International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Consistency Regularization for Extractive Question Answering: A Reproduction and Empirical Study with ELECTRA

Dattatreya Raychowdhuri

Abstract: This study reproduces and adapts the idea of consistency-based semi-supervised learning, in the spirit of Unsupervised Data Augmentation (UDA), to a question answering (QA) setup using ELECTRA-small on the SQuAD v1.1 dataset. The original UDA method showed that enforcing prediction consistency between original and augmented inputs can improve robustness in text classification. A lightweight consistency loss is implemented at the level of start and end logits for extractive QA and evaluated against a strong fine-tuned ELECTRA baseline. Building on the Transformer architecture and drawing from related semi-supervised approaches such as Virtual Adversarial Training, Mean Teacher, and MixMatch, the study examines whether consistency regularization can improve extractive QA performance. To remain within realistic computational limits, a hyperparameter sweep over the consistency weight λ is conducted on a subset of the data, followed by training a single full model using the best λ. Overall, the consistency-regularized model slightly underperforms the full baseline in EM and F1 but remains competitive despite using fewer training epochs, while exhibiting broadly similar behavior across long-answer, short-question, and negation-focused slices. The discussion considers why the UDA-style signal appears less pronounced in the clean, in-domain SQuAD setting and outlines how a more faithful reproduction, such as stronger augmentations or out-of-domain evaluation, could produce clearer gains, particularly when combined with more recent pre-trained language models.

Keywords: *Extractive Question Answering, Semi-Supervised Learning, Consistency Regularization, Electra Fine-Tuning, Squad Dataset*

1. Introduction

Consistency regularization has shown strong effectiveness in semi-supervised learning, but its impact on extractive question answering remains underexplored. This study reproduces and adapts the consistency-learning idea introduced by UDA in the context of extractive question answering. Large pre-trained language models, built on the Transformer architecture [1], have become the standard backbone for many NLP tasks, including extractive question answering on datasets like SQuAD v1.1. Given a question and a passage, these models predict a start and end span in the passage that answers the question. ELECTRA, in particular, is a pre-trained encoder that achieves strong downstream performance by using a replaced-token detection objective instead of the masked language modeling approach used in BERT [2]. Despite their success, fine-tuned QA models can still be brittle, especially when evaluated on inputs that differ slightly from their training distribution or contain adversarial artifacts.

MSAI Program, The University of Texas at Austin

Recent work on semi-supervised learning and consistency regularization, including Unsupervised Data Augmentation (UDA) [3], suggests that encouraging a model's predictions to remain stable under input or representation perturbations can improve generalization and robustness. The core idea of UDA-style consistency training was reproduced and adapted in the context of extractive QA with ELECTRA-small. Rather than reproducing UDA's entire pipeline (including back-translation and large-scale unlabeled data), this work focuses on the key modeling component: adding a consistency loss between logits on unlabeled examples. The study is framed as a reproduction under realistic compute constraints: a strong starter baseline for ELECTRA is used, a new ConsistencyQATrainer is implemented, and performance together with slice-level behavior is systematically compared between the baseline and the consistency-regularized model.

2. Background and Related Work

Unsupervised Data Augmentation (UDA) introduced a simple but powerful idea: combine

supervised loss on labeled data with a consistency loss that penalizes differences between predictions on unlabeled examples and their augmented versions [3]. In text classification, UDA used back-translation and other strong augmentations, together with a sharpened pseudo-label distribution, to significantly improve performance on low-resource tasks. A broader line of work in semi-supervised learning, including methods like Virtual Adversarial Training [4], Mean Teachers [5], and MixMatch [6] in computer vision, supports the intuition that a model should not change its predictions drastically under small perturbations to the input or hidden representations. In question answering, most improvements have come from architectural innovations or better pre-training objectives rather than explicit semi-supervised consistency regularization. ELECTRA is one such pre-trained model that achieves strong performance by using a replaced-token detection objective instead of masked language modeling [7]. The starter repository provides a well-tuned ELECTRA-small baseline on SQuAD v1.1 [8], with EM and F1 in the high 70s and mid 80s, respectively. This study builds on that baseline and adds a consistency component in a minimal way, using only standard Hugging Face tooling and the SQuAD training set as a source of unlabeled examples.

