Adversarial Natural Language Inference with Data Augmentation on Fact Verification Dataset: 3-class Classification task

Enrico Maria Aldorasi
Sapienza University of Rome
`aldorasi.2131576@studenti.uniroma1.it`

Abstract

This paper describes methodologies and results obtained in training different models on two train sets, one of which is the result of data augmentation techniques applied on the original train set, in testing them on the original test set and on an adversarial test set generated to be particularly complex by a human annotator, for Natural Language Inference (NLI) 3-class Classification task, to determine whether a hypothesis is true, false, or undetermined given a certain premise.

1 Dataset Description

This is a subset of FEVER, a hand-curated dataset for Fact Extraction and Verification built from Wikipedia, with pairs (claim, context), sensitive to the sentence order and annotates into true (Supported), false (Refuted), or undetermined (Not Enough Info). In this context dataset pairs are considered as (premise, hypothesis) and annotates into ENTAILMENT, CONTRADICTION, NEUTRAL. Here's the structure:

```
{
"id": The ID of the sample,
"premise": The context to be used in making the inference,
"hypothesis": The claim that is made concerning the premise,
"label": The annotated label for the claim. One of ENTAILMENT|CONTRADICTION|NEUTRAL
}
```

Each sample has also in its structure informations about Word Sense Disambiguation (WSD) and Semantic Role Annotations (SRL). The dataset is divided into train set (*51.1k rows*), validation set (*2.29k rows*) and test set (*2.29k rows*).

2 Model Selection

Two transformer-based models were chosen for this project:

RoBERTa-base: Robustly optimized BERT
DeBERTa-v3-large: Decoding - enhanced BERT

3 Model Architecture

This paper presents the implementation of two distinct architectures:

- Plain architecture use pre-trained transformer model to generate contextual embeddings from input text pairs (premise, hypothesis) followed by different layers.
  Pooling Layer extracts the embedding of the [CLS] token (the first token in the sequence) from the final hidden state of the transformer model.
  Classification layer applies a linear transformation to the pooled output to predict the NLI class labels (CONTRADICTION, NEUTRAL, ENTAILMENT).

- Enhanced architecture, builds on the Plain architecture, incorporate an other layer before the Classification layer to improve model regularization and robustness.
  Dropout layer randomly zeros some elements of the input tensor with a specified probability during training. This helps prevent overfitting and improves generalization.

4 Model Design

The model design involves carefully selected hyperparameters, including a **maximum sequence length** of *128*, a **batch size** of *16*, **accumulation steps** of *4*, a **learning rate** of *1e-5*, *20* **epochs** with **patience** set to *3* for early stopping and a **dropout rate** of *0.1*. The training process employs the AdamW optimizer. The *torch.cuda.amp* is utilized to enhance computation efficiency. Evaluation is performed on both original and adversarial test sets, with metrics such as **accuracy**, **precision**,

recall, and **F1-score** calculated to comprehensively evaluate model performance (view table 1).

5 Data Augmentation

Several data augmentation techniques were applied to the original train set to create new adversarial samples (view 1) on which train models and try to get better performance:

- Summarization: technique that generates a concise version of the original text while preserving its main meaning. In this implementation was used the BART (Bidirectional and Auto-Regressive Transformers);
- Hypernym Replacement: technique that replaces a specific word with an other one with a broader meaning that includes the meanings of the replaced word. To do this was used WordNet;
- Synonym Replacement: technique that replaces words with others that have the same or similar meanings. It was used to swap verbs. As for Hypernym, was used WordNet;
- Negation: technique that involves changing the meaning of a sentence to its opposite. After negating a verb in the hypothesis, this is particularly useful to convert entailment relationships to contradictions and vice versa.

These techniques are chosen randomly for each sample to ensure more generalization.

