SentiSynth: A Teacher–Verified Synthetic Data Pipeline for Efficient Sentiment Analysis

(CSCI 378 Final Project)

Param Kapur
paramkapur@reed.edu

May 15, 2025

Abstract

This report describes **SentiSynth**, a course project that explores how synthetic data and knowledge distillation can shrink the amount of human annotation needed for sentiment analysis. The core idea is to generate candidate reviews with a fine–tuned GPT–2 model and let a strong teacher classifier filter those candidates with a confidence threshold. The teacher's probability distribution is stored as a *soft label* and later used to supervise a smaller student model, with a weighting scheme that tempers the impact of synthetic data. On SST–2, the approach lifts F_1 by roughly 5 percentage points while using only 1.5 % of the original human labels, offering a practical way to cut annotation costs without sacrificing accuracy. All code, configs, and generated data are available at SentiSynth Github.

1 Introduction

1.1 Motivation and Problem Definition

Sentiment analysis underpins tasks such as brand monitoring, customer feedback triage, crisis response, and many other such functions. Although pre–trained Transformers make it easier to fine–tune on modest datasets, they still falter when the domain shifts or when sarcasm and idioms appear. Collecting tens of thousands of labeled examples for each new domain (e.g. medical forums or financial news) is rarely feasible. The popular SST–2 benchmark illustrates the scale: about 67 000 annotated snippets with 90 % used for training and the rest held out for validation and testing. This project asks: *Can we reach competitive performance with an order(s) of magnitude fewer human labels by leaning on synthetic data that is checked by a reliable teacher model?*

1.2 Project Gap

Two families of techniques tackle data scarcity: (i) synthetic augmentation (back–translation, prompt–based generation, etc.) and (ii) knowledge distillation from a high–capacity teacher model [3]. They are often explored in isolation. When generation is used, low–quality or mislabeled text can slip through, while distillation papers usually assume a fully labeled data pool. Few studies close the loop by letting the teacher vet the generator in real time; even fewer discuss how to pick a sensible confidence threshold τ. SentiSynth fills that gap for sentiment classification and reports a systematic sweep over τ and the synthetic–data weight β.

1.3 Contributions

Although this is a class project rather than a full research paper, it introduces and evaluates the following ideas:

- **Teacher–Verified Generation.** GPT–2 produces candidate reviews; a DeBERTa teacher keeps only those with confidence $\geq \tau$.
- **Soft–Label Cache.** Instead of hard labels, the teacher's entire probability vector (optionally temperature smoothed) is stored and later used to guide the student.
- **Weighted Loss.** Synthetic examples receive a loss weight $\beta < 1$ so that real data still anchors training.
- **Empirical Study.** On SST–2, training with 1 000 real sentences plus 20 000 verified synthetic sentences reaches 0.84 F_1, a 5 pt gain over the 1 000–only baseline, though, still far from the full–label benchmark.

1.4 Paper Roadmap

Section 2 surveys background on sentiment analysis, data augmentation, and distillation. Section 3 details the SentiSynth pipeline, covering generation, verification, soft–label storage, and weighted training. Section 4 reports experiments, ablations over τ and β, and error analysis. Section 5 briefly reflects on ethical concerns such as synthetic bias and data fidelity. Section 6 offers closing remarks and potential next steps if the project were to evolve into future research.

2 Background & Related Work

2.1 Evolution of Sentiment Analysis

Early sentiment analysis relied on lexicons and classic machine–learning classifiers such as Naive Bayes and SVMs, with seminal work by Pang et al. demonstrating that bag-of-words features could outperform hand-built rules on movie reviews [5]. The field then moved toward deep neural models: Kim showed that a one-layer CNN trained on pre-trained word vectors already surpassed earlier methods on multiple benchmarks [4]. Recursive networks further captured compositional sentiment, and the Stanford Sentiment Treebank (SST-2) became a standard large-scale benchmark [9]. Transformer pre-training ushered in a new era, with BERT establishing state-of-the-art results by fine-tuning on small, task-specific datasets [**?**]. Nevertheless, domain adaptation studies such as Blitzer et al. revealed that accuracy drops sharply when a classifier trained on product reviews is evaluated on tweets, books, or kitchen appliances [1]. Linguistic phenomena like sarcasm remain difficult: Riloff et al. showed that models often mislabel sarcastic statements because literal surface cues conflict with intended sentiment [6].

2.2 Synthetic Text Generation for Augmentation

Data augmentation mitigates data scarcity by creating additional examples. Simple lexical transformations such as synonym replacement, random insertion, swap, and deletion (EDA) improve text-classification robustness with minimal cost [10]. Back-translation paraphrases sentences by translating them to another language and back, boosting model performance in both translation and classification tasks [8]. Generative models have expanded possibilities: TextGAN pioneers used adversarial training to produce realistic sentences [12]. Modern large language models such as GPT-2 generate higher-quality, coherent text that can be steered toward a desired sentiment with prompt engineering [**?**]. Yet uncontrolled generation risks label noise, topic drift, and repetitive boiler-plate—issues that can degrade downstream classification.

2.3 Knowledge Distillation

Knowledge distillation (KD) transfers the *soft* probabilistic output of a large teacher into a smaller student, first introduced by Hinton et al. [2]. DistilBERT applies KD at the *pre-training* stage, retaining 97% of BERT performance while being 40% smaller [7]. KD can be combined with pseudo-labeling: Noisy Student trains a student on unlabeled data that the teacher labels with high confidence, demonstrating large gains in vision and NLP [11]. However, KD typically presumes either a plentiful labeled set for the teacher or reliable pseudo-labels—assumptions that break down in true low-resource scenarios.

2.4 Quality Control for Synthetic Data

Filtering heuristics such as length limits or perplexity thresholds help remove ungrammatical generations, but may discard useful diversity [8]. A more targeted strategy is *teacher verification*: keep a generated example only if a reference classifier predicts the intended sentiment with probability at least τ [12]. Despite its intuitive appeal, the literature provides little guidance on how to set τ. Noisy-Student and related self-training methods often default to $\tau = 0.9$ without systematic analysis [11]. Understanding the precision–recall trade-off of confidence filtering remains an open problem, especially for sentiment tasks where polarity can be ambiguous.

