How to Select One Among All ? An Extensive Empirical Study Towards the Robustness of Knowledge Distillation in Natural Language Understanding

Tianda Li[1], **Ahmad Rashid**[1], **Aref Jafari**[1,2], **Pranav Sharma**[2], **Ali Ghodsi**[3], **Mehdi Rezagholizadeh**[1]
[1]Huawei Noah's Ark Lab
[2]David R. Cheriton School of Computer Science, University of Waterloo
[3]Department of Statistics and Actuarial Science, University of Waterloo
{tianda.li, ahmad.rashid, mehdi.rezagholizadeh}@huawei.com; aref.jafari@uwaterloo.ca

Abstract

Knowledge Distillation (KD) is a model compression algorithm that helps transfer the knowledge of a large neural network into a smaller one. Even though KD has shown promise on a wide range of Natural Language Processing (NLP) applications, little is understood about how one KD algorithm compares to another and whether these approaches can be complimentary to each other. In this work, we evaluate various KD algorithms on in-domain, out-of-domain and adversarial testing. We propose a framework to assess the adversarial robustness of multiple KD algorithms. Moreover, we introduce a new KD algorithm, Combined-KD [1], which takes advantage of two promising approaches (better training scheme and more efficient data augmentation). Our extensive experimental results show that Combined-KD achieves state-of-the-art results on the GLUE benchmark, out-of-domain generalization, and adversarial robustness compared to competitive methods.

1 Introduction

Pre-trained language models have achieved impressive results on a wide variety of NLP problems (Peters et al., 2018; Devlin et al., 2019; Liu et al., 2019; Yang et al., 2020; Radford and Narasimhan, 2018; Radford et al., 2019). The rapidly increasing parameter size, however, has not only made the training process more challenging (Ghaddar and Langlais, 2019), but has also become an obstacle when deploying these models on edge devices. To address the over-parameterization and computation cost of state-of-the-art (SOTA) pre-trained language models, KD (Hinton et al., 2015) has emerged as a widely used model compression technique in the literature (Rogers et al., 2020).

Recent work on improving KD can be categorized into two directions: 1) Designing a better training scheme to help the student model learn efficiently from the teacher model. E.g., matching the student model's intermediate weights and attention matrices with the teacher's during the training (Sun et al., 2019; Wang et al., 2020; Passban et al., 2020) or designing progressive or curriculum based learning (Jafari et al., 2021; Sun et al., 2020; Mirzadeh et al., 2020) to overcome capacity gap (Mirzadeh et al., 2020) between teacher and student models. 2) Employing data-augmentation (Jiao et al., 2020; Fu et al., 2020; Rashid et al., 2021; Kamalloo et al., 2021) to improve KD by using more diverse training data. It is difficult to compare these methods since, typically, the teachers and students are initialized differently.

The robustness of KD also requires further investigation. Recent studies have revealed that the strong performance of neural networks in NLP can be partially attributed to learning spurious statistical patterns in the training set and even the SOTA models can make mistakes if a few words in their input are replaced (Jin et al., 2019; Li et al., 2020; Gao et al., 2018; Li et al., 2019). As a result, even though KD has achieved good performance in different downstream tasks (Jiao et al., 2020; Sanh et al., 2020; Sun et al., 2020), it is desirable to investigate if these KD methods learn semantic knowledge and are robust enough to retain their performance on an out-of-domain (OOD) dataset (McCoy et al., 2019; Zhang et al., 2019a) or under an adversarial attack (Jin et al., 2019; Gao et al., 2018). It would also be desirable to evaluate if different KD algorithms are complimentary and can be combined successfully.

Our contributions in this paper are as follows:

1. We compare KD algorithms for BERT compression initialized with the same teacher and student, and rank them against one another on the GLUE benchmark.

2. We conduct OOD and adversarial evaluation

[1]We will release our code at https://github.com/huawei-noah/KD-NLP.

to investigate the robustness of KD methods.

3. We propose a unified adversarial framework (UAF) that can evaluate adversarial robustness in a multi-model setting to fairly compare different models.

4. We introduce a new KD method named Combined-KD (ComKD) by taking advantage of data-augmentation and progressive training. Results show that our proposed ComKD not only achieves a new SOTA on the GLUE benchmark, but is also more robust compared to competitive KD baselines under OOD evaluation and adversarial attacks.

2 Related Work

2.1 Unsupervised pre-training

Unsupervised pre-training (Devlin et al., 2019; Liu et al., 2019; Yang et al., 2020) has been shown to be very effective in improving the performances of a wide range of NLP problems. Model performance has scaled well with larger number of parameters and training data (Raffel et al., 2020; Brown et al., 2020; Radford et al., 2019).

2.2 Knowledge distillation (KD)

Knowledge distillation (Hinton et al., 2015; Buciluă et al., 2006; Gao et al., 2018; Kamalloo et al., 2021; Rashid et al., 2020) has emerged as an important algorithm in language model compression (Jiao et al., 2020; Sanh et al., 2020; Sun et al., 2020). In the general setting, a larger model is employed as the teacher and a smaller model as the student, and the knowledge of the teacher is transferred to the student during the KD training. Specifically, in addition to a supervised training loss, the student also considers a distillation loss over the soft target probabilities of the teacher.

Sun et al. (2019) proposed distilling intermediate layer representation in addition to the regular distillation loss. Since the teacher typically has more layers than the student, the algorithm has to decide which layers to distil from and which to skip. Passban et al. (2020) overcome this challenge by designing an attention mechanism which fuses teacher-side information and takes each layer's significance into consideration. Jafari et al. (2021) identifies the capacity gap problem (Mirzadeh et al., 2020) i.e., as the teacher increases in size (and performance), the performance of a fixed size student will initially improve and then drop down. They propose to improve KD by using temperature to anneal the teacher's output gradually, then the student will be trained following the annealed output. Rashid et al. (2021) proposed adversarial data augmentation to improve KD. They train a generator to perturb data samples so as to increase the divergence between the student and teacher output.

