A List of Acronyms

This section serves as a reference for all acronyms used throughout the paper.

Table A.1: Overview of all acronyms used, in alphabetical order.

Acronym	Description
AM	Augmentation Model
BERT	Bidirectional Encoder Representations from Transformers (Devlin et al., 2019)
DAWSON	Data Augmentation using Weak Supervision On Natural Language
EDA	Easy Data Augmentation (Wei and Zou, 2019)
GLUE	General Language Understanding Evaluation benchmark (Wang et al., 2018)
GPU	Graphics Processing Unit
LAMB	Layer-wise Adaptive Moments (You et al., 2020)
LB	Lower Bound
MCC	Matthews Correlation Coefficient (Matthews, 1975)
MLM	Masked Language Modeling
QNLI	Question-Answering Natural Language Inference (Rajpurkar et al., 2016)
QQP	Quora Question Pairs (Iyer et al., 2017)
SBO	Span Boundary Objective (Joshi et al., 2020)
SE	Span Extraction
SQuAD	Stanford Queston Answering Dataset (Rajpurkar et al., 2016)
SST	Stanford Sentiment Treebank (Socher et al., 2013)
UB	Upper Bound
WS	Weak Supervision

B Implementation Details

The starting point for the augmentation model is a BERT-Base uncased, with $L = 12$ transformer blocks, $H = 768$ hidden size, and $A = 12$ attention heads, resulting in 110M parameters. This configuration is chosen as it is the most commonly used in the field, mainly because the larger version of BERT does not fit on most GPUs and smaller versions have only been recently introduced. We make use of the implementation from Huggingface[2], a library providing a common interface for all transformer-based models. We use the original model by Devlin et al. (2019), pre-trained on the BookCorpus dataset and the English version of Wikipedia. Our implementation is in TensorFlow. We make use of layerwise learning rates by using Layer-wise Adaptive Moments (LAMB) as optimizer. Proposed by You et al. (2020), LAMB is originally intended to speed up pre-training by allowing for larger batch sizes without loss in performance. However, You et al. (2020) found that LAMB also yields excellent performance for smaller batch sizes and is typically more consistent than the often used Adam with Weight Decay (Loshchilov and Hutter, 2018).

During training, we make use of *smart batching*. Attention is computed for every token in relation to every other token. Thus, including more tokens increases the number of relations exponentially. Within a batch, all sequences need to be padded to the same length such that they can be fitted into an non-ragged tensor. However, batches do not have to be the same shape. By first sorting the dataset based on string length, and shuffling locally within a range of 3-6 batch sizes as a rolling window to maintain randomness, the maximum sequence length per batch is optimized and computation time is decreased. After the batches have been created, they are shuffled for the training order. Smart batching is especially useful in a dataset with strongly heterogeneous sequence lengths, such as movie reviews, where one can leave a single word or an extensive essay. Decreasing the overall maximum sequence length results in a loss of information, while keeping the maximum sequence length larger results in many unnecessary computation for short reviews.

C End-to-End Example

For the step-by-step example, the methodology is applied to two movie reviews. The first review - the running example - is a *negative* review taken from the expert-labeled dataset:

“one relentlessly depressing situation"

The second review is taken from the large unlabeled dataset:

“even if you’ve never heard of chaplin, you’ll still be glued to the screen"

[2] https://huggingface.co/transformers/

STEP 1: SpanBERT

First, a BERT model, pre-trained by Devlin et al. (2019), is initialized with a SpanBERT head on top. No labels are required, wherefore all available data can be used. Suppose that for the examples, span lengths of $l = 1$ and $l = 2$ are drawn respectively. The starting point of the spans is random. In the example, the masks are placed as:

"one [MASK] depressing situation"

"even if you've never heard of [MASK] [MASK] still be glued to the screen"

The dataset is repeated 10 times, such that masks are placed at different places in a sentence. The masks are predicted both using the full sequence and just the boundary tokens, as explained in Section 3.1. In this manner, the model is trained for a predefined number of epochs. The SpanBERT pre-training both fits the model to the domain-specific data and improves augmentation.

STEP 2: Span-Extractive Pre-Training

Next, the model is trained on the unlabeled data only, using weak labels. Suppose that the weak supervision method estimates there is a probability of 60% that the unlabeled example is positive, that is $Pr(\tilde{y} = \textbf{positive}) = 0.6$. Both textual labels are prepended and the model is trained using span extraction. The sentence, with probabilistic labels, looks as follows:

*"**positive** (0.6) **negative** (0.4) even (0) if (0) you've (0) never (0) heard (0) of (0) chaplin, (0) you'll (0) still (0) be (0) glued (0) to (0) the (0) screen" (0)*

The model has to both determine the sentiment of the sentence, and select the token-label with the corresponding sentiment.

STEP 3.1: Target Analysis

In the previous step, a classifier is trained, that has not yet seen the sentiment of the expert-labeled review. As such, first the classifier is used to make a prediction for the sentence, which can be compared to the true label as analysis of the complexity. Suppose the classifier makes the following prediction:

*"**positive** (0.3) **negative** (0.6) one (0) relentlessly (0.1) depressing (0) situation" (0)*

The predicted probabilities to select a token are shown below the corresponding token. For illustration purposes, the classifier has also incorrectly assigned some probability to the non-label token "relentlessly". It is known that the correct label is negative, and the model has predicted the negative token is to be selected with 60% probability. Therefore the error - or complexity - is 40%, that is $e = 0.4$.

