Lecture 17: Unsupervised Learning

Alan Ritter

(many slides from Greg Durrett)
Wei Xu will present on Friday

No Class on December 4

Final Project Presentations are during the final exam time scheduled on December 12
What data do we learn from?

- Supervised settings:
  - Tagging: POS, NER
  - Parsing: constituency, dependency, semantic parsing
  - IE, MT, QA, ...

- Semi-supervised models
  - Word embeddings / word clusters (helpful for nearly all tasks)
  - Language models for machine translation
  - Learn linguistic structure from unlabeled data and use it?
This Lecture

- Discrete linguistic structure from generative models: unsupervised POS induction
  - Expectation maximization for learning HMMs

- Continuous structure with generative models: variational autoencoders

- Continuous structure with “discriminative” models: transfer learning
EM for HMMs
Recall: Hidden Markov Models

- Input $x = (x_1, ..., x_n)$  
  Output $y = (y_1, ..., y_n)$

- Observation $(x)$ depends only on current state $(y)$
- Multinomials: tag x tag transitions, tag x word emissions
- $P(x | y)$ is a distribution over all words in the vocabulary — not a distribution over features (but could be!)
Unsupervised Learning

- Can we induce linguistic structure? Thought experiment...

  a b a c c c c
  b a c c c

- What’s a two-state HMM that could produce this?

- What if I show you this sequence?

  a a b c c a a

- What did you do? Use current model parameters + data to refine your model. This is what EM will do
Part-of-Speech Induction

- Input $\mathbf{x} = (x_1, \ldots, x_n)$  
  Output $\mathbf{y} = (y_1, \ldots, y_n)$

- Assume we don’t have access to labeled examples — how can we learn a POS tagger?

- Key idea: optimize $P(\mathbf{x}) = \sum_{\mathbf{y}} P(\mathbf{y}, \mathbf{x})$  
  Generative model explains the data $\mathbf{x}$; the right HMM makes it look likely

- Optimizing marginal log-likelihood with no labels $\mathbf{y}$:

  $\mathcal{L}(x_1, \ldots, D) = \sum_{i=1}^{D} \log \sum_{\mathbf{y}} P(\mathbf{y}, x_i)$  
  non-convex optimization problem
Part-of-Speech Induction

- Input $x = (x_1, ..., x_n)$  Output $y = (y_1, ..., y_n)$

- Optimizing marginal log-likelihood with no labels $y$:
  \[
  \mathcal{L}(x_1, ..., D) = \sum_{i=1}^{D} \log \sum_{y} P(y, x_i)
  \]

- Can’t use a discriminative model; $\sum_{y} P(y|x) = 1$, doesn’t model $x$

- What’s the point of this? Model has inductive bias and so should learn some useful latent structure $y$ (clustering effect)

- EM is just one procedure for optimizing this kind of objective
Expectation Maximization

\[
\log \sum_y P(x, y|\theta) = \log \sum_y q(y) \frac{P(x, y|\theta)}{q(y)} \quad \text{Variational approximation } q \quad \text{is a trick we’ll return to later!}
\]

\[
\geq \sum_y q(y) \log \frac{P(x, y|\theta)}{q(y)} \quad \text{Jensen’s inequality (uses concavity of log)}
\]

\[
= \mathbb{E}_{q(y)} \log P(x, y|\theta) + \text{Entropy}[q(y)]
\]

- Condition on parameters \( \theta \)
- Can optimize this lower-bound on log likelihood instead of log-likelihood

Adapted from Leon Gu
**Expectation Maximization**

\[
\log \sum_y P(x, y|\theta) \geq \mathbb{E}_{q(y)} \log P(x, y|\theta) + \text{Entropy}[q(y)]
\]

- If \( q(y) = P(y|x, \theta) \), this bound ends up being tight.

- **Expectation-maximization:** alternating maximization of the lower bound over \( q \) and \( \theta \)
  
  - Current timestep = \( t \), have parameters \( \theta^{t-1} \)
  
  - **E-step:** maximize w.r.t. \( q \); that is, \( q^t = P(y|x, \theta^{t-1}) \)
  
  - **M-step:** maximize w.r.t. \( \theta \); that is, \( \theta^t = \arg\max_\theta \mathbb{E}_{q^t} \log P(x, y|\theta) \)

Adapted from Leon Gu
EM for HMMs

- Expectation-maximization: alternating maximization
  - E-step: maximize w.r.t. $q$; that is, $q^t = P(y|x, \theta^{t-1})$
  - M-step: maximize w.r.t. $\theta$; that is, $\theta^t = \arg\max_{\theta} \mathbb{E}_{q^t} \log P(x, y|\theta)$

