CS11-747 Neural Networks for NLP
Unsupervised and Semi-supervised Learning of Structure

Graham Neubig

Carnegie Mellon University
Language Technologies Institute

Site
https://phontron.com/class/nn4nlp2017/
Supervised, Unsupervised, Semi-supervised

• Most models handled here are **supervised** learning
  • Model $P(Y|X)$, at training time given both

• Sometimes we are interested in **unsupervised** learning
  • Model $P(Y|X)$, at training time given only $X$

• Or **semi-supervised** learning
  • Model $P(Y|X)$, at training time given both or only $X$
Learning Features vs. Learning Structure
Learning Features vs. Learning Discrete Structure

• Learning features, e.g. word/sentence embeddings:
  this is an example

• Learning discrete structure:
  this is an example →

  this is an example

• Why discrete structure?
  • We may want to model information flow differently
  • More interpretable than features?
Unsupervised Feature Learning (Review)

• When learning embeddings, we have an objective and use the intermediate states of this objective
  
• CBOW
  
• Skip-gram
  
• Sentence-level auto-encoder
  
• Skip-thought vectors
  
• Variational auto-encoder
How do we Use Learned Features?

• To solve tasks directly (Mikolov et al. 2013)

• And by proxy, knowledge base completion, etc., to be covered in a few classes

• To initialize downstream models
What About Discrete Structure?

- We can cluster words
- We can cluster words in context (POS/NER)
- We can learn structure
What is our Objective?

• Basically, a generative model of the data $X$

• Sometimes factorized $P(X|Y)P(Y)$, a traditional generative model

• Sometimes factorized $P(X|Y)P(Y|X)$, an autoencoder

• This can be made mathematically correct through variational autoencoder $P(X|Y)Q(Y|X)$
Clustering Words in Context
A Simple First Attempt

• Train word embeddings

• Perform k-means clustering on them

• Implemented in word2vec (-classes option)

• But what if we want single words to appear in different classes (same surface form, different values)
Hidden Markov Models

- Factored model of $P(X|Y)P(Y)$
- State→state **transition** probabilities
- State→word **emission** probabilities

$$P_T(JJ<s>) * P_T(NN|JJ) * P_T(NN|NN) * P_T(NN|LRB) * \ldots$$

$$P_E(\text{Natural}|JJ) * P_E(\text{Language}|JJ) * P_E(\text{Processing}|JJ) * \ldots$$
Unsupervised Hidden Markov Models

• Change label states to unlabeled numbers

\[ P_T(13|0) \times P_T(17|13) \times P_T(17|17) \times P_T(6|17) \times \ldots \]

\[ P_E(\text{Natural}|13) \times P_E(\text{Language}|17) \times P_E(\text{Processing}|17) \times \ldots \]

• Can be trained with forward-backward algorithm
Hidden Markov Models w/ Gaussian Emissions

• Instead of parameterizing each state with a categorical distribution, we can use a Gaussian (or Gaussian mixture)!

• Long the defacto-standard for speech

• Applied to POS tagging by training to emit word embeddings by Lin et al. (2015)
Featurized Hidden Markov Models (Tran et al. 2016)

- Calculate the transition/emission probabilities with neural networks!
  - **Emission**: Calculate representation of each word in vocabulary w/ CNN, dot product with tag representation and softmax to calculate emission prob
  - **Transition Matrix**: Calculate w/ LSTMs (breaks Markov assumption)

Figure 2: Computational graph of Char-CNN emission network. A character convolutional neural network is used to compute the weight of the linear layer for every minibatch.

Figure 3: A graphical representation of our LSTM transition network. Transition matrix $T_{t-1,t}$ from time step $t-1$ to $t$ is computed based on the hidden state of the LSTM at time $t-1$. 
CRF Autoencoders
(Ammar et al. 2014)

• Like HMMs, but more principled/flexible

• Predict potential functions for tags, try to reconstruct the input from the tags
A Simple Approximation: State Clustering (Giles et al. 1992)

- Simply train an RNN according to a standard loss function (e.g. language model)
- Then cluster the hidden states according to k-means, etc.
Unsupervised Phrase-structured Composition Functions
Soft vs. Hard Tree Structure

• Soft tree structure: use a differentiable gating function

• Hard tree structure: non-differentiable, but allows for more complicated composition methods
One Other Paradigm: Weak Supervision

- **Supervised**: given X, Y to model P(Y|X)
- **Unsupervised**: given X to model P(Y|X)
- **Weakly Supervised**: given X and V to model P(Y|X), under assumption that Y and V are correlated

- Note: different from multi-task or transfer learning because we are given no Y
- Note: different from supervised learning with latent variables, because we care about Y, not V
Gated Convolution
(Cho et al. 2014)

• Can choose whether to use left node, right node, or combination of both

• Trained using MT loss
Learning with RL
(Yogatama et al. 2016)

• Intermediate tree-structured representation for language modeling

• Predict that tree using shift-reduce parsing, sentence representation composed in tree-structured manner

• Reinforcement learning with supervised loss, prediction loss
Learning w/ Layer-wise Reductions (Choi et al. 2017)

- Choose one parent at each layer, reducing size by one
- Train using Gumbel-straightthrough reparameterization trick
- Faster and more effective than RL?
- Williams et al. (2017) find that this gives less trivial trees as well
Learning Dependencies
Phrase Structure vs. Dependency Structure