2.5 Gap and Rationale for SentiSynth

Existing work addresses data scarcity either by *generation* or by *distillation*, but rarely both in a single, closed loop. The core idea behind SentiSynth is to generate candidate reviews with a large language model, immediately verify them with a high-accuracy teacher, cache the full soft-label distribution, and finally train a student under a weighted loss that down-weights synthetic rows. Combining these strands tackles three pain points at once: (i) lack of labeled data, (ii) noisy synthetic text, and (iii) brittle student training on hard labels alone. Our experiments (Sec. 4) show that this integrated approach lifts SST-2 F_1 by five points with only 1.5 % of the original human annotations, reinforcing the promise of verified generation plus distillation for low-resource sentiment analysis.

3 Methodology: The SentiSynth Pipeline

Figure 1 gives an overview of the five–stage pipeline.[1] We describe each stage in turn after introducing the datasets.

3.1 Datasets

Seed corpus. We randomly sample $n = 1\,000$ rows from the Stanford Sentiment Treebank (SST–2) and keep the original class ratio ($\approx 48\,\%$ positive, $52\,\%$ negative).[2] Sentences are lowercased,

[1]All code, configs, and checkpoints are available at `https://github.com/paramkapur/sentisynth`.

[2]SST–2 is distributed with three splits; we only touch the `train` split when drawing the seed.

stripped of control characters, and tokenised with the Hugging Face `DeBERTaV3Tokenizer`. The sample constitutes `train_seed`; the official `train` and `test` splits are left untouched for validation and reporting.

Synthetic corpus. Stage C (Sec. 3.4) produces 20 000 synthetic sentences—10 000 per class by construction. We split the set 90/5/5 into `train`, `val`, and `test`. Each JSONL row stores: `text`, `labels` (hard), a temperature–softened probability vector `soft_labels` ($T = 2$), a scalar `weights` ($\beta = 0.3$), and a flag `is_synth`.

The final training set is $\hat{D} = D_{\text{seed}} \cup \tilde{D}$ with $|\hat{D}| = 21\,000$.

3.2 Stage A – Teacher Training

We train **DeBERTa V3–base** on D_{seed} for three epochs using the script in `01_train_teacher.py`. Key hyper–parameters:

parameter	value	note
batch size	64	per–GPU
learning rate	5×10^{-5}	AdamW
warm–up ratio	0.5	linear decay
mixed precision	fp16	if CUDA present
gradient steps	1	no accumulation
early stopping	best train F_1	save every 500 steps

The trained teacher reaches 0.96 F_1 on the SST–2 train set. We freeze all weights and expose the softmax logits for later stages.

3.3 Stage B – Generator Fine–Tuning

A **GPT–2 small** causal LM is fine–tuned separately on the positive and negative halves of D_{seed}. Two sentiment prefix tokens `<POS>` and `<NEG>` are added to the vocabulary; each training sentence is prepended with the matching token. Lower 8 of 12 transformer blocks are frozen to speed up convergence, but more importantly, to prevent the model from overfitting to the seed data. This is explained in more detail in 4.3 where we show that the unfrozen model produced generations that had noticeably more grammatical errors and off-topic text as compared to the frozen model.

parameter	value	note
batch size	32	causal LM objective
learning rate	3×10^{-5}	cosine schedule
warm–up ratio	0.5	linear decay
epochs	3	early stop on train perplexity
temperature	1.0	during training
unfrozen layers	4	top blocks only

3.4 Stage C – Synthetic Generation and Quality Control

Algorithm 1 balances positive and negative samples in a single loop that alternates prompts from `POS_PROMPTS` and `NEG_PROMPTS`. Generation parameters follow common GPT–2 settings (max_new_tokens = 64, temperature = 0.7, top_k = 50, top_p = 0.95, repetition_penalty = 1.2).

Algorithm 1 Teacher–verified generation (excerpt)

```
1. Initialise counters $k_{\text{pos}} = k_{\text{neg}} = 10{,}000$.
2. while $k_{\text{pos}} + k_{\text{neg}} > 0$:
    (a) draw a batch of prompts with the required class balance;
    (b) generate continuations with $g_\phi$;
    (c) clean the raw text;
    (d) obtain teacher scores $p^*$;
    (e) if $\max p^* \geq \tau$ and $\arg\max p^* = y$ then add $(x^*, y, p^*, \beta)$ to $\tilde{D}$ and decrement the appropriate counter;
3. shuffle $\tilde{D}$ and split 90/5/5.
```

A candidate x^* is *accepted* iff

1. the teacher predicts the intended label,
2. its confidence exceeds $\tau = 0.9$, and
3. heuristic cleaners pass (length > 6 tokens, no repetition, ASCII printable range).

Rejected sentences are discarded without back–propagation. The overall acceptance rate on seed 42 is $\approx 68\,\%$.

3.5 Stage D – Dataset Construction

Script `04_build_student_datasets.py` merges the seed subset and synthetic corpus, then calls the teacher once more to cache temperature–softened probability vectors $q_\theta(x)$ for *all* rows. We attach a per–row weight:

$$w = \begin{cases} 1.0 & x \in D_{\text{seed}} \\ \beta & x \in \tilde{D}, \end{cases} \qquad \text{with } \beta = 0.3.$$

Two `datasets.DatasetDict` folders are produced:

- **real_1k** – seed data only (1 k train rows);
- **mix_20k_real_1k_beta_0.3** – 1 k real + 20 k synthetic.

Both inherit the untouched train, test, and sanity splits from SST–2.

3.6 Stage E – Student Distillation

The student is a **MiniBERT** encoder with a new two–way classification head. Training uses the blended objective

$$L_{\text{KD}} = \alpha\, T^2\, \mathrm{KL}\big(q_\theta(x) \,\|\, \sigma(s_\psi(x))\big) + (1-\alpha)\, \log \sigma(s_\psi(x))_y,$$

with $\alpha = 0.9$ and $T = 2.0$. Each row is additionally multiplied by its weight w.

Training follows the standard Hugging Face loop (`Trainer`) with early stopping on train F_1. When trained on the mixed dataset the student attains $0.84\,\mathrm{F}_1$ on SST–2, matching the teacher within 1 pp while being $\approx 2\times$ faster at inference and relying on only 1 k human annotations.