- E-step: for an HMM: run forward-backward with the given parameters
- Compute $P(y_i = s|x, \theta^{t-1})$, $P(y_i = s_1, y_{i+1} = s_2|x, \theta^{t-1})$
  - tag marginals at each position
  - tag pair marginals at each position

- M-step: set parameters to optimize the crazy argmax term
M-Step

- Recall how we maximized log $P(x,y)$: read counts off data

  - $count(DT, \text{the}) = 1$  
    $P(\text{the} | DT) = 1$
  - $count(DT, \text{dog}) = 0$  
    $P(\text{dog} | DT) = 0$
  - $count(NN, \text{the}) = 0$  
    $P(\text{the} | NN) = 0$
  - $count(NN, \text{dog}) = 1$  
    $P(\text{dog} | NN) = 1$

- Same procedure, but maximizing $P(x,y)$ in expectation under $q$
  - $q$ means that $q$ specifies *fractional counts*

  - $count(DT, \text{the}) = 0.9$  
    $P(\text{the} | DT) = 0.75$
  - $count(DT, \text{dog}) = 0.3$  
    $P(\text{dog} | DT) = 0.25$
  - $count(NN, \text{the}) = 0.1$  
    $P(\text{the} | NN) = 0.125$
  - $count(NN, \text{dog}) = 0.7$  
    $P(\text{dog} | NN) = 0.875$
M-Step

- Same for transition probabilities

<table>
<thead>
<tr>
<th>Transition</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT—NN</td>
<td>0.6</td>
</tr>
<tr>
<td>DT—DT</td>
<td>0.1</td>
</tr>
<tr>
<td>NN—DT</td>
<td>0.2</td>
</tr>
<tr>
<td>NN—NN</td>
<td>0.1</td>
</tr>
</tbody>
</table>

- The transition probabilities:
  - $P(DT|DT) = \frac{1}{7}$
  - $P(NN|DT) = \frac{6}{7}$
  - $P(DT|NN) = \frac{2}{3}$
  - $P(NN|NN) = \frac{1}{3}$

The word sequence: the dog
How does EM learn things?

- Initialize (M-step 0):

  - Emissions
    
    \[
    \begin{align*}
    P(\text{the}|\text{DT}) &= 0.9 & P(\text{the}|\text{NN}) &= 0.05 \\
    P(\text{dog}|\text{DT}) &= 0.05 & P(\text{dog}|\text{NN}) &= 0.9 \\
    P(\text{marsupial}|\text{DT}) &= 0.05 & P(\text{marsupial}|\text{NN}) &= 0.05
    \end{align*}
    \]

  - Transition probabilities: uniform

- E-step 1: (all values are approximate)

  \[
  \begin{align*}
  \text{DT: } 0.95 & \quad \text{DT: } 0.05 & \text{DT: } 0.95 & \text{DT: } 0.5 \\
  \text{NN: } 0.05 & \quad \text{NN: } 0.95 & \text{NN: } 0.05 & \text{NN: } 0.5
  \end{align*}
  \]

  the \quad dog \quad the \quad marsupial

  uniform
How does EM learn things?

- E-step 1:
  - $\text{DT: 0.95}$  $\text{DT: 0.05}$  $\text{DT: 0.95}$  $\text{DT: 0.5}$
  - $\text{NN: 0.05}$  $\text{NN: 0.95}$  $\text{NN: 0.05}$  $\text{NN: 0.5}$
  - the  dog  the  marsupial

- M-step 1:
  - Emissions aren’t so different
  - Transition probabilities (approx): $P(\text{NN}|\text{DT}) = 3/4$, $P(\text{DT}|\text{DT}) = 1/4$
How does EM learn things?

- E-step 2:
  
  DT: 0.95  DT: 0.05  DT: 0.95  DT: 0.30
  NN: 0.05  NN: 0.95  NN: 0.05  NN: 0.70
  the      dog       the      marsupial

- M-step 1:
  
  - Emissions aren’t so different
  - Transition probabilities (approx): $P(\text{NN}|\text{DT}) = 3/4$, $P(\text{DT}|\text{DT}) = 1/4$
How does EM learn things?

- **E-step 2:**

  - DT: 0.95  DT: 0.05
  - NN: 0.05  **NN: 0.95**

  - the  dog

- **M-step 2:**

  - Emission $P(\text{marsupial}|\text{NN}) > P(\text{marsupial}|\text{DT})$
  - Remember to tag marsupial as NN in the future!
  - Context constrained what we learned! That’s how data helped us
How does EM learn things?

