Semantic-Augment: Augmenting the Semantic Space of Transformers Improves Generalization

Stanford CS224N Custom Project

Emirhan Kurtulus
Department of Computer Science
Stanford University
emirhank@stanford.edu

Abstract

Self-supervised pretraining has been a critical component for quality of the natural language processing methods. Because of the abundance of text data, the simple noising-denoising paradigm allowed the models to be scaled to trillions of parameters. This paradigm can be extended by adding consistency maximization and noise invariance. In this work, we propose a novel and simple framework to improve representation learning in language model pretraining by maximizing the similarity between two noised views of the same sequence in addition to the conventional masked language modeling loss. The proposed method improves performance on downstream tasks (0.9% average GLUE score, and 1.1% SWAG accuracy), while enlarging the representation space (10 to 20% higher feature alignment) of the model without incurring additional computational costs, and can be generalized to both encoder-only, encoder-decoder, and decoder-only model pretraining.

1 Key Information to include

• Mentor: Ekin Dogus Cubuk (Google Brain)
• External Collaborators: N/A
• Sharing project: N/A

2 Introduction

Large language models have been one of the driving factors in recent deep learning advances [1,2,3,4]. Since the introduction of the Transformer architecture [5], large language models (LLMs) have been all we need. Language models show great performance on a variety of tasks, from question answering [6] to common sense reasoning [7]. One of the main reasons for its effectiveness on a wide range of tasks [8] is the pre-training phase, in which the model can learn from a huge corpus of up to 500 billion tokens of unlabeled text [2]. The independence from labeled data allows training trillion-scale models [10].

Although they offer great advantages due to their performance on downstream tasks and their zero-shot generalization capabilities, they still have various problems with language understanding, ranging from representation deficiency [11] to alignment problems with human language understanding [12]. These problems can be mitigated by improving the pre-training phase to teach LLMs better language representations and improve their generalization.

For this purpose, we propose a new pretraining framework, which we call Semantic-Augment, that is easily applicable to encoder-only models [13], decoder-only models [14], and encoder-decoder models [15]. Here, we combine the advantages of denoising approaches with consistency regularization through noise invariance. Our framework performs two forward passes for different noised views of the same sequence, and applies feature similarity loss to these views in addition to the conventional
Figure 1: Semantic-Augment framework.

language modeling loss. Semantic-Augment improves the generalization of the models to downstream
tasks (+0.9% GLUE score and 1.1% SWAG accuracy), while expanding the feature space of the
models (approximately 10% higher alignment on STSB) and reducing representation deficiency.

3 Background / Related Work

Many studies focused on pretraining different LLM architectures. Therefore, the field is largely
dominated by the Transformer architecture [5]. As one of the first works on pretraining Transformers
on-scale, BERT [13] proposes training an encoder-only model to denoise the masked words in a
given sentence while predicting, for given two sentences, if the second entails the first. This approach
is later extended by BART [16] to encoder-decoder models by adding more noising strategies to
the pretraining stage while using a unidirectional encoder. They show that under right conditions, a
unidirectional encoder is as good as bidirectional for representation learning. Later, A whole series of
models for the sole purpose of next-token prediction were proposed by the GPT family [14, 17, 1].
GPT family shows us that scale enables many features such as few-shot learning and zero-shot task
generalization which motivated a set of works on scaling laws [18, 9], instruction tuning [19, 20],
and in-context learning [21, 22].

Another set of works is on sequence-level representation learning where the goal is to learn the
best features for a given sentence. Sentence-BERT [23] shows that using BERT embeddings for
semantic search is infeasible and proposes a self-supervised representation learning method for
adding a projection head on top of BERT which enforces feature-invariance against augmentations
such as adversarial perturbations, token/feature cutoff, and dropout. SimCSE [26] further simplifies
the pretraining stage by using different dropout masks for two branches and simply relying on dropout
features for invariance-maximization. DiffCSE [27] improves the method by incorporating generative
learning into self-supervised pretraining through replaced token detection similar to Electra [28].
PaSer [29] uses a fully generative pretraining objective by encoding different views of the same
sentence which are later used to recover the masked words.