2.3 Model Robustness Evaluation

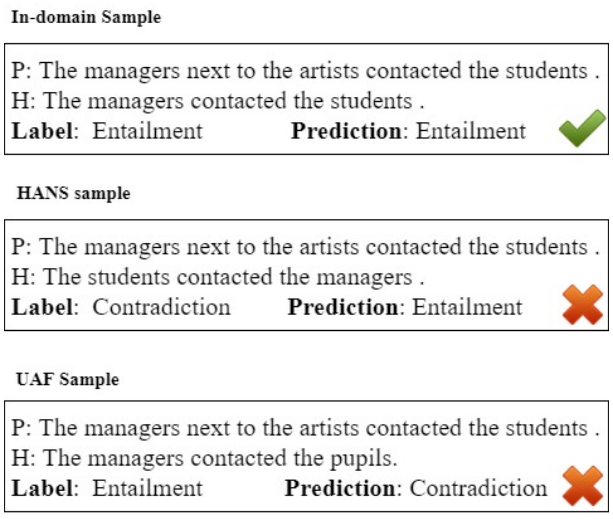
It has been demonstrated that models which are SOTA on different NLP applications, such as machine translation and natural language understanding, can be brittle to small perturbations of the data (Cheng et al., 2019; Belinkov and Bisk, 2017; McCoy et al., 2019). In our work we consider OOD tests and adversarial attacks.

2.3.1 Out-of-Domain test

The purpose of the OOD test is to change the distribution of dataset by applying fixed patterns to the original dataset. E.g., McCoy et al. (2019) used three heuristic rules to modify the MNLI evaluation set. Zhang et al. (2019a) proposed well-formed paraphrase and non-paraphrase pairs with high lexical overlap based on the original QQP (Wang et al., 2018) dataset. Glockner et al. (2018) introduced a natural language inference (NLI) test set by replacing a single word of a training instance using WordNet (Miller, 1995).

2.3.2 Adversarial Attack

Adversarial examples, which were first identified in computer vision (Goodfellow et al., 2015; Kurakin et al., 2016; Labaca-Castro et al., 2021), are small perturbations to data which are indiscernible for humans but can confuse a neural network classifier. The standard approach is to add gradient-based perturbation on continuous input spaces (Goodfellow et al., 2015; Kurakin et al., 2016). Recently, studies also explore the use of adversarial examples on NLP tasks, e.g., using adversarial examples to measure robustness against an adversarial attack (Jin et al., 2019), or adding adversarial examples during training process to help models improve in robustness and generalization (Zhu et al., 2020; Ghaddar et al., 2021a,b; Rashid et al., 2021). Jin et al. (2019) proposed a model dependent framework, textfooler, to generate adversarial samples to attack existing models. Different from previous rule-based frameworks, textfooler can automatically replace the most semantically important words based on a specific model's output.


In-domain Sample

P: The managers next to the artists contacted the students .
H: The managers contacted the students .
Label: Entailment **Prediction**: Entailment ✓

HANS sample

P: The managers next to the artists contacted the students .
H: The students contacted the managers .
Label: Contradiction **Prediction**: Entailment ✗

UAF Sample

P: The managers next to the artists contacted the students .
H: The managers contacted the pupils.
Label: Entailment **Prediction**: Contradiction ✗

Figure 1: Different model evaluations for Natural Language Inference. The model is evaluated on in-domain, out-of-domain and adversarial samples and makes an error on the latter two.

3 Methodology

First, we evaluate different KD algorithms on three kinds of test sets. In-domain testing where the training and test sets are from the same distribution, OOD test where the test set is specially designed to diagnose whether the model overfits to spurious lexical patterns and an adversarial test set to measure robustness to adversarial examples. Then we present our ComKD algorithm.

In figure 1, we present an example from Natural Language Inference where the model can predict an in-domain sample correctly, but makes mistakes on OOD (HANS) and adversarial (UAF) evaluation. Specifically, in this example, the HANS sample changes the object and the subject whereas the UAF sample replaces the most semantically important word.

3.1 In-domain test

We train on the GLUE benchmark (Wang et al., 2018) and use the provided evaluation sets as our in-domain test datasets. We evaluate both on the GLUE dev set and the test set.

3.2 Out-of-Domain test

We employ HANS (Zhang et al., 2019a) and PAWS (Zhang et al., 2019b) as our OOD test set. Models are trained on MNLI and QQP datasets respectively.

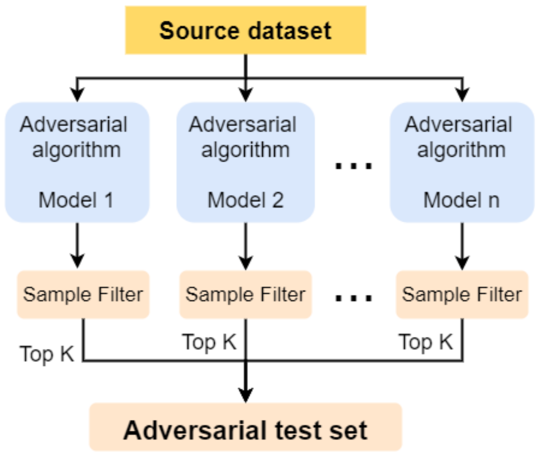


Figure 2: Unified Adversarial Framework

3.3 Adversarial Test

Adversarial attack is an effective way to test the robustness of a model. Current adversarial attacks, however, focus on single model attacks which can not be used to draw a comparison between different models directly. To deal with this, we propose a unified adversarial framework (UAF), presented in figure 2, which can help us fairly compare different KD algorithms with the same adversarial attack.