- Can think of $q$ as a kind of “fractional annotation”
- E-step: compute annotations (posterior under current model)
- M-step: supervised learning with those fractional annotations
- Initialize with some reasonable weights, alternate E and M until convergence
EM’s Lower Bound

\[ \mathcal{L}(x_1, \ldots, D) = \sum_{i=1}^{D} \log \sum_y P(y, x_i) \]

\[ \mathcal{L}(x_1, \ldots, D; \theta) \]

Initialize probabilities \( \theta \)

repeat

- Compute expected counts \( e \)
- Fit parameters \( \theta \)

until convergence

slide credit: Taylor Berg-Kirkpatrick
EM’s Lower Bound

\[ \mathcal{L}(x_1, \ldots, D) = \sum_{i=1}^{D} \log \sum_{y} P(y, x_i) \]

\[ \mathcal{L}(x_1, \ldots, D; \theta) \]

- **E-step**: compute \( q \) which gives this lower bound

\[ \text{repeat} \]
- Compute expected counts \( e \)
- Fit parameters \( \theta \)
\[ \text{until} \] convergence

- Initialize probabilities \( \theta \)

Slide credit: Taylor Berg-Kirkpatrick
EM’s Lower Bound

\[ \mathcal{L}(x_1, \ldots, D) = \sum_{i=1}^{D} \log \sum_y P(y, x_i) \]

- **M-step:** find maximum of lower bound

---

Initialize probabilities \( \theta 

repeat

- Compute expected counts \( e \)
- Fit parameters \( \theta \)

until convergence
EM’s Lower Bound

\[ \mathcal{L}(x_1, \ldots, D) = \sum_{i=1}^{D} \log \sum_{y} P(y, x_i) \]

\[ \mathcal{L}(x_1, \ldots, D; \theta) \]

Initialize probabilities \( \theta \)

repeat

- Compute expected counts \( e \)
- Fit parameters \( \theta \)

until convergence

- E-step 2: re-estimate \( q \)

slide credit: Taylor Berg-Kirkpatrick
EM’s Lower Bound

\[ \mathcal{L}(x_1, \ldots, D) = \sum_{i=1}^{D} \log \sum_y P(y, x_i) \]

Initialize probabilities \( \theta \)

repeat

- Compute expected counts \( e \)
- Fit parameters \( \theta \)

until convergence

\[ \mathcal{L}(x_1, \ldots, D; \theta) \]

- E-step 2: re-estimate \( q \)

slide credit: Taylor Berg-Kirkpatrick
EM’s Lower Bound

\[ \mathcal{L}(x_1, \ldots, D) = \sum_{i=1}^{D} \log \sum_{y} P(y, x_i) \]

\[ \mathcal{L}(x_1, \ldots, D; \theta) \]

Initialize probabilities \( \theta \)

repeat

- Compute expected counts \( e \)
- Fit parameters \( \theta \)

until convergence

slide credit: Taylor Berg-Kirkpatrick
EM’s Lower Bound

\[ \mathcal{L}(x_1, \ldots, D) = \sum_{i=1}^{D} \log \sum_y P(y, x_i) \]

Initialize probabilities \( \theta \)

repeat
- Compute expected counts \( e \)
- Fit parameters \( \theta \)
until convergence

slide credit: Taylor Berg-Kirkpatrick
EM’s Lower Bound

\[ \mathcal{L}(x_1, \ldots, D) = \sum_{i=1}^{D} \log \sum_y P(y, x_i) \]

\[ \mathcal{L}(x_1, \ldots, D; \theta) \]

Initialize probabilities \( \theta \)

repeat

- Compute expected counts \( e \)
- Fit parameters \( \theta \)

until convergence

slide credit: Taylor Berg-Kirkpatrick
Merialdo (1994): you have a whitelist of tags for each word

Learn parameters on $k$ examples to start, use those to initialize EM, run on 1 million words of unlabeled data

Tag dictionary + data should get us started in the right direction...
### Part-of-speech Induction

