Dependency parses for NLU

Christopher Potts

CS 244U: Natural language understanding
April 21
Syntactic structure: My dog will not go in the lake.

Treebank-style parsetree:
(Root
 (S
   (NP (PRP$ My) (NN dog))
   (VP (MD will) (RB not)
     (VP (VB go)
       (PP (IN in)
         (NP (DT the) (NN lake))))))

Dependencies:
poss(dog-2, My-1)
nsubj(go-5, dog-2)
aux(go-5, will-3)
neg(go-5, not-4)
root(ROOT-0, go-5)
prep(go-5, in-6)
det(lake-8, the-7)
pobj(in-6, lake-8)

Collapsed dependencies:
poss(dog-2, My-1)
nsubj(go-5, dog-2)
aux(go-5, will-3)
neg(go-5, not-4)
root(ROOT-0, go-5)
det(lake-8, the-7)
pobj_in(go-5, lake-8)
Simplified relationships, easier feature extraction

```
S
  NP
    NNP Gerald
  VP
    VBD gave
    NP
      NNS awards
    PP
      TO to
      NP
        NNS puppies

S
  NP
    NNP Gerald
  VP
    VBD gave
    NP
      NNS puppies
    NP
      NNS awards
```
Plan and goals

Goals

- Make the case for Stanford dependency structures (de Marneffe et al. 2006; de Marneffe and Manning 2008a,b; de Marneffe et al. 2013)
- Highlight some of the ways that semantic information is passed around inside sentences.
- Engage with other topics: VSMs, classifiers, and semantic parsing.
Plan and goals

Goals

- Make the case for Stanford dependency structures (de Marneffe et al. 2006; de Marneffe and Manning 2008a,b; de Marneffe et al. 2013)
- Highlight some of the ways that semantic information is passed around inside sentences.
- Engage with other topics: VSMs, classifiers, and semantic parsing.

Not covered here

The theory of parsing, the theory of semantic dependencies, or the details of mapping from phrase structure trees to dependencies. In short, we’re going to be consumers of dependencies, seeking to use them to get ahead in NLU.
Plan and goals

Goals

- Make the case for Stanford dependency structures (de Marneffe et al. 2006; de Marneffe and Manning 2008a,b; de Marneffe et al. 2013)
- Highlight some of the ways that semantic information is passed around inside sentences.
- Engage with other topics: VSMs, classifiers, and semantic parsing.

Not covered here

The theory of parsing, the theory of semantic dependencies, or the details of mapping from phrase structure trees to dependencies. In short, we’re going to be *consumers* of dependencies, seeking to use them to get ahead in NLU.

Plan

1. Get a feel for Stanford dependencies
2. Case study: advmod-based VSMs
3. Case study: dependencies as classifier features
4. Case study: capturing the semantic influence of negation
Dependency structures in NLU

Dependencies as the basis for features:

- Word-sense disambiguation (Lin 1998) [last year’s slides on WSD]
- Relation extraction (Snow et al. 2005; Mintz et al. 2009)
- Semantic role labeling (Surdeanu et al. 2008; Johansson and Nugues 2008)
- Semantic parsing (Liang et al. 2013)
- Detecting speaker commitment (hedging, etc.; de Marneffe et al. 2012)
- Forecasting public opinion (Lerman et al. 2008)
- Analysis of political debates (Balahur et al. 2009)
- Drug interactions (Percha et al. 2012)
- …
Updates from de Marneffe et al. 2013:
- New relations are boxed.
- Changed/deleted relations are in red, with notes
Stanford dependencies relation hierarchy
Stanford dependencies relation hierarchy
Stanford dependencies relation hierarchy
Stanford dependency construction

Ruled-based mapping from phrase structure trees to dependency graphs:

1. **Dependency extraction**: for each constituent, identify its *semantic* head and project the head upwards:

   ![Phrase Structure Tree]

   - **MD**: might
   - **VP**: have
   - **VBN**: escaped
Stanford dependency construction

Ruled-based mapping from phrase structure trees to dependency graphs:

1. **Dependency extraction**: for each constituent, identify its *semantic* head and project the head upwards:

   ![Dependency tree](image)

   - VP
     - MD
       - **might**
     - VP
       - VB
         - **have**
       - VP
         - VBN
           - escaped
Stanford dependency construction

Ruled-based mapping from phrase structure trees to dependency graphs:

1. Dependency extraction: for each constituent, identify its *semantic* head and project the head upwards:

```
VP[escaped]
  MD[might]
     might
  VP[escaped]
     VB[have]
      have
     VP[escaped]
      VBN[escaped]
      escaped
```

Relations determined:

- `aux(escaped, might)`
- `aux(escaped, have)`
- `dep(escaped, might)`

Rules might also deliver `dep(escaped, have)` always favor the most specific.
Stanford dependency construction

Ruled-based mapping from phrase structure trees to dependency graphs:

1. **Dependency extraction**: for each constituent, identify its *semantic* head and project the head upwards:

   - **MD[might]**
   - **VP[escaped]**
     - **VB[have]**
     - **VP[escaped]**
       - **VBN[escaped]**

   Relations determined:
   - aux(escaped, might)
   - aux(escaped, have)

2. **Dependency typing**: label each dependency pair with the most specific appropriate relation in terms of the dependency hierarchy.

   - relation: aux
   - parent: VP

   - Tregex pattern:
     - `VP < VP < (/^(?::TO|MD|VB.*|AUXG?)$/=target)`

   Relations determined:
   - dep(escaped, might)
   - dep(escaped, have)

Rules might also deliver

- dep(escaped, might)

Always favor the most specific.
Stanford dependencies: basic and collapsed

Quoting from the javadocs, trees/EnglishGrammaticalRelations.java:

The “collapsed” grammatical relations primarily differ as follows:

- Some multiword conjunctions and prepositions are treated as single words, and then processed as below.
- Prepositions do not appear as words but are turned into new “prep” or “prepc” grammatical relations, one for each preposition.
- Conjunctions do not appear as words but are turned into new “conj” grammatical relations, one for each conjunction.
- The possessive “’s” is deleted, leaving just the relation between the possessor and possessum.
- Agents of passive sentences are recognized and marked as agent and not as prep_by.
Stanford tools