At the intersection of pretraining and representation learning, there are a set of works that try to
combine language modeling and representation learning. In pre-transformer era, CVT [30] increased
the supervision of the models by adding auxiliary tasks for different views of which an LSTM model
makes predictions. COCO-LM [31] shows that Transformer models have a squeezed representation
space and mitigates the problem by jointly doing language modeling and sentence-level representation
learning. TaCL [32] takes this a step further by learning a token-based features, opposed to sentence
level, while doing language modeling. DialogueCSE [33] extends such work to the dialogue domain
by learning dialogue-level features through encoding features at a sentence level but aggregating the features at individual turns level and applying contrastive loss between turns.

4 Approach

Semantic-Augment combines representation learning and language modeling. During training, our framework enforces feature invariance over different noised view of the same sequence. As shown in Figure 1, our framework consists of three parts:

- **Two stochastic noising methods** generate two different noised views of the same sequence (e.g., “Semantic-Augment rules NLP” → (“Semantic-Augment rules [MASK]”, “Semantic-Augment [MASK] NLP”). This is applicable to all Transformer-based architectures, since the main difference between them in pretraining is noising method.

- **A language model** that generates pre-logits features used for feature similarity loss.

- **Pairwise feature similarity and supervised loss functions** are used to enforce feature invariance in learning conventional language modeling.

In each optimization step, we take a sample of size $N$, where each sample is a sequence of the same length. Then, using random noising, we generate two views of each sequence, resulting in total of $2N$ samples. For all sequences, we extract logits and features from the neural network. We apply a similarity loss ($L_2$) between the features of the augmented views of the same sequence to enforce invariance, and we also compute cross entropy loss. We provide a general overview of our framework in NumPy [34] in Figure 2. Given input $x$, logits $f(x)$, labels $y$, features of the first augmented views $v_1 = v_1(x)$, features of the second augmented views $v_2 = v_2(x)$, supervised loss $\ell$, and feature similarity loss weight $w$ (we call this hyperparameter tied-weight), the loss function of Semantic-Augment is:

$$L_{\text{Semantic-Aug}} = \sum_{i} \ell(f(v_1(x)), y) + w\|v_1(x) - v_2(x)\|^2$$ (1)

There are several ways to apply feature similarity loss, since language models output a feature vector for all words in a given sequence. In this work, we only consider the use of the loss of similarity
between the features of CLS tokens only, at the word level, where the individual features of all words are included in the L2 loss, and at the sentence level, where a sentence-level feature is constructed by an average of the word-level features before applying the loss. We refer to the models trained using our method as Tied-X, where X is the name of the model (e.g., Tied-BERT). Since our approach can be viewed as maximizing feature similarity in semantic space, we call our method Semantic-Augment.

4.1 Baselines

Pre-training BERT is an expensive task (about 40 Nvidia V100 days). Therefore, we perform our experiments using a simplified Cramming [35] setting where the models are limited by a single GPU day, in our case 1 Nvidia A100 day. Since the proposed method almost doubles the runtime, we consider two baselines. The first is the model trained for twice the number of steps, while the second is the proposed setting where the similarity loss is zero. The version with zero similarity loss also shows the benefits of our method over Batch Augmentation [36], i.e., the noise invariance caused by different augmented views of the same sequence is examined in tw=0 model.

Our main comparison metric is the mean GLUE [37] and SWAG [38] performance as well as the feature space distribution. Also, we do not compare our method to COCO-LM [31], even if it is really relevant, since their computational budget is significantly higher than ours (20 A100 days versus 1 A100 day).

4.2 Implementation Details

All of our experiments were performed using the Pytorch [39] deep learning framework. Training

5 Experiments and Analysis

In this section, we present our baselines and compare our results to the baseline, analyzing the benefits of Semantic-Augment.