After selecting the adversarial algorithm and source dataset, each model that is used in the evaluation will apply the same adversarial algorithm to generate adversarial samples. To keep the quality of generated samples during the generation, a quality score will be employed to rank the adversarial samples. The quality score will be computed follow the function below:

$$Score = cos(Model_n(X), Model_n(X')), \quad (1)$$

Where $Model_n$ is the model that generates the adversarial sample, X is the original sample, X' is the generated adversarial sample. We calculate the cosine distance between the hidden state of the two [CLS] tokens to get the quality score. Intuitively, adversarial examples are not expected to be too similar to the original sample or the models can easily distinguish it. On the other hand, the adversarial example can not be too distant from the original sample, as it will compromise model quality. As a result, for the sample filter step, two threshold values λ_{up} and λ_{down} will be used to filter the adversarial samples. Only the sample with a quality score in the range of $(\lambda_{down}, \lambda_{up}]$ will be kept. Finally, we collect top K best samples from each sample filter to complete our adversarial test set.

3.4 Combined Knowledge Distillation

According to the current results in the literature, MATE-KD (Rashid et al., 2021) and Annealing-KD (Jafari et al., 2021) are two of the best methods from the two family of KD algorithms. We will attempt to combine their strengths and evaluate whether it will improve the performance overall.

The teacher logit annealing of Annealing-KD is specially interesting because it addresses the capacity gap problem. Pre-trained language models are constantly increasing in size and larger model tend to perform better on downstream tasks. As demonstrated by Mirzadeh et al. (2020), the performance of a fixed student does not necessarily scale with a better teacher. The adversarial algorithm in MATE-KD, on the other hand, augments data which is designed to probe parts of the teacher function not explored by the training data. Other advantages of these methods are that they only distil the teacher logits (as opposed to the weights and attention maps), do not introduce additional hyper-parameters, and perform well empirically.

We employ a masked language model (MLM) generator for data augmentation. The generator is trained to produce samples which maximize the divergence between the teacher and the student logits. Additionally, the generator fixes most of the text and only generates the masked tokens so that the text does not diverge too much from the training distribution.

The object function can be formulated as:

$$X' = G_{\phi}(Mask(X)) \tag{2}$$

$$\max_{\phi}(\mathcal{L}_G(\phi)) = D_{MSE}(T(X'), S_{\theta}(X')), \tag{3}$$

where $X = \{x_i\}_{i=1}^{T}$ is the input sequence and i is sequence length. $Mask(.)$ is a function that randomly masks tokens of the input sequence X. In practice, we mask 30% percent of tokens. $G_{\phi}(.)$ is the adversarial generator network with parameter ϕ, $T(.)$ and $S_{\theta}(.)$ are the teacher and student respectively, D_{MSE} is the mean squared error.

The student is trained in two phases. During phase 1, the student model will only learn from the teacher. During phase 2, however, the student model will learn from the ground-truth label.

In phase 1, we anneal the teacher logits inspired by Jafari et al. (2021). Note that the student logits are not annealed. The annealing schedule progressively moves from a lower temperature to a temperature of 1. We thus move from a smoother distribution to a sharper softmax distribution. For phase 1, we train to minimize the following losses:

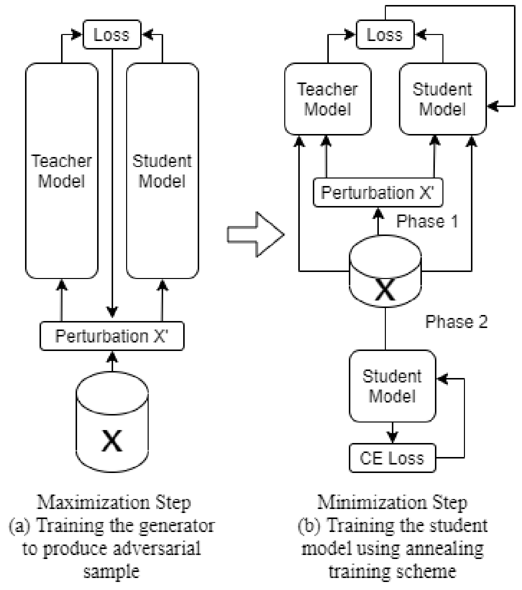


Figure 3: llustration of the maximization and minimization steps of ComKD. For Maximization step, a generator will be trained to generate adversarial samples to maximize the difference between teacher model's and student model's output. For the minimization step, the annealing training scheme will be employed and the student model will learn to match the teacher output on both the original and the perturbed input.

$$\mathcal{L}_{\text{ADV}} = D_{MSE}(t \cdot T(X'), S_{\theta}(X')), \tag{4}$$

$$\mathcal{L}_{\text{KD}} = D_{MSE}(t \cdot T(X), S_{\theta}(X)), \tag{5}$$

where T is the teacher network, S is the student network, θ is the set of student parameters, X' is the augmented sample and t is the temperature. A max_T hyperparameter will be introduced to calculate t. If the epoch number during the training is smaller than max_T, then $t = \frac{epoch}{max_T}$[2]; otherwise, $t = 1$. The total loss in this phase is $\mathcal{L}_{\text{ADV}} + \mathcal{L}_{\text{KD}}$.

For phase 2, the student model $S_{\theta}(.)$ will only learn from original data, and we employ cross entropy (CE) loss as our objective function. The complete algorithm can be found in Appendix B.

