Distilling Large Language Models for Twitter Sentiment Analysis

Yufeng Wu
Williams College sw20@williams.edu

Emma Neil
Williams College emn3@williams.edu

Abstract

We implement Hinton et al's "knowledge distillation" technique using soft and hard labels to train a small, Deep Averaging Network (15,000 parameters) for sentiment analysis from a large, roBERTa-based model (123 million parameters) using Twitter data. We find some significant improvement in the accuracy of a student model distilled from a large teacher model over a model trained without teacher data.

1 Introduction

Model distillation describes the process of compressing a large, complex model ("teacher model") into a smaller model ("student model") with minimal loss in performance. The promise of model distillation is that by asking the student model to mimic the behavior of a giant, pre-trained teacher model, the student model is able to recover some of the performance of the teacher model and beat a baseline approach that does not rely on a teacher model.

There are increasingly many reasons to develop smaller language models. First, compressed versions of large language models are more widely accessible–they can be run by organizations and individuals with fewer computational and financial resources. Second, running inference on compute-intensive models has a large carbon footprint. Powering and maintaining data centers for large language models requires huge amounts of electricity and water. Given the urgent threat of climate change, smaller models have the benefits of using less (fossil-fuel-burning) energy. Third, in smaller machines, such as IoT devices, it is only possible to deploy smaller models. As a result, having well-performing small models can empower more downstream NLP applications.

In this paper, we implement a two-layer Deep Averaging Network (DAN) for sentiment analysis which is distilled from the pre-trained Twitter-roBERTa-base-sentiment (Loureiro et al., 2022). We evaluate this model by comparing it with three baseline models: 1) Logistic Regression with Bag-of-Words assumption, 2) Logistic Regression with GloVE-embedded input, and 3) Non-distilled DAN. Our evaluation finds 2.7% improvement in the test accuracy of a student model distilled from the teacher model over a model trained without distillation. From a teacher model that achieves 72.9% accuracy on the test dataset, we build a distilled student model capable of correctly predicting sentiment for 62.9% of test tweets.

2 Related Work

Knowledge distillation is a special branch of model distillation. Other distillation methods include weight-sharing, low-rank factorization, pruning, and quantization (Xu and McAuley, 2022). In our project, we test the assumption that soft labels act as regularizers in the learning process, as suggested by Hinton et al (Hinton et al., 2015), and compare its accuracy to a baseline model of the same architecture with a normal, non-distilled loss.

In their landmark paper, Distilling the Knowledge in a Neural Network, Hinton et al. discuss a new method for transferring knowledge from a large neural network to a much shallower one (Hinton et al., 2015). This method relies on two important concepts: soft labels and hard labels. The soft label is a vector of probabilities over the output classes created by passing the last layer of output through the softmax function. The authors suggest a loss function consisting a weighted sum of the loss with respect to soft and hard labels, so that the student model learns from both the ground truth labels and the behavior of the teacher model. The authors suggest that the use of soft labels prevents small, specialist student models from over-fitting.

3 Dataset

We use TweetEval (Barbieri et al., 2020), a commonly used Twitter benchmark composed of seven heterogeneous tweet classification tasks, including irony, hate, offensive, stance, emoji, emotion, and sentiment, all framed as multi-class classification problems.

In our project, we only use the sentiment section of the dataset, which contains train, test, and development sets with 45,615, 12,284, and 2,000 rows respectively. Each row contains a tweet and a sentiment label that is either 0 (negative), 1 (neutral), or 2 (positive).

Dataset	Negative	Neutral	Positive
Train	15.5%	45.3%	39.2%
Dev	15.6%	43.4%	41.0%
Test	32.3%	48.3%	19.4%

Table 1: Percentage of sentiment labels in train, dev, and test Sets. E.g., the first cell indicates that 15.5% of the tweets in the training set are labeled as negative.

Notice the class imbalance issue in the dataset as outlined in Tabel 1. In the train and development sets, the tweets are predominantly neutral and positive. In the test set, the class imbalance switches, so the majority of tweets are neutral and negative. This feature of the dataset allows us to better assess the robustness of our models.

Previous work have shown that using the same data that the teacher is trained on for distillation results in better accuracy for the student model (Hinton et al., 2015). Since the pre-trained teacher model uses the TweetEval dataset (Barbieri et al., 2020), we also use it to distill the student. The teacher achieved 72.9% accuracy on its test set. Our goal is to distill a small model that tries to accomplish similar accuracy as the teacher model.