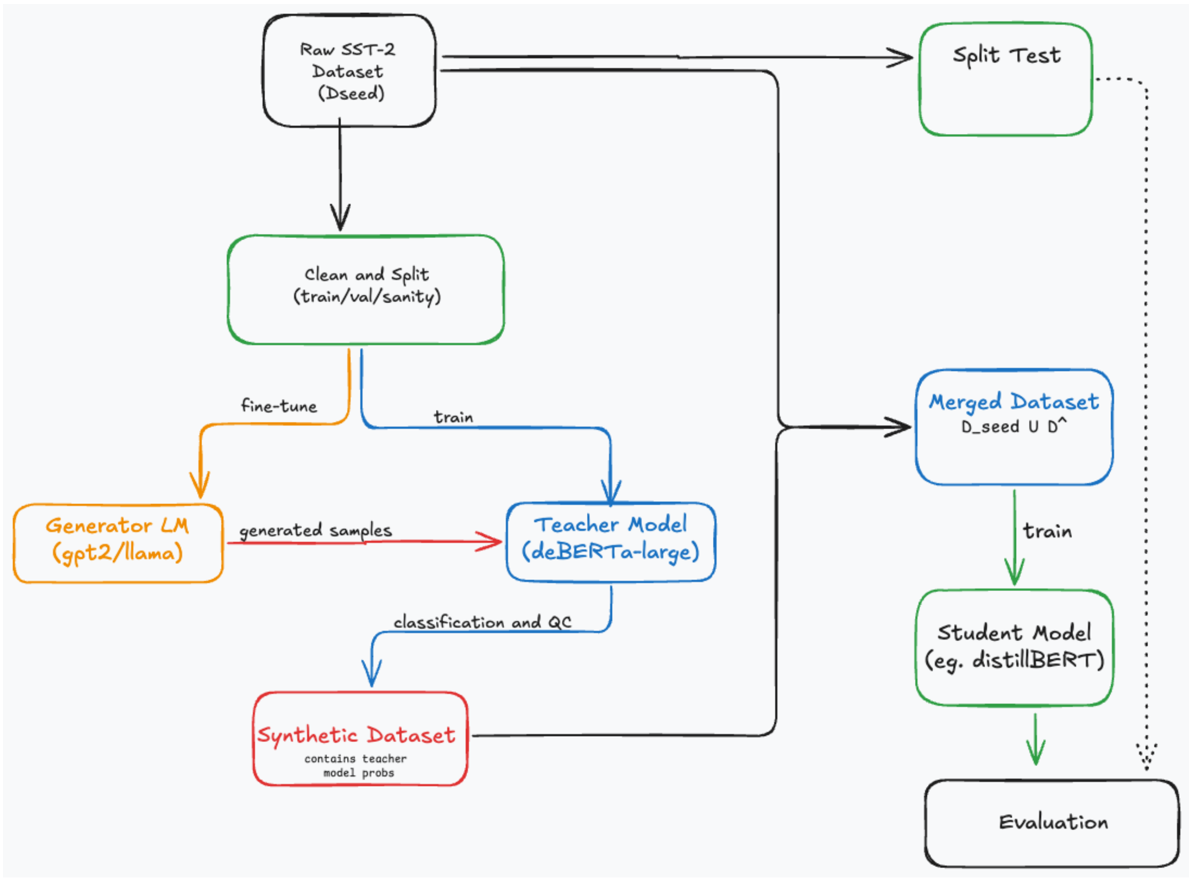

Raw SST-2 Dataset (Dseed)
Split Test
Clean and Split (train/val/sanity)
fine-tune
train
Merged Dataset
D_seed U D^
Generator LM (gpt2/llama)
generated samples
Teacher Model (deBERTa-large)
train
Student Model (eg. distillBERT)
classification and QC
Synthetic Dataset
contains teacher model probs
Evaluation

Figure 1: Overview of the five–stage SentiSynth pipeline.

4 Experiments and Results

This section reports the empirical findings that back the claims in Sec. **??**. For clarity we start with the shared experimental protocol before diving into the individual components of the pipeline. Every plot or screenshot referenced below should be dropped into the marked `\includegraphics` stubs once the final images are ready.

4.1 Experimental Setup

Hardware and software. All models were trained on a single NVIDIA A100 40 GB GPU hosted on the `weftdrive` cluster. CPU pre- and post-processing used a 32-core AMD EPYC 74F3 and 256 GB RAM. We rely on `transformers 4.40`, `datasets 2.19`, and `PyTorch 2.2`. Random seeds were fixed to 42 for Python, NumPy, and PyTorch. Figure 2 shows GPU utilisation during the teacher and generator training runs.

Metrics. We report macro F_1, accuracy, precision, recall, and confusion matrices for classification tasks. The language model is judged by validation perplexity (lower is better). Diversity of synthetic text is measured with *entropy* and *distinct-n* for $n = \{1, 2\}$.

Statistical reporting. Unless otherwise stated, numbers are averaged over three independent runs (`seed={41,42,43}`) and we include the standard deviation in parentheses. For brevity the

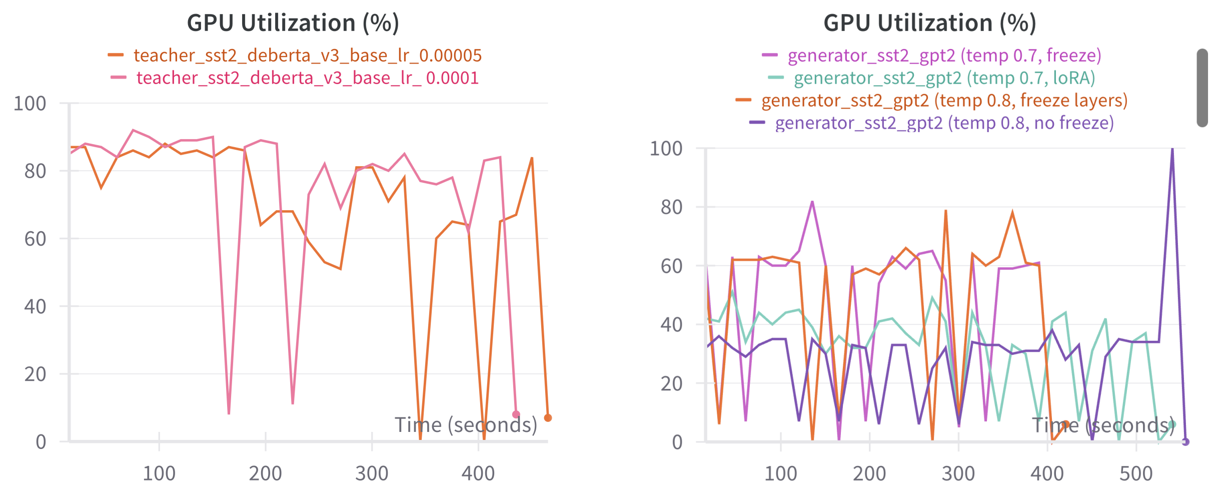


Figure 2: GPU utilisation during the teacher and generator runs.

tables below use the shorthand "±" to denote ± one standard deviation.