4 Experiment

4.1 Data

For in-domain test, we evaluate previous KD models as well as our proposed ComKD model on nine

[2]During the training, epoch is from 1 to $Epoch_{max}$ that set as hyperparameter.

datasets of the General Language Understanding Evaluation (GLUE) (Wang et al., 2019) benchmark which includes classification and regression tasks. These datasets can be broadly divided into 3 families of problems: 1) Single sentence tasks which include linguistic acceptability (CoLA) and sentiment analysis (SST-2). 2) Similarity and paraphrasing tasks which include paraphrasing (MRPC and QQP) and a regression task (STS-B). 3) Inference tasks which include Natural Language Inference (MNLI, WNLI, RTE) and Question Answering (QNLI).

For OOD test, we employ HANS (Zhang et al., 2019a) and PAWS (Zhang et al., 2019b) as our OOD test set. Models are first trained on MNLI and QQP dataset respectively.

For UAF test, we used GLUE benchmark dev set as our source data, and textfooler (Jin et al., 2019) as our adversarial algorithm. λ_{down} is set to 0.5 and λ_{up} is set to 0.99. Please note that the textfooler can only be applied to classification tasks, so we do not include results on STS-B. We also exclude WNLI because the dataset size is too small. The statistics of the UAF test sets are shown in Table 1. It's it notable that the size of UAF test set for BERT-base and RoBERTa-large is different because the number of models that participate in the test is also different.

4.2 Evaluation metrics

On GLUE, we follow the setting of the GLUE leaderboard (Wang et al., 2019). Specifically, CoLA is evaluated by Matthews correlation coefficient (MCC), STS-B is evaluated by Pearson correlations, MRPC is evaluated by F1 score, and the rest of the datasets are evaluated by accuracy. UAF on GLUE employs the same metrics. For OOD test, both F1 score and accuracy are used to evaluate QQP and PAWS dataset, and we use accuracy on HANS and MNLI.

	CoLA	MNLI	MRPC	QQP	QNLI	RTE	SST-2
BERT-base	1200	6000	1200	6000	6000	1200	1200
RoBERTa-large	1000	5000	1000	5000	5000	1000	1000

Table 1: Dataset size for the UAF test sets

4.3 Experimental Setting

We evaluate the different KD methods on two settings. In the first setting, the teacher model is BERT-base (Devlin et al., 2019) and the student model is initialized with the weights of DistilBERT (Sanh et al., 2020), which consists of 6 layers with a hidden dimension of 768 and 8 attention heads. We find two student model initialization strategies in the literature. Methods such as PKD (Sun et al., 2019) and ALP-KD (Passban et al., 2020) initialize the weights of the student model with a subset of the teacher weights. Other methods such as Annealing-KD (Jafari et al., 2021) and MATE-KD (Rashid et al., 2021) initialize the student model with a pre-trained one such as DistilBERT. We also present a version of ALP-KD which is initialized with a pre-trained model. The pre-trained models are taken from the authors release. The teacher and student are 110M and 66M parameters respectively with a vocabulary size of 30,522 extracted the using Byte Pair Encoding (BPE) (Sennrich et al., 2016) tokenization method.

On the second setting, the teacher model is RoBERTa-large (Liu et al., 2019) and the student is initialized with the weights of DistillRoBERTa (Sanh et al., 2020). RoBERTa-large consists of 24 layers with a hidden dimension of 1024 and 16 attention heads for a total of 355 million parameters. We use the pre-trained model from Huggingface (Wolf et al., 2019). The student consists of 6 layers, 768 hidden dimension, 8 attention heads, with 82 million parameters. Both models have a vocabulary size of 50,265 extracted using BPE. The model hyperparameter and training details are listed in Appendix A.

For the UAF tests we set K to be 1000 for the larger datasets (MNLI, QQP and QNLI) and 200 for the rest.

5 Evaluation

5.1 In-domain test

On Table 2, we present the result of the KD algorithms on GLUE when the teacher is BERT-base. We present an additional baseline for data augmentation following Rashid et al. (2021) that adds the data augmentation from TinyBERT (Jiao et al., 2020) to Vanilla-KD. We observe that all the methods improve on the Vanilla-KD results. On the methods which introduce intermediate layer distillation, ALP-KD performs better than PKD. Moreover, initializing ALP-KD with DistilBERT is better than initializing it with the teacher weights. MATE-KD, which employs adversarial data augmentation, performs the best among baseline methods followed by Annealing-KD which anneals the teacher weights. Our proposal, ComKD, which

	Method	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	Avg. score
Baseline	BERT-base	59.5	84.6	90.6	91.5	91.0	68.2	93.1	88.4	80.3
	DistilBERT	51.3	82.1	90.1	89.2	88.5	59.9	91.3	86.9	77.3
	Vanilla-KD	47.3	82.8	89.5	89.9	90.5	66.0	90.4	86.7	77.7
New training scheme	PKD	45.7	82.1	89.3	89.3	90.7	68.2	91.5	88.6	77.9
	ALP-KD	47.0	81.9	89.2	89.7	90.7	68.6	91.9	88.6	78.2
	ALP-KD (DistilBERT)	51.8	82.9	89.9	89.9	91.2	67.5	91.4	87.3	78.7
	Annealing-KD	55.2	83.8	90.2	89.8	91.2	67.9	92.1	87.5	79.3
Data augmentation	Tinybert + Aug	55.2	82.1	87.0	89.7	89.5	68.6	91.9	87.8	78.7
	MATE-KD	**60.4**	84.5	90.5	**91.2**	**91.4**	70.0	**92.2**	88.5	80.5
Ours	ComKD	59.4	**84.7**	**91.4**	90.7	**91.4**	**71.8**	91.7	**89.1**	**80.7**

Table 2: GLUE dev result for different KD models (BERT). The score for the WNLI task is 56.3 for all models and is included in Avg. score. Bold number are the best performance reached by 6-layer models in this table.

