CS 533: Natural Language Processing

More Pretrained Transformers, Latent-Variable Generative Models

Karl Stratos

Rutgers University
Review: Pretrained Transformers

- Language models with Transformer architecture

- **Unsupervised transfer learning** (aka. "self-supervised" learning)
  1. Pretrain on a ton of raw text
  2. Finetune on a downstream task with modest supervision

- Enormous improvement over baselines trained from scratch on many NLU tasks

- Landmark: BERT (Devlin et al., 2019)
  - **Masked language modeling** (MLM)
  - "this is too [MASK] to fit" → "big"
  - Amenable to the full force of deep bidirectional self-attention in Transformer encoders
Some BERT Extensions

- **RoBERTa** *(Liu et al., 2019)*
  - A Robustly optimized BERT pretraining approach
  - Same as BERT but much more thoroughly optimized
  - Dynamic masking, no next sentence prediction (i.e., only MLM loss), BPE instead of wordpiece tokenization (thus language agnostic), trained with larger batch sizes for longer on more data
  - Very significant improvement, e.g., GLUE score
    - BERT (340m parameters): 80.5
    - RoBERTa (355m parameters): 88.1
    - Human: 87.1

- **ALBERT** *(Lan et al., 2019)*
  - A Lite BERT
  - Reduce number of parameters by: (1) Token embedding dimension bottleneck (≪ hidden dimension), (2) Tying Transformer parameters across layers
  - Catch: The model is smaller but slower! Larger hidden dim
  - GLUE score 89.4 with ensembling
Pretraining Encoder-Decoder Models

- BERT only pretrains a Transformer encoder
  - Limited to simple downstream tasks like text classification, tagging, span finding

- Critically, cannot be directly used for text generation

- How can we pretrain a Transformer decoder?
  - Can certainly just train it as a standard left-to-right LM (e.g., GPTs). But then no deep bidirectional self-attention
  - Is there a way to pretrain encoder & decoder jointly and get the best of both worlds?
BART (Lewis et al., 2019)

- Pronounced *bahrt* (vs. *burt* for BERT)
- Transformer encoder-decoder model trained as a *denoising autoencoder*
  - **Input.** \text{Corrupt}(\text{text})
  - **Output.** \text{text}

- Special cases
  - \text{Corrupt}(\text{text}) = \emptyset: \approx \text{GPT}
  - \text{Corrupt}(\text{text}) = \text{MaskTokens}(\text{text}): \approx \text{BERT}
  - \text{Corrupt}(\text{text}) = \text{Permute}(\text{text}): \approx \text{XLNet} (Yang et al., 2019)

- Great deal of flexibility in noise. Example: “text infilling”, a span-level generalization of MLM
  - Span lengths sampled from \text{Poisson}(\lambda = 3), entire span replaced by single [MASK], e.g.,

\[
\text{Corrupt(There Is No Plan to Stop Chemical Weapons in Syria)} \\
= \text{There Is No Plan to [MASK] in Syria}
\]

- Model must learn to infer span lengths in denoising
BART Pretraining

- Best of both worlds
  1. Encoder: Deep bidirectional self-attention over corrupted text
  2. Decoder: Autoregressive prediction of uncorrupted text

- Explored a variety of noise schemes

- Token/span masking is again found to be crucial
- Final choice: Text infilling + sentence-level shuffling
- No single noise best for all: Performance highly task-dependent. E.g., for perplexity null corruption (plain left-to-right LM) sometimes best.
BART Finetuning

- Text-level classification
  1. Feed input text to encoder (if sentence pair, concatenated)
  2. Feed the same text to decoder conditioning on the encoding
  3. Use the last top hidden state of the decoder to classify

- Token-level classification (e.g., SQuAD-style QA, tagging): Same as text-level classification, only use top decoder hidden states as contextual token embeddings
- Conditional text generation: Directly finetune
- MT: Add a few randomly initialized encoder layers at input.
Details of BART

- Number of parameters $406m$ (vs. $355m$ of RoBERTa which has 24 encoder layers)
  - 12 Transformer encoder/decoder layers, dimension 1024
  - GPT-2 style BPE tokenization: Shared embs $E \in \mathbb{R}^{50265 \times 1024}$

