Lecture 19: BERT Variants

Week 6, Lecture 2 - Improvements and Optimizations

PSYC 51.07: Models of Language and Communication

Winter 2026

Today's Agenda 📋

1. 🚀 **RoBERTa**: Robustly Optimized BERT
2. 🔬 **ALBERT**: A Lite BERT with parameter sharing
3. ⚡ **DistilBERT**: Knowledge distillation for efficiency
4. 🔌 **ELECTRA**: Replace Token Detection
5. 📊 **Comparative Analysis**: When to use which variant
6. 💻 **Practical Considerations**: Model selection guide

Goal: Understand improvements to BERT and choose the right model

BERT's Limitations ⚠️

What could be improved?

1. **Training Procedure**
    - Some choices seemed arbitrary

- NSP task might not be useful
- Static masking (same masks every epoch)

2. **Model Size**
    - 110M (Base) or 340M (Large) parameters

- Large memory footprint
- Slow inference

3. **Training Efficiency**
    - Only 15% of tokens are predicted

RoBERTa: Robustly Optimized BERT 🚀

Key idea: Better training = Better performance

RoBERTa's Improvements (Liu et al. 2019):

1. **Remove NSP Task**
    - Next Sentence Prediction hurt performance
- Use only Masked Language Modeling
- Full sentences (don't need sentence pairs)

2. **Dynamic Masking**
    - Generate masking pattern every time
- BERT: static masks (same for every epoch)
- More diverse training signal

RoBERTa Results 📈

Consistent improvements over BERT

SQuAD 2.0	**83.1**	**89.4**	**+6.3**
MNLI	86.7	90.2	+3.5
SST-2	94.9	96.4	+1.5
RACE	72.0	83.2	+11.2

Key Findings:

- Dynamic masking better than static
- More data + longer training = better results
- RoBERTa-Large matches or beats BERT-Large on all tasks
- Sometimes huge gains (RACE: +11.2 points!)

Dynamic vs Static Masking 🎭

How masking patterns are generated

Static Masking (BERT):

Preprocessing:

- Mask tokens once
- Save masked dataset
- Same masks every epoch

Epoch 1: "My [MASK] is cute"

Epoch 2: "My [MASK] is cute"

Epoch 3: "My [MASK] is cute"

Problem:

ALBERT: A Lite BERT 🔬

Key idea: Parameter sharing for efficiency

ALBERT's Innovations (Lan et al. 2019):

1. Factorized Embedding

```
1# BERT: Direct embedding
2# 30K vocab × 768 hidden = 23M params
3bert_embed = nn.Embedding(30000, 768)
4
5# ALBERT: Two-step embedding
6# 30K × 128 + 128 × 768 = 3.8M + 0.1M
7albert_embed = nn.Embedding(30000, 128)
8albert_project = nn.Linear(128, 768)
9# Savings: 83% fewer embedding params!
```

2. Cross-Layer Sharing

ALBERT Parameter Efficiency 📊

Dramatic parameter reduction!

ALBERT-base	12	768	12M
ALBERT-large	24	1024	18M
ALBERT-xlarge	24	2048	60M
ALBERT-xxlarge	12	4096	235M

Key Observations:

- ALBERT-base: than BERT-base
- Can train much larger hidden sizes with same memory
- ALBERT-xxlarge: 4096 hidden dim, still only 235M params
- Trade-off: fewer params but similar computation (layer sharing)

Cross-Layer Parameter Sharing 🔗

How ALBERT achieves parameter efficiency

BERT (No Sharing):

```
1class BERT:
2    def __init__(self):
3        # Each layer has unique parameters
4        self.layers = [
5            TransformerLayer() for _ in range(12)
6        ]
7        # 12 × 7M params = 85M params
8
9    def forward(self, x):
10        for layer in self.layers:
11            x = layer(x)  # Different weights
12        return x
```

DistilBERT: Knowledge Distillation ⚡

Key idea: Train small model to mimic large model

```
# Knowledge Distillation Training Loop
teacher = BertModel.from_pretrained("bert-base")  # 12 layers, frozen
student = DistilBertModel(num_layers=6)            # 6 layers, trainable

for batch in training_data:
    # Teacher provides "soft targets" (probability distributions)
    with torch.no_grad():
        teacher_logits = teacher(batch)  # e.g., [0.7, 0.2, 0.1, ...]

    # Student tries to match teacher's distribution
    student_logits = student(batch)

    # Distillation loss: KL divergence between distributions
    # Temperature T=2 softens the distribution (more informative)
    loss_distill = KL_divergence(
        softmax(student_logits / T),
```

DistilBERT: Knowledge Distillation ⚡

```
    loss_mlm = masked_lm_loss(student_logits, labels)

    # Combined loss
    loss = 0.5 * loss_distill + 0.5 * loss_mlm
```

...continued

Why soft targets work: Teacher's "wrong" predictions contain information (e.g., "dog" → "cat" more likely than "car")

Reference: Sanh et al. (2019) - "DistilBERT, a distilled version of BERT"

DistilBERT Results 📊

Significant efficiency gains!