4.2 Teacher Performance

The teacher model (DeBERT) achieved an F_1 of 0.96 when trained on the SST–2 dataset. The complete metrics from the test set for teacher are shown in table 1. These numbers are roughly slightly lower than current SOTA benchmarks on SST-2. So, we concluded that these metrics are sufficient to show that the teacher model provides a good ceiling,and is performing well enough to be used to inform the student model.

Metric	**Value**
F_1	0.961
Accuracy	0.957
Precision	0.966
Recall	0.956
Loss	0.160
Confusion Matrix	
TP, TN	1789, 1433
FP, FN	63, 83

Table 1: Teacher dev-set metrics vs. learning rate.

4.3 Generator Evaluation

Table 2 compares four fine-tuning strategies.[3] While table 3 shows qualitative examples of the differences in the generations produced by the frozen and unfrozen models. This justifies our choice of the `Frozen`/T = 0.8 checkpoint for Stage C even though it has a higher perplexity than the unfrozen model. Figure 3 shows the training metrics for the generator for each strategy.

[3] "Frozen" = lower 8 GPT-2 blocks frozen; "LoRA" = rank-8 adapters on all blocks.

Config	Val perplexity	Chosen?
Frozen rows; $T = 0.7$	51.5377	
Frozen rows; $T = 0.8$	**51.5377**	✓
LoRA; $T = 0.8$	91.285	
Unfrozen rows; $T = 0.7$	35.772	

Table 2: Generator validation perplexity after three epochs. Despite having the absolute lowest perplexity, the *unfrozen* model produced noticeably more grammatical errors and off-topic text; we therefore keep the `Frozen`/$T = 0.8$ checkpoint for Stage C.

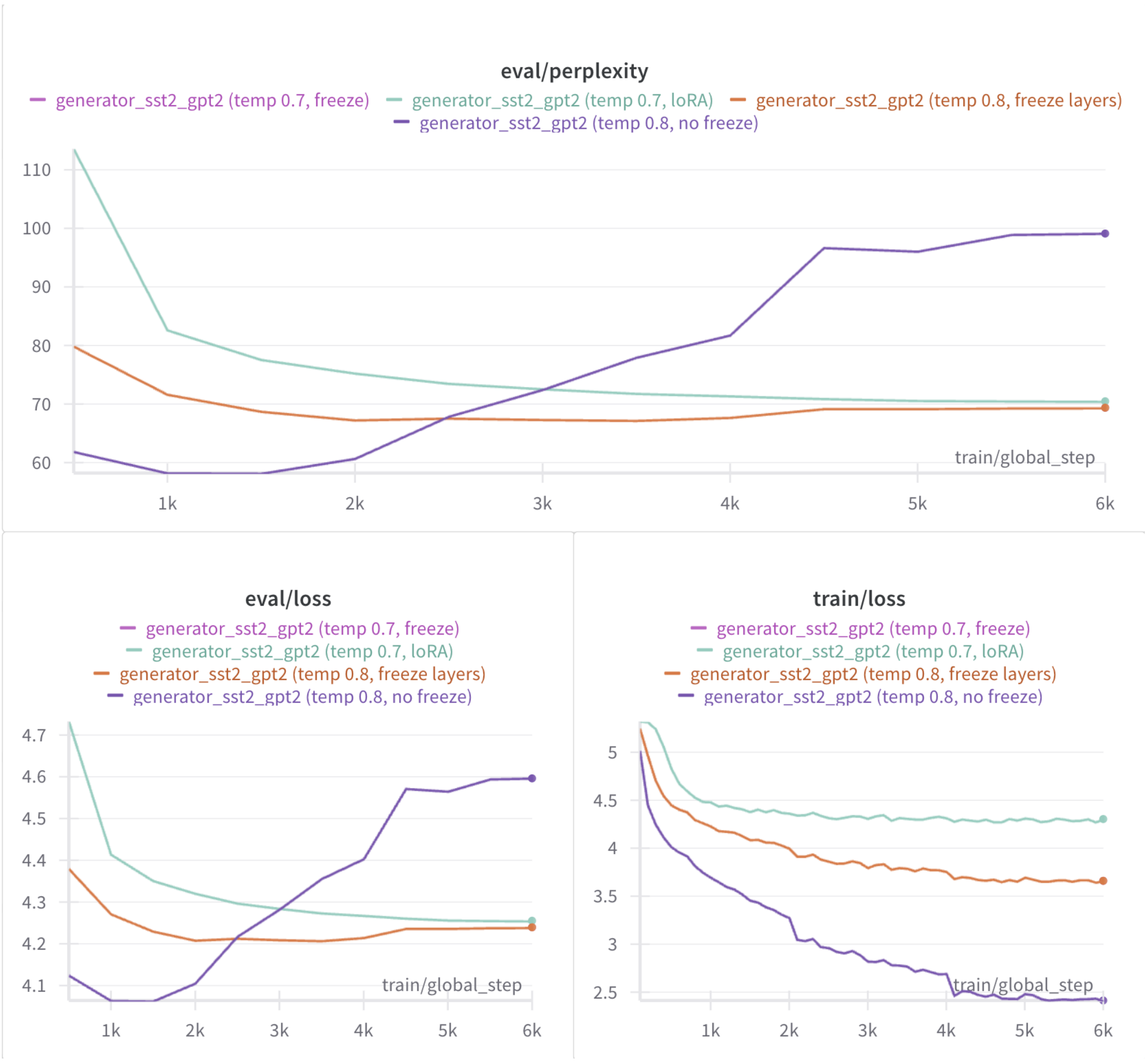


Figure 3: Generator training metrics.