• Previous methods attempt to learn representations of phrases in tree-structured manner

• We might also want to learn dependencies, that tell which words depend on others
Dependency Model w/ Valence (Klein and Manning 2004)

• Basic idea: top-down dependency based language model that generates left and right sides, then stops

I saw a girl with a telescope

• For both the right and left side, calculate whether to continue generating words, and if yes generate

e.g., a slightly simplified view for word “saw”

\[
P_d(<\text{cont}> | \text{saw}, \leftarrow, \text{false}) \times P_w(l | \text{saw}, \leftarrow, \text{false}) \times \\
P_d(<\text{stop}> | \text{saw}, \leftarrow, \text{true}) \times \\
P_d(<\text{cont}> | \text{saw}, \rightarrow, \text{false}) \times P_w(\text{girl} | \text{saw}, \leftarrow, \text{false}) \times \\
P_d(<\text{cont}> | \text{saw}, \rightarrow, \text{true}) \times P_w(\text{with} | \text{saw}, \leftarrow, \text{true}) \times \\
P_d(<\text{stop}> | \text{saw}, \leftarrow, \text{true})
\]
Unsupervised Dependency Induction w/ Neural Nets (Jiang et al. 2016)

• Simple: parameterize the decision with neural nets instead of with count-based distributions

• Like DMV, train with EM algorithm
Learning Dependency Heads w/ Attention (Kuncoro et al. 2017)

• Given a phrase structure tree, what child is the head word, the most important word in the phrase?

• Idea: create a phrase composition function that uses attention: examine if attention weights follow heads defined by linguistics

<table>
<thead>
<tr>
<th>Noun phrases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canadian (0.09) Auto (0.31) Workers (0.2) union (0.22) president (0.18)</td>
</tr>
<tr>
<td>no (0.29) major (0.05) Eurobond (0.32) or (0.01) foreign (0.01) bond (0.1) offerings (0.22)</td>
</tr>
<tr>
<td>Saatchi (0.12) client (0.14) Philips (0.21) Lighting (0.24) Co. (0.29)</td>
</tr>
<tr>
<td>nonperforming (0.18) commercial (0.23) real (0.25) estate (0.1) assets (0.25)</td>
</tr>
<tr>
<td>the (0.1) Jamaica (0.1) Tourist (0.03) Board (0.17) ad (0.20) account (0.40)</td>
</tr>
<tr>
<td>the (0.0) final (0.18) hour (0.81)</td>
</tr>
<tr>
<td>their (0.0) first (0.23) test (0.77)</td>
</tr>
<tr>
<td>Apple (0.62) , (0.02) Compaq (0.1) and (0.01) IBM (0.25)</td>
</tr>
<tr>
<td>both (0.02) stocks (0.03) and (0.06) futures (0.88)</td>
</tr>
<tr>
<td>NP (0.01) , (0.0) and (0.98) NP (0.01)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Verb phrases</th>
<th>Prepositional phrases</th>
</tr>
</thead>
<tbody>
<tr>
<td>buying (0.31) and (0.25) selling (0.21) NP (0.23)</td>
<td>ADVP (0.14) on (0.72) NP (0.14)</td>
</tr>
<tr>
<td>ADVP (0.27) show (0.29) PRT (0.23) PP (0.21)</td>
<td>ADVP (0.05) for (0.54) NP (0.40)</td>
</tr>
<tr>
<td>pleaded (0.48) ADJP (0.23) PP (0.15) PP (0.08) PP (0.06)</td>
<td>ADVP (0.02) because (0.73) of (0.18) NP (0.07)</td>
</tr>
<tr>
<td>received (0.33) PP (0.18) NP (0.32) PP (0.17)</td>
<td>such (0.31) as (0.65) NP (0.04)</td>
</tr>
<tr>
<td>cut (0.27) NP (0.37) PP (0.22) PP (0.14)</td>
<td>from (0.39) NP (0.49) PP (0.12)</td>
</tr>
<tr>
<td>to (0.99) VP (0.01)</td>
<td>of (0.97) NP (0.03)</td>
</tr>
<tr>
<td>were (0.77) n’t (0.22) VP (0.01)</td>
<td>in (0.93) NP (0.07)</td>
</tr>
<tr>
<td>did (0.39) n’t (0.60) VP (0.01)</td>
<td>by (0.96) S (0.04)</td>
</tr>
<tr>
<td>handle (0.09) NP (0.91)</td>
<td>at (0.99) NP (0.01)</td>
</tr>
<tr>
<td>VP (0.15) and (0.83) VP 0.02</td>
<td>NP (0.1) after (0.83) NP (0.06)</td>
</tr>
</tbody>
</table>
Other Examples
Learning about Word Segmentation from Attention (Boito et al. 2017)

- We want to learn word segmentation in an unsegmented language
- Simple idea: we can inspect the attention matrices from a neural MT system to extract words
Learning Segmentations w/ Reconstruction Loss  (Elsner and Shain 2017)

- Learn segmentations of speech/text that allow for easy reconstruction of the original
- Idea: consistent segmentation should result in easier-to-reconstruct segments
- Train segmentation using policy gradient
Learning Language-level Features (Malaviya et al. 2017)

- All previous work learned features of a single sentence

- Can we learn features of the whole language? e.g. Typology: what is the canonical word order, etc.

- A simple method: train a neural MT system on 1017 languages, and extract its representations
Questions?