Trees!
All grades are now Pass / No Record.
II / TQE status is not affected.

Midterm has been turned into a homework assignment,
and no peer grading on future assignments.
Deadlines are flexible.

Let us know if you’re having trouble accessing videos.

Stay safe!
Recap: labels and sequences
Predicting labels

\[ S = W_2^T f(W_1^T x) \]

\[ f(W_1^T x) = h_1 \]

\[ W_2^T h_1 = S \]

\[ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \text{ input} \]

\[ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \text{ “hidden layer”} \]

\[ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \text{ output} \]
Predicting sequences: n-gram models

$sphinx$ of $black$ quartz 

$[.03]$
Predicting sequences: n-gram models

$sphinx$ of $black$ $quartz$
Predicting sequences: neural networks

judge

my

vow
Labeling sequences: HMMs & CRFs

**HMM:** \[ p(O, Q) = \prod_t p(q_t | q_{t-1}) \ p(o_t | q_t) \]
CRF: \[ p(O, Q) = \frac{1}{Z} \exp\left\{ \sum_t a^\top \phi(q_t, q_{t-1}) + b^\top \phi(o_t \mid q_t) \right\} \]
Labeling sequences: HMMs & CRFs

\[ p(O, Q) = \frac{1}{Z} \exp\left\{ \sum_{t} a^T \phi(q_t, q_{t-1}) + b^T \phi(o_t \mid q_t) \right\} \]
Labeling sequences: neural networks

[CLS] cheap and delicious [SEP] would definitely buy again
Sequence-to-sequence models

ENCODER

DECODER

\text{in} \quad \text{horto} \quad [\text{SEP}] \quad \text{Caecilius} \quad \text{is} \quad \text{in}
Other structures
Syntax

I ate spaghetti with meatballs

I ate spaghetti with a fork
I ate spaghetti with meatballs

I ate spaghetti with a fork
she thinks that the food here is delicious

Useful to distinguish between statements and beliefs, even in simple NLP problems!
Semantics

```
forbid(send(find(type=email), find(keyword=dean))
```

don’t send the email to the dean
forbid(send(find(type=email), find(keyword=dean)))
Semantics

```
forbid(send(find(type=email), find(keyword=dean))
```

```
\lambda x. send(find(type=email), x)
```

```
send(find(type=email), find(keyword=dean))
```

```
don't send the email to the dean
```
Representation Learning for Text-level Discourse Parsing

Yangfeng Ji
School of Interactive Computing
Georgia Institute of Technology
jiyfeng@gatech.edu

Jacob Eisenstein
School of Interactive Computing
Georgia Institute of Technology
jacobe@gatech.edu

Abstract
Text-level discourse parsing is notoriously difficult, as distinctions between discourse relations require subtle semantic judgments that are not easily captured using standard features. In this paper, we present a representation learning approach, in which we transform surface features into a latent space that facilitates RST discourse parsing. By combining the machinery of large-margin transition-based structured prediction with representation learning, our method jointly learns to parse discourse while at the same time learning a discourse-driven projection of surface features. The resulting shift-reduce discourse parser obtains substantial improvements over the previous state-of-the-art in predicting relations and nuclearity on the RST Treebank.

1 Introduction
Discourse structure describes the high-level organization of text or speech. It is central to a number of high-impact applications, such as text summarization (Louis et al., 2010), sentiment analysis (Vóll and Taboada, 2007; Somasundaran et al., 2009), question answering (Ferrucci et al., 2010), and automatic evaluation of student writing (Miltsakaki and Kukich, 2004; Burstein et al., 2013). Hierarchical discourse representations such as Rhetorical Structure Theory (RST) are particularly useful because of the computational applicability of tree-shaped discourse structures (Taboada and Mann, 2006), as shown in Figure 1.