	RoBERTa-large	DistilRoBERTa	Vanilla-KD	Annealing-KD	MATE-KD	ComKD (Ours)
CoLA	68.1	59.7	60.9	61.7 (54.0)	65.9 (56.0)	**67.4 (58.6)**
RTE	86.3	69.7	71.1	73.6 (73.7)	75.0 (75.0)	**80.1 (76.6)**
MRPC	91.9	90.1	90.2	90.6 (86.0)	91.9 (**90.2**)	**93.0** (89.7)
STS-B	92.3	88.3	88.8	89.0 (86.8)	90.4 (88.0)	**91.5 (88.5)**
SST-2	96.4	89.8	92.5	93.1 (93.6)	94.1 (94.9)	**95.2 (95.1)**
QNLI	94.6	89.1	91.3	92.5 (90.8)	**94.6** (92.1)	91.7 (**92.6**)
QQP	91.5	90.4	91.6	91.5 (81.2)	91.5 (81.2)	**91.9 (81.4)**
MNLI-m	90.2	81.9	84.1	85.3 (84.4)	85.8 (85.2)	**87.2 (85.9)**
WNLI	56.3	56.3	56.3	56.3 (65.1)	56.3 (65.1)	56.3 (65.1)
Avg. score	85.3	79.5	80.8	81.4 (79.8)	82.7 (80.8)	**83.9 (81.5)**

Table 3: Dev set results on GLUE benchmark (RoBERTa). Annealing-KD, MATE-KD and ComKD results in paranthesis is the leaderboard test result. Bold number are the best performance reached by 6-layer models in this table.

	MNLI-m (Dev)	HANS	QQP-dev (Acc)	PAWSqqp(ACC)	QQP-dev (F1)	PAWSqqp (F1)
RoBERTa-large	90.2	78.2	91.5	43.3	88.8	48.8
DistilRoBERTa	83.8	58.6	91.2	34.8	88.2	44.1
MATE-KD	86.3	63.6	**92.0**	**38.3**	**89.2**	**46.4**
Annealing-KD	84.5	61.2	91.6	35.8	88.7	44.6
ComKD (Ours)	**87.2**	**68.6**	91.6	35.2	88.7	45.0

Table 4: OOD test result (Bold numbers are the best performance reached by 6-layer models in this table)

combines both adversarial data augmentation and annealing training scheme outperforms all these methods. The results of MATE-KD indicate that data augmentation is a successful strategy on all datasets and performs particularly well on the smaller ones such as CoLA, STS-B and RTE.

We evaluate the best performing baselines, Annealing-KD and MATE-KD, as well as our method on the RoBERTa setting. Here, the teacher is RoBERTa-large and the student is initialized with the weights of DistilRoBERTa. Table 3 presents the dev set results and the test set results (in paranthesis) on GLUE. We see two interesting trends; First, the results follow the same pattern as the previous setup where ComKD is the best, followed by MATE-KD, Annealing-KD and Vanilla-KD. Second, we see a larger gap between our algorithm and MATE-KD. In contrast to MATE-KD we anneal the teacher logits and this has shown to alleviate the capacity gap problem (Jafari et al., 2021), i.e. a larger capacity difference between the teacher and the student makes distillation more difficult. When learning from a larger teacher, annealing the logits as well as data augmentation both improve KD.

5.2 Out-of-Domain test

We conduct OOD test for RoBERTa-large teacher setting and the are results shown in Table 4.

Specifically, ComKD performs better than MATE-KD on HANS dataset. MATE-KD get better performance on PAWS. To some extend, data augmentation does help the model perform good on OOD tests. Here, we see that the gap between the teacher and student is much larger on the OOD

	BERT-base	DistilBERT	ALP-KD (DistilBERT)	Annealing-KD	MATE-KD	ComKD (Ours)
CoLA	39.2	24.6	26.8	25.8	35.5	**39.7**
MNLI	89.8	87.3	89.4	90.6	**91.7**	90.9
MRPC	91.8	91.0	89.0	92.0	91.3	**92.1**
QNLI	91.1	87.9	89.6	90.0	**90.8**	90.1
QQP	90.3	86.1	84.1	85.6	85.5	**87.4**
RTE	74.5	72.4	77.3	**81.2**	75.9	74.7
SST-2	84.4	82.3	82.1	83.8	83.8	**84.5**
Avg. score (by dataset)	80.2	75.9	77.0	78.4	79.2	**79.9**
Avg. score (by sample size)	86.6	83.0	83.8	84.9	85.6	**85.9**

Table 5: Unified adversarial framework test for BERT-base teacher. Bold number are the best performance reached by 6-layer models in this table.

	RoBERTa-large	DistilRoBERTa	Annealing-KD	MATE-KD	ComKD (Ours)
CoLA	14.7	5.0	2.4	**6.6**	4.9
MNLI	37.0	36.6	36.6	37.0	**37.5**
MRPC	94.9	90.7	88.7	**94.2**	93.4
QNLI	94.2	90.8	92.4	**92.9**	92.8
QQP	89.3	86.2	87.9	87.2	**88.1**
RTE	77.4	69.7	**73.4**	69.2	71.5
SST-2	87.6	81.8	81.8	82.9	**84.0**
Avg. score (by dataset)	70.7	65.8	66.2	67.1	**67.5**
Avg. score (by sample size)	72.5	69.2	70.0	70.4	**70.8**

Table 6: Unified adversarial framework test for RoBERTa-large teacher. Bold number are the best performance reached by 6-layer models in this table.

datasets compared to the in-domain testing. Thus, when evaluating the performance of KD and evaluating the gap between teacher and student, we should consider perturbed datasets in addition to the in-domain testing.