6 Results

The main purpose in analysing the results obtained is to focus on the differences in the performance of the trained models on both the original dataset and the augmented dataset, evaluating them on both test sets.
RoBERTa-base with plain architecture trained on the original dataset, achieves an accuracy of *75%* on the original test set and an accuracy of *56%* on the adversarial test set. When trained on the augmented dataset it achieves an accuracy of *74%* on the original test set and an accuracy of *58%* on the adversarial test set.
RoBERTa-base with enhanced architecture trained on the original dataset, achieves an accuracy of *74%* on the original test set and an accuracy of *55%* on the adversarial test set. When trained on the augmented dataset it achieves an accuracy of *74%* on the original test set and an accuracy of *58%* on the adversarial test set.
DeBERTa-v3-large with enhanced architecture trained on the original dataset, achieves an accuracy *77%* on the original test set and an accuracy of *64%* on the adversarial test set.
When trained on the augmented dataset it achieves an accuracy of *78%* on the original test set and an accuracy of *64%* on the adversarial test set.
The results demonstrate that training on the augmented dataset generally helps models perform better on the adversarial test set, although it may not significantly improve performance on the original test set. Additionally, the introduction of the dropout layer in the enhanced architecture for the RoBERTa-base model does not substantially affect the results, suggesting that regularization may not be a critical factor in this context. DeBERTa-v3-large shows superior performance compared to RoBERTa-base, especially on the adversarial test set, indicating its robustness in handling more challenging examples.

7 Instructions to Execute Code

To execute Python scripts, navigate to the project directory and run the following command to install the required libraries:

```
> pip install -r requirements.txt
```

Script to create the augmented set:

```
> python 2131576-augment.py
```

The augmented dataset will be automatically saved in the script directory and some samples will be printed.
Script to train and to test models can be executed exploiting the argparse library.
To train one of the models on original data:

```
> python 2131576-main.py train --data
[original] --model [roberta/deberta]
--architecture [plain/enhanced]
```

To train one of the models on the Augmented data:

```
> python 2131576-main.py train --data
[augmented] --augmented data
[path/to/augmented.jsonl] --model
[roberta/deberta] --architecture
[plain/enhanced]
```

At the end of the training, the trained model will be saved into the script directory and will be ready to be tested:

```
> python 2131576-main.py test --data
[original/adversarial]
```