<table>
<thead>
<tr>
<th>Number of tagged sentences used for the initial model</th>
<th>0</th>
<th>100</th>
<th>2000</th>
<th>5000</th>
<th>10000</th>
<th>20000</th>
<th>all</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Iter</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>77.0</td>
<td>90.0</td>
<td><strong>95.4</strong></td>
<td>96.2</td>
<td>96.6</td>
<td>96.9</td>
<td>97.0</td>
</tr>
<tr>
<td>1</td>
<td>80.5</td>
<td>92.6</td>
<td>95.8</td>
<td>96.3</td>
<td>96.6</td>
<td>96.7</td>
<td>96.8</td>
</tr>
<tr>
<td>2</td>
<td>81.8</td>
<td>93.0</td>
<td>95.7</td>
<td>96.1</td>
<td>96.3</td>
<td>96.4</td>
<td>96.4</td>
</tr>
<tr>
<td>3</td>
<td>83.0</td>
<td>93.1</td>
<td>95.4</td>
<td>95.8</td>
<td>96.1</td>
<td>96.2</td>
<td>96.2</td>
</tr>
<tr>
<td>4</td>
<td>84.0</td>
<td>93.0</td>
<td>95.2</td>
<td>95.5</td>
<td>95.8</td>
<td>96.0</td>
<td>96.0</td>
</tr>
<tr>
<td>5</td>
<td>84.8</td>
<td>92.9</td>
<td>95.1</td>
<td>95.4</td>
<td>95.6</td>
<td>95.8</td>
<td>95.8</td>
</tr>
<tr>
<td>6</td>
<td>85.3</td>
<td>92.8</td>
<td>94.9</td>
<td>95.2</td>
<td>95.5</td>
<td>95.6</td>
<td>95.7</td>
</tr>
<tr>
<td>7</td>
<td>85.8</td>
<td>92.8</td>
<td>94.7</td>
<td>95.1</td>
<td>95.3</td>
<td>95.5</td>
<td>95.5</td>
</tr>
<tr>
<td>8</td>
<td>86.1</td>
<td>92.7</td>
<td>94.6</td>
<td>95.0</td>
<td>95.2</td>
<td>95.4</td>
<td>95.4</td>
</tr>
<tr>
<td>9</td>
<td>86.3</td>
<td>92.6</td>
<td>94.5</td>
<td>94.9</td>
<td>95.1</td>
<td>95.3</td>
<td>95.3</td>
</tr>
<tr>
<td>10</td>
<td>86.6</td>
<td>92.6</td>
<td>94.4</td>
<td>94.8</td>
<td>95.0</td>
<td>95.2</td>
<td>95.2</td>
</tr>
</tbody>
</table>

- Small amounts of data > large amounts of unlabeled data
- Running EM *hurts* performance once you have labeled data

Merialdo (1994)
Two Hours of Annotation

<table>
<thead>
<tr>
<th>Human Annotations</th>
<th>0. No EM</th>
<th>1. EM only</th>
<th>2. With LP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial data</td>
<td>T</td>
<td>K</td>
<td>U</td>
</tr>
<tr>
<td>KIN tokens A</td>
<td>72</td>
<td>90</td>
<td>58</td>
</tr>
<tr>
<td>KIN types A</td>
<td>63</td>
<td>77</td>
<td>32</td>
</tr>
<tr>
<td>MLG tokens B</td>
<td>74</td>
<td>89</td>
<td>49</td>
</tr>
<tr>
<td>MLG types B</td>
<td>71</td>
<td>87</td>
<td>46</td>
</tr>
<tr>
<td>ENG tokens A</td>
<td>63</td>
<td>83</td>
<td>38</td>
</tr>
<tr>
<td>ENG types A</td>
<td>66</td>
<td>76</td>
<td>37</td>
</tr>
<tr>
<td>ENG tokens B</td>
<td>70</td>
<td>87</td>
<td>44</td>
</tr>
<tr>
<td>ENG types B</td>
<td>69</td>
<td>83</td>
<td>38</td>
</tr>
</tbody>
</table>

- Kinyarwanda and Malagasy (two actual low-resource languages)
- Label propagation (technique for using dictionary labels) helps a lot, with data that was collected in two hours

Garrette and Baldridge (2013)
Variational Autoencoders
Continuous Latent Variables

- For discrete latent variables $y$, we optimized: 
  \[ P(x) = \sum_{y} P(y, x) \]

- What if we want to use continuous latent variables?
  \[ P(z, x) = P(z)P(x|z) \]
  \[ P(x) = \int P(z)P(x|z) \, \partial z \]

- Can use EM here when $P(z)$ and $P(x|z)$ are Gaussians

- What if we want $P(x|z)$ to be something more complicated, like an LSTM with $z$ as the initial state?
Deep Generative Models

\[
P(z, x) = P(z)P(x|z)
\]

- \(z\) is a latent variable which should control the generation of the sentence, maybe capture something about its topic
Deep Generative Models

\[
\log \int_z P(x, z|\theta) = \log \int_z q(z) \frac{P(x, z|\theta)}{q(z)} \geq \int_z q(z) \log \frac{P(x, z|\theta)}{q(z)} \quad \text{Jensen}
\]

\[
= \mathbb{E}_{q(z|x)}[-\log q(z|x) + \log P(x, z|\theta)]
\]

\[
= \mathbb{E}_{q(z|x)}[\log P(x|z, \theta)] - \text{KL}(q(z|x)\|P(z))
\]

“make the data likely under q” “make q close to the prior” (discriminative)

- KL divergence: distance metric over distributions (more dissimilar <=> higher KL)
Variational Autoencoders

\[ \mathbb{E}_{q(z|x)}[\log P(x|z, \theta)] - \text{KL}(q(z|x) \| P(z)) \]