Unfortunately, the performance of discourse parsing is still relatively weak: the state-of-the-art F-measure for text-level relation detection in the RST Treebank is only slightly above 55% (Joty et al., 2013). While recent work has introduced increasingly powerful features (Feng and Hirst, 2012) and inference techniques (Joty et al., 2013), discourse relations remain hard to detect, due in part to a long tail of “alternative lexicalizations” that can be used to realize each relation (Prasad et al., 2010). Surface and syntactic features are not capable of capturing what are fundamentally semantic distinctions, particularly in the face of relatively small annotated training sets.

In this paper, we present a representation learning approach to discourse parsing. The core idea of our work is to learn a transformation from a bag-of-words surface representation into a latent space in which discourse relations are easily identifiable. The latent representation for each discourse unit can be viewed as a discriminatively-trained vector-space representation of its meaning. Alternatively, our approach can be seen as a nonlinear learning algorithm for incremental structure prediction, which overcomes feature sparsity through effective parameter tying. We consider several alternative methods for transforming the original features, corresponding to different ideas of the meaning and role of the latent representation.

Our method is implemented as a shift-reduce discourse parser (Marcu, 1999; Sagae, 2009). Learning is performed as large-margin transition-based structure prediction (Taskar et al., 2003), while at the same time jointly learning to project the surface representation into latent space. The

Figure 1: An example of RST discourse structure.
Why trees?

“Simplest” formal generative process that provides hierarchical relationships and long-distance dependencies:

My aunt gave me a microscope.

My aunt’s sister gave me a microscope.

My aunt’s sister, who works at the NIH, gave me a microscope.

My aunt’s sister, who works at a little-known constituent institute of the...
Syntax in ten minutes
Constituents

Key idea from previous examples: some sentence fragments “stick together”—can be moved around, replaced, and modified without affecting meaning / grammaticality:

I ate spaghetti with meatballs

I ate

I ate it

It was spaghetti with meatballs that I ate
Constituents

Some fragments are harder to manipulate:

I ate spaghetti with meatballs

I ate meatballs ✗ (meaning changed)

It was ate spaghetti with that I meatballs ✗ (not grammatical)
Constituents

Not just things:

I ate spaghetti with a fork

I ate spaghetti

It was with a fork that I ate spaghetti
Constituents & Types

[I [wrote [an email [to [the dean]]]]]
Constituents & Types

[ event ]

[ action ]

[ thing ]

[ relationship ]

[ thing ]

[l [wrote [an email [to [the dean]]]]]]

???
Lots of research on the exact form of this hierarchy. For most NLP applications: entities, events, relations.
Types & semantics

forbid(send(find(type=email), find(keyword=dean))
don’t send the email to the dean
Context free grammars

Just like in HMMs, we’d like to define some joint distribution over sentences and underlying structures, and reason about marginals and conditionals.

What’s the right distribution over trees and sentences?
Context free grammars

A *sentence* might consist of an entity and an action.  

\[ I \] \[ swallowed the spider \]
A sentence might consist of an entity and an action.

\[ I \text{[swallowed the spider]} \]

\[ S \rightarrow NP \ VP \quad \text{a Noun Phrase followed by a Verb Phrase make a Sentence} \]

A sentence might just consist of an action.

\[ \text{[eat the spider]} \]

\[ S \rightarrow VP \quad \text{a Verb Phrase makes a Sentence} \]
The image contains text about Context free grammars. The grammatical rules are as follows:

- **S** → **NP VP | VP**
  - "or"