The Stanford parser is distributed with starter Java code for parsing your own data. It also has a flexible command-line interface. Some relevant commands:

# Map plain text to dependency structures:

code:
```java
java -mx3000m -cp stanford-parser.jar edu.stanford.nlp.parser.lexparser.LexicalizedParser
-outputFormat "typedDependencies" englishPCFG.ser.gz textFile
```

# Map tagged data to dependency structures:

code:
```java
java -mx3000m -cp stanford-parser.jar edu.stanford.nlp.parser.lexparser.LexicalizedParser
-outputFormat "typedDependencies" -tokenized -tagSeparator / englishPCFG.ser.gz taggedFile
```

# Map phrase-structure trees to Stanford collapsed dependencies
(change -collapsed to -basic for collapsed versions):

code:
```java
java -cp stanford-parser.jar edu.stanford.nlp.trees.EnglishGrammaticalStructure
-treeFile treeFile -collapsed
```

Software/docs: http://nlp.stanford.edu/software/lex-parser.shtml
Graphviz

Graphviz is free graphing software that makes it easy to visualize dependency structures: http://www.graphviz.org/

digraph g {
  /* Nodes */
  "Al-1" [label="Al"];  
  "said-2" [label="said"];  
  "that-3" [label="that"];  
  "it-4" [label="it"];  
  "was-5" [label="was"];  
  "raining-6" [label="raining"];  
  /* Edges */
  "said-2" -> "Al-1" [label="nsubj"];  
  "raining-6" -> "that-3" [label="complm"];  
  "raining-6" -> "it-4" [label="nsubj"];  
  "raining-6" -> "was-5" [label="aux"];  
  "said-2" -> "raining-6" [label="ccomp"];  
}
Argument structure

- This section reviews the way basic constituents are represented in Stanford dependency structures.
- I concentrate on the most heavily used relations.
- To understand the less-used ones, consult the dependencies manual (de Marneffe and Manning 2008a) and play around with examples using the online parser demo:

  http://nlp.stanford.edu:8080/parser/index.jsp
Verbal structures
Verbal structures: intransitive and transitive

**Intransitive**

- **Al escaped.**
  - `escaped`
  - `nsbj`
  - `Al`

- **Al might escape.**
  - `escape`
  - `nsbj`
  - `aux`
  - `Al`
  - `might`

- **Al might have escaped.**
  - `escaped`
  - `nsbj`
  - `aux`
  - `aux`
  - `Al`
  - `might`
  - `have`

- **Al might have been escaping.**
  - `escaping`
  - `nsbj`
  - `aux`
  - `aux`
  - `aux`
  - `Al`
  - `might`
  - `have`
  - `been`

**Transitive**

- **Sue saw stars.**
  - `saw`
  - `nsbj`
  - `dobj`
  - `Sue`
  - `stars`

- **Gerald gave puppies awards.**
  - `gave`
  - `nsbj`
  - `dobj`
  - `Gerald`
  - `puppies`
  - `awards`

- **Gerald gave awards to puppies basic.**
  - `gave`
  - `nsbj`
  - `dobj`
  - `prep`
  - `Gerald`
  - `awards`
  - `to`
  - `pobj`
  - `puppies`

- **Gerald gave awards to puppies collapsed.**
  - `gave`
  - `nsbj`
  - `dobj`
  - `prep_to`
  - `Gerald`
  - `awards`
  - `puppies`
Verbal structures: sentential complements

**Tensed**

Al said that it was raining.

```plaintext
said
   /\  
  /   \ 
 nsubj ccomp
Al   raining
    /\    
   /  
complm nsubj aux
that  it  was
```

**Infinitival**

Kim wants to win.

**Basic**

```plaintext
wants
   /\  
  /   \ 
 nsubj xcomp
Kim  win
    /\    
   /  
 aux
  to
```

**Collapsed**

```plaintext
wants
   /\  
  /   \ 
 nsubj xcomp
Kim  win
 xsubj aux
   /\  
  /   \ 
 to
```
Nominal structures

### Basic

**Proper name**
- Sam

**Quantifier**
- Everyone

**Determiner**
- student
- the

**Possessive**
- bike
- Sam

### Modified

**Adjective**
- student
- happy

**Prepositional**
- the
- happy
- of

**Relative clause**
- the
- linguistics
- won
- who
Modification

Predicative constructions

Basic

Lexical pred

Lexical

Small clause

Adverbs

wonderfully happy

surprisingly amazingly happy

not surprisingly happy

in no way happy
Coordination: conj and cc

Nominals (here, nsubj)

Ivan and Penny left.

Basic

Collapsed

Verb phrases

Nobody sang and danced.

Basic

Collapsed
advmod dependencies

totally open  o------------------ o  tall, short
lower closed  o------------------ o  wet, bent
upper closed  o------------------● o  pure, straight
totally closed●------------------● o  opaque, open

Adverbs for distinguishing scales

- Maximality: completely, fully, totally, absolutely, 100%, perfectly, ...
- Proportion: half, mostly, most of the way, two-thirds, three-sevenths, ...
- Minimality: slightly, somewhat, partially, ...

<table>
<thead>
<tr>
<th>Adverb</th>
<th>Totally open</th>
<th>Totally closed</th>
<th>Upper closed</th>
<th>Lower closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximality</td>
<td>*</td>
<td>✓</td>
<td>✓</td>
<td>*</td>
</tr>
<tr>
<td>Proportion</td>
<td>*</td>
<td>✓</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Minimality</td>
<td>*</td>
<td>✓</td>
<td>*</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table: Summary of adverb patterns.

(Kennedy and McNally 2005; Kennedy 2007; Syrett and Lidz 2010)
Gigaword NYT (h/t to Nate Chambers for the parsing!)

