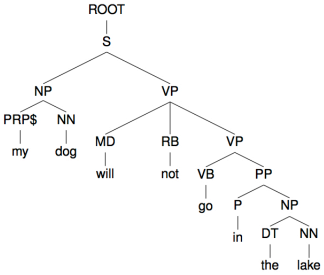
Dependency parses for NLU

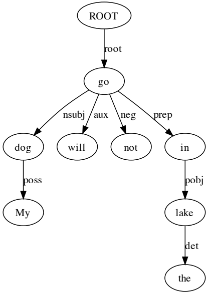
Christopher Potts

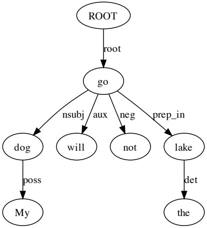
CS 244U: Natural language understanding
April 21

Syntactic structure: *My dog will not go in the lake.*

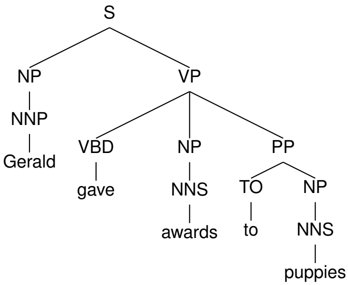
Treebank-style parsetree	Dependencies	Collapsed dependencies
(ROOT (S (NP (PRP$ My) (NN dog)) (VP (MD will) (RB not) (VP (VB go) (PP (IN in) (NP (DT the) (NN lake))))) (. .)))	poss(dog-2, My-1) nsubj(go-5, dog-2) aux(go-5, will-3) neg(go-5, not-4) root(ROOT-0, go-5) prep(go-5, in-6) det(lake-8, the-7) pobj(in-6, lake-8)	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) prep_in(go-5, lake-8)

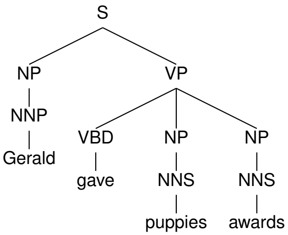


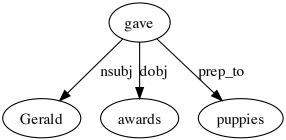


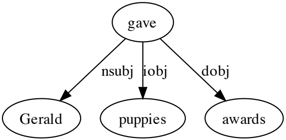


Simplified relationships, easier feature extraction









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

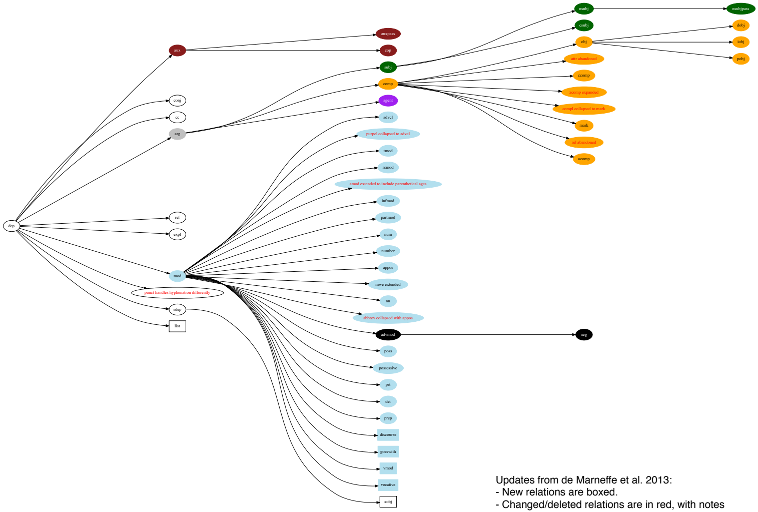
1. Get a feel for Stanford dependencies
2. Case study: advmod-based VSMs
3. Case study: dependencies as classifier features
4. Case study: capturing the semantic influence of negation

Dependency structures in NLU

Dependencies as the basis for features:

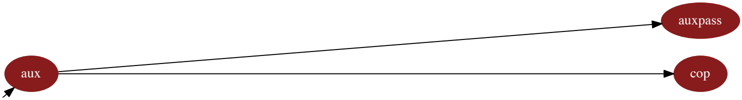
- Word-sense disambiguation (Lin 1998) [last year's slides on WSD]
- Relation extraction (Snow et al. 2005; Mintz et al. 2009)
- Semantic role labeling (Surdeanu et al. 2008; Johansson and Nugues 2008)
- Semantic parsing (Liang et al. 2013)
- Detecting speaker commitment (hedging, etc.; de Marneffe et al. 2012)
- Forecasting public opinion (Lerman et al. 2008)
- Analysis of political debates (Balahur et al. 2009)
- Drug interactions (Percha et al. 2012)
- ...