- Pretraining
  - Noise: Text infilling + sentence-level shuffling. Input is a document. 30% tokens masked, sentences shuffled.
  - Closely follows RoBERTa: Same pretraining data (160gb of news, books, stories, web), $500k$ updates w/ batch size 8000

- Classification result: Matches RoBERTa
  - BART’s generation capabilities don’t come at the expense of classification performance

- At the same time, significant improvement on conditional text generation
  - Abstractive summarization (R1): CNN/DailyMail 42.13 $\rightarrow$ 44.16, XSum 38.81 $\rightarrow$ 45.14
  - MT (BLEU): WMT16 Ro-En 36.80 $\rightarrow$ 37.96
T5 (Raffel et al., 2020)

- **Text-To-Text Transfer Transformer**
  - Concurrent work with BART on pretraining Transformer encoder-decoder model
  - Also based on large-scale denoising autoencoding, using a carefully cleaned version of the Common Crawl web scrapes
  - Additionally pretrained on a diverse set of *supervised* tasks framed as seq2seq problems

- Complementary insights confirming BART’s findings
  - Denoising encoder-decoder more effective than decoder LM
  - For noise, token masking crucial

- One of the very top performers on GLUE/SuperGLUE
  - 11 billion parameters: 90.3 GLUE, 89.3 SuperGLUE
Multilingual/Domain-Specific Pretrained Transformers

- **Multilingual BERT**: Released along with the original BERT
  - Same as BERT but trained on a union of Wikipedia dumps in 104 languages
  - Enables zero-shot cross-lingual model transfer (Pires et al., 2019): Finetune in language $A$, evaluate in language $B$

- **Multilingual BART (mBART)** (Liu et al., 2020)
  - Same as BART but trained on 25 languages extracted from Common Crawl with language identifier

- Directly transferrable to MT tasks, huge improvement (esp for low-resource languages)

- **Domain specific BERTs**: BioBERT (Lee et al., 2019) for biomedical text, SciBERT (AI2) for scientific text
The Model Size Problem

▶ Pretrained LMs growing rapidly in size

▶ Impossible to train except industry, difficult to use
▶ Focus of NLP shifted too much on sheer engineering (brainless usage of larger models)
▶ Also bad for the environment: Training a BERT on GPU emits as much CO₂ as a trans-American flight (Strubell et al., 2019)
Model Compression/Knowledge Distillation (KD)

- KD: Train a big “teacher” model $p_{\text{teacher}}$, learn a small “student” model $p_\theta$ by minimizing

\[ J(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{y \in Y} p_{\text{teacher}}(y|x_i) \log p_\theta(y|x_i) \]

- Form of regularization, in particular label smoothing
  - If $p_{\text{teacher}}(y_i|x_i) = 1$ then back to usual cross entropy. Can be controlled by softmax temperature (Hinton et al., 2015)
  - Big models have capacity to induce broader patterns, make small models mimic rather than figure out on their own

- Example: DistilBERT (Sanh et al., 2020)
  - Teacher: BERT-base (110m). Student: BERT-base with half of layers removed (67m)
  - 40% smaller, 60% faster, GLUE score down by 79.5 $\rightarrow$ 77.0

- Can also sample from teacher (e.g., if $y$ is a sequence)
  - KD: Use teacher predictions not gold labels (Kim and Rush, 2016)
What Does a Pretrained LM Know?

- **Probing.** Freeze pretrained model, train a classifier on top for simplified linguistic tasks (POS tagging, NER, semantic role labeling, etc.)
  - The more it “contains” linguistic knowledge, the better probing performance
- Easily solved even with small-scale pretraining
- In contrast, NLU tasks require billions of pretraining tokens before working

(Tenney et al., 2019) (Zhang et al., 2020)
Introducing Latent Variables in Generative Models

- Generative models (e.g., LMs) define $p_\theta(x)$
  - The only random variable is observation $x$
- Idea: Introduce additional variable $z$ and explicitly model an unseen generative process
  - We believe the process to be true (or at least useful for something), even though we don’t observe it

(Original Image: 4edges/Wikimedia Commons)
Latent-Variable Generative Models (LVGMs)