5.1 Experimental Setting

Following [13], we pretrain our models using the Bookcorpus [41] and a 2022 Wikipedia dump [2] datasets, where the Wikipedia dump is approximately the same size as the original model. We do not apply any preprocessing, except for tokenization, which is done uncased.
Figure 4: Feature similarity error comparison between our model and our baseline. All STSB means evaluation on both train and dev set. STSB > X denotes the case where samples with semantic overlap bigger than X are sampled. Feature similarity error is the L2 error between the cosine similarity output by the model and the STSB label.

<table>
<thead>
<tr>
<th>STSB &gt; 0.89</th>
<th>STSB &gt; 0.86</th>
<th>STSB &gt; 0.88</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0 / 0.89</td>
<td>0.0 / 0.86</td>
<td>0.08 / 0.88</td>
</tr>
</tbody>
</table>

Table 1: Some sentences with low semantic similarity sampled from STSB dataset. On the top row, the first number is the semantic similarity label given in the dataset, while the second is the sentence-level cosine similarity between the features output by the BERT-base model for sentence-1 and sentence-2.
In pre-training, we train a BERT-base model for 80,000 steps over 800, of which a linear learning rate warm-up is applied. We use a sequence length of 128, a learning rate of 1e-4, a weight decay of 0.01, a global batch size of 1024, and for increased stability we clip the gradient to ensure the norm is 1 using a linear learning rate scheduler. Following [42], we omit the next sentence prediction loss and only do masked language modeling for all our experiments. We use no special attention mechanisms or implementation tricks, and make no changes to the BERT architecture to ensure that our gains are not due to the software lottery [43]. We omit curriculum learning and batch warmup in Cramming setting.

We evaluate the downstream performance of our models using two settings: GLUE [37] and SWAG [38]. Due to a limited computational budget, for pretraining, we run a hyperparameter search over 1, 3, 5, 7, 10 for tied-weight. For the downstream tasks, we perform a hyperparameter search over learning rates {5e-5, 4e-5, 3e-5, 2e-5}, the batch sizes {16, 32}, and weight decay 0.1 to GLUE for 3 epochs, while for SWAG we only train using batch size of 16, a weight decay of 0.01, and a learning rate of 2e-5 for 2 epochs.

### 5.2 GLUE

GLUE tasks are fine-tuning datasets that contain benchmarks for domains ranging from sentiment classification to semantic similarity. In our evaluation, which follows the original BERT evaluation, we exclude the WNLI task because of its problematic train/dev/set split [3]. In Table 2, we present the performances of our model and the baselines. We see that our model generalizes better to the downstream tasks, outperforming the baseline by 0.9% of the average GLUE score. This is to be expected in the sense that our approach increases noise invariance and implicitly introduces better semantic understanding. This clearly shows that the benefits of Semantic-Augment.

### 5.3 Question-Answering

We evaluate the question-answering capabilities of our model on SWAG [38] dataset. It includes 113k adversarially-curated multiple choice questions covering a rich set of domains. The goal of this task is to choose one of the given four choices that is semantically the most compatible with the given sentence. For this task, Semantic-Augment significantly outperforms its baseline by 1.1% accuracy.

### 5.4 Feature Distribution

Representation deficiency is the problem where the representation space of the model is significantly limited [31] in the sense that the model uses only a small range of its output distribution. This results in features for different sentences being very close to each other. For two completely random sentences, the sentence-level features output from BERT baseline have a high cosine similarity. We provide examples for this case in Table 1.

### Table 2: Results on GLUE dataset for the baseline models and Tied-BERT. We report the designated metrics given in [37]. All compared models are trained for 1 Nvidia A100 day. Reported results are single-task single-model finetuning.

