–28, Nusa Dua, Bali. Association for Computational Linguistics.

Matteo Gabburo, Nicolaas Paul Jedema, Siddhant Garg, Leonardo F. R. Ribeiro, and Alessandro Moschitti. 2024b. Measuring retrieval complexity in question answering systems.

Matteo Gabburo, Rik Koncel-Kedziorski, Siddhant Garg, Luca Soldaini, and Alessandro Moschitti. 2022. Knowledge transfer from answer ranking to answer generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9481–9495, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Siddhant Garg and Alessandro Moschitti. 2021. Will this question be answered? question filtering via answer model distillation for efficient question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7329–7346, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Siddhant Garg, Thuy Vu, and Alessandro Moschitti. 2019. Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection. In *AAAI Conference on Artificial Intelligence*.

Felix Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with lstm. *Neural computation*, 12:2451–71.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. *Transactions of the Association for Computational Linguistics (TACL)*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague. Association for Computational Linguistics.

Yoav Goldberg. 2017. *Neural Network Methods for Natural Language Processing*, volume 37 of *Synthesis Lectures on Human Language Technologies*. Morgan & Claypool, San Rafael, CA.

Travis Goodwin, Max E. Savery, and Dina Demner-Fushman. 2020. Towards zero shot conditional summarization with adaptive multi-task fine-tuning. *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, 2020:3215–3226.

Alex Graves, Abdel rahman Mohamed, and Geoffrey E. Hinton. 2013. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778.

Zishan Guo, Renren Jin, Chuang Liu, Yufei Huang, Dan Shi, Supryadi, Linhao Yu, Yan Liu, Jiaxuan Li, Bojian Xiong, and Deyi Xiong. 2023. Evaluating large language models: A comprehensive survey.

Shivanshu Gupta, Yoshitomo Matsubara, Ankit Chadha, and Alessandro Moschitti. 2023. Cross-lingual knowledge distillation for answer sentence selection in low-resource languages. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 14078–14092, Toronto, Canada. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *ArXiv*, abs/2004.10964.

Rishav Hada, Varun Gumma, Adrian Wynter, Harshita Diddee, Mohamed Ahmed, Monojit Choudhury, Kalika Bali, and Sunayana Sitaram. 2024. Are large language model-based evaluators the solution to scaling up multilingual evaluation? In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1051–1070, St. Julian's, Malta. Association for Computational Linguistics.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing.

Vincent J. Hellendoorn and Premkumar Devanbu. 2017. Are deep neural networks the best choice for modeling source code? In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2017, page 763–773, New York, NY, USA. Association for Computing Machinery.

Abram Hindle, Earl T. Barr, Zhendong Su, Mark Gabel, and Premkumar Devanbu. 2012. On the naturalness of software. In *Proceedings of the 34th International Conference on Software Engineering*, ICSE '12, page 837–847. IEEE Press.

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

Sepp Hochreiter and Jürgen Schmidhuber. 1997a. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Sepp Hochreiter and Jürgen Schmidhuber. 1997b. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Chao-Chun Hsu, Eric Lind, Luca Soldaini, and Alessandro Moschitti. 2021. Answer generation for retrieval-based question answering systems. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4276–4282, Online. Association for Computational Linguistics.

Ryu Iida, Canasai Kruengkrai, Ryo Ishida, Kentaro Torisawa, Jong-Hoon Oh, and Julien Kloetzer. 2019. Exploiting background knowledge in compact answer generation for why-questions. In *AAAI*.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

F. Jelinek. 1990. *Self-organized language modeling for speech recognition*, page 450–506. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Tianbo Ji, Chenyang Lyu, Gareth J. F. Jones, Liting Zhou, and Yvette Graham. 2022. Qascore—an unsupervised unreferenced metric for the question generation evaluation. *Entropy*, 24.

Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L'elio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *ArXiv*, abs/2310.06825.

Dan Jurafsky and James H. Martin. 2009. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Pearson Prentice Hall, Upper Saddle River, N.J.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 39–48, New York, NY, USA. Association for Computing Machinery.

Seohyun Kim, Jinman Zhao, Yuchi Tian, and Satish Chandra. 2020. Code prediction by feeding trees to transformers.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference of Learning Representations*.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer.

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

Sachin Kumar, Antonios Anastasopoulos, Shuly Wintner, and Yulia Tsvetkov. 2021. Machine translation into low-resource language varieties. *arXiv preprint arXiv:2106.06797*.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 957–966. JMLR.org.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations.

Ivano Lauriola and Alessandro Moschitti. 2021a. Answer sentence selection using local and global context in transformer models. In *ECIR 2021*.

Ivano Lauriola and Alessandro Moschitti. 2021b. Answer sentence selection using local and global context in transformer models. In *European Conference on Information Retrieval*.