To further compare the ComKD and MATE-KD, we introduce the UAF tests which the evaluation sets are generated by each model itself.

5.3 Adversarial Attack

In order to compare the adversarial robustness of each KD method, we conduct UAF tests.

As introduced in Section 3.3 and Section 4.1, we use GLUE datasets as source data and textfooler as the adversarial algorithm. The textfooler algorithm, for a given trained model and dataset, first computes an importance score of the tokens in a sentence. A token is more important if removing it has a greater impact on the model output. Then, it replaces the important tokens with its closest synonyms. In our setting, textfooler will replace at most 15% of the tokens in a sequence with their synonyms.

Different from OOD tests which is pre-defined, the evaluation set for UAF test is generated by the tested models themselves. A robust model is expected to handle both adversarial samples that are generated by itself and adversarial samples generated by other models. It is notable that the results on BERT setting cannot be compared with the results on RoBERTa setting, because the test sets are different.

To fairly compare the model's performance, we also show two kind of average scores. The first is average by dataset and this is similar to how GLUE evaluates by averaging the performance on all datasets. The second one is average by sample size where we do a weighted average and weigh the result on each dataset by its size. Thus, larger datasets receive a greater weight.

Tables 5 and 6 present the UAF results for the BERT-base teacher setting and the RoBERTa-large teacher setting respectively. We observe that ComKD outperforms other 6-layer methods on average for both settings and achieves a higher score on four out of seven datasets on the BERT setting and three out of seven on the RoBERTa setting. Similar to the OOD results, we observe that the gap between the teacher and student is larger on the UAF test compared to the in-domain test. On

Index	Sample	Label	RoBERTa-large	DistilRoBERTa	Annealing-KD	MATE-KD	ComKD (Ours)
1	**P**: Yes , you 've done very well , young man **H**: No , you have not done very adequately	C	C	C	C	C	C
2	**P**: All of the islands are now officially and proudly part of France , not colonies as they were for some three centuries **H**: The island agreed to join France instead of being colony	E	N	N	N	N	E
3	**P**: I guess history repeats itself , Jane **H**: I truely think the past situation shown history repeats itself	E	N	N	N	E	E
4	**P**: Case study evaluations **H**: Independent cases studies assessments	E	E	N	N	N	N
5	**P**: Pretty good newspaper uh **H**: I thinks this is a good newspaper , and the comics section is my favourite	E	N	N	N	N	N

Table 7: Some error sample from RoBERTa-large teacher setting UAF test (MNLI). C is contradiction, N is neutral, E is entailment.

	RoBERTa large	DistilRoBERTa	Annealing-KD	MATE-KD	ComKD (Ours)
Add end mark	36.4	36.0	36.3	36.4	36.7
Remove end mark	37.1	36.7	36.5	37.1	37.4

Table 8: Model performance on UAF (MNLI) after remove or add end mark. (RoBERTa-large)

the BERT setting ComKD and MATE-KD achieved a higher score than the teacher.

The performance of all the KD algorithms is consistent with the trend on the in-domain testing. ALP-KD performance is again lower than the other techniques. Overall, our experiments conclude that structural approaches for fine-tuning are not as effective as data-augmentation and progressive learning.

5.4 Error Analysis

In this section, we analyze the error of UAF (MNLI) test.

Table 7 shows some of the UAF samples generated by RoBERTa based models on MNLI. For the sample 2, only ComKD can predict correctly. Even though this sample has overlap between the premise and hypothesis, these models don't predict entailment directly, which indicates the prediction decisions don't only rely on the word overlap. The semantics of words are also important.

For sample 3, both MKD and ComKD can predict correctly, and other models, however, can not. In this sample, there is a length mismatch of premise and hypothesis, as a result, it is harder to predict. For sample 4, Only RoBERTa-large can predict correctly. To get the correct prediction in this sample, the models need to understand that "evaluations" has the same meaning here as "assessments". Sample 5 is a sample that none of models predict correctly. Again, there is a length mismatch of premise and hypothesis.

We also investigate the influence of punctuation on the RoBERTa-large teacher setting. As shown in table 8, we make two variants of the UAF (MNLI) dataset. For the first setting, we add "." for all the samples that don't have end mark. For the second setting, all the samples' end mark will be removed. According to the table, the end marks do influence the performance of models. Again ComKD perform better than other models in both settings. We also list some samples to show the influence of punctuation in Appendix C.

5.5 Further Discussion

To find out how KD methods work differently, we looked at the UAF test result (Shown in Figure 9) of MNLI dataset, and further analysed the contradiction, entailment and neutral classes. We can see that *data augmentation based* KD methods (ComKD and MateKD) have higher precision on Contradiction label samples, which means that these model can not be easily confused by negation words since the recall is close for most of KD methods. We see a higher precision in entailment class for ALP-KD, Annealing-KD and ComKD. We also found that KD models perform better than finetune students on each label's f1 score. In summary, *data augmentation based* KD tend to classify Contradiction labels, on the other hand, *better training scheme* KD models prefer to classify Entailment labels. We also see the same trend on In-domain

UAF	Contradiction			Entailment			Neutral		
	R	P	F1	R	P	F1	R	P	F1
ALP-KD	0.926	0.894	0.907	0.829	**0.965**	0.892	0.923	0.839	0.879
Annealing-KD	**0.946**	0.893	0.919	0.858	0.960	0.894	0.917	0.875	0.895
BERT-base	0.937	0.882	0.908	0.844	0.949	0.894	0.907	0.864	0.885
ComKD (Ours)	0.940	0.903	0.921	0.856	0.961	0.905	**0.929**	0.869	0.897
DistilBERT	0.924	0.859	0.890	0.819	0.923	0.868	0.881	0.847	0.864
MATE-KD	**0.946**	**0.918**	**0.932**	**0.878**	0.948	**0.911**	0.923	**0.885**	**0.904**

Table 9: Models detailed performance on UAF (MNLI) test.