- The followed by a noun makes an entity

- **NP** → **the N**
- **N** → **cat | dog | spider | cheesecake | democracy**

- A verb and an optional entity make an action

- **VP** → **V | V NP**
- **V** → **eat | eats | run | differentiate | ...**
A sample from our CFG

\[
S \rightarrow \text{NP VP} \mid \text{VP} \\
\text{NP} \rightarrow \text{the N} \\
\text{N} \rightarrow \text{cat} \mid \text{dog} \mid \text{spider} \mid \text{cheesecake} \mid \text{democracy} \\
\text{VP} \rightarrow \text{V} \mid \text{V NP} \\
\text{V} \rightarrow \text{eat} \mid \text{eats} \mid \text{run} \mid \text{differentiate} \mid \ldots
\]
A sample from our CFG

\[
\begin{align*}
S & \rightarrow \text{NP VP} \mid \text{VP} \\
\text{NP} & \rightarrow \text{the N} \\
\text{N} & \rightarrow \text{cat} \mid \text{dog} \mid \text{spider} \mid \text{cheesecake} \mid \text{democracy} \\
\text{VP} & \rightarrow \text{V} \mid \text{V NP} \\
\text{V} & \rightarrow \text{eat} \mid \text{eats} \mid \text{run} \mid \text{differentiate} \mid \ldots
\end{align*}
\]

<table>
<thead>
<tr>
<th>S</th>
<th>the cat VP</th>
<th>the cat eat the N</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP VP</td>
<td>the cat V NP</td>
<td>the cat eat the democracy</td>
</tr>
<tr>
<td>the N VP</td>
<td>the cat eat NP</td>
<td></td>
</tr>
</tbody>
</table>
A sample from our CFG

\[
\begin{align*}
S & \rightarrow \text{NP} \hspace{1mm} \text{VP} \mid \text{VP} \\
\text{NP} & \rightarrow \text{the} \hspace{1mm} \text{N} \\
\text{N} & \rightarrow \text{cat} \mid \text{dog} \mid \text{spider} \mid \text{cheesecake} \mid \text{democracy} \\
\text{VP} & \rightarrow \text{V} \mid \text{V} \hspace{1mm} \text{NP} \\
\text{V} & \rightarrow \text{eat} \mid \text{eats} \mid \text{run} \mid \text{differentiate} \mid \ldots
\end{align*}
\]

\[
\text{the \hspace{1mm} cat \hspace{1mm} eat \hspace{1mm} the \hspace{1mm} democracy}
\]
What about other languages?

and also for it generally hasn’t money

_and in most cases he has no money for it either_

[McDonald et al 2005]
What about other languages?

*a taky na to většinou nemá peníze*

and also for it generally hasn’t money

and in most cases he has no money for it either

can’t draw a constituency tree!
a taky na to většinou nemá peníze
Probabilistic grammars
CFGs as generative models

Noun
CFGs as generative models
CFGs as generative models

Noun

Modal

Verb

I

can

run
CFGs as generative models
CFGs as generative models

```
S → NP VP
  NP → the cat
  V → eat
  NP → the democracy
```
A probabilistic context free grammar (PCFG) consists of

1. A set of nonterminal symbols $N$
2. A set of terminal symbols $T$
3. A set of rules $R$
4. A set of rule probabilities $p(r \in R | n \in N)$
A rule consists of

1. A left hand symbol
2. A sequence of right-hand symbols

\[
\begin{align*}
\text{LHS} & \quad \rightarrow \quad \text{RHS} & \quad \text{prob} \\
S & \quad \rightarrow \quad \text{NP} \quad \text{VP} & \quad 0.75
\end{align*}
\]

such that \( \sum_{\text{rules with LHS symbol A}} p(\text{rule} \mid \text{LHS}) = 1 \)
Queries: joint probability

The cat eats

\[ p(T, S) \]

joint prob. of tree and sentence

\[ p(S \rightarrow \text{NP VP}) \cdot p(\text{NP} \rightarrow \text{D N}) \]

\[ p(\text{D} \rightarrow \text{the}) \cdot p(\text{N} \rightarrow \text{cat}) \ldots \]
Queries: best tree

\[ \text{argmax}_T \ p(T \mid S) \]
most probable tree given sent.

\[ p(S \rightarrow \text{NP VP}) \cdot p(\text{NP} \rightarrow \text{D N}) \]
\[ p(\text{D} \rightarrow \text{the}) \cdot p(\text{N} \rightarrow \text{cat}) \ldots \]
Queries: sentence marginal

\[ p(S) \]

prob. of sentence under any tree

(there are \(\frac{(2n)!}{(n+1)!n!}\) unlabeled binary trees over \(n\) words...)