Available in list format (tab-separated values):

http://www.stanford.edu/class/cs224u/restricted/data/gigawordnyt-advmod.tsv.zip
Or: /afs/ir/class/cs224u/WWW/restricted/data/gigawordnyt-advmod.tsv.zip

Pairs advmod(X, Y) with counts:

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>Y</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>end</td>
<td>here</td>
<td>98434</td>
</tr>
<tr>
<td>2</td>
<td>well</td>
<td>as</td>
<td>84031</td>
</tr>
<tr>
<td>3</td>
<td>longer</td>
<td>no</td>
<td>74486</td>
</tr>
<tr>
<td>4</td>
<td>far</td>
<td>so</td>
<td>71853</td>
</tr>
<tr>
<td>5</td>
<td>much</td>
<td>so</td>
<td>71460</td>
</tr>
<tr>
<td>6</td>
<td>now</td>
<td>right</td>
<td>66373</td>
</tr>
<tr>
<td>7</td>
<td>much</td>
<td>too</td>
<td>66264</td>
</tr>
<tr>
<td>8</td>
<td>much</td>
<td>how</td>
<td>64794</td>
</tr>
<tr>
<td>9</td>
<td>said</td>
<td>also</td>
<td>62588</td>
</tr>
<tr>
<td>10</td>
<td>year</td>
<td>earlier</td>
<td>60290</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>3211133</td>
<td>scuff</td>
<td>how</td>
<td>1</td>
</tr>
</tbody>
</table>
## Gigaword NYT (h/t to Nate Chambers for the parsing!)

### Dependent × parent matrix: raw counts

<table>
<thead>
<tr>
<th></th>
<th>when</th>
<th>also</th>
<th>just</th>
<th>now</th>
<th>more</th>
<th>so</th>
<th>even</th>
<th>how</th>
<th>where</th>
<th>as</th>
</tr>
</thead>
<tbody>
<tr>
<td>is</td>
<td>17663</td>
<td>21310</td>
<td>10853</td>
<td>46433</td>
<td>2094</td>
<td>8204</td>
<td>8388</td>
<td>14546</td>
<td>22985</td>
<td>2039</td>
</tr>
<tr>
<td>have</td>
<td>20657</td>
<td>20156</td>
<td>18757</td>
<td>31288</td>
<td>2162</td>
<td>7508</td>
<td>13003</td>
<td>4184</td>
<td>12573</td>
<td>1572</td>
</tr>
<tr>
<td>was</td>
<td>26976</td>
<td>10634</td>
<td>8253</td>
<td>3014</td>
<td>1265</td>
<td>4025</td>
<td>5644</td>
<td>6554</td>
<td>11818</td>
<td>1920</td>
</tr>
<tr>
<td>said</td>
<td>19695</td>
<td>62588</td>
<td>3984</td>
<td>4953</td>
<td>923</td>
<td>4933</td>
<td>6198</td>
<td>575</td>
<td>4209</td>
<td>608</td>
</tr>
<tr>
<td>much</td>
<td>207</td>
<td>145</td>
<td>4184</td>
<td>474</td>
<td>10079</td>
<td>71460</td>
<td>421</td>
<td>64794</td>
<td>140</td>
<td>46174</td>
</tr>
<tr>
<td>are</td>
<td>11546</td>
<td>14212</td>
<td>4929</td>
<td>23470</td>
<td>2418</td>
<td>7591</td>
<td>4779</td>
<td>7952</td>
<td>19832</td>
<td>1214</td>
</tr>
<tr>
<td>get</td>
<td>19342</td>
<td>4004</td>
<td>8474</td>
<td>5811</td>
<td>1401</td>
<td>2657</td>
<td>5930</td>
<td>14477</td>
<td>6840</td>
<td>718</td>
</tr>
<tr>
<td>do</td>
<td>8299</td>
<td>1550</td>
<td>7908</td>
<td>9899</td>
<td>2733</td>
<td>37339</td>
<td>2915</td>
<td>14474</td>
<td>2376</td>
<td>598</td>
</tr>
<tr>
<td>'s</td>
<td>7811</td>
<td>9488</td>
<td>8815</td>
<td>13779</td>
<td>1371</td>
<td>3949</td>
<td>4293</td>
<td>1690</td>
<td>6281</td>
<td>1500</td>
</tr>
<tr>
<td>had</td>
<td>16854</td>
<td>16247</td>
<td>7039</td>
<td>3128</td>
<td>1512</td>
<td>1703</td>
<td>7930</td>
<td>1735</td>
<td>6936</td>
<td>1742</td>
</tr>
</tbody>
</table>

### Dependent × parent matrix: positive PMI with contextual discounting

<table>
<thead>
<tr>
<th></th>
<th>when</th>
<th>also</th>
<th>just</th>
<th>now</th>
<th>more</th>
<th>so</th>
<th>even</th>
<th>how</th>
<th>where</th>
<th>as</th>
</tr>
</thead>
<tbody>
<tr>
<td>is</td>
<td>0.00</td>
<td>0.04</td>
<td>0.00</td>
<td>1.12</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.16</td>
<td>0.65</td>
<td>0.00</td>
</tr>
<tr>
<td>have</td>
<td>0.00</td>
<td>0.30</td>
<td>0.48</td>
<td>1.05</td>
<td>0.00</td>
<td>0.00</td>
<td>0.38</td>
<td>0.00</td>
<td>0.36</td>
<td>0.00</td>
</tr>
<tr>
<td>was</td>
<td>0.23</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.40</td>
<td>0.00</td>
</tr>
<tr>
<td>said</td>
<td>0.00</td>
<td>1.56</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>much</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.11</td>
<td>2.01</td>
<td>0.00</td>
<td>2.09</td>
<td>0.00</td>
<td>1.80</td>
<td>0.00</td>
</tr>
<tr>
<td>are</td>
<td>0.00</td>
<td>0.17</td>
<td>0.00</td>
<td>0.98</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.09</td>
<td>1.04</td>
<td>0.00</td>
</tr>
<tr>
<td>get</td>
<td>0.32</td>
<td>0.00</td>
<td>0.21</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.12</td>
<td>1.00</td>
<td>0.28</td>
<td>0.00</td>
</tr>
<tr>
<td>do</td>
<td>0.00</td>
<td>0.00</td>
<td>0.14</td>
<td>0.42</td>
<td>0.00</td>
<td>1.77</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>'s</td>
<td>0.00</td>
<td>0.07</td>
<td>0.25</td>
<td>0.75</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.20</td>
<td>0.00</td>
</tr>
<tr>
<td>had</td>
<td>0.22</td>
<td>0.65</td>
<td>0.06</td>
<td>0.00</td>
<td>0.00</td>
<td>0.45</td>
<td>0.00</td>
<td>0.34</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
### Adverbs

