Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (T5)

Vishal Ramesh

27 December 2025

This paper presents a systematic empirical survey of transfer learning techniques in Natural Language Processing (NLP). Its primary contribution is not a novel architecture, but rather a **unified framework** that treats every text processing problem—classification, translation, summarization, and regression—as a "text-to-text" generation task.

By unifying the input/output format, the authors were able to rigorously compare different model architectures, pre-training objectives, and dataset filtering methods. The study culminates in the release of the **Colossal Clean Crawled Corpus (C4)** and the **T5 model**, which, when scaled to 11 billion parameters, achieved state-of-the-art results on benchmarks including GLUE, SQuAD, and SuperGLUE.

1 Core Methodology: The Unified Text-to-Text Framework

Prior to T5, transfer learning was fragmented. Models like BERT were used for classification (outputting class labels), while Seq2Seq models were used for translation. T5 standardizes this by forcing all tasks to accept a text sequence as input and generate a text sequence as output.

1.1 Task Formatting

The model uses a "task prefix" to distinguish between operations. This unified interface allows the same loss function (Cross-Entropy), hyperparameters, and decoding strategy to be used universally.

- **Machine Translation:**
  - *Input:* `translate English to German: That is good.`
  - *Target:* `Das ist gut.`
- **Classification (e.g., MNLI):**
  - *Input:* `mnli premise: I hate pigeons. hypothesis: My feelings towards pigeons are filled with animosity.`
  - *Target:* `entailment`
  - *Note:* The model does not output a class ID (0/1/2). It is trained to generate the literal string "entailment", "contradiction", or "neutral".
- **Regression (e.g., STS-B):**
  - *Input:* `stsb sentence1: The cat sat. sentence2: The cat lay down.`
  - *Target:* `3.8`
  - *Mechanism:* Continuous float values are rounded to the nearest 0.2 increment and converted to string literals (e.g., 2.57 becomes "2.6").

2 Baseline Configuration

- **Architecture:** Standard encoder-decoder Transformer (similar to BERT-Base on both sides)
- **Model Size:** 220M parameters (base model)
- **Training Scale:** 2^{35} tokens (34 billion tokens)
- **Vocabulary:** 32,000 wordpieces using SentencePiece
  - Trained on a mixture: 10 parts English + 1 part each of German, French, and Romanian
- **Learning Rate:** Inverse square root decay schedule
- **Training Steps:** 2^{19} steps (524k steps)
- **Batch Size:** 128 sequences (with experiments also testing 2048)

3 Data Engineering: The C4 Dataset

To support massive scale, the authors created the **Colossal Clean Crawled Corpus (C4)**, a 750 GB dataset derived from the Common Crawl.

3.1 Cleaning Heuristics

The paper emphasizes that model performance is strictly limited by data quality. Raw web text is noisy, containing code, menus, and error messages. Key filtering heuristics included:

- **Punctuation:** Retaining only lines ending in terminal punctuation (., ?, !, ").
- **Length:** Discarding pages < 5 sentences and lines < 5 words.
- **Deduplication:** Removing any 3-sentence span that appeared more than once in the dataset to prevent memorization of boilerplate text.
- **Language Detection:** Filtering for English text with $> 99\%$ probability.
- **Content Safety:** Removal of "List of Dirty, Naughty, Obscene or Otherwise Bad Words".

3.2 Dataset Ablation Study

Beyond C4, the authors systematically compared multiple unlabeled datasets to understand the impact of data source and filtering:

3.2.1 Datasets Tested

1. **C4 (Colossal Clean Crawled Corpus):** 750 GB, heavily filtered
2. **Unfiltered C4:** Same source, minimal filtering
3. **RealNews-like:** News articles from credible sources
4. **WebText-like:** Reddit-scraped data (GPT-2 style)
5. **Wikipedia + Toronto Books Corpus:** Traditional NLP corpus

3.2.2 Findings

- **Domain-specific pre-training** helped on matching downstream tasks (e.g., RealNews improved news classification)
- **In-domain pre-training on small datasets** led to overfitting
- **C4's filtering heuristics** provided consistent improvements across diverse tasks
- Unfiltered C4 underperformed, validating the importance of data quality over raw scale

4 Empirical Study: Architecture & Objectives

The authors conducted an extensive ablation study to determine the optimal model structure and pre-training objective.