4 Methods

We implement knowledge distillation technique (Hinton et al., 2015) to compress a pre-trained large language model trained for sentiment analysis into a smaller architecture (Iyyer et al., 2015). In order to evaluate the effectiveness of this method, we also train three baseline models:

- Logistic Regression with Bag-of-Words assumption
- Logistic Regression with vector-embedded input
- Non-distilled DAN of same architecture as the distilled student model

Tokenization Analyzing tweets requires a carefully chosen tokenizer, since they often contain misspelled words, made-up words, hashtags, emojis, user handles, and other symbols. It is important to choose a tokenization method which recognizes hashtags as single entities.

The large model we chose, cardiffnlp/twitter-roberta-base-sentiment available on HuggingFace (Barbieri et al., 2020), tokenizes input using a non-publicly-available built-in tokenizer which outputs encoded "input_ids" and an "attention_mask" for its transformer architecture. Because the tokenizer's encoding mechanism is not documented, we were not able to directly use the teacher's tokenizer to process the input data during student training and testing. Instead, we use NLTK's Twitter Tokenizer with a flag to remove handles from text (denoted by "@" before a word) (Bird et al., 2009). It is not ideal to compare a student's sentiment classification of a tweet to a teacher's performance on the same tweet if the two models tokenize differently, since the performance gap between the student and the teacher might be attributable to not just their architectural differences and the distillation process, but also differences in tokenization. This is a limitation of our project. The following example shows how our chosen NLTK Tokenizer works:

Input tweet: "*described the whole plot of #FantasticBeasts to baba* In the end he's like: ’so, where was Harry?’ 😩"

Output tokens: [’*’, ’described’, ’the’, ’whole’, ’plot’, ’of’, ’#fantasticbeasts’, ’to’, ’baba’, ’*’, ’in’, ’the’, ’end’, "he's", ’like’, ’:’, "’", ’so’, ’,’, ’where’, ’was’, ’harry’, ’?’, "’", ’😩’]

The tokenized tweets have the following out-of-vocabulary (OOV) rates on the pre-trained GloVE embeddings, which we discuss in detail below.

Dataset	% OOV
Train	8.43%
Dev	8.29%
Test	11.51%

Table 2: Percentage of out-of-vocabulary tokens in train, dev, and test sets on GloVE embeddings, after tokenized the all inputs with the NLTK's Twitter Tokenizer.

Embedding We embed tokenized input into a vector space for each example in the TweetEval dataset. We chose to use a version of of pre-trained Global Vectors for Word Representation (GloVE) embeddings specifically trained on a dataset of 2 billion Tweets (Pennington et al., 2014). GloVE is an unsupervised learning algorithm for obtaining vector embeddings, trained on a matrix of token co-occurence in a corpus. We chose a version of this pre-trained, twitter-specific embedding tool which encodes each token into a 50-dimensional vector. Other available versions include 100-, 200-, and 25-dimensional vectors, but we chose the 50-dimensional version because it encodes enough contextual information and also allows us to keep the number of parameters in the student model relatively small.

To standardize the the input length of tweets in our dataset, we set the maximum number of tokens in each tweet to be 128. Any token arrays with fewer than 128 tokens are "padded" with the "<PAD>" token. Any token arrays with more than 128 tokens are truncated so that only the first 128 tokens are embedded. Furthermore, any tokens outside of the GloVE vocabulary are embedded as a "the" token. This is a heuristic–"the" tends to be neutral and should not sway the overall sentiment of a tweet.

Teacher Model We use Twitter-roBERTa-base-sentiment trained on approximately 124 million tweets and finetuned for English sentiment analysis with the TweetEval dataset, as explained in Section 3 (Loureiro et al., 2022). The model outputs labels 0 (negative), 1 (neutral), or 2 (positive) and achieves 72.9% accuracy on the test set of TweetEval. The model contains approximately 123 million parameters. roBERTa (robustly optimized BERT Pretraining Approach) (Liu et al., 2019) is a version of BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) which is trained for longer on bigger batches, longer sequences, and with more data. Both BERT and roBERTa use a transformer architecture with token masking during training time. Unlike BERT, roBERTa does not perform next sentence prediction, and its token masking pattern changes dynamically during training.