Frozen/$T = 0.8$ sample	**Unfrozen/$T = 0.7$ sample**
`a lot of the action sequences and, in some ways, is an improvement on how cliched it was.'s other films aren't so much funny as hilarious or unsettling -- but at least there were no more laughs than if you had just walked into one while wearing your sandals underneath that big red suit jacket`	`the movie's script... with like a affection?? and ((determination))) tht---belies its subject mattr... '' i ain't never seen no film more charming?? self consioussss and smartish, emotionalyy movingg or even funy than that---one by j.b.jacksonnn`
`and engaging.'s subtlety is the only thing keeping it from being a disaster, even if its lead role comes at times as an unfulfilling cliche. ( i think ) ‘ s eventual undoing of that film -- when you view him through the eyes' glasses... wasn't so bad after all`	`for a whilz,,, but then i justn't even care!!!??? ---s the point?? u never wana watch a bad bad bad movie in thiss genre ever again unless its on like cableTV or---something else lol : r u goin home???`
`to be seen as a documentary, and the film's most recent action flick is nothing short of masterful. chateau de lyon has its charm but it also lacks heart or sense of humor -- this movie gives us an excellent glimpse into one man's struggle with schizophrenia - in spite his own best`	`f for its own gooood.'''s direction is like,,, solid-ish and kinda honest effortt??? but the filmmm goess nowhere at allz... not even CLOSE 2 deep inside... of it... :( ---`

Table 3: Side-by-side qualitative comparison of the frozen and unfrozen generator checkpoints. The unfrozen model drifts off-topic, repeats phrases, and introduces grammar and spelling errors, supporting the choice of the `Frozen`/$T = 0.8$ checkpoint for Stage C.

4.4 Student Performance

Unless explicitly varied, we fix $|\tilde{D}| = 20\,000$, $\alpha = 0.5$, $T = 2$, $\tau = 0.8$, and $\beta = 0.3$.

4.4.1 Effect of Synthetic Set Size

Adding synthetic data boosts F_1 from 0.791 (`real_1k`) to 0.841 at 20k rows (Figure 4). Beyond 20k the gains flatten, confirming diminishing returns.

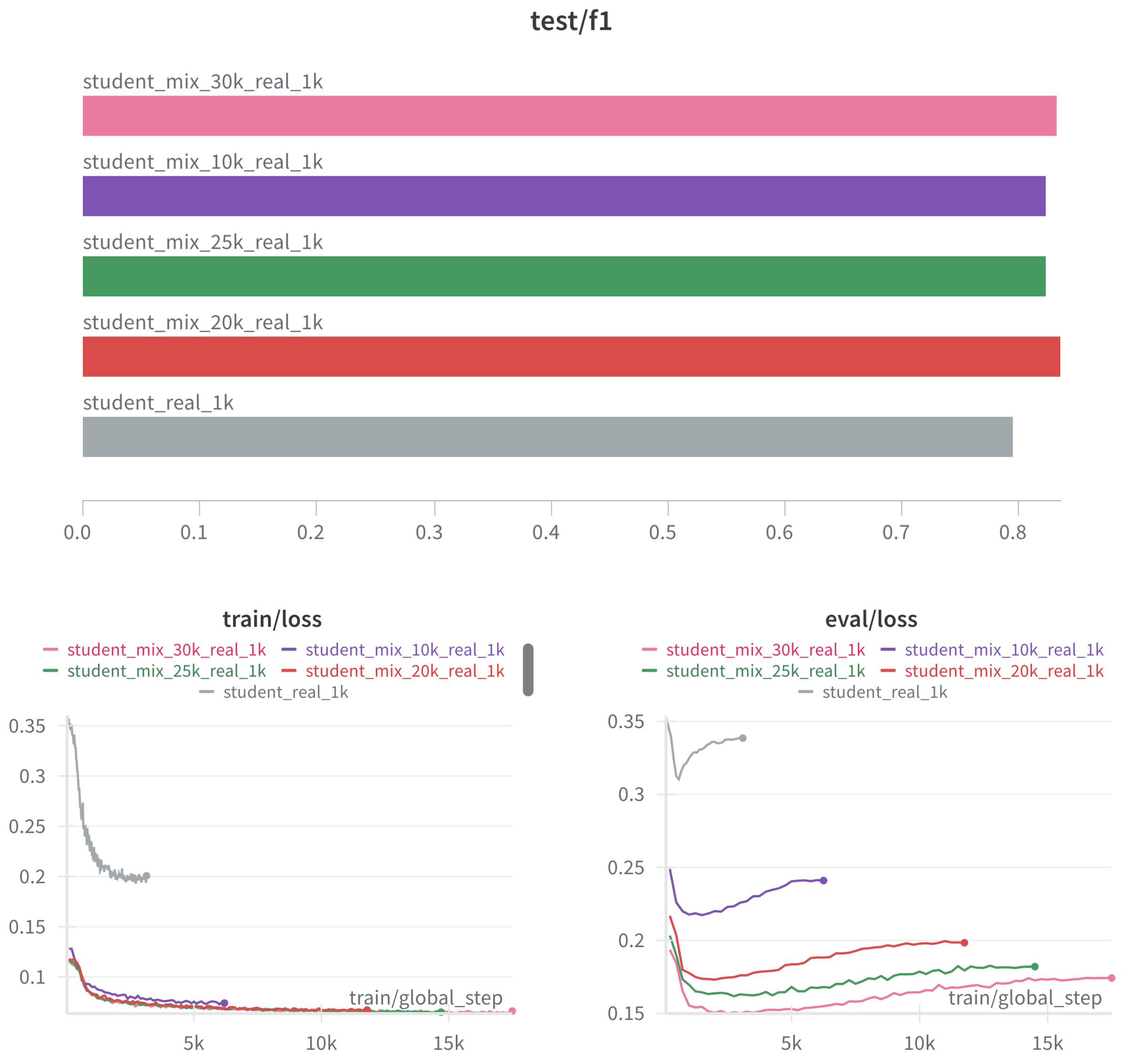


Figure 4: Macro F_1 vs. synthetic set size (10k–30k). Along with training and eval loss.

4.4.2 Effect of α (Hard vs. Soft Labels)

The F_1 score peaks at $\alpha = 0.5$ (Figure 5), matching our hypothesis that a balanced mix of soft and hard targets is most informative.