4.1 Architecture Comparison

Three architectural variants were tested:

1. **Encoder-Decoder (Standard Transformer):** Fully-visible attention in the encoder; causal attention in the decoder.
2. **Language Model (GPT-style):** Decoder-only stack with causal attention.
3. **Prefix LM:** A hybrid decoder-only architecture.
    - *Mechanism:* Uses fully-visible attention over the "input" segment (like BERT) and switches to causal attention for the "target" segment (like GPT).

Finding: The **Encoder-Decoder** architecture consistently outperformed the others. Although the Prefix LM was competitive, the explicit separation of reading (encoding) and writing (decoding) proved most effective for the text-to-text format.

4.2 Pre-training Objective

The study compared Language Modeling (predict next word), Deshuffling, and Denoising objectives.
Finding: Span-Corruption (Denoising) was optimal.

- *Mechanism:* Instead of masking single tokens (BERT), T5 masks contiguous **spans** of text and replaces them with unique sentinel tokens (`<X>`, `<Y>`).
- *Efficiency:* The target sequence consists *only* of the missing spans, making training computationally cheaper than regenerating the full sentence.
- *Parameters:* A corruption rate of 15% with an average span length of 3 tokens yielded the best results.

4.3 Complete Pre-training Objective Comparison

The paper tested a comprehensive range of unsupervised objectives beyond the three main categories:

BERT-style Masking Variants:

1. **Replace Corrupted Spans:** Entire spans replaced with mask tokens, model predicts all corrupted tokens.
2. **Replace with Sentinel:** Spans replaced with unique sentinel tokens (<X>, <Y>), target only contains the corrupted tokens.
3. **Drop Corrupted Tokens:** Simply removes corrupted tokens without replacement.

Corruption Hyperparameters:

- **Corruption rates tested:** 10%, 15%, 25%, 50%
- **Mean span lengths tested:** 2, 3, 5, 10 tokens
- **Optimal configuration:** 15% corruption with mean span length of 3 tokens

Key Insight: The "Replace Span" approach (span corruption with sentinels) was optimal because:

- The target sequence contains only the corrupted spans (more efficient than full reconstruction)
- Sentinel tokens provide position information
- Balances between too easy (single token masking) and too difficult (very long spans)

5 Training & Scaling Strategies

5.1 Fine-Tuning vs. Parameter Efficiency

The authors investigated methods to adapt the pre-trained model to downstream tasks efficiently.

- **Full Fine-Tuning:** Updating all parameters.
- **Adapter Layers:** Freezing the main model and training small dense-ReLU-dense blocks inserted between layers.

Finding: While **Adapter Layers** offer significant memory savings during training, they consistently underperformed compared to **Full Fine-Tuning**. To match performance, adapters had to be scaled up significantly, negating their efficiency benefits.

5.2 Architecture Modifications

T5 introduces several architectural refinements to the standard Transformer:

Key Modifications:

1. **Relative Position Embeddings:** Instead of absolute position encodings, T5 uses relative position biases that allow better length generalization
2. **Simplified Layer Normalization:**
    - Applied outside residual connections (pre-norm)
    - Bias terms removed for efficiency

3. **Parameter Sharing:** Experiments showed that sharing parameters between encoder and decoder halved the parameter count with minimal performance degradation, though separate parameters performed slightly better

Computational Efficiency: Despite having roughly 2x the parameters of a decoder-only model, the encoder-decoder architecture has similar computational cost because:

- The encoder uses fully-visible attention (cheaper than causal)
- The decoder only processes the typically shorter target sequence with causal attention

5.3 Multi-Task Learning & Sampling

To train a single model on multiple tasks simultaneously, data imbalance must be addressed.