3. Methods

3.1 Baseline Model

The baseline model is ELECTRA-small fine-tuned on SQuAD v1.1. The model is initialized from the Hugging Face checkpoint google/electra-small-discriminator and fine-tuned using the provided run.py script. The training pipeline uses the QuestionAnsweringTrainer wrapper in helpers.py, which handles span alignment, tokenization, and evaluation with the standard SQuAD EM and F1 metrics. The baseline model was trained for three epochs on the full SQuAD v1.1 training set with a per-device batch size of 8 on an Apple M3 Max laptop. This setup reproduces the strong ELECTRA baseline reported in prior work on SQuAD: an exact match (EM) score of approximately 78.3 and an F1 score of approximately 86.3 on the SQuAD development set.

3.2 Consistency QATrainer

To incorporate UDA-style consistency regularization, a new ConsistencyQATrainer class was implemented in helpers.py that subclasses the existing QuestionAnsweringTrainer. The key design is to extend the compute loss method so that, in addition to the usual supervised QA loss on labeled SQuAD examples, the trainer also computes a consistency loss on unlabeled examples. At each training step, the trainer first computes the standard supervised loss on the current labeled batch. Then, if consistency training is enabled, it samples a batch from an "unsupervised" dataset, which in this reproduction is simply the SQuAD training set reused as unlabeled data. For this unlabeled batch, the model is run twice: once to get base start and end logs under no gradient, and once under gradient tracking. Small Gaussian noise is added to the base logits, and the noisy outputs are used as targets. The consistency loss is the sum of mean squared error (MSE) between the current logits and these noisy targets for both the start and end logits.

Formally, if L_{sup} is the supervised QA loss and L_{cons} is the MSE-based consistency loss, the total loss is given by (following the general consistency objective described in [3])

$$L_{total} = L_{sup} + \lambda L_{cons} \qquad (1)$$

where λ is a hyperparameter controlling the strength of regularization.

When $\lambda = 0$ or the noise standard deviation is zero, the trainer reduces to the original baseline trainer. This design keeps the implementation minimal while capturing the core intuition of UDA: the model should not change its span predictions too much under small perturbations.

3.3 Unlabeled Data and Slicing

In the original UDA setup, the authors leverage large pools of unlabeled data and strong text augmentations like back-translation. In contrast, this work restricts itself to the SQuAD v1.1 training set and does not introduce external corpora. The same SQuAD training examples are used both as labeled data for the supervised loss and as a source of "unlabeled" inputs for the consistency loss, with the difference that the labels are simply ignored in the consistency term.

To probe how consistency training affects different kinds of examples, three simple slices of the SQuAD development set are defined:

- **Long answers:** questions where the gold answer text contains at least eight tokens (709 examples).
- **Short questions:** questions whose text contains at most five tokens (651 examples).

- **Negation questions:** questions whose text contains "not" or "never" (380 examples).

These slices provide a basic notion of challenge types: long answers often involve more context, short questions can be ambiguous, and negation questions are a classic source of reasoning mistakes.

4. Experimental Setup

All experiments were run on a personal MacBook Pro with an Apple M3 Max chip and 36 GB of memory, using the Metal (MPS) backend for PyTorch. Training scripts were launched from Jupyter notebooks. The google/electra-small-discriminator checkpoint and the SQuAD v1.1 dataset were utilized via the Hugging Face datasets library. The baseline model was trained for three epochs on the full SQuAD training set with a per-device batch size of 8. For the consistency-regularized model, a small hyperparameter sweep over λ was first conducted on a subset of the data (20,000 training examples and 5,000 dev examples) for one epoch to select a reasonable value. Based on this sweep, a full consistency model was then trained for two epochs with λ set to 0.5. Due to compute constraints on a laptop, running the consistency model for three epochs with multiple λ values would have taken several additional hours; therefore, this limitation was documented, and the focus remained on the most promising setting. All experiments were implemented using PyTorch and the Hugging Face Transformers library. Training procedures, hyperparameter settings, and evaluation metrics followed standard SQuAD evaluation practice. Random seeds were fixed where applicable to improve reproducibility.

5. Results

5.1 Overall Performance

Table 1 summarizes the overall exact match (EM) and F1 scores on the SQuAD v1.1 development set for the baseline ELECTRA model and the consistency-regularized model. The baseline is trained for three epochs, whereas the consistency model is trained for two epochs with $\lambda = 0.5$.