**Generative model (test):**

1. \( z \sim P(z) \)
2. \( x \)

**Autoencoder (training):**

1. Input \( x \)
2. \( q(z|x) \) distribution over \( z \)
3. Generative model
4. \( x \) Maximize \( P(x|z, \theta) \)

---

Miao et al. (2015)
Training VAEs

\[ \mathbb{E}_{q(z|x)}[\log P(x|z, \theta)] - KL(q(z|x) || P(z)) \]

- Choose \( q \) to be Gaussian with parameters that are computed from \( x \)
  \[ q = N(\mu(x), \text{diag}(\sigma^2(x))) \]
- \( \mu \) and \( \sigma \) are computed from an LSTM over \( x \), call their parameters \( \phi \)
- How to handle the expectation? Sampling

Autoencoder (training):

\[ \phi \]

“inference network”

\[ \theta \]

generative model

Miao et al. (2015)
Training VAEs

For each example $\mathbf{x}$

Compute $q$ (run forward pass to compute mu and sigma)

For some number of samples

Sample $z \sim q$

Compute $P(\mathbf{x}|z)$ and compute loss

Backpropagate to update phi, theta

Autoencoder (training):

\[
\begin{align*}
&\phi \\
\rightarrow & q(z|x) \\
& \theta \\
\rightarrow & \text{generative model}
\end{align*}
\]
Autoencoders

- Inference network (q) is the encoder and generator is the decoder

- Same computation graph as VAE, add KL divergence term to make the objective the same

- Another interpretation: train an autoencoder and add Gaussian noise

Gaussian noise

the movie was great + <s> the movie was good [STOP]
What does gradient encourage latent space to do?

\[ \mathbb{E}_{q(z|x)}[\log P(x|z, \theta)] + KL(q(z|x)\|P(z)) \]
What do VAEs do?

- Let us encode a sentence and generate similar sentences:

<table>
<thead>
<tr>
<th>INPUT</th>
<th>MEAN</th>
<th>SAMPLE 1</th>
<th>SAMPLE 2</th>
<th>SAMPLE 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>we looked out at the setting sun</td>
<td>they were laughing at the same time</td>
<td>i went to the kitchen</td>
<td>how are you doing?</td>
<td></td>
</tr>
<tr>
<td>i went to the kitchen</td>
<td>i went to my apartment</td>
<td>i looked around the room</td>
<td>what are you doing?</td>
<td></td>
</tr>
<tr>
<td>i turned back to the table</td>
<td></td>
<td></td>
<td>&quot;are you sure?&quot;</td>
<td></td>
</tr>
</tbody>
</table>

- Style transfer: also condition on sentiment, change sentiment

- ...or use the latent representations for semi-supervised learning

Positive

⇒ ARAE
great indoor mall.
⇒ Cross-AE
no smoking mall.

Positive

⇒ ARAE
it has a great atmosphere, with wonderful service.
⇒ Cross-AE
it has no taste, with a complete jerk.

Positive

⇒ ARAE
it has a great horrible food and run out service.
⇒ Cross-AE

Bowman et al. (2016), Zhao et al. (2017)
BERT
Goals of Unsupervised Learning

- We want to use unlabeled data, but EM “requires” generative models. Are models like this really necessary?

- word2vec: predict nearby word given context. This wasn’t generative, but the supervision is free...

- Language modeling is a “more contextualized” form of word2vec
ELMo

- Generative model of the data!
- Train one model in each direction on 1B words, use the LSTM hidden states as context-aware token representations

\[ P(x_i | x_1, \ldots, x_{i-1}) = \text{LSTM}(x_1, \ldots, x_{i-1}) \]
Recall: Self-Attention

- Each word forms a “query” which then computes attention over each word

\[ \alpha_{i,j} = \text{softmax}(x_i^T x_j) \quad \text{scalar} \]

\[ x'_i = \sum_{j=1}^{n} \alpha_{i,j} x_j \quad \text{vector = sum of scalar * vector} \]

- Multiple “heads” analogous to different convolutional filters. Use parameters \( W_k \) and \( V_k \) to get different attention values + transform vectors

\[ \alpha_{k,i,j} = \text{softmax}(x_i^T W_k x_j) \quad x'_{k,i} = \sum_{j=1}^{n} \alpha_{k,i,j} V_k x_j \]

Vaswani et al. (2017)
Recall: Transformers

» Augment word embedding with position embeddings, each dim is a sine/cosine wave of a different frequency. Closer points = higher dot products

» Works essentially as well as just encoding position as a one-hot vector

Vaswani et al. (2017)
BERT

AI2 made ELMo in spring 2018, GPT was released in summer 2018, BERT came out October 2018

Three major changes compared to ELMo:
- Transformers instead of LSTMs (transformers in GPT as well)
- Bidirectional <=> Masked LM objective instead of standard LM
- Fine-tune instead of freeze at test time
ELMo is a unidirectional model (as is GPT): we can concatenate two unidirectional models, but is this the right thing to do?