Student Model We implement a Deep Averaging Network (DAN), a neural network architecture proposed by Iyyer et al. for sentiment analysis (Iyyer et al., 2015). The authors suggest applying an average, an unordered composition function, to the text embedding input. We use this technique, but because of time constraints chose not to implement the paper's second major contribution, word dropout, which randomly drops some tokens before computing the sentence average embedding. Our final student model contains an input layer of dimension 50, two hidden layers, of dimension 128 and 64, respectively, and an output layer of dimension 3. We also apply dropout on edge weights throughout the network at a rate of 25%. We use mini-batch gradient descent with a batch size of 100 and apply an Adam optimizer which adapts learning rates for each parameter for faster training (Kingma and Ba, 2017). As for linear activation function, we choose to use a leaky ReLU, whose curve does not contain zero-slope values. Like the teacher model, the final layer of the student model outputs a probability distribution over labels 0 (negative), 1 (neutral), and 2 (positive).

The student model is trained to minimize the distillation loss, a weighted average of the cross-entropy loss with respect to the teacher's soft labels and the cross-entropy loss with respect to the hard labels. See Section 2 for more information on soft and hard labels. The equation for the loss, then, is as follows.

total_loss $= soft\_loss + hard\_loss$
soft_loss $= (1 - soft\_label\_weight) *$
$loss\_fn(log\_probs\_batch, Y\_batch)$
hard_loss $= soft\_label\_weight *$
$loss\_fn(log\_probs\_batch, soft\_labels\_batch)$,
loss_fn $=$ Cross Entropy Loss

Baseline Models

Logistic Regression with BOW The first model is a simple, multi-class logistic regression with Bag-of-Words inputs. We use sklearn's CountVectorizer (part of the feature_extraction.text package) to convert an example into a sparse matrix of token counts based on the vocabulary of the training corpus (Pedregosa et al., 2011). We use sklearn's LogisticRegression (part of the linear_model package) with the default solver (limited-memory BFGS; calculates approximation to the Hessian) and a maximum of 5,000 iterations (Pedregosa et al., 2011).

Logistic Regression, GloVE The second model is the same as the previous baseline, with the exception that it takes in GloVE-embedded input.

Non-distilled DAN Finally, we train a model of the same architecture as the student model to isolate the effect of distillation on the student's performance.

Fine-tuning We fine-tune the hyper-parameters of our student DAN using grid-search. We searched through 192 combinations of hyper-parameters, including learning rate, dropout probability, soft label weight, hidden layer sizes, and batch size. Then, the hyper-parameter sets are ranked based on the best dev accuracy achieved during the gradient descent process. The five best combinations of hyper-parameters are shown in Table 3.

5 Results and Evaluation

Table 3 shows the top 5 hyper-parameter configurations with the highest accuracy on the development set. Each achieved its highest accuracy on the development set at an iteration before 10,000, indicating that 10,000 iterations of training is enough for these models to converge. For reproducibility, we set the random seed of PyTorch and Numpy as a fixed integer value in our code. We proceed with the set of hyper-parameters that achieved the highest development accuracy of 62.7%, as shown in the first row of Table 3.

Next, we fix all the hyper-parameters as indicated above except for soft label weight and run another experiment to analyze how the model accuracy change as we turn the soft label weight higher or lower. Figure 1 shows how the train and development accuracy changes as the soft label weight increases from 0 to 1, with step size equals 0.1. When the soft label weight is set to 0, the model completely ignores information from the soft labels generated by the teacher model. Hence, setting the soft label weight parameter to 0 creates a baseline Deep Averaging Network (DAN) that does not use the distillation technique, while the rest of the model configuration is exactly the same as the student model trained via distillation. When the soft label weight is set to 1, the student model's only objective becomes minimizing the differences between its predictions (each as a set of softmax probabilities across three labels) to the soft labels. In other words, when the soft label weight is 1, the model ignores information from the hard labels. When the weight is set in between 0 and 1, the student model learns a weighted combination from both soft and hard labels.

As Figure 1 suggests, varying the soft label weight does not significantly impact the model's accuracy, at least within the range we explored. Compared to the baseline DAN with soft label weight equal to 0, the best distillation model improves

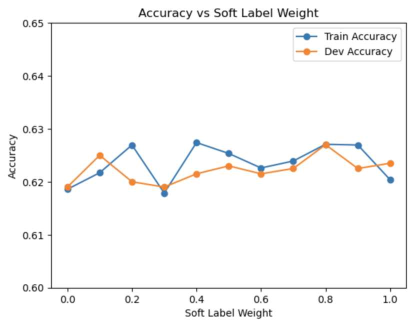

Figure 1: Training student model with different soft label weights, fixing other hyper-parameters as the best set found by grid search

the development accuracy by approximately 0.004 (0.4%). Therefore, the effectiveness of model distillation seems very little in our experimental setup.