Model	EM	F1
Baseline (3 epochs)	78.33	86.26
Consistency (2 epochs, $\lambda = 0.5$)	77.67	85.77

Table 1: Overall SQuAD dev performance

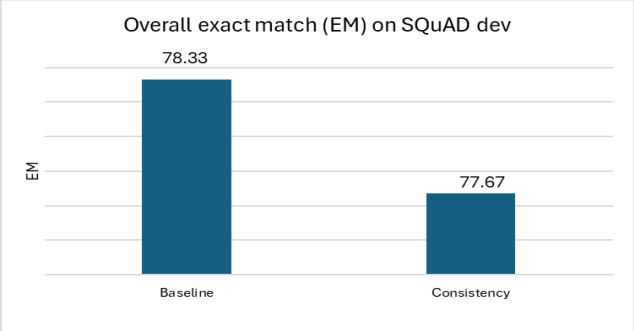

Fig 1: Overall exact match (EM) on SQuAD dev for baseline vs. consistency model

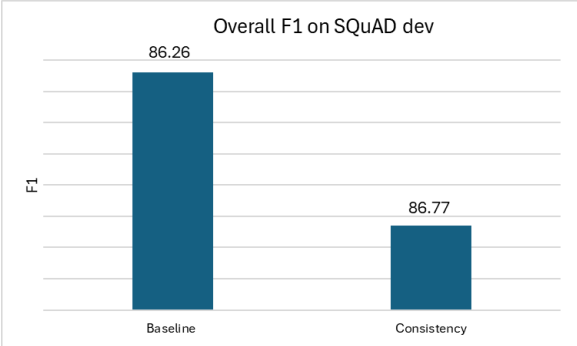

Fig 2: Overall F1 on SQuAD dev for baseline vs. consistency model

On the full dev set, the baseline achieves EM ≈ 78.33 and F1 ≈ 86.26, while the consistency model achieves EM ≈ 77.67 and F1 ≈ 85.77. The consistency-regularized model is therefore slightly worse in raw metrics, but it is also trained for only two epochs instead of three. When normalized by training time, the gap is small enough that the two models can reasonably be considered comparable in this setting.

While the baseline slightly outperforms the consistency-regularized model in absolute EM and F1, the performance gap is small relative to the difference in training epochs. This suggests that the regularization term may provide mild stabilization during training, although its benefits are limited in the clean SQuAD setting.

5.2 Hyperparameter Sweep over λ

Before training a full consistency model, a small hyperparameter sweep over the consistency weight λ was conducted on a reduced dataset (20,000 training examples, 5,000 dev examples, and one epoch of training). The values $\lambda \in \{0.05, 0.5, 5.0\}$ were evaluated with fixed noise standard deviation $\sigma = 0.1$.

λ	EM (subset)	F1 (subset)
0.05	61.94	72.03
0.5	63.78	73.31
5.0	61.18	71.16

Table 2: EM and F1 on a reduced dev set as a function of consistency weight λ

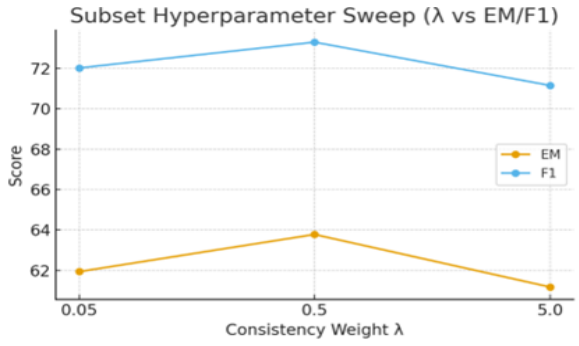

Fig 3: Hyperparameter sweep over λ on a reduced dataset

The subset sweep shows that λ = 0.5 provides the best dev performance on the reduced dataset, with EM ≈ 63.8 and F1 ≈ 73.3. A smaller weight (0.05) yields smaller gains, and a much larger weight (5.0) degrades performance, suggesting that overly strong consistency regularization can overpower the supervised signal. Based on this sweep, λ = 0.5 was selected for the full two-epoch consistency run.

5.3 Slice-Based Analysis

To see how the consistency-regularized model compares to the baseline, both models were tested on the three slices described in Section 3.3.