ELMo reprs look at each direction in isolation; BERT looks at them jointly.

A stunning ballet dancer, Copeland is one of the best performers to see live.

Devlin et al. (2019)
How to learn a “deeply bidirectional” model? What happens if we just replace an LSTM with a transformer?

Transformer LMs have to be “one-sided” (only attend to previous tokens), not what we want
Masked Language Modeling

- How to prevent cheating? Next word prediction fundamentally doesn't work for bidirectional models, instead do masked language modeling.

- BERT formula: take a chunk of text, predict 15% of the tokens.
  - For 80% (of the 15%), replace the input token with [MASK].
  - For 10%, replace with random.
  - For 10%, keep same.

Devlin et al. (2019)
Next “Sentence” Prediction

- Input: [CLS] Text chunk 1 [SEP] Text chunk 2
- 50% of the time, take the true next chunk of text, 50% of the time take a random other chunk. Predict whether the next chunk is the “true” next
- BERT objective: masked LM + next sentence prediction

```
```

Devlin et al. (2019)
BERT Architecture

- **BERT Base**: 12 layers, 768-dim per wordpiece token, 12 heads. Total params = 110M
- **BERT Large**: 24 layers, 1024-dim per wordpiece token, 16 heads. Total params = 340M
- Positional embeddings and segment embeddings, 30k word pieces
- This is the model that gets **pre-trained** on a large corpus

Devlin et al. (2019)
What can BERT do?

- CLS token is used to provide classification decisions
- Sentence pair tasks (entailment): feed both sentences into BERT
- BERT can also do tagging by predicting tags at each word piece

Devlin et al. (2019)
What can BERT do?

- How does BERT model this sentence pair stuff?
- Transformers can capture interactions between the two sentences, even though the NSP objective doesn’t really cause this to happen.

[CLS] A boy plays in the snow [SEP] A boy is outside
What can BERT NOT do?

- BERT **cannot** generate text (at least not in an obvious way)

- Not an autoregressive model, can do weird things like stick a [MASK] at the end of a string, fill in the mask, and repeat

- Masked language models are intended to be used primarily for “analysis” tasks
Fine-tuning BERT

- Fine-tune for 1-3 epochs, batch size 2-32, learning rate 2e-5 - 5e-5
- Large changes to weights up here (particularly in last layer to route the right information to [CLS])
- Smaller changes to weights lower down in the transformer
- Small LR and short fine-tuning schedule mean weights don’t change much
- More complex “triangular learning rate” schemes exist

(b) Single Sentence Classification Tasks: SST-2, CoLA
# Fine-tuning BERT

![Fine-tuning BERT Table](image)

- BERT is typically better if the whole network is fine-tuned, unlike ELMo

---

Peters, Ruder, Smith (2019)
## Evaluation: GLUE

<table>
<thead>
<tr>
<th>Corpus</th>
<th></th>
<th>Train</th>
<th></th>
<th>Test</th>
<th>Task</th>
<th>Metrics</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoLA</td>
<td></td>
<td>8.5k</td>
<td>1k</td>
<td>1k</td>
<td>acceptability</td>
<td>Matthews corr.</td>
<td>misc.</td>
</tr>
<tr>
<td>SST-2</td>
<td></td>
<td>67k</td>
<td>1.8k</td>
<td>1.8k</td>
<td>sentiment</td>
<td>acc.</td>
<td>movie reviews</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Single-Sentence Tasks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MRPC</td>
<td></td>
<td>3.7k</td>
<td>1.7k</td>
<td>1.7k</td>
<td>paraphrase</td>
<td>acc./F1</td>
<td>news</td>
</tr>
<tr>
<td>STS-B</td>
<td></td>
<td>7k</td>
<td>1.4k</td>
<td>1.4k</td>
<td>sentence similarity</td>
<td>Pearson/Spearman corr.</td>
<td>misc.</td>
</tr>
<tr>
<td>QQP</td>
<td></td>
<td>364k</td>
<td>391k</td>
<td>391k</td>
<td>paraphrase</td>
<td>acc./F1</td>
<td>social QA questions</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Similarity and Paraphrase Tasks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MNLI</td>
<td></td>
<td>393k</td>
<td>20k</td>
<td>20k</td>
<td>NLI</td>
<td>matched acc./mismatched acc.</td>
<td>misc.</td>
</tr>
<tr>
<td>QNLI</td>
<td></td>
<td>105k</td>
<td>5.4k</td>
<td>5.4k</td>
<td>QA/NLI</td>
<td>acc.</td>
<td>Wikipedia</td>
</tr>
<tr>
<td>RTE</td>
<td></td>
<td>2.5k</td>
<td>3k</td>
<td>3k</td>
<td>NLI</td>
<td>acc.</td>
<td>news, Wikipedia</td>
</tr>
<tr>
<td>WNLNI</td>
<td></td>
<td>634</td>
<td>146</td>
<td>146</td>
<td>coreference/NLI</td>
<td>acc.</td>
<td>fiction books</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Inference Tasks</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Wang et al. (2019)
Results