<table>
<thead>
<tr>
<th>absolutely</th>
<th>certainly</th>
<th>never</th>
<th>recently</th>
<th>somewhat</th>
<th>quickly</th>
</tr>
</thead>
<tbody>
<tr>
<td>utterly</td>
<td>definitely</td>
<td>not</td>
<td>subsequently</td>
<td>slightly</td>
<td>swiftly</td>
</tr>
<tr>
<td>totally</td>
<td>surely</td>
<td>maybe</td>
<td>ago</td>
<td>considerably</td>
<td>soon</td>
</tr>
<tr>
<td>truly</td>
<td>probably</td>
<td>either</td>
<td>since</td>
<td>decidedly</td>
<td>gradually</td>
</tr>
<tr>
<td>completely</td>
<td>obviously</td>
<td>ever</td>
<td>later</td>
<td>extremely</td>
<td>rapidly</td>
</tr>
<tr>
<td>equally</td>
<td>undoubtedly</td>
<td>yes</td>
<td>shortly</td>
<td>terribly</td>
<td>slowly</td>
</tr>
<tr>
<td>quite</td>
<td>necessarily</td>
<td>why</td>
<td>previously</td>
<td>very</td>
<td>eventually</td>
</tr>
<tr>
<td>obviously</td>
<td>indeed</td>
<td>would</td>
<td>first</td>
<td>markedly</td>
<td>immediately</td>
</tr>
<tr>
<td>really</td>
<td>clearly</td>
<td>simply</td>
<td>when</td>
<td>equally</td>
<td>promptly</td>
</tr>
<tr>
<td>whatsoever</td>
<td>therefore</td>
<td>pray</td>
<td>already</td>
<td>more</td>
<td>fast</td>
</tr>
</tbody>
</table>

### Adjectives

<table>
<thead>
<tr>
<th>happy</th>
<th>sad</th>
<th>tall</th>
<th>full</th>
<th>straight</th>
<th>closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>excited</td>
<td>painful</td>
<td>large</td>
<td>empty</td>
<td>largest</td>
<td>closing</td>
</tr>
<tr>
<td>pleased</td>
<td>frustrating</td>
<td>wide</td>
<td>tight</td>
<td>straightforward</td>
<td>shut</td>
</tr>
<tr>
<td>nice</td>
<td>tragic</td>
<td>steep</td>
<td>complete</td>
<td>twice</td>
<td>sealed</td>
</tr>
<tr>
<td>comfortable</td>
<td>depressing</td>
<td>strong</td>
<td>crowded</td>
<td>best</td>
<td>halted</td>
</tr>
<tr>
<td>silly</td>
<td>ugly</td>
<td>thin</td>
<td>over</td>
<td>certain</td>
<td>corp.</td>
</tr>
<tr>
<td>proud</td>
<td>embarrassing</td>
<td>lucky</td>
<td>solid</td>
<td>steady</td>
<td>suspended</td>
</tr>
<tr>
<td>good</td>
<td>beautiful</td>
<td>quick</td>
<td>smooth</td>
<td>ordinary</td>
<td>retired</td>
</tr>
<tr>
<td>nervous</td>
<td>dumb</td>
<td>good</td>
<td>dark</td>
<td>decent</td>
<td>canceled</td>
</tr>
<tr>
<td>uncomfortable</td>
<td>unfortunate</td>
<td>high</td>
<td>filled</td>
<td>smooth</td>
<td>ending</td>
</tr>
</tbody>
</table>
Latent Semantic Analysis

1. Apply singular value decomposition to the PPMI+discounting matrix.
2. Inspect singular values; settle on 25 dimensions:

3. For rows (dependents): $R[1:25, \ldots] \times S[1:25, 1:25]$
Latent Semantic Analysis

1. Apply singular value decomposition to the PPMI+discounting matrix.
2. Inspect singular values; settle on 25 dimensions:

Some adverb neighbors (cosine distance, PPMI + discounting + LSA)

### Adverbs without LSA (repeated from earlier)

<table>
<thead>
<tr>
<th>absolutely</th>
<th>certainly</th>
<th>never</th>
<th>recently</th>
<th>somewhat</th>
<th>quickly</th>
</tr>
</thead>
<tbody>
<tr>
<td>utterly</td>
<td>definitely</td>
<td>not</td>
<td>subsequently</td>
<td>slightly</td>
<td>swiftly</td>
</tr>
<tr>
<td>totally</td>
<td>surely</td>
<td>maybe</td>
<td>ago</td>
<td>considerably</td>
<td>soon</td>
</tr>
<tr>
<td>truly</td>
<td>probably</td>
<td>either</td>
<td>since</td>
<td>decidedly</td>
<td>gradually</td>
</tr>
<tr>
<td>completely</td>
<td>obviously</td>
<td>ever</td>
<td>later</td>
<td>extremely</td>
<td>rapidly</td>
</tr>
<tr>
<td>equally</td>
<td>undoubtedly</td>
<td>yes</td>
<td>shortly</td>
<td>terribly</td>
<td>slowly</td>
</tr>
<tr>
<td>quite</td>
<td>necessarily</td>
<td>why</td>
<td>previously</td>
<td>very</td>
<td>eventually</td>
</tr>
<tr>
<td>obviously</td>
<td>indeed</td>
<td>would</td>
<td>first</td>
<td>markedly</td>
<td>immediately</td>
</tr>
<tr>
<td>really</td>
<td>clearly</td>
<td>simply</td>
<td>when</td>
<td>equally</td>
<td>promptly</td>
</tr>
<tr>
<td>whatsoever</td>
<td>therefore</td>
<td>pray</td>
<td>already</td>
<td>more</td>
<td>fast</td>
</tr>
</tbody>
</table>