- **Strategy: Temperature-Scaled Mixing**.
- *Problem:* Large tasks (Translation) drown out small tasks (GLUE) if sampled by size. Equal sampling causes overfitting on small tasks.
- *Solution:* The probability of sampling a task is raised to the power of $1/T$.
  - $T = 1$: Proportional sampling.
  - $T = \infty$: Equal sampling.
- **T5 Approach:** Using $T = 2$ or similar artificially boosts the sampling rate of low-resource tasks without allowing them to dominate.

5.4 Pre-training + Fine-tuning vs Multi-task Learning

The paper compares two paradigms for leveraging multiple tasks:

Approach 1: Pre-training then Fine-tuning (Standard Transfer Learning)

- First pre-train on unsupervised objective (span corruption)
- Then fine-tune separately on each downstream task
- **Advantage:** Each task gets a specialized model

Approach 2: Multi-task Learning

- Train on all supervised tasks simultaneously from scratch (no unsupervised pre-training)
- Use temperature-scaled mixing to balance tasks
- **Advantage:** Single model handles all tasks

Finding: Pre-training + Fine-tuning consistently outperformed multi-task learning from scratch. The unsupervised pre-training phase provides crucial general-purpose representations that are difficult to learn from supervised data alone, even when using diverse tasks.

However, the paper also explored **multi-task pre-training** (unsupervised + supervised tasks during pre-training), which closed some of the gap but still underperformed compared to the traditional two-stage approach.

5.5 Scaling Laws (The "Bitter Lesson")

The paper posed a resource allocation question: Given a fixed 4x increase in compute, is it better to train longer, use larger batches, or build a larger model?
Finding: Increasing model size provided the most consistent gains. This finding led to the development of **T5-11B** (11 billion parameters), which achieved SOTA performance simply by virtue of its scale, reinforcing the "Bitter Lesson" that general methods scaling with compute often outperform hand-engineered optimizations.

5.6 Model Size Variants

T5 was released in multiple sizes to study scaling behavior systematically:

Model	Parameters	d_model	d_ff	Layers	Heads
T5-Small	60M	512	2048	6	8
T5-Base	220M	768	3072	12	12
T5-Large	770M	1024	4096	24	16
T5-3B	3B	1024	16384	24	32
T5-11B	11B	1024	65536	24	128

Scaling Observations:

- Performance improved consistently across all model sizes
- The 11B model required minimal hyperparameter tuning beyond the base configuration
- Larger models showed better sample efficiency (required less data to reach comparable performance)

5.7 Compute Allocation: The 4x Experiment

When given a fixed 4x increase in computational budget, three strategies were compared:

1. **Train 4x longer:** Use the same model for 4x training steps
2. **Increase batch size 4x:** Use larger batches (2048 vs 128)
3. **Increase model size 4x:** Scale from Base (220M) to Large (770M)

Result: Increasing model size provided the best performance gains, reinforcing the principle that model capacity is the primary driver of capability improvement given sufficient data.

6 Key Results & Conclusion

The final T5-11B model achieved:

- **GLUE:** 90.3 (SOTA).
- **SuperGLUE:** 88.9 (SOTA), nearing human baseline (89.8).
- **SQuAD:** State-of-the-art on Exact Match scores.

6.1 Evaluation Methodology

To ensure rigorous and reproducible results, the paper employed careful evaluation practices:

- **Validation-based selection:** All hyperparameters and design choices were selected based on validation set performance to avoid overfitting to test sets
- **Early stopping:** Training stopped when validation performance plateaued to prevent overfitting
- **Aggregate metrics:** For benchmarks like GLUE and SuperGLUE, reported the mean score across all constituent tasks
- **Multiple runs:** Key results were verified across multiple random seeds
- **Baseline comparisons:** Every modification was compared against the consistent baseline configuration

This systematic approach distinguishes T5 from papers that report only the best configuration without showing the full landscape of what was tested.

Conclusion: T5 represents a consolidation of transfer learning knowledge. By simplifying the interface (Text-to-Text) and maximizing the scale of data (C4) and parameters (11B), it established a robust baseline for modern NLP. While newer models like Gemini have shifted toward decoder-only, multimodal architectures, T5 remains a standard for functional, high-precision text generation tasks.