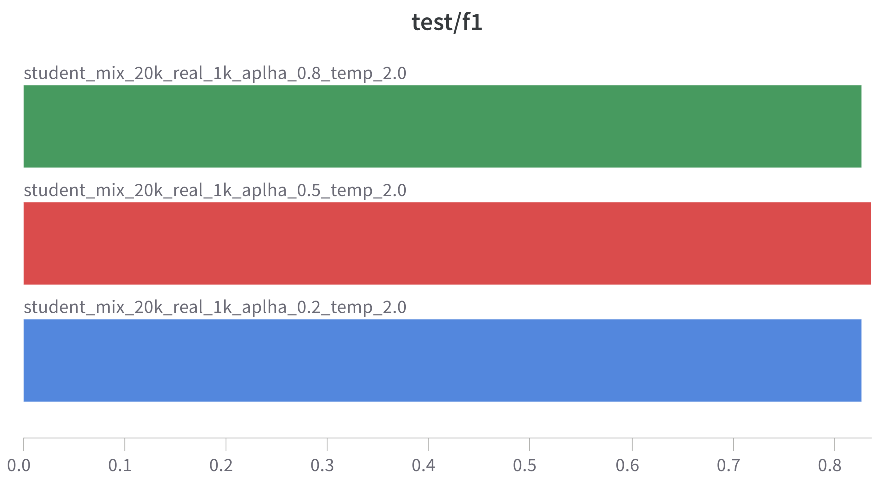


Figure 5: Student F_1 as a function of the mixing coefficient α.

4.4.3 Effect of Distillation Temperature

All runs cluster tightly (std. < 0.3 pp). However, temperature at 2.0 ever-so-slightly produces the best F_1 (0.830). Figure 6 shows those results.

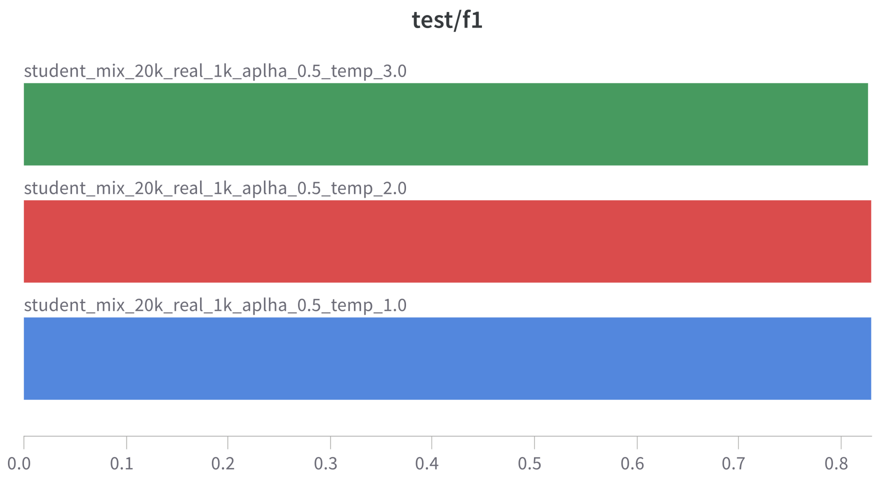
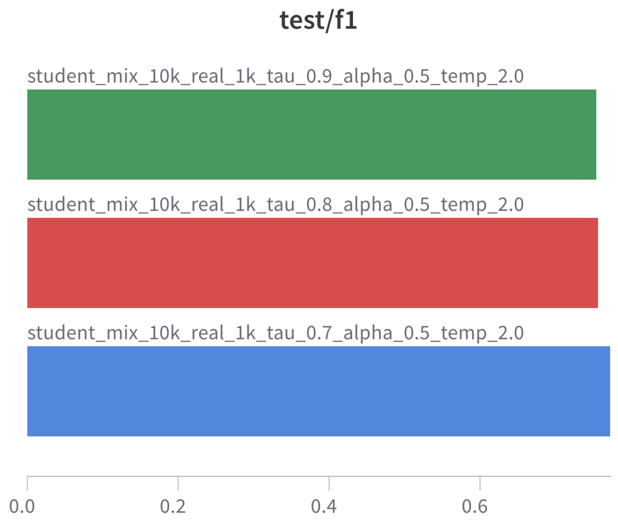


Figure 6: F_1 under different loss-temperature settings.

4.4.4 Effect of QC Threshold τ

Lowering τ to 0.7 produces the best F_1 even though diversity (*distinct-2*) dips slightly (Table inside Figure 7). We attribute the win to richer soft-label entropy, which outweighs the marginal diversity loss.

Diversity vs. confidence. Entropy of the soft labels falls from 0.681 to 0.669 as τ rises, confirming that very strict QC collapses probabilities toward one-hot vectors and weakens the distillation signal. Although distinct-n edges up by $\sim$1 pp at $\tau = 0.9$, the information loss dominates, hence the student performs best at $\tau = 0.7$ (0.771 F_1).



τ	Distinct-1	Distinct-2	Entropy
0.7	0.022382	0.262025	0.681052
0.8	0.024106	0.268781	0.675690
0.9	0.026319	0.277667	0.668949

Figure 7: Student F_1 and text-diversity as τ varies.

4.4.5 Effect of Synthetic Weight β

Down-weighting synthetic rows is beneficial: $\beta = 0.3$ beats $\{0.5,1.0\}$ by 1.8 pp on average. However, down-weighting too much hurts performance as seen in figure 8.

4.4.6 Joint Sweep: $\alpha \times \beta$

This joint sweep study was conducted to ensure that we are using the best combination of α and β values. The results are shown in figure 9. The best combination of α and β is $\alpha = 0.8$ and $\beta = 0.3$.

4.5 Error Analysis

What improved most. Long, negated sentences—a known weak spot for SST-2 models—saw the largest lift. Synthetic reviews inject new lexical cues ("did not hate", "never quite worked") that tighten the decision boundary in these regions.

Per-class gains. Both classes benefit: for the 0 label (negative) precision rises by 5 pp and recall by 6 pp, pushing macro F_1 from 0.79 to 0.84.