- $p_\theta$ defining a joint distribution over observation $x \in X$ and latent variable $z \in Z$

$$p_\theta(x, z) = \kappa_\theta(x | z) \times \pi_\theta(z)$$

- Very general definition
  - Can be discrete, continuous, or mixed
  - $x$ can be structured, $z$ can be structured, or both

- Why introduce latent variables?
  1. Clear generative story: Sample $z \sim \pi_\theta(z)$, then $x \sim \kappa_\theta(\cdot | z)$
  2. Marginal observation distribution can be more expressive
  3. Latent variables can be useful: Controllable generation (i.e., change $z$ to get $x$ we want), $z$ natural representation of $x$
Marginal Observation Distribution

- LVGM defines a marginal distribution $m_\theta$ over $\mathcal{X}$
  - If $z$ is discrete: $m_\theta(x) = \sum_{z \in \mathcal{Z}} p_\theta(x, z)$
  - If $z$ is continuous: $m_\theta(x) = \int_{z \in \mathcal{Z}} p_\theta(x, z) dz$
  - If $z$ is mixed: sum/integrate out appropriate dimensions

- $m_\theta$ can express a larger family of distributions
- Example: Bimodal distribution over $\mathcal{X} = \mathbb{R}$ cannot be expressed by any single Gaussian $\mathcal{N}(\mu, \sigma^2)$
  
  ![Bimodal distribution diagram]

- But can be expressed by a mixture of two Gaussians:

  $$m_\theta(x) = \pi_1 \mathcal{N}(\mu_1, \sigma_1^2)(x) + \pi_2 \mathcal{N}(\mu_2, \sigma_2^2)(x)$$

  Discrete latent variable $\mathcal{Z} = \{1, 2\}$
Better Explanation of Data

- Suppose iid samples from unknown population over \( \{a, b\}^{10} \) look like
  
  \[
  x^{(1)} = (a, a, a, a, a, a, a, a, a, a) \quad x^{(2)} = (b, b, b, b, b, b, b, b, b, b)
  \]
  
  \[
  x^{(3)} = (a, a, a, a, a, a, a, a, a, a) \quad x^{(4)} = (b, b, b, b, b, b, b, b, b, b)
  \]
  
  \[
  x^{(5)} = (a, a, a, a, a, a, a, a, a, a) \quad x^{(6)} = (b, b, b, b, b, b, b, b, b, b)
  \]

- Bag-of-words model \( p_\theta(x) = \prod_{j=1}^{10} p_\theta(x_j) \)?
  
  - The model’s independence assumption is clearly wrong!
  - Poor data fit: At most \( p_\theta(x^{(i)}) = 2^{-10} < 0.001 \) for each \( i \)

- LVGM \( m_\theta(x) = \sum_{z \in \{1, 2\}} \pi_\theta(z) \times \prod_{j=1}^{10} \kappa_\theta(x_j | z) \)
  
  - The model makes the right assumption (draw a latent “topic” \( z \) and draw observation conditioned on \( z \)).
  - Can achieve \( m_\theta(x^{(i)}) = 2^{-1} \) for each \( i \) with only twice more parameters
  - Also likely to generalize better (i.e., higher log likelihood of future samples)
Example LVGMs

- **HMMs:** \( z \in \mathcal{Z}^T \) (unobserved label sequence), \( x \in \mathcal{V}^T \) (sentence)

\[
p_\theta(x, z) = \prod_{t=1}^{T+1} t_\theta(z_t | z_{t-1}) \times \prod_{t=1}^T o_\theta(x_t | z_t)
\]

- **Gaussian LM:** \( z \in \mathbb{R}^d \) ("thought vector"), \( x \in \mathcal{V}^T \) (sentence)

\[
p_\theta(x, z) = \mathcal{N}(0_d, I_{d \times d})(z) \times \prod_{t=1}^{T+1} p_\theta(x_t | x_{<t}, z)
\]

- **Document hashing:** \( z \in \{0, 1\}^d \) ("hash code"), \( x \in \mathbb{R}^V \) (TFIDF document encoding)

\[
p_\theta(x, z) = \prod_{j=1}^d \text{Bernoulli}(\lambda_j)(z_j) \times \prod_{k=1}^V p_\theta(x_k | z)
\]
Marginal Log Likelihood