In-domain	Contradiction			Entailment			Neutral		
	R	P	F1	R	P	F1	R	P	F1
ALP-KD	0.847	0.850	0.848	0.812	0.892	0.850	0.831	0.753	0.790
Annealing-KD	0.857	0.847	0.852	0.839	0.885	0.861	0.818	0.782	0.799
BERT-base	**0.867**	0.858	**0.862**	0.840	0.895	0.867	0.833	0.788	0.810
ComKD (Ours)	0.859	0.861	0.860	0.842	**0.899**	**0.870**	**0.840**	0.783	**0.811**
DistilBERT	0.837	0.824	0.831	0.824	0.865	0.844	0.795	0.767	0.781
MATE-KD	0.858	**0.864**	0.861	**0.855**	0.876	0.866	0.819	**0.793**	0.806

Table 10: Models detailed performance on In-domain (MNLI) test.

test result (Shown in Figure 10). In both UAF and In-domain results, *data augmentation based* KD methods outperform *better training scheme* KD methods, which indicates that the student trained with data augmentation can achieve a better robustness compared with new training scheme strategy.

5.6 Conclusion

In this work, we conduct in-domain, OOD and UAF test to investigate the robustness of current KD methods. Results show that the KD models' are more robust than fine-tuned student models but less robust than teacher model. In general, the robustness ranking of each KD methods is consistent with GLUE benchmark average score. Specifically, the student trained with data augmentation can achieve a better robustness compared with new training scheme strategy. Moreover, we also verify that the two strategies of KD methods can be combined together to get a more robust KD model. Our newly proposed ComKD not only outperforms all of the KD methods and achieves SOTA results on the GLUE benchmark, but can also achieve better robustness according to the OOD and the UAF tests.

Acknowledgements

We thank MindSpore [3], a new deep learning computing framework, for partial support of this work.

References

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541.

Yong Cheng, Lu Jiang, and Wolfgang Macherey. 2019. Robust neural machine translation with doubly adversarial inputs. *arXiv preprint arXiv:1906.02443*.

[3]https://www.mindspore.cn/

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

Jie Fu, Xue Geng, Zhijian Duan, Bohan Zhuang, Xingdi Yuan, Adam Trischler, Jie Lin, Chris Pal, and Hao Dong. 2020. Role-wise data augmentation for knowledge distillation.

J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56.

Abbas Ghaddar and Philippe Langlais. 2019. Contextualized word representations from distant supervision with and for ner. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 101–108.

Abbas Ghaddar, Philippe Langlais, Ahmad Rashid, and Mehdi Rezagholizadeh. 2021a. Context-aware adversarial training for name regularity bias in named entity recognition. *Transactions of the Association for Computational Linguistics*, 9:586–604.

Abbas Ghaddar, Phillippe Langlais, Mehdi Rezagholizadeh, and Ahmad Rashid. 2021b. End-to-end self-debiasing framework for robust NLU training. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1923–1929, Online. Association for Computational Linguistics.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. *CoRR*, abs/1805.02266.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.

Aref Jafari, Mehdi Rezagholizadeh, Pranav Sharma, and Ali Ghodsi. 2021. Annealing knowledge distillation.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is bert really robust? natural language attack on text classification and entailment. *arXiv preprint arXiv:1907.11932*.

Ehsan Kamalloo, Mehdi Rezagholizadeh, Peyman Passban, and Ali Ghodsi. 2021. Not far away, not so close: Sample efficient nearest neighbour data augmentation via MiniMax. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3522–3533, Online. Association for Computational Linguistics.

Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *CoRR*, abs/1607.02533.

Raphael Labaca-Castro, Luis Muñoz-González, Feargus Pendlebury, Gabi Dreo Rodosek, Fabio Pierazzi, and Lorenzo Cavallaro. 2021. Universal adversarial perturbations for malware.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against bert using bert.

Tianda Li, Xiaodan Zhu, Quan Liu, Qian Chen, Zhigang Chen, and Si Wei. 2019. Several experiments on investigating pretraining and knowledge-enhanced models for natural language inference. *CoRR*, abs/1904.12104.

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.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

G. Miller. 1995. Wordnet: a lexical database for english. *Commun. ACM*, 38:39–41.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. 2020. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5191–5198.

Peyman Passban, Yimeng Wu, Mehdi Rezagholizadeh, and Qun Liu. 2020. ALP-KD: Attention-Based Layer Projection for Knowledge Distillation.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

A. Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer.

Ahmad Rashid, Vasileios Lioutas, Abbas Ghaddar, and Mehdi Rezagholizadeh. 2020. Towards zero-shot knowledge distillation for natural language processing. *arXiv preprint arXiv:2012.15495*.

Ahmad Rashid, Vasileios Lioutas, and Mehdi Rezagholizadeh. 2021. MATE-KD: Masked adversarial TExt, a companion to knowledge distillation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1062–1071, Online. Association for Computational Linguistics.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for bert model compression.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *arXiv preprint arXiv:2002.10957*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2020. Xlnet: Generalized autoregressive pretraining for language understanding.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019a. PAWS: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1298–1308, Minneapolis, Minnesota. Association for Computational Linguistics.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019b. PAWS: paraphrase adversaries from word scrambling. *CoRR*, abs/1904.01130.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020. Freelb: Enhanced adversarial training for natural language understanding.