Moving forward, we continue to use the best hyper-parameter configuration specified in the first row of Table 3, since the additional grid-search above on soft label weights also suggests 0.8 to be the optimal value for our model.

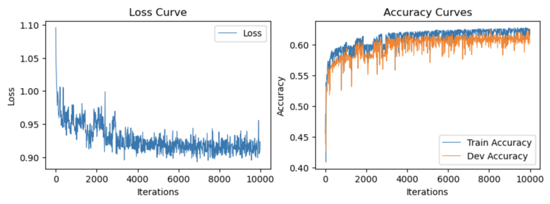

Figure 2: Loss Curve and Accuracy Curves of the final student model

Figure 2 shows the loss curve and accuracy curves during the training process of the final student model. Fluctuations in loss and accuracy are expected since we are using mini-batch stochastic gradient descent, which does not guarantee a decrease of loss over the entire training data at each step. The model's performance noticeably improves from iteration 0 to 3,000 and begins to plateau beyond iteration 5,000, confirming that the training process indeed converges within 10,000 iterations.

Table 4 compares the performance of the best student model and the three baselines. Surprisingly, logistic regression using a Bag-of-Words assumption achieves the highest development accuracy at 67.8%. However, while notably worse than baseline 1 on the development set, the distilled student model achieves the highest accuracy on the test set. We hypothesize that we see a better training and de-

Ranking	LR	DP	SLW	BSize	Dim1	Dim2	Train Acc.	Dev Acc.	Iter.
1	0.01	0.25	0.8	1000	128	64	62.6%	62.7%	9840
2	0.01	0.00	0.8	1000	64	32	62.1%	62.6%	6800
3	0.01	0.00	0.8	1000	128	32	62.9%	62.6%	9710
4	0.01	0.25	0.2	500	64	64	62.0%	62.5%	8500
5	0.01	0.00	0.8	1000	64	64	62.7%	62.5%	8010

Table 3: Top 5 Hyper-parameter Configurations that achieved the highest accuracy on the development (dev) set. LR: Learning Rate, DP: Dropout Probability, SLW: Soft Label Weight, BSize: Batch Size, Dim1: Hidden Layer 1 Dimension, Dim2: Hidden Layer 2 Dimension, Acc.: Accuracy, Dev Iter.: Iteration at which the highest Dev Accuracy is achieved.

velopment accuracy on baseline 1 but a higher test accuracy on our distilled model for the following reasons.

First, in all models except for baseline 1, we convert each token into a vector embedding and then average the token embeddings across each tweet to create a sentence embedding. This averaging process may dilute the signals in a tweet that express particularly strong sentiment.

Second, as presented in Table 1, the test set has a different distribution of class labels compared to the training and development sets. Since unregularized logistic regression is prone to over-fitting compared to a more complex model such as a feed-forward neural network, it is reasonable that baseline 1 has a significant decrease in performance on the test set.

To further evaluate these four models, we compute the error rate of each model on each sentiment class, as presented in Figure 3. For each class label of a particular model, its error rate is calculated as the number of false negatives divided by the number of true positives of that class. In other words, the error rate indicates the proportion of missed predictions for given sentiment class. A lower error rate indicates a superior ability of the model to accurately classify instances of that sentiment.

All models have a significantly lower error rate for predicting the neutral class, which is expected because there are close to 50% neutral examples in train and development sets, as shown in Table 1. Therefore, all four models have seen relatively more neutral examples and are incentivized to prioritize identifying neutral classes in order to achieve a higher accuracy. Due to class imbalance, all models struggle to identify negative labels.

In addition, a major difference between the Best Student Model (distilled DAN) and Baseline 3 (non-distilled DAN) is that the distilled version produces a more balanced error rate across the three