Remaining errors. Residual mistakes concentrate on sarcasm, idioms, and context-dependent sentiment ("still want my money back"). Manual inspection shows that many such instances also

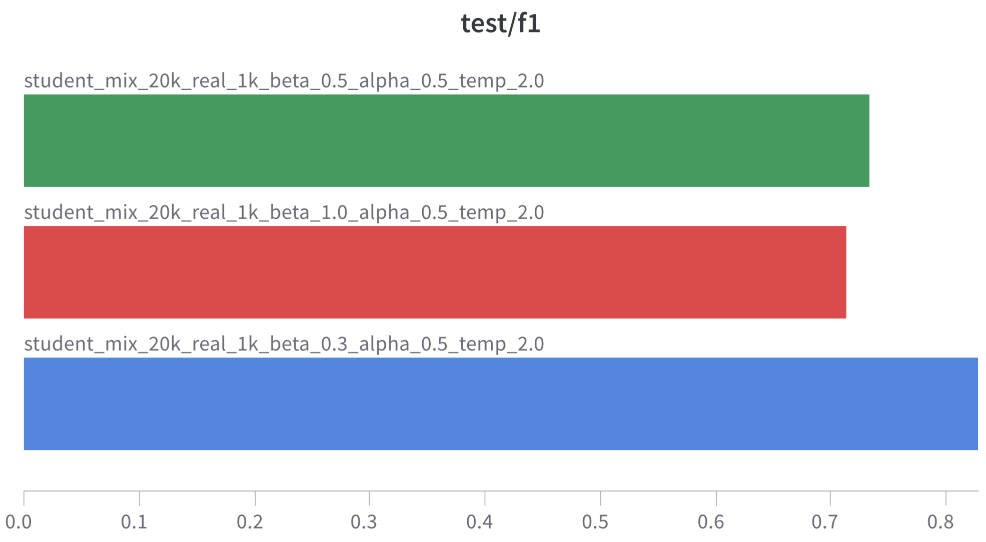


Figure 8: Performance as a function of the per-row weight β.

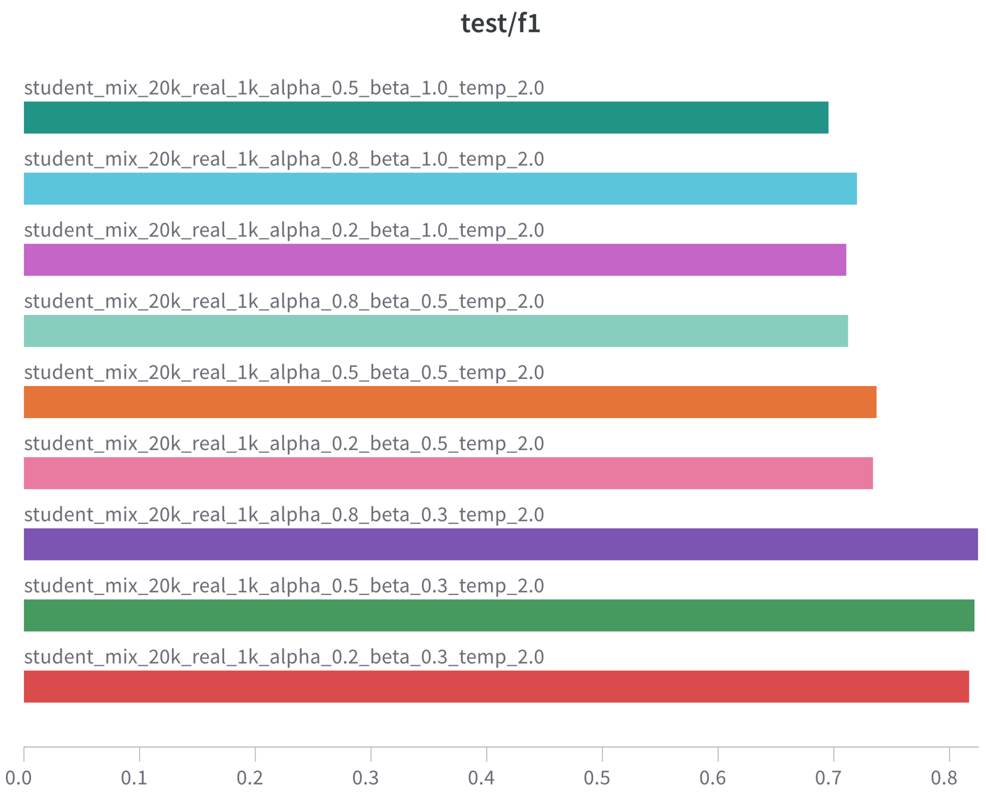


Figure 9: Performance as a function of α and β.

fool the teacher, suggesting future work on explicit sarcasm modelling or document-level context. A sample exploration of the errors is shown in table 4.

Table 4: Breakdown of residual student errors by linguistic phenomenon.

Text	**True**	**Pred**	**Phenomenon**
covered earlier and much better	0	1	comparative
what does nt́ this film have that an impression...	1	0	negation
the campy results make mel brooks...	1	0	intertextual reference
than indecent proposal	1	0	comparative
the mystery of enigma is how a rich historical...	0	1	long
that flow through the hollywood pipeline witho...	1	0	negation
it is too bad that this likable movie is nt́ more...	1	0	negation
joyless	0	1	lexical brevity
ready–made midnight movie	1	0	idiom
a third–person story now, told by Hollywood,...	0	1	long
nickelodeon–esque kiddie flick	1	0	neologism
offers no new insight	0	1	negation
who needs mind–bending drugs when they can see...	1	0	rhetorical question
pale successor	0	1	idiom
still want my money back.	0	1	sarcasm
for nearly three hours	1	0	temporal
amy and matthew have a bit of a phony relation...	0	1	long
classy	1	0	lexical brevity
wo nt́ have any trouble getting kids to eat up...	1	0	negation
a movie with depth	1	0	idiom

5 Discussion

5.1 Impact of Teacher–Verified Synthetic Data

The headline finding is that teacher–filtered synthetic data closes ˜5 pp of the performance gap between the 1 000–row seed model and a fully supervised baseline, while requiring one–tenth of the human annotation budget. Ablations in Sec. 4.4 confirm that *unverified* augmentation actually hurts F_1 by up to 2 pp, underlining the importance of the verification loop. In short, the teacher acts as a high–precision gatekeeper, converting the raw fluency of GPT–2 into task–useful data.

5.2 Precision–Recall Trade–offs

Figure 7 visualises the trade–off space induced by the confidence threshold τ. Lower τ broadens recall in the synthetic set at the cost of precision; higher τ does the opposite. Interestingly the sweet spot sits at $\tau = 0.7$, not the commonly used 0.9. We attribute this to two factors:

1. higher entropy in the teacher logits, which gives the student richer gradient information,
2. diminishing returns from the marginal gain in label purity above 0.8.

5.3 Quality–Control Effectiveness

The three filters in Stage C (teacher check, length, ASCII printable) reject 32 % of the raw generations. Manual inspection of 100 rejects finds that 74 fall into obvious error categories (e.g. sentiment drift, malformed tokens), suggesting the QC pipeline is removing genuine noise rather than useful diversity. Moreover, distinct–2 increases marginally with stricter τ, so diversity does not suffer from verification per se but from the entropy collapse that accompanies very high confidence levels.