Not only the prediction, but also the attention to the tokens is saved. The label tokens are removed, and the remaining probabilities re-weighted such that again, their sum is equal to 100%. For the example, the attention probabilities for the remaining tokens are:

"one (0.1) relentlessly (0.2) depressing (0.4) situation" (0.3)

STEP 3.2: Span-Extractive Fine-Tuning

After the target analysis, the model is trained further on the expert-labeled data. The fine-tuning procedure is identical to the pre-training, except that, instead of weak labels, the ground truth labels are used:

*"**positive** (0) **negative** (1) one (0) relentlessly (0) depressing (0) situation" (0)*

Note that the span-extractive pre-training step already trained the classifier to only select the label tokens and generalize beyond the noisy labels as much as possible. During the fine-tuning step, only the relationship between the token-label and corresponding sentiment has to be strengthened further.

STEP 4: Augmentation Training

The final training step is the conditional augmentation task. Only high-quality textual labels may be prepended, therefore the model is trained on the expert-labeled data only. The tokens are masked randomly:

*"**negative** one relentlessly depressing [MASK]"*

Because of the pre-training steps, the augmentation model will allocate more attention to the textual label, and make use of the sentiment of the sentence when predicting for the masked token. Like in the SpanBERT step, the dataset is repeated 10 times with different masks.

STEP 5: Heterogeneous Augmentation

The last step is the actual augmentation that is used to create the augmented dataset. Different than in the augmentation training, the dataset is not necessarily repeated, therefore the tokens are not masked using a uniform distribution, but with the attention probabilities from the target analysis, in order to increase the probability that relevant words are masked. In this case, the token "depressing", with the largest probability, is drawn:

*"**negative** one relentlessly [MASK] situation"*

When selecting a replacement token, a lower, and upper bound are used. We empirically set the general upper bound (UB) to 0.95 and lower bound (LB) to 0.8. Recall the error from the target analysis is 0.4. The observation-specific lower bound is:

$$\text{LB}_s = 0.8 + (0.95 - 0.8) * 0.4 = 0.86$$

When predicting a masked token, for each token in the vocabulary, a probability is estimated. We sort the tokens based on their respective probabilities, and calculate the cumulative probability up to each token. The tokens are only considered candidates if the associated cumulative probability is within the bounds of UB and LB_s. In the table below, the probabilities for all tokens in the vocabulary are given. Only the tokens in between the dotted lines are within bounds.

Token	Probability	Cumulative
Horrible	0.05	1.00
Bad	0.04	0.95
Sad	0.03	0.91
Boring	0.02	0.88
Lame	0.02	0.86
Mediocre	0.01	0.84
⋮	⋮	⋮
Parachute	0.00	0.00
Catalogue	0.00	0.00

Table C.1: Candidate tokens for the masked token in the example.

For the remaining tokens, their probabilities are re-scaled to again sum up to 100%. In the case of this example, there are 4 candidate tokens. The final token is sampled using the re-scaled probabilities. In this case, the augmented sample is:

*"one relentlessly **boring** situation"*

where the token *"boring"* is selected as replacement. As the number of masked tokens and the probability of the replacements are known, a probabilistic label is calculated to account for the added uncertainty in the data. The total number of tokens is $N = 4$, the number of replaced tokens is $K = 1$, and the probability of *"boring"* is 0.02. Thus, the probability of the augmented review's label is:

$$Pr(y^* = \textbf{negative}) = max\left(\frac{4 - 1 + 0.02}{4}, 0.50\right) = 0.755$$

If the probabilistic labels were to be omitted, $Pr(y^* = \textbf{negative})$ would be equal to 1. The augmented review is added to a new dataset of augmented observations. After all expert-labeled data is augmented, both datasets are combined. Finally, a new classifier - of any kind - may be trained on the combined dataset, using a normal classification formulation.

D Weak Supervision Simulation

In Figure D.1, a histrogram of draws for a simulated weak supervision confidence distribution are shown. The random draws have fat tails, such that a lot of noise is present during pre-training.

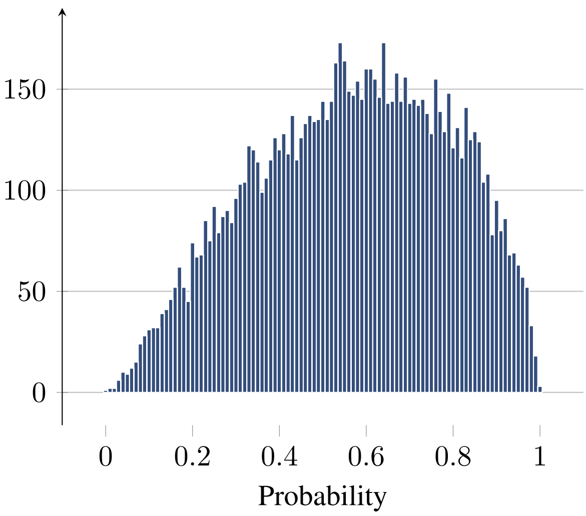

Figure D.1: Histogram of random draws of a Beta distribution with $\mu = 0.57$ and $\sigma^2 = 0.05$, for the weak supervision simulation. The draws are task-independent. In the histogram, the distribution of 10,000 draws is shown.