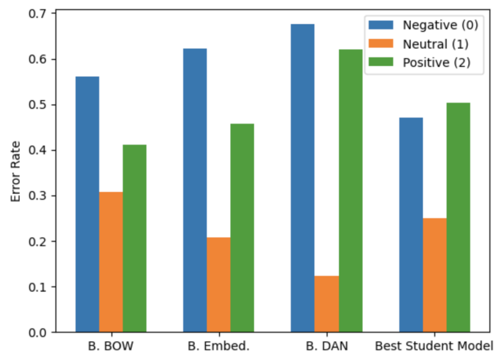

Figure 3: Class-specific error rates of different models. B.: Baseline, Embed.: Vector Embedding

sentiment classes. This could be explained by the fact that the student model learns partially from the teacher model. The teacher model is more robust as it is over-parameterized and fine-tuned from a general-purpose model. Hence, as Hinton et al. suggest, by emulating the soft labels of the teacher model, the student model acquires some degree of its robustness (Hinton et al., 2015). In contrast, the non-distilled Baseline DAN is less robust due to class imbalance within the training data. Consequently, it falls short of the Best Student Model's performance on test set, even though they achieved very similar accuracy (less than 1% difference) on the training and development sets.

6 Conclusion

While the baseline logistic regression model with Bag-of-Words input achieved the highest accuracy on the training and development sets, the distilled DAN outperformed all baselines on the test set. After analyzing the class distribution over the training, development, and test sets, we conclude that the superior performance of the distilled DAN can be attributed to its enhanced robustness through distillation.

Model	**Train Acc.**	**Dev Acc.**	**Test Acc.**
Baseline 1: Logistic + BOW	**90.9%**	**67.8%**	59.0%
Baseline 2: Logistic + Vector Embedding	61.5%	61.2%	61.0%
Baseline 3: Non-Distilled DAN	62.4%	61.9%	60.2%
Best Student Model (Distilled DAN)	62.6%	62.7%	**62.9%**

Table 4: Performance of Different Models. Acc.: Accuracy, Logistic: Logistic Regression, BOW: Bag-of-Words, DAN: Deep Averaging Network

Limitations As mentioned in 4, we were unable to use the same tokenizer for the teacher and student models. Furthermore, 8-11% of tokens in the input data were not in the vocabulary of the pre-trained GloVE embeddings. It is possible that with identical tokenization and a lower percentage of out-of-vocabulary tokens, our distilled student model may have performed better. Similarly, while we believe that using the GloVE embedding for "the" as the placeholder for all out-of-vocabulary tokens would not affect the predicted sentiment, choosing a different token might shift model output.

Future Work We are interested in the possibility of building a multi-task model which is distilled from multiple large, single-task language models. This type of model inspired by Li et al.'s 2022 paper An Unsupervised Multiple-Task and Multiple-Teacher Model for Cross-lingual Named Entity Recognition (Li et al., 2022). The authors propose a two-teacher model for training a named-entity recognition model to understand entities in two languages. The student model loss function is compared to a weighted average of the teachers hard and soft labels. Such a model is a natural extension of our project.

Because of time constraints, we did not test all reasonable hyper-parameter combinations. We are interested in examining the accuracy of a Distilled Deep Averaging Network given the following variable hyper-parameters:

- Number of hidden layers (3-5)
- Dropout on embedded input layer (Iyyer et al., 2015)
- Dropout weight on hidden layers (more variation)

We also propose a few measures to further test the performance of a distilled model. First, we suggest future researchers to re-weight the sentiment classes to address the dataset's class imbalance and the resulting off-balanced error rates. Second, we suggest training a student model of a different architecture or with different optimizations. For example, perhaps a different composition function would retain more information from the input text, or a higher number of hidden layers would improve training accuracy. Third, we suspect that the working with non-Twitter data might yield better results because the training corpus would likely contain fewer out-of-vocabulary and non-word tokens.

7 Acknowledgements

We thank Prof. Katie Keith for distilling her knowledge into us. We thank ChatGPT for helping with LaTex table formatting.

References

Francesco Barbieri, Jose Camacho-Collados, Leonardo Neves, and Luis Espinosa-Anke. 2020. Tweeteval: Unified benchmark and comparative evaluation for tweet classification. *arXiv preprint arXiv:2010.12421*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. 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 1681–1691.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Zhuoran Li, Chunming Hu, Xiaohui Guo, Junfan Chen, Wenyi Qin, and Richong Zhang. 2022. An unsupervised multiple-task and multiple-teacher model for cross-lingual named entity recognition. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 170–179.

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.

Daniel Loureiro, Francesco Barbieri, Leonardo Neves, Luis Espinosa Anke, and Jose Camacho-Collados. 2022. Timelms: Diachronic language models from twitter. *arXiv preprint arXiv:2202.03829*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D 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.

Canwen Xu and Julian McAuley. 2022. A survey on model compression for natural language processing. *arXiv preprint arXiv:2202.07105*.