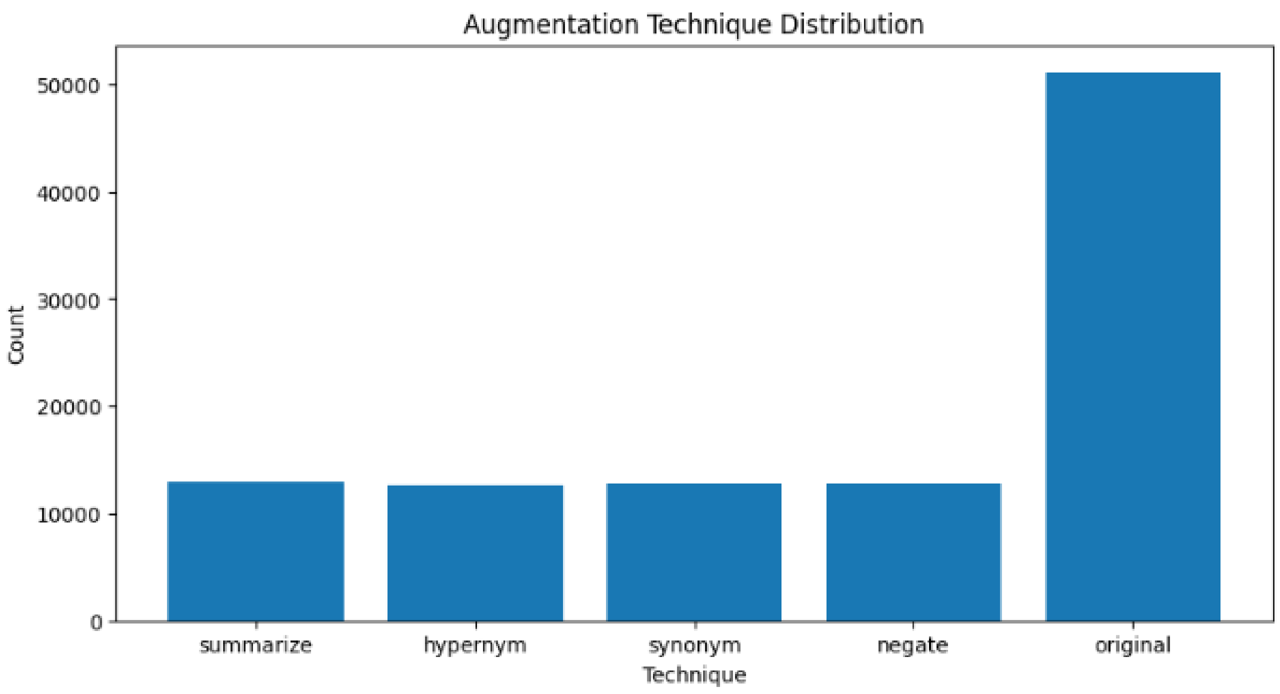

Figure 1: Augmentation Techniques Distribution

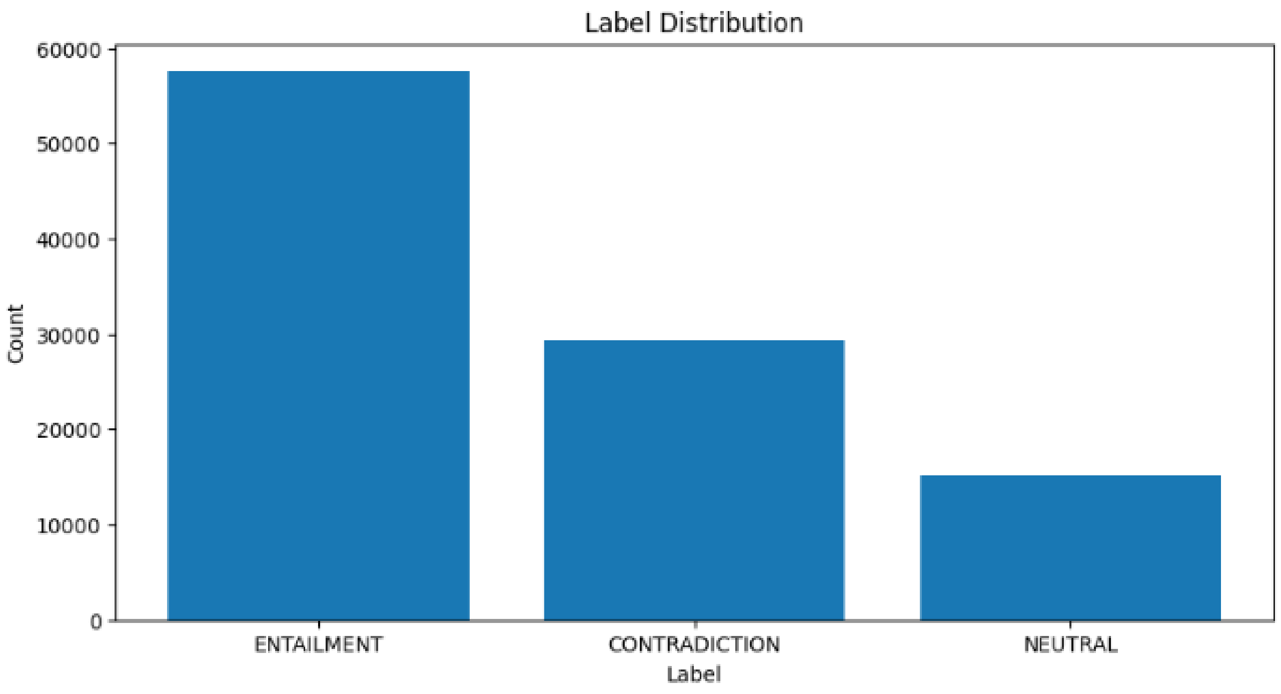

Figure 2: Label Distribution Data Augmentation

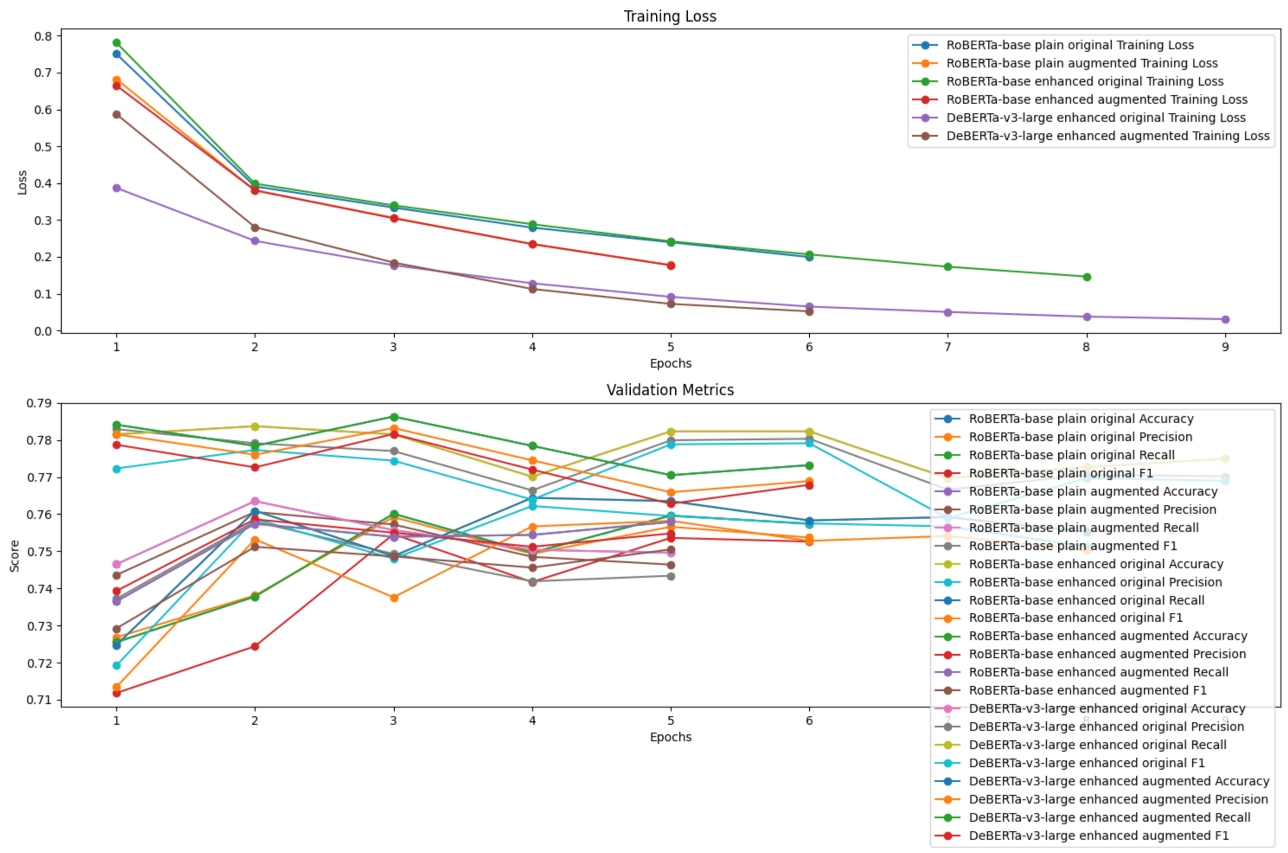

Figure 3: Training Results

Model	Architecture	Train-set	Test-set	Accuracy	Precision	Recall	F1-score
RoBERTa-base	plain	original	original	0.7495	0.7460	0.7495	0.7454
RoBERTa-base	plain	original	adversarial	0.5608	0.5598	0.5608	0.5594
RoBERTa-base	plain	augmented	original	0.7407	0.7371	0.7407	0.7345
RoBERTa-base	plain	augmented	adversarial	0.5786	0.5953	0.5786	0.5803
RoBERTa-base	enhanced	original	original	0.7433	0.7398	0.7433	0.7391
RoBERTa-base	enhanced	original	adversarial	0.5460	0.5707	0.5460	0.5477
RoBERTa-base	enhanced	augmented	original	0.7411	0.7361	0.7411	0.7337
RoBERTa-base	enhanced	augmented	adversarial	0.5816	0.5805	0.5816	0.5810
DeBERTa-v3-large	enhanced	original	original	0.7683	0.7640	0.7683	0.7638
DeBERTa-v3-large	enhanced	original	adversarial	0.6350	06434	0.6350	0.6347
DeBERTa-v3-large	enhanced	augmented	original	0.7783	0.7749	0.7783	0.7750
DeBERTa-v3-large	enhanced	augmented	adversarial	0.6350	0.6478	0.6350	0.6367

Table 1: Comparison of Model Performance Metrics