Inference Speed	1x	1.6x	60% faster
GLUE Score	79.6	77.0	97% retained

Performance on Specific Tasks:

SST-2 (Acc)	94.9	92.7
MNLI (Acc)	86.7	82.2

When to Use DistilBERT:

- Production deployment (latency-critical)
- Edge devices (limited memory)

ELECTRA: Efficient Learning 🔌

Key idea: Learn from all tokens, not just 15%

```
1# ELECTRA Training: Generator + Discriminator setup
2sentence = "The chef cooked a delicious meal"
3masked   = "The chef [MASK] a delicious meal"
4
5# Small generator (like BERT) fills in masks
6generator_output = generator(masked)
7# Generator predicts: "ate" (plausible but wrong)
8
9corrupted = "The chef ate a delicious meal"
10
11# Discriminator classifies EACH token: original or replaced?
12discriminator_output = discriminator(corrupted)
13# Output per token: [orig, orig, REPLACED, orig, orig, orig]
14
15# Loss computed on ALL tokens (not just 15%!)
16labels = [0, 0, 1, 0, 0, 0]  # 1 = replaced
```

ELECTRA Benefits ⚡

More efficient pre-training

Advantages:

1. **Sample Efficiency**
    - Learn from all tokens (100%) vs only masked (15%)
- Reaches same performance with less data
- Faster convergence
2. **Better Performance**
    - ELECTRA-Small outperforms BERT-Small
- ELECTRA-Base competitive with BERT-Large
- With same compute, ELECTRA is better

BERT Variants Comparison 📊

Summary of key variants

p{3cm}p{3cm}} Model	Key Innovation	Advantages	Best For

General Guidelines:

- **Best quality:** RoBERTa-Large
- **Best efficiency:** DistilBERT
- **Limited memory:** ALBERT
- **Limited training budget:** ELECTRA
- **Good default:** RoBERTa-Base or BERT-Base

Other Notable BERT Variants 🌟

The BERT family keeps growing!

1. **DeBERTa (Microsoft, 2020)**
    - Disentangled attention (separate content and position)
- Enhanced mask decoder
- State-of-the-art on SuperGLUE

2. **ERNIE (Baidu, 2019)**
    - Entity-level and phrase-level masking
- Knowledge enhancement
- Strong on Chinese NLP tasks

3. **SpanBERT (Facebook, 2019)**
    - Mask random spans instead of random tokens

Model Selection Guide 🎯

How to choose the right model for your task

```
1Start -> Quality or Speed? -> RoBERTa-Large -> Memory? -> DistilBERT -> ALBERT
```

Additional Considerations:

- **Domain:** Consider domain-specific pre-trained models (BioBERT, SciBERT, etc.)
- **Language:** Multilingual? Use mBERT, XLM-R
- **Task type:** Generation? Consider BART/T5 instead

Using Different BERT Variants 💻

Easy switching with HuggingFace

```
from transformers import AutoModel, AutoTokenizer

# BERT
bert_model = AutoModel.from_pretrained("bert-base-uncased")
bert_tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")

# RoBERTa
roberta_model = AutoModel.from_pretrained("roberta-base")
roberta_tokenizer = AutoTokenizer.from_pretrained("roberta-base")

# ALBERT
albert_model = AutoModel.from_pretrained("albert-base-v2")
albert_tokenizer = AutoTokenizer.from_pretrained("albert-base-v2")

# DistilBERT
distilbert_model = AutoModel.from_pretrained("distilbert-base-uncased")
```

Using Different BERT Variants 💻

```
21electra_tokenizer = AutoTokenizer.from_pretrained("google/electra-base-discrim
22
23# All have the same API!
24inputs = tokenizer("Hello, my dog is cute", return_tensors="pt")
25outputs = model(**inputs)
```

...continued

Benchmarking BERT Variants: Worked Example 💻

Practical comparison on sentiment analysis

```
import time
from transformers import pipeline

# Load different models for sentiment analysis
models = {
    "bert-base": "textattack/bert-base-uncased-SST-2",
    "distilbert": "distilbert-base-uncased-finetuned-sst-2-english",
    "albert": "textattack/albert-base-v2-SST-2",
}

test_texts = ["This movie was fantastic!", "I hated every minute of it."] * 10

for name, model_id in models.items():
    pipe = pipeline("sentiment-analysis", model=model_id)

    start = time.time()
```

Discussion Questions 💭

1. **Training vs Architecture:**
    - RoBERTa shows training matters. Is architecture overrated?

- How much can we improve with just better training?
- What's the right balance?

2. **Parameter Efficiency:**
    - ALBERT shares all layers. Why does this work?

- What are the limits of parameter sharing?
- Is there a "sweet spot"?

3. **Knowledge Distillation:**
    - Why does student learn better from teacher than from labels?

- What information is in the soft probabilities?

Looking Ahead 🔮

Today we learned:

- RoBERTa: Better training matters
- ALBERT: Parameter sharing for efficiency
- DistilBERT: Knowledge distillation
- ELECTRA: Replace token detection
- How to choose the right variant

Next lecture (Lecture 17 - Applications of Encoder Models):

From theory to practice! 🚀

Summary 🎯

Key Takeaways:

1. **RoBERTa**
    - Training procedure matters as much as architecture
- Remove NSP, dynamic masking, more data = better results
2. **ALBERT**
    - Parameter sharing dramatically reduces model size
- Factorized embeddings for efficiency
- 89% fewer parameters than BERT
3. **DistilBERT**
    - Knowledge distillation for deployment

References 📚

Essential Papers:

- **Liu et al. (2019)** - "RoBERTa: A Robustly Optimized BERT Pretraining Approach"
- **Lan et al. (2019)** - "ALBERT: A Lite BERT for Self-supervised Learning"
- **Sanh et al. (2019)** - "DistilBERT, a distilled version of BERT"
- **Clark et al. (2020)** - "ELECTRA: Pre-training Text Encoders as Discriminators"
- **He et al. (2020)** - "DeBERTa: Decoding-enhanced BERT with Disentangled Attention"

Resources:

- HuggingFace Model Hub: https://huggingface.co/models
- Papers With Code: BERT variants leaderboard
- Model Cards: Detailed documentation for each variant

Questions? 🙋

Discussion Time

Topics for discussion:

- BERT variants and improvements
- Model selection strategies
- Knowledge distillation
- Parameter efficiency
- Implementation questions

Thank you! 🙏

Next: Applications and Real-World Use Cases!