A Model training details

For all of the baseline models (Annealing-KD, MATE-KD, ALP-KD, PKD, Vanilla-KD) mentioned, we strictly follow the hyperparameters that are introduced in the original paper. For the DistilBERT student ALP-KD, we only change the initialization of student. The training manner and hyperparemeter tuning is in consistence with original ALP-KD. For the hyperparameters of ComKD that are listed in table 11, we manually tuned these based on MateKD. Adam optimizer will be applied to train the ComKD. The generator model that employed in ComKD is following setting of the MATE-KD. Specifically, for BERT-base teacher setting, a 4-layer Bert-mini model is used. For RoBERT-large teacher setting, a distilroberta-base model is used. We trained all models using a single NVIDIA V100 GPU. All experiments were run using the PyTorch [4] framework.

	batchsize	lr	ep1	ep2	T
CoLA	32	7e-6	100	10	10
MNLI	32	2e-5	30	10	10
MRPC	32	7e-6	200	10	10
QNLI	32	2e-5	100	10	10
QQP	32	2e-5	30	10	10
RTE	32	6e-6	200	10	10
SST-2	32	1e-5	100	10	10
STS-B	32	2e-5	100	10	10

Table 11: Hyperparameters for ComKD. ep1 and ep2 is corresponding to the training epochs of phrase 1 and phrase 2. T is max temperature

B Combined-KD Detailed Algorithm

In this section, we list Combined-KD details in algorithm 1.

C Discussion of the influence of punctuation

Some samples of prediction label change of add/remove end mark are shown on Table 12. Most models will not change the prediction after we remove or add a end mark except for Annealing-KD and DistilBERT.

Interestingly, the Annealing-KD can handle the sample 1 and sample 2 correctly after we add the end mark. DistilBERT will also give correct answer for sample 4 and sample 5. These phenomenons indicate that punctuation will give the models a hint to correctly do a classification, and the models make use of it.

[4] https://pytorch.org/

Algorithm 1: Combined Knowledge Distillation

```
end
Finetuned Teacher: $T(\cdot)$
pre-trained Student: $S(\cdot;\theta)$
Generator model: $G(\cdot;\phi)$
dataset: $D$
/* Generator training steps, every S steps, max
   temperature, learning rate                                */
Parameter: $S_g, S, max_T, \eta$
$step \leftarrow 0$
/* learning temperature                                      */
$temp \leftarrow 1$
for $batch \leftarrow D$ do
    $X, Y \leftarrow batch$ ;
    $step \leftarrow step \mod S$
    # Adversarial Step ;
    $X^m \leftarrow$
    $X{=}[x_1,\dots,x_n]$
    $p \sim$ unif$(0,1)$, Mask$(x_i \in X, p_i)$ ;
    /* predict logit only for the masked tokens              */
    $X_{\text{logits}} \leftarrow G(X^m \phi)$ ;
    $X' \leftarrow$ Gumbel-Softmax$(X_{\text{logits}})$ ;
    if $step < S_g$ then
        $\mathcal{L}_G \leftarrow MSE(T(X'), S(X';\theta))$ ;
        /* update generator parameters                       */
        $\phi \leftarrow \phi - \eta \frac{\partial \mathcal{L}_G}{\partial \phi}$ ;
    else
        # Knowledge Distillation ;
        if Phase = 1 then
            $T_{X'} \leftarrow \frac{temp}{max_T} T(X')$
            $\mathcal{L}_{\text{ADV}} \leftarrow MSE(T_{X'}, S(X';\theta))$ ;
            $T_X \leftarrow \frac{temp}{max_T} T(X)$
            $\mathcal{L}_{\text{KD}} \leftarrow MSE(T_X, S(X;\theta))$ ;
            $\mathcal{L} \leftarrow \frac{1}{2}\mathcal{L}_{\text{ADV}} + \frac{1}{2}\mathcal{L}_{\text{KD}}$ ;
            $\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}}{\partial \theta}$ ;
            if $temp \neq max_T$ then
                $temp \leftarrow temp + 1$
            end
        else
            $\mathcal{L} \leftarrow$ CE$(S(X;\theta), Y)$
            $\mathcal{L} \leftarrow$ CE$(S_\theta(X), Y)$ /* update student
                parameters                                   */
            $\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}}{\partial \theta}$ ;
        end
        decay $\eta$ ;
end
```

Index	Sample	Label	BERT	DistilBERT	ALP-KD*	AKD*	MKD*	ComKD
1	**P**: I just stopped where I was **H**: He felt very sick	E	E	N	N	N	N	N
1 (Add end mark)	**P**: I just stopped where I was . **H**: He felt very sick .	E	E	N	N	**E**	N	N
2	**P**: the census of 1931 served as an alarm signal for the malay national consciousness **H**: there was n't any censuses in malaysia prior to 1940	C	C	N	N	N	C	C
2 (Add end mark)	**P**: the census of 1931 served as an alarm signal for the malay national consciousness . **H**: there was n't any censuses in malaysia prior to 1940 .	C	C	N	N	**C**	C	C
3	**P**: oh , what a fool i feel ! **H**: I am beyond pride	C	E	N	N	N	C	C
3 (Add end mark)	**P**: oh , what a fool i feel ! **H**: I am beyond pride .	C	E	**E**	N	N	C	C
4	**P**: No , don't answer **H**: Don't say a word .	E	E	C	E	E	E	E
4 (Add end mark)	**P**: No , don't answer . **H**: Don't say a word .	E	E	**E**	E	E	E	E
5	**P**: how long has he been in his present position **H**: what length of time has he held the current position ?	E	E	N	E	E	E	E
5 (remove end mark)	**P**: how long has he been in his present position **H**: what length of time has he held the current position	E	E	**E**	E	E	E	E

Table 12: Details of prediction label change of add/remove end mark. ALP-KD* is ALP-KD (DistilBERT), AKD* is Annealing-KD and MKD* is MATE-KD.