<table>
<thead>
<tr>
<th>System</th>
<th>MNLI-(m/mm) 392k</th>
<th>QQP 363k</th>
<th>QNLI 108k</th>
<th>SST-2 67k</th>
<th>CoLA 8.5k</th>
<th>STS-B 5.7k</th>
<th>MRPC 3.5k</th>
<th>RTE 2.5k</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-OpenAI SOTA</td>
<td>80.6/80.1</td>
<td>66.1</td>
<td>82.3</td>
<td>93.2</td>
<td>35.0</td>
<td>81.0</td>
<td>86.0</td>
<td>61.7</td>
<td>74.0</td>
</tr>
<tr>
<td>BiLSTM+ELMo+Attn</td>
<td>76.4/76.1</td>
<td>64.8</td>
<td>79.9</td>
<td>90.4</td>
<td>36.0</td>
<td>73.3</td>
<td>84.9</td>
<td>56.8</td>
<td>71.0</td>
</tr>
<tr>
<td>OpenAI GPT</td>
<td>82.1/81.4</td>
<td>70.3</td>
<td>88.1</td>
<td>91.3</td>
<td>45.4</td>
<td>80.0</td>
<td>82.3</td>
<td>56.0</td>
<td>75.2</td>
</tr>
<tr>
<td>BERT\textsubscript{BASE}</td>
<td>84.6/83.4</td>
<td>71.2</td>
<td>90.1</td>
<td>93.5</td>
<td>52.1</td>
<td>85.8</td>
<td>88.9</td>
<td>66.4</td>
<td>79.6</td>
</tr>
<tr>
<td>BERT\textsubscript{LARGE}</td>
<td>\textbf{86.7/85.9}</td>
<td>\textbf{72.1}</td>
<td>\textbf{91.1}</td>
<td>\textbf{94.9}</td>
<td>\textbf{60.5}</td>
<td>\textbf{86.5}</td>
<td>\textbf{89.3}</td>
<td>\textbf{70.1}</td>
<td>\textbf{81.9}</td>
</tr>
</tbody>
</table>

- Huge improvements over prior work (even compared to ELMo)
- Effective at “sentence pair” tasks: textual entailment (does sentence A imply sentence B), paraphrase detection

Devlin et al. (2018)
RoBERTa

“Robustly optimized BERT”

‣ 160GB of data instead of 16 GB

‣ Dynamic masking: standard BERT uses the same MASK scheme for every epoch, RoBERTa recomputes them

‣ New training + more data = better performance

<table>
<thead>
<tr>
<th>Model</th>
<th>data</th>
<th>bsz</th>
<th>steps</th>
<th>SQuAD (v1.1/2.0)</th>
<th>MNLI-m</th>
<th>SST-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoBERTa</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>with BOOKS + WIKI</td>
<td>16GB</td>
<td>8K</td>
<td>100K</td>
<td>93.6/87.3</td>
<td>89.0</td>
<td>95.3</td>
</tr>
<tr>
<td>+ additional data (§3.2)</td>
<td>160GB</td>
<td>8K</td>
<td>100K</td>
<td>94.0/87.7</td>
<td>89.3</td>
<td>95.6</td>
</tr>
<tr>
<td>+ pretrain longer</td>
<td>160GB</td>
<td>8K</td>
<td>300K</td>
<td>94.4/88.7</td>
<td>90.0</td>
<td>96.1</td>
</tr>
<tr>
<td>+ pretrain even longer</td>
<td>160GB</td>
<td>8K</td>
<td>500K</td>
<td><strong>94.6/89.4</strong></td>
<td><strong>90.2</strong></td>
<td><strong>96.4</strong></td>
</tr>
<tr>
<td>BERT_Large</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>with BOOKS + WIKI</td>
<td>13GB</td>
<td>256</td>
<td>1M</td>
<td>90.9/81.8</td>
<td>86.6</td>
<td>93.7</td>
</tr>
</tbody>
</table>
GPT/GPT2
OpenAI GPT/GPT2