### Adverbs with LSA (25 dimensions)

<table>
<thead>
<tr>
<th>absolutely</th>
<th>certainly</th>
<th>never</th>
<th>recently</th>
<th>somewhat</th>
<th>quickly</th>
</tr>
</thead>
<tbody>
<tr>
<td>utterly</td>
<td>surely</td>
<td>you</td>
<td>subsequently</td>
<td>palpably</td>
<td>swiftly</td>
</tr>
<tr>
<td>truly</td>
<td>definitely</td>
<td>maybe</td>
<td>later</td>
<td>decidedly</td>
<td>soon</td>
</tr>
<tr>
<td>totally</td>
<td>probably</td>
<td>just</td>
<td>d.calif</td>
<td>seeming</td>
<td>prematurely</td>
</tr>
<tr>
<td>manifestly</td>
<td>doubt</td>
<td>yes</td>
<td>ago</td>
<td>any</td>
<td>instantly</td>
</tr>
<tr>
<td>wholly</td>
<td>undoubtedly</td>
<td>ok</td>
<td>r.ohio</td>
<td>slightly</td>
<td>immediately</td>
</tr>
<tr>
<td>patently</td>
<td>necessarily</td>
<td>q</td>
<td>shortly</td>
<td>congenitally</td>
<td>speedily</td>
</tr>
<tr>
<td>hardly</td>
<td>importantly</td>
<td>pray</td>
<td>first</td>
<td>distinctly</td>
<td>eventually</td>
</tr>
<tr>
<td>indisputably</td>
<td>doubtless</td>
<td>hey</td>
<td>d.mo</td>
<td>visibly</td>
<td>gradually</td>
</tr>
<tr>
<td>flat.out</td>
<td>secondly</td>
<td>anyway</td>
<td>since</td>
<td>sufficiently</td>
<td>slowly</td>
</tr>
</tbody>
</table>
Some adjective neighbors (cosine distance, PPMI + discounting + LSA)

### Adjectives without LSA (repeated from earlier)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>happy</td>
<td>sad</td>
<td>tall</td>
<td>full</td>
<td>straight</td>
<td>closed</td>
</tr>
<tr>
<td>excited</td>
<td>painful</td>
<td>large</td>
<td>empty</td>
<td>largest</td>
<td>closing</td>
</tr>
<tr>
<td>pleased</td>
<td>frustrating</td>
<td>wide</td>
<td>tight</td>
<td>straightforward</td>
<td>shut</td>
</tr>
<tr>
<td>nice</td>
<td>tragic</td>
<td>steep</td>
<td>complete</td>
<td>twice</td>
<td>sealed</td>
</tr>
<tr>
<td>comfortable</td>
<td>depressing</td>
<td>strong</td>
<td>crowded</td>
<td>best</td>
<td>halted</td>
</tr>
<tr>
<td>silly</td>
<td>ugly</td>
<td>thin</td>
<td>over</td>
<td>certain</td>
<td>corp.</td>
</tr>
<tr>
<td>proud</td>
<td>embarrassing</td>
<td>lucky</td>
<td>solid</td>
<td>steady</td>
<td>suspended</td>
</tr>
<tr>
<td>good</td>
<td>beautiful</td>
<td>quick</td>
<td>smooth</td>
<td>ordinary</td>
<td>retired</td>
</tr>
<tr>
<td>nervous</td>
<td>dumb</td>
<td>good</td>
<td>dark</td>
<td>decent</td>
<td>canceled</td>
</tr>
<tr>
<td>uncomfortable</td>
<td>unfortunate</td>
<td>high</td>
<td>filled</td>
<td>smooth</td>
<td>ending</td>
</tr>
</tbody>
</table>

### Adjectives with LSA (25 dimensions)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>happy</td>
<td>sad</td>
<td>tall</td>
<td>full</td>
<td>straight</td>
<td>closed</td>
</tr>
<tr>
<td>nice</td>
<td>ugly</td>
<td>thick</td>
<td>light</td>
<td>normal</td>
<td>suspended</td>
</tr>
<tr>
<td>terrible</td>
<td>scary</td>
<td>deep</td>
<td>flat</td>
<td>free</td>
<td>shut</td>
</tr>
<tr>
<td>strange</td>
<td>weird</td>
<td>loud</td>
<td>calm</td>
<td>flat</td>
<td>retired</td>
</tr>
<tr>
<td>cute</td>
<td>strange</td>
<td>bright</td>
<td>dry</td>
<td>natural</td>
<td>halted</td>
</tr>
<tr>
<td>scary</td>
<td>tragic</td>
<td>cheap</td>
<td>smooth</td>
<td>certain</td>
<td>replaced</td>
</tr>
<tr>
<td>wild</td>
<td>nasty</td>
<td>tight</td>
<td>quiet</td>
<td>conventional</td>
<td>stopped</td>
</tr>
<tr>
<td>excited</td>
<td>dumb</td>
<td>fast</td>
<td>cool</td>
<td>routine</td>
<td>cleared</td>
</tr>
<tr>
<td>cool</td>
<td>boring</td>
<td>hot</td>
<td>soft</td>
<td>benign</td>
<td>locked</td>
</tr>
<tr>
<td>special</td>
<td>odd</td>
<td>quick</td>
<td>steady</td>
<td>reasonable</td>
<td>sealed</td>
</tr>
</tbody>
</table>
t-SNE (van der Maaten and Geoffrey 2008) 2d embedding of the PPMI+discounting matrix: adverbs
t-SNE (van der Maaten and Geoffrey 2008) 2d embedding of the PPMI+discounting matrix: adverbs
t-SNE (van der Maaten and Geoffrey 2008) 2d embedding of the PPMI+discounting matrix: adverbs
t-SNE (van der Maaten and Geoffrey 2008) 2d embedding of the PPMI+discounting matrix: adverbs
t-SNE (van der Maaten and Geoffrey 2008) 2d embedding of the PPMI+discounting matrix: dependents
t-SNE (van der Maaten and Geoffrey 2008) 2d embedding of the PPMI+discounting matrix: dependents
t-SNE (van der Maaten and Geoffrey 2008) 2d embedding of the PPMI+discounting matrix: dependents
t-SNE (van der Maaten and Geoffrey 2008) 2d embedding of the PPMI+discounting matrix: dependents
Adverbial constructions

From a large collection of online product reviews:

<table>
<thead>
<tr>
<th>Modifiers</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>much more</td>
<td>4724</td>
</tr>
<tr>
<td>even more</td>
<td>4334</td>
</tr>
<tr>
<td>not very</td>
<td>2723</td>
</tr>
<tr>
<td>far more</td>
<td>2490</td>
</tr>
<tr>
<td>not too</td>
<td>2458</td>
</tr>
<tr>
<td>just plain</td>
<td>2117</td>
</tr>
<tr>
<td>just too</td>
<td>1938</td>
</tr>
<tr>
<td>very very</td>
<td>1819</td>
</tr>
<tr>
<td>not only</td>
<td>1771</td>
</tr>
<tr>
<td>way too</td>
<td>1594</td>
</tr>
<tr>
<td>little more</td>
<td>1508</td>
</tr>
<tr>
<td>not really</td>
<td>1422</td>
</tr>
<tr>
<td>just not very</td>
<td>216</td>
</tr>
<tr>
<td>just too damn</td>
<td>89</td>
</tr>
<tr>
<td>really not very</td>
<td>82</td>
</tr>
<tr>
<td>not only very</td>
<td>79</td>
</tr>
<tr>
<td>only slightly less</td>
<td>66</td>
</tr>
<tr>
<td>still not very</td>
<td>65</td>
</tr>
<tr>
<td>actually not too</td>
<td>58</td>
</tr>
<tr>
<td>still pretty darn</td>
<td>49</td>
</tr>
</tbody>
</table>
Classifier hypothesis: dependency edges beat bigrams

\[
\begin{align*}
\text{det}(\text{movie}, \text{This}) & \quad \mapsto \quad 1 \\
\text{nsubj}(\text{good}, \text{movie}) & \quad \mapsto \quad 1 \\
\text{aux}(\text{good}, \text{does}) & \quad \mapsto \quad 1 \\
\text{neg}(\text{good}, \text{not}) & \quad \mapsto \quad 1 \\
\text{cop}(\text{good}, \text{seem}) & \quad \mapsto \quad 1 \\
\end{align*}
\]

\[
\begin{align*}
\langle s \rangle \text{ This'} & \quad \mapsto \quad 1 \\
\text{This movie'} & \quad \mapsto \quad 1 \\
\text{movie does'} & \quad \mapsto \quad 1 \\
\text{does not'} & \quad \mapsto \quad 1 \\
\text{not seem'} & \quad \mapsto \quad 1 \\
\text{seem good'} & \quad \mapsto \quad 1 \\
\text{good } \langle /s \rangle '\ & \quad \mapsto \quad 1 \\
\end{align*}
\]

**Figure:** This movie does not seem good

\[
\begin{align*}
\text{det}(\text{scenery}, \text{the}) & \quad \mapsto \quad 1 \\
\text{nsubj}(\text{spectacular}, \text{scenery}) & \quad \mapsto \quad 1 \\
\text{cop}(\text{spectacular}, \text{was}) & \quad \mapsto \quad 1 \\
\text{conj\_but}(\text{spectacular}, \text{distracting}) & \quad \mapsto \quad 1 \\
\end{align*}
\]

\[
\begin{align*}
\langle s \rangle \text{ The'} & \quad \mapsto \quad 1 \\
\text{The scenery'} & \quad \mapsto \quad 1 \\
\text{scenery was'} & \quad \mapsto \quad 1 \\
\text{was spectacular'} & \quad \mapsto \quad 1 \\
\text{spectacular but'} & \quad \mapsto \quad 1 \\
\text{but distracting'} & \quad \mapsto \quad 1 \\
\text{distracting } \langle /s \rangle '\ & \quad \mapsto \quad 1 \\
\end{align*}
\]

**Figure:** This scenery was spectacular but distracting
Positive/negative sentiment with IMDB reviews

20K positive and 20K negative reviews from this collection:
http://ai.stanford.edu/~amaas/data/sentiment/

<sentence>
  <str>honestly, this is the worst franchise exploitation train wreck since .. .</str>
  <dep>[advmod(wreck-10, honestly-1), nsubj(wreck-10, this-3), ... ]</dep>
</sentence>

<sentence>
  <str>predator: requiem disaster .</str>
  <dep>[nn(disaster-4, requiem-3), dep(predator-1, disaster-4)]</dep>
</sentence>

.
.
.

Data and my code (using Python/sklearn):
http://www.stanford.edu/class/cs224u/code/depvssbigram.zip
Experimental set-up

Logistic Regression (MaxEnt) classifier. For each feature set:

Data and my code (using Python/sklearn):
http://www.stanford.edu/class/cs224u/code/depvsbigram.zip
Experimental set-up

Logistic Regression (MaxEnt) classifier. For each feature set:

1. Feature extraction: texts to vectors of feature counts.

Data and my code (using Python/sklearn):

http://www.stanford.edu/class/cs224u/code/depvsbigram.zip
Experimental set-up

Logistic Regression (MaxEnt) classifier. For each feature set:

1. Feature extraction: texts to vectors of feature counts.
2. Randomly split the data:

Data and my code (using Python/sklearn):
http://www.stanford.edu/class/cs224u/code/depvsbigram.zip
Experimental set-up

Logistic Regression (MaxEnt) classifier. For each feature set:

1. Feature extraction: texts to vectors of feature counts.
2. Randomly split the data:
   - 50% dev-set

Data and my code (using Python/sklearn):
http://www.stanford.edu/class/cs224u/code/depvsbigram.zip
Experimental set-up

Logistic Regression (MaxEnt) classifier. For each feature set:

1. Feature extraction: texts to vectors of feature counts.
2. Randomly split the data:
   - 50% dev-set
   - 50% eval-set

Data and my code (using Python/sklearn):
http://www.stanford.edu/class/cs224u/code/depvsbigram.zip
Experimental set-up

Logistic Regression (MaxEnt) classifier. For each feature set:

1. Feature extraction: texts to vectors of feature counts.
2. Randomly split the data:
   - 50% dev-set
   - 50% eval-set
3. With the dev-set, find the top 5000 most informative features (using a $\chi^2$ test of association) and the best regularization regime (L1 vs. L2, regularization strength in [0.1, 2]).

Data and my code (using Python/sklearn):
http://www.stanford.edu/class/cs224u/code/depvsbigram.zip
Experimental set-up

Logistic Regression (MaxEnt) classifier. For each feature set:

1. Feature extraction: texts to vectors of feature counts.
2. Randomly split the data:
   - 50% dev-set
   - 50% eval-set
3. With the dev-set, find the top 5000 most informative features (using a $\chi^2$ test of association) and the best regularization regime (L1 vs. L2, regularization strength in [0.1, 2]).
4. With the eval-set, evaluate the best model via 10-fold cross-validation.