The diagram shows a tree structure with nodes labeled 'NP', 'VP', 'S', 'D', 'N', and 'V', with words 'the', 'cat', and 'eats' branching from them.
Parsing
Chomsky normal form

Notational convenience: only binary trees.

Every rule has one of these forms:

Nonterminal $\rightarrow$ Terminal
Nonterminal $\rightarrow$ Nonterminal  Nonterminal

(Can always get rules into this form by introducing new NTs)
Warmup: the CKY algorithm

Is the string $S$ generated by CFG $G$?

$$S \rightarrow NP \ VP$$
$$NP \rightarrow D \ N$$
$$N \rightarrow \text{cat} \ | \ \text{eats}$$
$$VP \rightarrow \text{eats} \ | \ \text{sings}$$
$$D \rightarrow \text{the} \ | \ a$$

the  
cat  
eats
Warmup: the CKY algorithm

1. Fill in bottom row with NTs that can generate observed words

\[
\begin{align*}
S & \rightarrow \text{NP VP} \\
\text{NP} & \rightarrow \text{D N} \\
\text{N} & \rightarrow \text{cat | eats} \\
\text{VP} & \rightarrow \text{eats | sings} \\
\text{D} & \rightarrow \text{the | a}
\end{align*}
\]
Warmup: the CKY algorithm

2. Fill in second row with NTs that generate a symbol in each child

S → NP VP
NP → D N
N → cat | eats
VP → eats | sings
D → the | a
3. Fill in higher rows with NTs that generate a symbol *any pair of non-overlapping* children

\[
\begin{align*}
S & \rightarrow NP \ VP \\
NP & \rightarrow D \ N \\
N & \rightarrow cat \, | \, eats \\
VP & \rightarrow eats \, | \, sings \\
D & \rightarrow the \, | \, a
\end{align*}
\]
Warmup: the CKY algorithm

3. Fill in higher cells with NTs that generate symbols in any pair of non-overlapping children

The CKY algorithm is a dynamic programming algorithm for parsing sentences into their constituent parts. It constructs a tree-like structure to represent the dependencies between words in a sentence. The algorithm relies on the following rules:

- **S → NP VP**
- **NP → D N**
- **N → cat | eats**
- **VP → eats | sings**
- **D → the | a**

The diagram above illustrates the process of filling in the CKY tableau. Each cell in the tableau represents a non-terminal symbol (NT) that can be derived from the input sentence. The tableau is built by applying the rules of the grammar to the input sentence, starting from the root symbol S and working down to the terminal symbols D, N, and VP. The process involves comparing the symbols in the input sentence to the symbols generated by the NTs in the tableau, and filling in the tableau with the appropriate NT when a match is found.

The tableau is constructed by considering all possible pairs of non-overlapping children of each symbol. For example, in the sentence "the cat eats," the non-overlapping pairs of children for the symbol S are (NP, VP), where NP contains "the" and VP contains "cat eats." The NT NP is filled in the tableau as D N, and the NT VP is filled in as eats | sings.

The final tableau represents the parse tree of the sentence, showing how the sentence is composed of its constituent parts.
Warmup: the CKY algorithm

4. If the top cell contains the start symbol, the string is generated.

\[
\begin{array}{c}
\text{D} \\
\text{N} \\
\text{NP} \\
\text{S}
\end{array}
\]

\[
\begin{array}{c}
\text{the} \\
\text{cat} \\
\text{eats} \\
\text{NP, VP}
\end{array}
\]

\[
\begin{align*}
S & \rightarrow \text{NP VP} \\
\text{NP} & \rightarrow \text{D N} \\
\text{N} & \rightarrow \text{cat | eats} \\
\text{VP} & \rightarrow \text{eats | sings} \\
\text{D} & \rightarrow \text{the | a}
\end{align*}
\]
What parse assigns highest prob. to $S$ under the PCFG $G$?