- Training objective: Maximize marginal log likelihood (MLL)

\[ L(\theta) = \mathbb{E}_{x \sim \text{pop}} [\log m_{\theta}(x)] \]

(Equivalent to cross entropy minimization, but convenient to frame as maximization for later)

- Requires the ability to calculate marginal probability of \( x \!\!:\)

\[ m_{\theta}(x) = \mathbb{E}_{z \sim \pi_{\theta}} [\kappa_{\theta}(x | z)] \]

- Sometimes we can calculate it exactly (best scenario)
  - \( z \) is discrete and \( \mathcal{Z} \) is small: \( m_{\theta}(x) = \sum_{z \in \mathcal{Z}} p_{\theta}(x, z) \) directly computable
  - \( p_{\theta} \) makes Markov assumptions: \( m_{\theta}(x) \) computable by dynamic programming (e.g., forward algorithm for HMMs)

- In general, we need to approximate by sampling
Variance Reduction by Importance Sampling

- Have an \( x \in \mathcal{X} \), would like to estimate \( L_x(\theta) = \log m_\theta(x) \)

- Naive scheme: Draw \( K \) iid samples \( z^{(1)} \ldots z^{(K)} \sim \pi_\theta \) and use
  \[
  \hat{L}_x^K(\theta) = \frac{1}{K} \sum_{k=1}^{K} \log \kappa_\theta(x|z^{(k)})
  \]
  - Unbiased: As \( K \to \infty \) we have \( \hat{L}_x^K(\theta) \to L_x(\theta) \)
  - Problem: High variance. What if \( \kappa_\theta(x|z^*) = 1 \) for a single \( z^* \in \mathcal{Z} \) but \( \pi_\theta(z^*) \) is tiny?

- Reduce variance by introducing an inference network (aka. approximate posterior) \( q_\phi(z|x) \) that tells us which \( z \) is “important” for \( x \)

- For any choice of \( q_\phi \) (full support)

  \[
  L_x(\theta) \overset{(1)}{=} \log \mathbb{E}_{z \sim q_\phi(\cdot|x)} \left[ \frac{p_\theta(x, z)}{q_\phi(z|x)} \right] \overset{(2)}{=} \mathbb{E}_{z \sim q_\phi(\cdot|x)} \left[ \log \frac{p_\theta(x, z)}{q_\phi(z|x)} \right]
  \]

  (1) Importance sampling, (2) Jensen’s inequality (log is concave)
ELBO

- **Evidence Lower Bound**

\[
\text{ELBO}(\theta, \phi) = \mathbb{E}_{x \sim \text{pop}, \, z \sim q_\phi(\cdot|x)} \left[ \log \frac{p_\theta(x, z)}{q_\phi(z|x)} \right] \leq L(\theta)
\]

- A variational lower bound on MLL ("variational" means optimization-based)
- We are learning *three* distributions
  1. Prior \( \pi_\theta(z) \)
  2. Conditional likelihood \( \kappa_\theta(x|z) \)
  3. Approximate posterior \( q_\phi(z|x) \): This is an "optimization assistant".
- In fact, the gap is precisely

\[
L(\theta) - \text{ELBO}(\theta, \phi) = D_{\text{KL}}(q_\phi||\omega_\theta)
\]

where \( \omega_\theta(z|x) = \frac{p_\theta(x,z)}{m_\theta(x)} \) is the true posterior probability, thus

\[
\text{ELBO}(\theta, \phi) = L(\theta) \iff q_\phi(z|x) = \omega_\theta(z|x) \quad \forall x, z
\]
Exact Relationship Between ELBO and MLL

\[ L(\theta) = \mathbb{E}_{x \sim \text{pop}} \left[ \log m_\theta(x) \right] \]

\[ = \mathbb{E}_{x \sim \text{pop}, \ z \sim q_\phi(\cdot|x)} \left[ \log m_\theta(x) \right] \]

\[ = \mathbb{E}_{x \sim \text{pop}, \ z \sim q_\phi(\cdot|x)} \left[ \log \frac{m_\theta(x)\omega_\theta(z|x)q_\phi(z|x)}{\omega_\theta(z|x)q_\phi(z|x)} \right] \]

\[ = \mathbb{E}_{x \sim \text{pop}, \ z \sim q_\phi(\cdot|x)} \left[ \log \frac{p_\theta(x, z)}{q_\phi(z|x)} \right] + \mathbb{E}_{x \sim \text{pop}, \ z \sim q_\phi(\cdot|x)} \left[ \log \frac{q_\phi(z|x)}{\omega_\theta(z|x)} \right] \]

\[ \text{ELBO}(\theta, \phi) \quad \text{and} \quad D_{KL}(q_\phi||\omega_\theta) \]
Variational Autoencoders (VAEs) (Kingma and Welling, 2014)