Data and my code (using Python/sklearn):
http://www.stanford.edu/class/cs224u/code/depvsbigram.zip
Experimental set-up

Logistic Regression (MaxEnt) classifier. For each feature set:

1. Feature extraction: texts to vectors of feature counts.
2. Randomly split the data:
   - 50% dev-set
   - 50% eval-set
3. With the dev-set, find the top 5000 most informative features (using a $\chi^2$ test of association) and the best regularization regime (L1 vs. L2, regularization strength in [0.1, 2]).
4. With the eval-set, evaluate the best model via 10-fold cross-validation.
5. F1 as the primary evaluation statistic; non-parametric Wilcoxon rank-sums test to compare differences for statistical significance.

Data and my code (using Python/sklearn):
http://www.stanford.edu/class/cs224u/code/depvsbigram.zip
Results

Figure: Results of 10-fold cross-validation. Error bars are standard errors. All pairs of models are statistically different ($p < 0.001$).

<table>
<thead>
<tr>
<th>Features</th>
<th>Penalty</th>
<th>Prior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unigrams</td>
<td>L2</td>
<td>0.1</td>
</tr>
<tr>
<td>Bigrams</td>
<td>L2</td>
<td>0.2</td>
</tr>
<tr>
<td>Dependencies</td>
<td>L2</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Discussion

- Ceiling effect?
- Loss of information as a result of dependencies tokenization?
- Sparsity induced by the interlocking dependency relations?
- ...
Negation

- Negation is frequent, systematic, and semantically potent.
- Let’s see if we can use dependencies to get a grip on what it means and how it interacts with its fellow constituents.
- The lessons learned should generalize to a wide range of semantic relations and operations, many of which we will study during the unit on semantic composition.
Tracking the influence of negation: semantic scope

A few examples (of many):

I didn’t enjoy it.  

I never enjoy it.  

I don’t think I will enjoy it.
Scope domains

Parse trees

Op  Scope domain for Op

NP

Op  Scope domain for Op

PP

NP

Op

Dependencies. ‘rel’ should exclude certain non-scope relations.

Negation generalized: downward monotonicity

**Definition (Upward monotonicity)**

An operator $\delta$ is upward monotone iff for all expressions $\alpha$ in the domain of $\delta$:

if $\alpha \subseteq \beta$, then $(\delta \alpha) \subseteq (\delta \beta)$

**Definition (Downward monotonicity)**

An operator $\delta$ is downward monotone iff for all expressions $\alpha$ in the domain of $\delta$:

if $\alpha \subseteq \beta$, then $(\delta \beta) \subseteq (\delta \alpha)$
Negation generalized: downward monotonicity

Definition (Upward monotonicity)

An operator $\delta$ is upward monotone iff for all expressions $\alpha$ in the domain of $\delta$:

if $\alpha \subseteq \beta$, then $(\delta\alpha) \subseteq (\delta\beta)$

Definition (Downward monotonicity)

An operator $\delta$ is downward monotone iff for all expressions $\alpha$ in the domain of $\delta$:

if $\alpha \subseteq \beta$, then $(\delta\beta) \subseteq (\delta\alpha)$

A student smoked.

A Swedish student smoked. A student smoked cigars.
Negation generalized: downward monotonicity

**Definition (Upward monotonicity)**

An operator $\delta$ is upward monotone iff for all expressions $\alpha$ in the domain of $\delta$:

$$\text{if } \alpha \subseteq \beta, \text{ then } (\delta \alpha) \subseteq (\delta \beta)$$

**Definition (Downward monotonicity)**

An operator $\delta$ is downward monotone iff for all expressions $\alpha$ in the domain of $\delta$:

$$\text{if } \alpha \subseteq \beta, \text{ then } (\delta \beta) \subseteq (\delta \alpha)$$

A student smoked.

A Swedish student smoked. A student smoked cigars.
Negation generalized: downward monotonicity

**Definition (Upward monotonicity)**

An operator $\delta$ is upward monotone iff for all expressions $\alpha$ in the domain of $\delta$:

if $\alpha \subseteq \beta$, then $(\delta \alpha) \subseteq (\delta \beta)$

**Definition (Downward monotonicity)**

An operator $\delta$ is downward monotone iff for all expressions $\alpha$ in the domain of $\delta$:

if $\alpha \subseteq \beta$, then $(\delta \beta) \subseteq (\delta \alpha)$

A student smoked.

$\Rightarrow$  $\Leftrightarrow$

A Swedish student smoked. A student smoked cigars.

No student smoked.

No Swedish student smoked. No student smoked cigars.
Negation generalized: downward monotonicity

**Definition (Upward monotonicity)**

An operator $\delta$ is upward monotone iff for all expressions $\alpha$ in the domain of $\delta$:

if $\alpha \subseteq \beta$, then $(\delta \alpha) \subseteq (\delta \beta)$

**Definition (Downward monotonicity)**

An operator $\delta$ is downward monotone iff for all expressions $\alpha$ in the domain of $\delta$:

if $\alpha \subseteq \beta$, then $(\delta \beta) \subseteq (\delta \alpha)$

A student smoked.

A Swedish student smoked. A student smoked cigars.

No student smoked.

No Swedish student smoked. No student smoked cigars.
Negation generalized: downward monotonicity

**Definition (Upward monotonicity)**

An operator $\delta$ is upward monotone iff for all expressions $\alpha$ in the domain of $\delta$:

$$\text{if } \alpha \subseteq \beta, \text{ then } (\delta \alpha) \subseteq (\delta \beta)$$

**Definition (Downward monotonicity)**

An operator $\delta$ is downward monotone iff for all expressions $\alpha$ in the domain of $\delta$:

$$\text{if } \alpha \subseteq \beta, \text{ then } (\delta \beta) \subseteq (\delta \alpha)$$

A student smoked.

\[\Rightarrow\]

A Swedish student smoked. A student smoked cigars.

No student smoked.

\[\Rightarrow\]

No Swedish student smoked. No student smoked cigars.

Every student smoked.