James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. 2022. FNet: Mixing tokens with Fourier transforms. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4296–4313, Seattle, United States. Association for Computational Linguistics.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *KR*. AAAI Press.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. *ArXiv*, abs/2005.11401.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020c. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Jian Li, Yue Wang, Michael R. Lyu, and Irwin King. 2018. Code completion with neural attention and pointer networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, IJCAI-2018. International Joint Conferences on Artificial Intelligence Organization.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

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

Stuart P. Lloyd. 1982. Least squares quantization in pcm. *IEEE Trans. Inf. Theory*, 28:129–136.

Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin B. Clement, Dawn Drain, Daxin Jiang, Duyu Tang, Ge Li, Lidong Zhou, Linjun Shou, Long Zhou, Michele Tufano, Ming Gong, Ming Zhou, Nan Duan, Neel Sundaresan, Shao Kun Deng, Shengyu Fu, and Shujie Liu. 2021. Codexglue: A machine learning benchmark dataset for code understanding and generation. *CoRR*, abs/2102.04664.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015a. Multi-task sequence to sequence learning. *CoRR*, abs/1511.06114.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China. Association for Computational Linguistics.

Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Expansion via prediction of importance with contextualization. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1573–1576, New York, NY, USA. Association for Computing Machinery.

C. D. Manning, P. Raghavan, and H. Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.

Vaibhav Mavi, Anubhav Jangra, and Adam Jatowt. 2022. A survey on multi-hop question answering and generation. *ArXiv*, abs/2204.09140.

Antonio Valerio Miceli Barone and Rico Sennrich. 2017. A parallel corpus of python functions and documentation strings for automated code documentation and code generation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 314–319, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Tomas Mikolov, Martin Karafiát, Lukáš Burget, Jan Honza Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Sewon Min, Jordan Boyd-Graber, Chris Alberti, Danqi Chen, Eunsol Choi, Michael Collins, Kelvin Guu, Hannaneh Hajishirzi, Kenton Lee, Jennimaria Palomaki, Colin Raffel, Adam Roberts, Tom Kwiatkowski, Patrick Lewis, Yuxiang Wu, Heinrich Küttler, Linqing Liu, Pasquale Minervini, Pontus Stenetorp, Sebastian Riedel, Sohee Yang, Minjoon Seo, Gautier Izacard, Fabio Petroni, Lucas Hosseini, Nicola De Cao, Edouard Grave, Ikuya Yamada, Sonse Shimaoka, Masatoshi Suzuki, Shumpei Miyawaki, Shun Sato, Ryo Takahashi, Jun Suzuki, Martin Fajcik, Martin Docekal, Karel Ondrej, Pavel Smrz, Hao Cheng, Yelong Shen, Xiaodong Liu, Pengcheng He, Weizhu Chen, Jianfeng Gao, Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Wen-tau Yih. 2021. Neurips 2020 efficientqa competition: Systems, analyses and lessons learned. In *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*, volume 133 of *Proceedings of Machine Learning Research*, pages 86–111. PMLR.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. Compositional questions do not necessitate multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4249–4257, Florence, Italy. Association for Computational Linguistics.

Alireza Mohammadshahi, Thomas Scialom, Majid Yazdani, Pouya Yanki, Angela Fan, James Henderson, and Marzieh Saeidi. 2022. Rquge: Reference-free metric for evaluating question generation by answering the question. *ArXiv*, abs/2211.01482.

Gordon E. Moore. 1965. Cramming more components onto integrated circuits. *Electronics*, 38(8).

Benjamin Muller, Luca Soldaini, Rik Koncel-Kedziorski, Eric Lind, and Alessandro Moschitti. 2022. Cross-lingual open-domain question answering with answer sentence generation. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 337–353, Online only. Association for Computational Linguistics.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. In *CoCo@NIPS*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Rodrigo Nogueira and Kyunghyun Cho. 2020. Passage re-ranking with bert.

Jekaterina Novikova, Ondrej Dusek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for nlg. In *Conference on Empirical Methods in Natural Language Processing*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5835–5847, Online. Association for Computational Linguistics.

Lawrence R. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020a. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020b. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018a. Know what you don't know: Unanswerable questions for squad.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018b. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016b. Squad: 100,000+ questions for machine comprehension of text.

Surangika Ranathunga, En-Shiun Annie Lee, Marjana Prifti Skenduli, Ravi Shekhar, Mehreen Alam, and Rishemjit Kaur. 2023. Neural machine translation for low-resource languages: A survey. *ACM Computing Surveys*, 55(11):1–37.

Veselin Raychev, Pavol Bielik, and Martin Vechev. 2016. Probabilistic model for code with decision trees. In *Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*, OOPSLA 2016, page 731–747, New York, NY, USA. Association for Computing Machinery.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4512–4525, Online. Association for Computational Linguistics.

Ehud Reiter. 2018a. A Structured Review of the Validity of BLEU. *Computational Linguistics*, 44(3):393–401.

Ehud Reiter. 2018b. A structured review of the validity of BLEU. *Computational Linguistics*, 44(3):393–401.

S. E. Robertson and Sparck K. Jones. 1976. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3:333–389.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1599–1613, Minneapolis, Minnesota. Association for Computational Linguistics.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Aliaksei Severyn and Alessandro Moschitti. 2016. Modeling relational information in question-answer pairs with convolutional neural networks.