<table>
<thead>
<tr>
<th></th>
<th>MNLI</th>
<th>SST-2</th>
<th>STSB</th>
<th>RTE</th>
<th>QNLI</th>
<th>QQP</th>
<th>MRPC</th>
<th>CoLA</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>79.6</td>
<td>90.4</td>
<td>85.2</td>
<td>58.1</td>
<td>89.1</td>
<td>86.3</td>
<td>87.9</td>
<td>44.8</td>
<td>77.7</td>
</tr>
<tr>
<td>Tied-BERT</td>
<td>80.4</td>
<td>90.0</td>
<td>85.1</td>
<td>59.9</td>
<td>87.8</td>
<td>86.0</td>
<td>88.0</td>
<td>46.5</td>
<td>78.0</td>
</tr>
<tr>
<td>Tied-BERT</td>
<td>80.7</td>
<td>90.8</td>
<td>85.7</td>
<td>60.3</td>
<td>89.4</td>
<td>86.9</td>
<td>89.1</td>
<td>45.7</td>
<td>78.6</td>
</tr>
</tbody>
</table>

### Table 3: Results on SWAG question-answering dataset. All compared models are pretrained for a single A100 day.

<table>
<thead>
<tr>
<th></th>
<th>baseline</th>
<th>Tied-BERT (tw=0)</th>
<th>Tied-BERT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>63.5</td>
<td>63.9</td>
<td>64.6</td>
</tr>
</tbody>
</table>
Since Semantic-Augment aims to enlarge the feature space of networks, we evaluate the features of our model using a benchmark we developed. For our benchmark, we use the STSB dataset, which contains sentence pairs and a manually annotated sentence similarity label. We evaluate the STSB examples without fine-tuning on STSB examples to see the differences caused by the pretraining stage. Since we do not fine-tune the model on train set, we evaluate on both train and dev sets.

Our benchmark consists of two different scenarios. The first is to compare the feature similarity error for the sentence pairs from the STSB dataset for a variety of sentence similarity brackets {all, above \{0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5\}. In addition to examining fine-grained feature similarity, we also examine feature dissimilarity between random sentences samples from the Bookcorpus dataset. We perform this analysis for all layers considering that for a sentence-pair, feature similarity should be aligned for all layers.

In Figure 3, we show the feature dissimilarity for 10,000 randomly sampled sentence pairs from the Bookcorpus dataset. We report the feature dissimilarity results for all layers. Since we are aiming to distinguish between two unrelated sentences, we use feature dissimilarity as a metric (1 - cosine similarity). As expected, the last layer is closer to the golden truth, but for all layers the features of Tied-BERT are significantly closer to the golden truth. For the first layer the difference is about 9%, while for the last layer the difference is almost 10%. This is a clear indication that Semantic-Augment prevents the model from assigning a high semantic overlap score to unrelated sentences.

Our results on the STSB train and dev set are depicted in Figure 4. We not only evaluate our models on the entire STSB dataset, but also create subsets with different semantic overlap regimes. We show that the Semantic Augment model has significantly more matching features than its baseline model for all layers and all regimes. It is worth noting that the alignment benefits of our model also apply to the early layers, showing that there is a general feature alignment throughout the internal representation.

6 Conclusion

Currently, language models assign extremely high scores to totally unrelated sentences and, unlike their image processing counterparts, pre-training language models does not maximize noise invariance, resulting in suboptimal performance. In this work, we present Semantic-Augment, a simple framework for unifying representation learning and language modeling. It significantly improves the performance of the models on GLUE and SWAG datasets while allowing the model to create a broader and more expressive feature space, thus avoiding representation deficiency. Moreover, Semantic-Augment can be implemented with only a few lines of additional code, making it an easy-to-use framework.

Our framework is easily applicable to all Transformer-based pretraining tasks, as it simply depends on the noising methods already used in pretraining. In the future, we aim to investigate the use of this framework for encoder-decoder and decoder-only architectures and its effectiveness when applied to Instruction-Tuning [19]. Given that our model creates a better representation space, another research direction would be to analyze whether our method minimizes model biases and thus potential harms.

References