Every Swedish student smoked. Every student smoked cigars.
Negation generalized: downward monotonicity

**Definition (Upward monotonicity)**

An operator $\delta$ is upward monotone iff for all expressions $\alpha$ in the domain of $\delta$:

$$\text{if } \alpha \subseteq \beta, \text{ then } (\delta \alpha) \subseteq (\delta \beta)$$

**Definition (Downward monotonicity)**

An operator $\delta$ is downward monotone iff for all expressions $\alpha$ in the domain of $\delta$:

$$\text{if } \alpha \subseteq \beta, \text{ then } (\delta \beta) \subseteq (\delta \alpha)$$

A student smoked.  

$\Rightarrow$  

A Swedish student smoked.  

A student smoked cigars.

No student smoked.  

$\Rightarrow$  

No Swedish student smoked.  

No student smoked cigars.

Every student smoked.  

$\Rightarrow$  

Every Swedish student smoked.  

Every student smoked cigars.
Negation generalized: downward monotonicity

**Definition (Upward monotonicity)**

An operator $\delta$ is upward monotone iff for all expressions $\alpha$ in the domain of $\delta$:

$$\text{if } \alpha \subseteq \beta, \text{ then } (\delta\alpha) \subseteq (\delta\beta)$$

**Definition (Downward monotonicity)**

An operator $\delta$ is downward monotone iff for all expressions $\alpha$ in the domain of $\delta$:

$$\text{if } \alpha \subseteq \beta, \text{ then } (\delta\beta) \subseteq (\delta\alpha)$$

A student smoked.

\[\Rightarrow\]

A Swedish student smoked.

\[\Rightarrow\]

A student smoked cigars.

No student smoked.

\[\Rightarrow\]

No Swedish student smoked.

\[\Rightarrow\]

No student smoked cigars.

Every student smoked.

\[\Rightarrow\]

Every Swedish student smoked.

\[\Rightarrow\]

Every student smoked cigars.

Few students smoked.

Few Swedish students smoked.

Few students smoked cigars.
Negation generalized: downward monotonicity

Definition (Upward monotonicity)

An operator $\delta$ is upward monotone iff for all expressions $\alpha$ in the domain of $\delta$:

if $\alpha \subseteq \beta$, then $(\delta \alpha) \subseteq (\delta \beta)$

Definition (Downward monotonicity)

An operator $\delta$ is downward monotone iff for all expressions $\alpha$ in the domain of $\delta$:

if $\alpha \subseteq \beta$, then $(\delta \beta) \subseteq (\delta \alpha)$

A student smoked.

$\Rightarrow$ $\nLeftarrow$

A Swedish student smoked. A student smoked cigars.

No student smoked.

$\Rightarrow$ $\nLeftarrow$

No Swedish student smoked. No student smoked cigars.

Every student smoked.

$\Rightarrow$ $\nLeftarrow$

Every Swedish student smoked. Every student smoked cigars.

Few students smoked.

$\Rightarrow$ $\nLeftarrow$

Few Swedish students smoked. Few students smoked cigars.
Marking the scope of negation

A few examples (of many):

- `the movie was not very good .`
- `i rarely enjoy horror movies .`
- `i do n't think that is a good idea .`
Approximation with tokenized strings

I’d be remiss if I didn’t point out that the effects of negation can be nicely approximated by a string-level operation (Das and Chen 2001; Pang et al. 2002).

1. Tokenize in a way that isolates and preserves clause-level punctuation. Starter Python tokenizer:
   
   http://sentiment.christopherpotts.net/code-data/happyfuntokenizing.py

2. Append a \_NEG suffix to every word appearing between a negation and a clause-level punctuation mark.

3. A negation is any word matching this regex:

   ```regex
   (?:(?:never|no|nothing|nowhere|noone|none|not|
         havent|hasnt|hadnt|cant|couldnt|shouldnt|
         wont|wouldnt|dont|doesn't|didn't|isnt|arent|aint
   )\$)
   ```
Predicting the effects of negation using IMDB user-supplied reviews

Outside the scope of negation

- **good** – 732,963 tokens
  - Cat = 0.01 (p = 0.152)
  - Cat^2 = -0.02 (p < 0.001)

- **bad** – 254,146 tokens
  - Cat = -0.2 (p < 0.001)
  - Cat^2 = 0.01 (p < 0.001)

- **excellent** – 136,404 tokens
  - Cat = 0.22 (p < 0.001)

- **terrible** – 45,470 tokens
  - Cat = -0.28 (p < 0.001)
  - Cat^2 = 0.02 (p < 0.001)
Predicting the effects of negation using IMDB user-supplied reviews

Outside the scope of negation

**good** – 732,963 tokens

Cat = 0.01 (p = 0.152)
Cat^2 = -0.02 (p < 0.001)

**bad** – 254,146 tokens

Cat = -0.2 (p < 0.001)
Cat^2 = 0.01 (p < 0.001)

**excellent** – 136,404 tokens

Cat = 0.22 (p < 0.001)
Cat^2 = 0.01 (p < 0.001)

**terrible** – 45,470 tokens

Cat = -0.28 (p < 0.001)
Cat^2 = 0.02 (p < 0.001)

In the scope of negation

**neg(good)** – 169,772 tokens

Cat = -0.06 (p < 0.001)
Cat^2 = -0.01 (p < 0.001)

**neg(bad)** – 113,865 tokens

Cat = -0.14 (p < 0.001)
Cat^2 = -0.02 (p = 0.011)

**neg(excellent)** – 10,393 tokens

Cat = 0.15 (p < 0.001)
Cat^2 = 0.00 (p = 0.152)

**neg(terrible)** – 9,936 tokens

Cat = -0.25 (p < 0.001)
Generalizing further still: commitment and perspective

Overview

- Whereas $\text{neg}(p)$ entails that $p$ is not factual,
- speech and attitude predicates are semantically consistent with $p$ and its negation,
- though the pragmatics is a lot more complicated; (de Marneffe et al. 2012).

Examples

1. The dictator claimed that no citizens were injured.
2. The Red Cross claimed that no citizens were injured.
3. They said it would be horrible, but they were wrong: I loved it!!!

How might we get a grip on the semantic effects of these predicates?
References I


de Marneffe, Marie-Catherine; Bill MacCartney; and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, 449–454. ACL.


Mintz, Mike; Steven Bills; Rion Snow; and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 1003–1011. Suntec, Singapore: Association for Computational Linguistics.