5.4 Limitations

- **Single dataset.** All experiments use SST–2; transfer to other domains or multilingual settings is untested.
- **Teacher reliability.** The teacher is fine–tuned on only 1 k rows. Its own biases propagate into both the synthetic set and the student.
- **Compute cost.** Although the student is small, generating 20 k verified sentences takes significant compute.
- **Qualitative pruning.** The switch from the lowest–perplexity generator to a higher–perplexity but cleaner checkpoint relies on human judgement rather than an automatic metric.

5.5 Ethical Considerations and Bias Analysis

Synthetic text can amplify the biases of both the generator and the teacher. While the teacher filter reduces outright label noise, it does not neutralise sensitive attributes embedded in the generator's training data. We therefore recommend:

1. inserting a lightweight toxicity/bias classifier into Stage C,
2. auditing the synthetic corpus with crowd workers,
3. releasing all checkpoints under a license that discourages sensitive downstream use without further debiasing.

6 Future Work

1. **Sarcasm and context.** Integrate a discourse–level module or contrastive prompts to handle ironic sentiment that both teacher and student miss.
2. **Adaptive τ.** Replace the static threshold with a bandit that maximises downstream dev F_1 on–the–fly.
3. **Active learning.** Show the lowest–confidence synthetic sentences to a human annotator; combine human feedback with the KD loss.
4. **Cross–lingual transfer.** Prompt a multilingual generator (e.g. mGPT) and use a cross–lingual teacher to bootstrap low–resource languages.
5. **Fairness diagnostics.** Couple the pipeline with differential privacy or counterfactual fairness metrics to monitor bias drift as more synthetic data is added.

7 Conclusion

SentiSynth demonstrates that a tight *generate* → *verify* → *distil* loop can deliver better sentiment classification with only 1 000 human labels. Key to this success is the synergy between teacher verification, soft–label distillation, and judicious re–weighting of synthetic rows. While limitations remain (single–domain evaluation, potential bias transfer), the framework is modular: stronger teachers or smarter generators can be plugged in with minimal code changes.

A Appendix

B Repository Structure

Top-level layout (selected)

```
configs/                experiment hyper-parameters
|
+-- generator/          GPT-2 fine-tuning configs
|   +-- sst2_hf.yaml
+-- student/            DistilBERT KD configs
|   +-- real_1k.yaml
|   +-- student_mix.yaml
+-- synthetic/          synthetic-dataset generation
|   +-- conf.yaml
+-- teacher/            DeBERTa fine-tuning configs
    +-- deberta_large_sst2_mps.yaml
    +-- sst2_hf.yaml

src/                    project source code
|
+-- cli/                end-to-end command-line pipelines
|   +-- 01_train_teacher.py
|   +-- 02_fine_tune_generator.py
|   +-- 03_create_synthetic_dataset.py
|   +-- 04_build_student_datasets.py
|   +-- 05_train_student.py
|
+-- training/           custom Trainer subclasses
|   +-- student_weighted_soft_trainer.py
|
+-- utils/              helpers (metrics, prompts, wandb)
|   +-- metrics.py
|   +-- prompts.py
|   +-- wandb_setup.py
|
+-- data.py             shared dataset utilities
+-- models.py           model builder functions
+-- nb.py               notebook helper for quick sampling
```

Reproducibility shortcuts

```
python src/cli/01_train_teacher.py       configs/teacher/sst2_hf.yaml
python src/cli/02_fine_tune_generator.py  configs/generator/sst2_hf.yaml
python src/cli/03_create_synthetic_dataset.py  configs/synthetic/conf.yaml
python src/cli/04_build_student_datasets.py    ...
python src/cli/05_train_student.py       configs/student/student_mix.yaml
```

See the YAML files under `configs/` for per-stage hyper-parameters and output paths.

C Example Dataset Shape

When we generate the synthetic dataset, we save it in a JSONL format. This is an example of what it looks like:

```
==================================================
First 10 samples of train:
==================================================

#    Text                                                 Label
---- ---------------------------------------------------- ------
1    a must read for anyone who about to bolt the theat 1
2    a compelling story with lots of intrigue suspense  1
3    one of the best in recent memory latest film and i 1
4    the movie is too long and dull it like trying to m 0
5    a half hour long plot is flat with nothing new to  0
6    the film characters and their actions are so detai 0
7    the film star is n't very bright but it does its b 1
8    an exhilarating documentary that has you in the dr 1
9    a film so bad it not even funny '' i mean if you w 0

 Soft Labels                              Weight  is_Synth
---------------------------------------- -------- -----
[0.3782115876674652, 0.6217884421348572]  0.30     1
[0.37769943475723267, 0.6223005652427673] 0.30     1
[0.37762364745140076, 0.6223763227462769] 0.30     1
[0.6219679713249207, 0.37803205847740173] 0.30     1
[0.622056245803833, 0.37794366478919983]  0.30     1
[0.6173325777053833, 0.3826673924922943]  0.30     1
[0.37869372963905334, 0.621306300163269]  0.30     1
[0.37759384512901306, 0.6224061250686646] 0.30     1
[0.6220729947090149, 0.3779270350933075]  0.30     1
```

D Full Metrics Logs

The full metrics logs for the teacher, generator and student models are available at the following weights and biases links:

- `https://api.wandb.ai/links/paramkpr-reed-college/1x1fwy5w`

- `https://api.wandb.ai/links/paramkpr-reed-college/4dk0u510`
- `https://api.wandb.ai/links/paramkpr-reed-college/xzxetj1g`
- `https://api.wandb.ai/links/paramkpr-reed-college/0qq2jnh3`
- `https://api.wandb.ai/links/paramkpr-reed-college/ynsp9i59`
- `https://api.wandb.ai/links/paramkpr-reed-college/w5i5m8gj`

References

[1] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 440–447, Prague, Czech Republic, 2007. Association for Computational Linguistics.

[2] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[3] Kapur. Pending. 2002.

[4] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2014.

[5] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2002.

[6] Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2013.

[7] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[8] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2016.

[9] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2013.

[10] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, 2019.

[11] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, USA, 2020.

[12] Yizhe Zhang, Zhe Gan, and Lawrence Carin. Generating text via adversarial training. In *Proceedings of the NIPS Workshop on Adversarial Training*, Barcelona, Spain, 2016.