‣ “ELMo with transformers” (works better than ELMo)

‣ Train a single unidirectional transformer LM on long contexts

‣ GPT2: trained on 40GB of text collected from upvoted links from reddit

‣ 1.5B parameters — by far the largest of these models trained as of March 2019

‣ Because it's a language model, we can generate from it

Radford et al. (2019)

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Layers</th>
<th>$d_{model}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>117M</td>
<td>12</td>
<td>768</td>
</tr>
<tr>
<td>345M</td>
<td>24</td>
<td>1024</td>
</tr>
<tr>
<td>762M</td>
<td>36</td>
<td>1280</td>
</tr>
<tr>
<td>1542M</td>
<td>48</td>
<td>1600</td>
</tr>
</tbody>
</table>
Miley Cyrus was caught shoplifting from Abercrombie and Fitch on Hollywood Boulevard today.

The 19-year-old singer was caught on camera being escorted out of the store by security guards.

The singer was wearing a black hoodie with the label ‘Blurred Lines’ on the front and ‘Fashion Police’ on the back.

Scroll down for video

Shoplifting: Miley Cyrus was caught shoplifting from Abercrombie and Fitch on Hollywood Boulevard today (pictured)

The singer was wearing a black hoodie with the label ‘Blurred Lines’ on the front and ‘Fashion Police’ on the back

The singer was also wearing a pair of black-rimmed glasses, a black jacket, black jeans and black sandals.

She was carrying a pair of black and white striped gloves and a small black bag.
Open Questions

1) How novel is the stuff being generated? (Is it just doing nearest neighbors on a large corpus?)

2) How do we understand and distill what is learned in this model?

3) How do we harness these priors for conditional generation tasks (summarization, generate a report of a basketball game, etc.)

4) Is this technology dangerous? (OpenAI has only released 774M param model, not 1.5B yet)
Pre-Training Cost (with Google/AWS)

- BERT: Base $500, Large $7000
- Grover-MEGA: $25,000
- XLNet (BERT variant): $30,000 — $60,000 (unclear)

This is for a single pre-training run...developing new pre-training techniques may require many runs

*Fine-tuning* these models can typically be done with a single GPU (but may take 1-3 days for medium-sized datasets)

Pushing the Limits

- NVIDIA: trained 8.3B parameter GPT model (5.6x the size of GPT-2)

- Arguable these models are still underfit: larger models still get better held-out perplexities

NVIDIA blog (Narasimhan, August 2019)
Colossal Cleaned Common Crawl: 750 GB of text

We still haven't hit the limit of bigger data being useful

Raffel et al. (October 23, 2019)
BART

- Sequence-to-sequence BERT variant: permute/make/delete tokens, then predict full sequence autoregressively
- For downstream tasks: feed document into both encoder + decoder, use decoder hidden state as output
- Good results on dialogue, summarization tasks

Lewis et al. (October 30, 2019)
Analysis
What does BERT learn?

- Heads on transformers learn interesting and diverse things: content heads (attend based on content), positional heads (based on position), etc.

Clark et al. (2019)
What does BERT learn?

- **Direct objects** attend to their verbs
  - 86.8% accuracy at the `dobj` relation

- **Noun modifiers** (e.g., determiners) attend to their noun
  - 94.3% accuracy at the `det` relation

- **Coreferent** mentions attend to their antecedents
  - 65.1% accuracy at linking the head of a coreferent mention to the head of an antecedent

Still way worse than what supervised systems can do, but interesting that this is learned organically

Clark et al. (2019)
Probing BERT

- Try to predict POS, etc. from each layer. Learn mixing weights
  \[ h_{i,\tau} = \gamma_{\tau} \sum_{\ell=0}^{L} s_{\tau}^{(\ell)} h_{i}^{(\ell)} \]
  representation of wordpiece \( i \) for task \( \tau \)

- Plot shows \( s \) weights (blue) and performance deltas when an additional layer is incorporated (purple)

- BERT “rediscover the classical NLP pipeline”: first syntactic tasks then semantic ones

Tenney et al. (2019)
Compressing BERT

- Remove 60+% of BERT’s heads with minimal drop in performance
- DistilBERT (Sanh et al., 2019): nearly as good with half the parameters of BERT (via knowledge distillation)

(b) Evolution of accuracy on the MultiNLI-matched validation set when heads are pruned from BERT according to $I_h$ (solid blue) and accuracy difference (dashed green).

Michel et al. (2019)
Open Questions

- BERT-based systems are state-of-the-art for nearly every major text analysis task

- These techniques are here to stay, unclear what form will win out

- Role of academia vs. industry: no major pretrained model has come purely from academia

- Cost/carbon footprint: a single model costs $10,000+ to train (though this cost should come down)