▶ **VAE.** Maximizing ELBO written as an autoencoding objective

$$
\text{ELBO}(\theta, \phi) = \mathbb{E}_{x \sim \text{pop}, z \sim q_\phi(\cdot|x)} \left[ \log \frac{\kappa_\theta(x|z)\pi_\theta(z)}{q_\phi(z|x)} \right] \\
= \mathbb{E}_{x \sim \text{pop}, z \sim q_\phi(\cdot|x)} [\log \kappa_\theta(x|z)] - \mathbb{E}_{x \sim \text{pop}} \left[ D_{\text{KL}}(q_\phi(\cdot|x) || \pi_\theta) \right]
$$

- **Reconstruction term** large if $q_\phi(\cdot|x)$ encodes $x$ into $z$ well and $\kappa_\theta(\cdot|z)$ decodes $z$ back to $x$ well
- **Regularization term** small if $q_\phi(\cdot|x) \approx \pi_\theta$ in expectation
Example: Gaussian VAE for Language Modeling

- Continuous latent space $\mathcal{Z} = \mathbb{R}^d$
- Observation space $\mathcal{X} = \mathcal{V}^*$ (i.e., all sentences)
- Model
  - Prior: $\pi_\theta = \mathcal{N}(0_d, I_{d \times d})$ (no learnable parameters)
  - Conditional likelihood: $\kappa_\theta(x|z) = \prod_{t=1}^{T+1} \kappa_\theta(x_t|x_{<t}, z)$. This can be any conditional word distribution that additionally conditions on $z \in \mathbb{R}^d$
- Inference network: $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \text{diag}(\sigma^2_\phi(x)))$ where
  \[
  \begin{bmatrix}
  \mu_\phi(x) \\
  \sigma^2_\phi(x)
  \end{bmatrix} = \underbrace{\text{enc}_\phi(x)} \quad \in \mathbb{R}^{2d}
  \]
  any sentence encoder (e.g., LSTM last state)

- The Gaussian parameterization enables a particularly effective estimation of ELBO
  - KL between Gaussians: Closed form
  - Differentiable sampling by reparameterization trick:
    \[ z \sim \mathcal{N}(\mu, \sigma^2) \iff z = \mu + \sigma \cdot \epsilon \text{ where } \epsilon \sim \mathcal{N}(0, 1) \]
Example: Gaussian VAE for Language Modeling (Cont.)

- ELBO for a single sentence $x$ for clarity

- KL term:

$$D_{KL}(q_{\phi}(\cdot| x)||\pi_{\theta}) = D_{KL}(\mathcal{N}(\mu_{\phi}(x), diag(\sigma_{\phi}^2(x)))||\mathcal{N}(0_d, I_{d \times d}))$$

$$= \frac{1}{2} \left( \sum_{i=1}^{d} [\sigma_{\phi}^2(x)]_i + [\mu_{\phi}(x)]_i^2 - 1 - \log[\sigma_{\phi}^2(x)]_i \right)$$

- Reconstruction term: Single-sample estimation, $\epsilon \sim \mathcal{N}(0_d, I_{d \times d})$

$$\mathbb{E}_{z \sim q_{\phi}(\cdot| x)} [\log \kappa_{\theta}(x| z)] \approx \log \kappa_{\theta}(x| \mu_{\phi}(x) + \sigma_{\phi}(x) \odot \epsilon)$$

$$\hat{R}_x(\theta, \phi)$$

- Take a gradient step on $\beta D_{KL}(q_{\phi}(\cdot| x)||\pi_{\theta}) - \hat{R}_x(\theta, \phi)$. 