$S \rightarrow$ NP VP | NP N

NP \rightarrow D N

N \rightarrow cat | eats

VP \rightarrow eats | sings

D \rightarrow the | a
1. Fill in bottom row with prob. that each NT generates word

- **D**: 0.5
- **N**: 0.8
- **VP**: 0.5
- **N**: 0.2

```
S → NP VP | NP N
   1
NP → D N
   0.8 0.2
N → cat | eats
   0.9 0.1
VP → eats | sings
   0.5 0.5
D → the | a
   0.5 0.5
```
2. Fill in higher rows with highest-scoring product of child probs. times rule prob.

\[ 0.5 \times 0.8 \times 1 = 0.4 \]

**Rules:**
- **S** → **NP VP** | **NP N**
- **NP** → **D N**
- **N** → **cat** | **eats**
- **VP** → **eats** | **sings**
- **D** → **the** | **a**
2. Fill in higher rows with highest-scoring product of child probs. times rule prob.
3. The score for $S$ in the top cell is the score of the best parse.
Q: what is the most probable tree for a given sentence?

\[ \delta(s, i, j) \text{ highest-scoring tree with root } s \text{ covering words } i:j \]

**base case:**

\[ \delta(s, i, i + 1) = p(s \rightarrow w_i) \]

**inductive case:**

\[ \delta(s, i, j) = \max_{k \in [i+1, j-1]} \max_{s',s''} p(s \rightarrow s's'') \delta(s', i, k) \delta(s'', k, j) \]
Q: what is the most probable assignment of tags to observations?

\[ \arg\max_Q p(Q \mid O) \]

\[ \delta(t,j) = \max_i \delta(t-1,i) a_{ij} b_j(o_t) \]

\[ \delta(1,j) = \pi(j) b_j(o_1) \]
The inside algorithm for CFGs

Q: what is the marginal probability of a sentence given a tree?

$$\sum_T p(T, S)$$

$$\beta(s, i, j)$$ probability of all parses with root s covering words i:j

base case:

$$\beta(s, i, i + 1) = p(s \rightarrow w_i)$$

inductive case:

$$\beta(s, i, j) = \sum_{k \in [i+1, j-1]} \sum_{s', s''} p(s \rightarrow s's'') \beta(s', i, k) \beta(s'', k, j)$$
Tree-structured CRFs

Instead of scores $p(A \rightarrow B \ C)$, use an arbitrary scorer

$$w^\top \phi(A, B, C, i, j)$$

(can be different in each cell & look at full sentence sentence)

Works just like the HMM version!

$\beta(S, 0, |S|)$ is the partition function
Learning
Supervised learning

For PCFGs—given a treebank, estimate by counting:

\[ p(S \rightarrow NP \ VP) = \frac{\#(S \rightarrow NP \ VP)}{\#(S)} \]
Supervised learning

For PCFGs—given a treebank, estimate by counting:

\[ p(S \rightarrow NP \; VP) = \frac{\#(S \rightarrow NP \; VP)}{\#(S)} \]

This doesn’t work very well: basic syntactic categories are too coarse.
Supervised learning: lexicalization

Idea: enrich nonterminal alphabet with information about the most important **word** underneath:

I ate **spaghetti** with **meatballs**

[e.g. Collins 97]
Supervised learning: Markovization

Idea: enrich nonterminal alphabet with more information about the local tree structure:

I ate *spaghetti* with *meatballs*

[e.g. Klein 03]
Supervised learning: features & NNs

Idea: Use the CRF version

\[ P(T) \propto \exp \left\{ \sum_{A \rightarrow B \, C, i, k, j} w^\top \phi(A, B, C, i, k, j) \right\} \]

and give \( \phi \) features like “\( A = \text{NP} \) and \( j:k \) contains fork”
(or make it a neural network)
## Supervised parsing: what’s still hard?