Noam M. Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *ArXiv*, abs/1804.04235.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *CoRR*, abs/1909.08053.

Chenglei Si, Chen Zhao, and Jordan Boyd-Graber. 2021. What's in a name? answer equivalence for open-domain question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9623–9629, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Luca Soldaini and Alessandro Moschitti. 2020. The cascade transformer: an application for efficient answer sentence selection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5697–5708, Online. Association for Computational Linguistics.

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

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Proc. Interspeech 2012*, pages 194–197.

Alexey Svyatkovskiy, Shao Kun Deng, Shengyu Fu, and Neel Sundaresan. 2020. Intellicode compose: code generation using transformer. *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering.*

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651, New Orleans, Louisiana. Association for Computational Linguistics.

Chuanqi Tan, Furu Wei, Qingyu Zhou, Nan Yang, Bowen Du, Weifeng Lv, and Ming Zhou. 2018. Context-aware answer sentence selection with hierarchical gated recurrent neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(3):540–549.

Harish Tayyar Madabushi, Mark Lee, and John Barnden. 2018. Integrating question classification and deep learning for improved answer selection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3283–3294, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Thuy Vu and Alessandro Moschitti. 2021a. AVA: an automatic eValuation approach for question answering systems. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5223–5233. Association for Computational Linguistics.

Thuy Vu and Alessandro Moschitti. 2021b. Multilingual answer sentence reranking via automatically translated data. *CoRR*, abs/2102.10250.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. 2022. GPL: Generative pseudo labeling for unsupervised domain adaptation of dense retrieval. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2345–2360, Seattle, United States. Association for Computational Linguistics.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 22–32, Prague, Czech Republic. Association for Computational Linguistics.

Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. 2020a. Linformer: Self-attention with linear complexity. Cite arxiv:2006.04768.

Xin Wang, Yasheng Wang, Fei Mi, Pingyi Zhou, Yao Wan, Xiao Liu, Li Li, Hao Wu, Jin Liu, and Xin Jiang. 2021a. Syncobert: Syntax-guided multi-modal contrastive pre-training for code representation.

Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. 2021b. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021*.

Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. 2019. Multi-passage BERT: A globally normalized BERT model for open-domain 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)*, pages 5878–5882, Hong Kong, China. Association for Computational Linguistics.

Zizhen Wang, Yixing Fan, Jiafeng Guo, Liu Yang, Ruqing Zhang, Yanyan Lan, Xueqi Cheng, Hui Jiang, and Xiaozhao Wang. 2020b. Match^2: A matching over matching

model for similar question identification. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 559–568, New York, NY, USA. Association for Computing Machinery.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Anna Wegmann and Dong Nguyen. 2021. Does it capture STEL? a modular, similarity-based linguistic style evaluation framework. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7109–7130, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Aaron Steven White, Pushpendre Rastogi, Kevin Duh, and Benjamin Van Durme. 2017. Inference is everything: Recasting semantic resources into a unified evaluation framework. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 996–1005, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Terry Winograd. 1972. *Understanding Natural Language*. Academic Press, Inc., USA.

Peng Xu, Davis Liang, Zhiheng Huang, and Bing Xiang. 2021. Attention-guided generative models for extractive question answering. *ArXiv*, abs/2110.06393.

An Yang, Kai Liu, Jing Liu, Yajuan Lyu, and Sujian Li. 2018a. Adaptations of ROUGE and BLEU to better evaluate machine reading comprehension task. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 98–104, Melbourne, Australia. Association for Computational Linguistics.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy J. Lin. 2019a. End-to-end open-domain question answering with bertserini. In *North American Chapter of the Association for Computational Linguistics*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019b. Xlnet: Generalized autoregressive pretraining for language understanding. In *Neural Information Processing Systems*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018b. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. *CoRR*, abs/2007.14062.

Sina Zarrieß, Henrik Voigt, and Simeon Schüz. 2021. Decoding methods in neural language generation: A survey. *Information*, 12(9).

Zhiyuan Zeng, Jiaze Chen, Weiran Xu, and Lei Li. 2021. Gradient-based adversarial factual consistency evaluation for abstractive summarization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4102–4108, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis : A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Xiaoyu Zhang, Yishan Li, Jiayin Wang, Bowen Sun, Weizhi Ma, Peijie Sun, and Min Zhang. 2024. Large language models as evaluators for recommendation explanations.

Zeyu Zhang, Thuy Vu, Sunil Gandhi, Ankit Chadha, and Alessandro Moschitti. 2022a. Wdrass: A web-scale dataset for document retrieval and answer sentence selection. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 4707–4711, New York, NY, USA. Association for Computing Machinery.

Zeyu Zhang, Thuy Vu, and Alessandro Moschitti. 2021. Joint models for answer verification in question answering systems. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3252–3262, Online. Association for Computational Linguistics.

Zeyu Zhang, Thuy Vu, and Alessandro Moschitti. 2022b. Double retrieval and ranking for accurate question answering. *CoRR*, abs/2201.05981.