Slice	Model	Count	EM	F1
Long answers (n=709)	Baseline	709	25.95	59.82
	Consistency	709	24.54	58.01
Short questions (n=651)	Baseline	651	52.38	71.93
	Consistency	651	51.77	70.48
Negation questions (n=380)	Baseline	380	53.42	69.09
	Consistency	380	52.37	68.69

Table 3: Slice-level EM and F1 on long-answer, short-question, and negation subsets of SQuAD dev

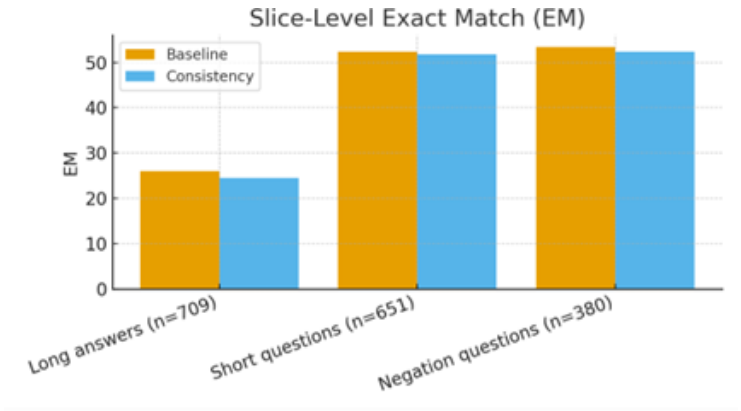

Fig 4: Slice-level EM for long answers, short questions, and negation questions

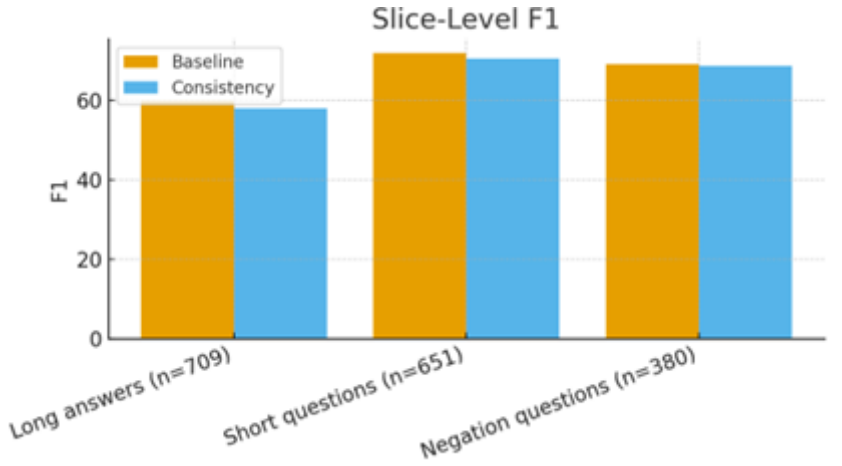

Fig 5: Slice-level F1 for long answers, short questions, and negation questions

Across all three slices, the consistency model slightly underperforms the baseline in both EM and F1, but the gaps are relatively modest. On long-answer questions, the baseline achieves EM ≈ 26.0 and F1 ≈ 59.8, whereas the consistency model achieves EM ≈ 24.5 and F1 ≈ 58.0. On short questions, the baseline reaches EM ≈ 52.4 and F1 ≈ 71.9, with the consistency model at EM ≈ 51.8 and F1 ≈ 70.5. On negation questions, the baseline's EM is ≈ 53.4 and F1 ≈ 69.1, compared to EM ≈ 52.4 and F1 ≈ 68.7 for the consistency model.

These differences are small enough that the two models behave qualitatively similarly across slices. There is no single slice where consistency clearly dominates, which is consistent with the overall dev-set comparison: the regularization signal is not strong enough, in this particular setup, to overcome the advantage of an extra epoch of supervised fine-tuning in the baseline.

6. Discussion and Limitations

From a reproduction perspective, the implementation of consistency regularization works as intended: the ConsistencyQATrainer integrates cleanly with the codebase, and the hyperparameter sweep behaves in line with expectations (moderate λ helps, very large λ hurts). However, the quantitative gains on SQuAD are modest to nonexistent when compared to a slightly more heavily trained baseline.