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Occurrences</th>
<th>Nodes Involved</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>PP Attachment</td>
<td>846</td>
<td>1455</td>
<td>1.7</td>
</tr>
<tr>
<td>Single Word Phrase</td>
<td>490</td>
<td>490</td>
<td>1.0</td>
</tr>
<tr>
<td>Clause Attachment</td>
<td>385</td>
<td>913</td>
<td>2.4</td>
</tr>
<tr>
<td>Adverb and Adjective Modifier Attachment</td>
<td>383</td>
<td>599</td>
<td>1.6</td>
</tr>
<tr>
<td>Different Label</td>
<td>377</td>
<td>754</td>
<td>2.0</td>
</tr>
<tr>
<td>Unary</td>
<td>347</td>
<td>349</td>
<td>1.0</td>
</tr>
<tr>
<td>NP Attachment</td>
<td>321</td>
<td>597</td>
<td>1.9</td>
</tr>
<tr>
<td>NP Internal Structure</td>
<td>299</td>
<td>352</td>
<td>1.2</td>
</tr>
<tr>
<td>Coordination</td>
<td>209</td>
<td>557</td>
<td>2.7</td>
</tr>
<tr>
<td>Unary Clause Label</td>
<td>185</td>
<td>200</td>
<td>1.1</td>
</tr>
<tr>
<td>VP Attachment</td>
<td>64</td>
<td>159</td>
<td>2.5</td>
</tr>
<tr>
<td>Parenthetical Attachment</td>
<td>31</td>
<td>74</td>
<td>2.4</td>
</tr>
<tr>
<td>Missing Parenthetical</td>
<td>12</td>
<td>17</td>
<td>1.4</td>
</tr>
<tr>
<td>Unclassified</td>
<td>655</td>
<td>734</td>
<td>1.1</td>
</tr>
</tbody>
</table>

*Spaghetti with a fork*

*[[world oil] prices]*

[Kummerfeld, 2016]
Unsupervised learning

<table>
<thead>
<tr>
<th>Model</th>
<th>$F_1$</th>
<th>Training/Test PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Trees</td>
<td>19.5</td>
<td>—</td>
</tr>
<tr>
<td>Right Branching</td>
<td>39.5</td>
<td>—</td>
</tr>
<tr>
<td>Scalar PCFG (unsupervised)</td>
<td>&lt; 35.0</td>
<td>&gt; 350</td>
</tr>
</tbody>
</table>

worse than assuming every tree looks like this: 😳

[Kim et al. 2018]
Unsupervised learning: embeddings

<table>
<thead>
<tr>
<th>Model</th>
<th>$F_1$</th>
<th>Training/Test PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Trees</td>
<td>19.5</td>
<td>–</td>
</tr>
<tr>
<td>Right Branching</td>
<td>39.5</td>
<td>–</td>
</tr>
<tr>
<td>Scalar PCFG</td>
<td>&lt; 35.0</td>
<td>&gt; 350</td>
</tr>
<tr>
<td>Neural PCFG</td>
<td>52.6</td>
<td>≈ 250</td>
</tr>
</tbody>
</table>

“Grammar embeddings": $p(A \rightarrow B \ C) \propto \exp\{v_A^\top f(v_B, v_C)\}$

[Kim et al. 2018]
What about neural nets?

**best:**

$$\exp\left\{ \sum_{(A \rightarrow B \ C, i, k, j)} w^T \phi(A, B, C, v_i, v_k, v_j) \right\} / Z$$

**almost as good:**

train an independent cell classifier that takes $v_{i,j,k}$ as input
Next class: advanced language modeling