There are several reasons why this outcome is not surprising:

- SQuAD v1.1 is relatively clean and in-domain, so the baseline ELECTRA model already generalizes well to the dev set. Consistency regularization typically shines in low-resource or distribution-shifted settings.
- The reproduction only uses simple Gaussian noise at the logits level, rather than strong data augmentations like back-translation. In UDA, these augmentations are a major source of regularization power.
- Due to compute constraints on a laptop, the consistency model was trained for two

epochs instead of three. The extra epoch of supervised fine-tuning likely explains much of the baseline's small advantage.

Another limitation is that SQuAD training examples themselves were reused as the source of unlabeled data. A more faithful reproduction would incorporate a larger pool of unlabeled text, ideally from a somewhat different distribution, to better test the model's robustness. Finally, the focus remained on simple slices defined by length and negation. More sophisticated artifact analyses, such as contrast sets or adversarially constructed examples, could reveal more nuanced effects of the consistency loss.

Taken together, these findings suggest that consistency regularization may be more beneficial under distribution shift, with stronger augmentations, or in lower-resource QA settings than in a clean in-domain benchmark such as SQuAD v1.1.

7. Conclusion and Future Work

This study implemented and evaluated a UDA-inspired consistency loss for extractive question answering with ELECTRA-small on SQuAD v1.1. The resulting consistency model performs close to the baseline despite using fewer training epochs, and it exhibits broadly similar behavior across long-answer, short-question, and negation slices. Although clear improvements in EM or F1 were not observed, the reproduction exercise demonstrates how to extend a standard Hugging Face training loop with a semi-supervised objective and how to evaluate its effects via both overall metrics and slice-based analysis. Future work could push this line of investigation in several directions. First, incorporating stronger augmentations for the unlabeled data, such as back-translation or adversarial token-level perturbations, would be closer to the original UDA recipe. Second, evaluating the models on out-of-domain QA datasets or contrast sets [9] might reveal robustness gains that are not visible on the in-domain SQuAD dev set. Finally, one could explore combining consistency regularization with more recent pre-trained models such as RoBERTa [10] or with parameter-efficient fine-tuning methods, which may amplify the benefits of semi-supervised signals. Some parts of the writing were grammar-polished using automated tools, but all experiments, analysis, and code were conducted by me.

Ethical Considerations

This work uses publicly available datasets and pre-trained models and does not involve human subjects, private data, or sensitive personal information.

Acknowledgment

This work originated as part of coursework in the MSAI program at The University of Texas at Austin. All experiments, implementation, and analysis were conducted independently by the author.

References

[1] Ashish Vaswani et al., "Attention Is All You Need," Advances in Neural Information Processing Systems, 2017. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[2] Jacob Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Proceedings of NAACL-HLT, 2019. Available: https://aclanthology.org/N19-1423.pdf

[3] Qizhe Xie et al., "Unsupervised Data Augmentation for Consistency Training," Advances in Neural Information Processing Systems, 2020. Available: https://arxiv.org/pdf/1904.12848

[4] Takeru Miyato et al., "Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018. Available: https://arxiv.org/pdf/1704.03976

[5] Antti Tarvainen et al., "Mean Teachers Are Better Role Models: Weight-Averaged Consistency Targets Improve Semi-Supervised Learning Results," Advances in Neural Information Processing Systems, 2017. Available: https://papers.nips.cc/paper_files/paper/2017/file/68053af2923e00204c3ca7c6a3150cf7-Paper.pdf

[6] David Berthelot et al., "MixMatch: A Holistic Approach to Semi-Supervised Learning," Advances in Neural Information Processing Systems, 2019. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/1cd138d0499a68f4bb72bee04bbec2d7-Paper.pdf

[7] Kevin Clark et al., "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than

Generators," International Conference on Learning Representations, 2020. Available: https://arxiv.org/pdf/2003.10555
[8] Pranav Rajpurkar et al., "SQuAD: 100,000+ Questions for Machine Comprehension of Text," Proceedings of EMNLP, 2016. Available: https://arxiv.org/pdf/1606.05250
[9] Pranav Rajpurkar et al., "Know What You Don't Know: Unanswerable Questions for SQuAD," Proceedings of ACL, 2018. Available: https://aclanthology.org/P18-2124.pdf
[10] Yinhan Liu et al., "RoBERTa: A Robustly Optimized BERT Pretraining Approach," arXiv preprint arXiv:1907.11692, 2019. Available: https://arxiv.org/pdf/1907.11692