Stanford dependencies relation hierarchy

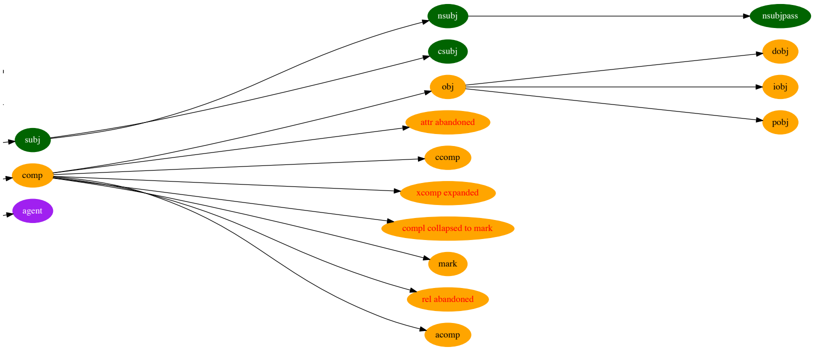


Updates from de Marneffe et al. 2013:
- New relations are boxed.
- Changed/deleted relations are in red, with notes

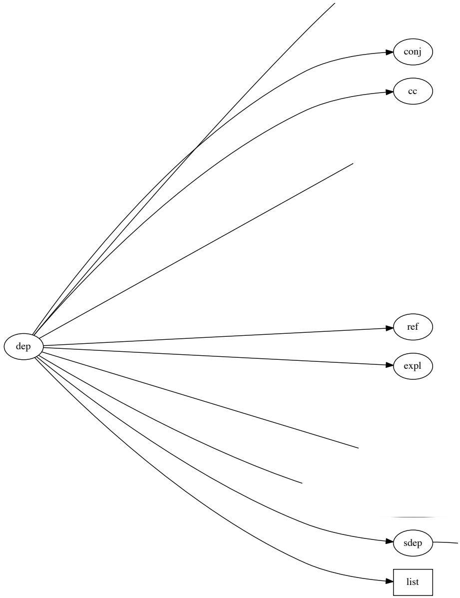
Stanford dependencies relation hierarchy



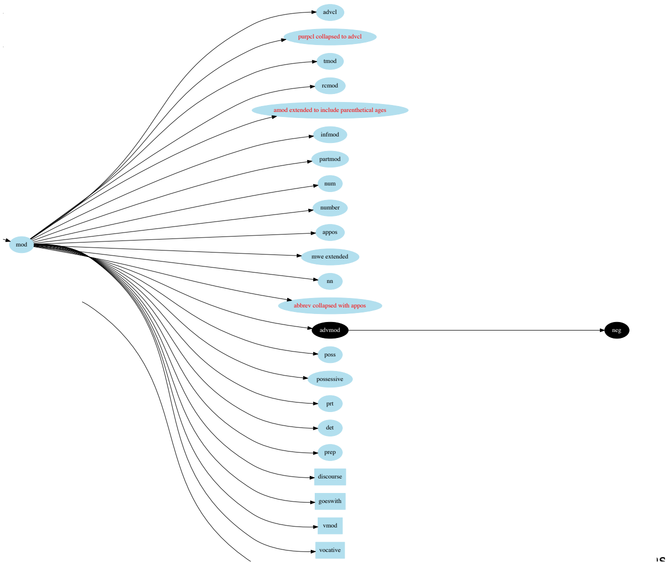
Stanford dependencies relation hierarchy



Stanford dependencies relation hierarchy



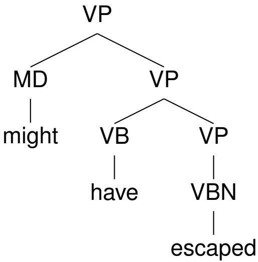
Stanford dependencies relation hierarchy



Stanford dependency construction

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

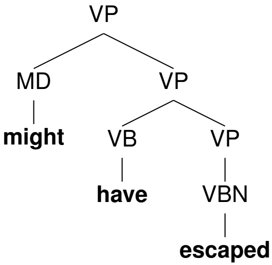
1. **Dependency extraction**: for each constituent, identify its *semantic* head and project the head upwards:



Stanford dependency construction

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

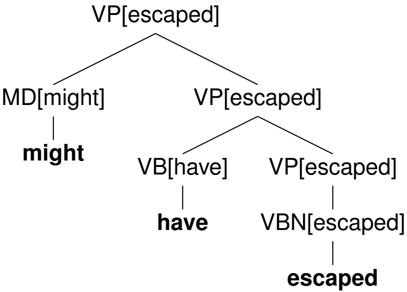
1. **Dependency extraction**: for each constituent, identify its *semantic* head and project the head upwards:



Stanford dependency construction

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

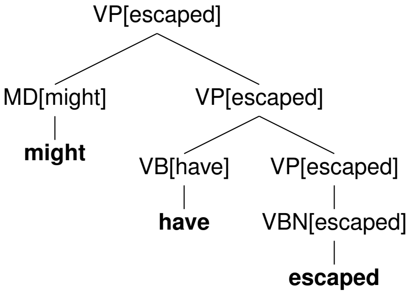
1. **Dependency extraction**: for each constituent, identify its *semantic* head and project the head upwards:



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:



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:

```
aux(escaped, might)
aux(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

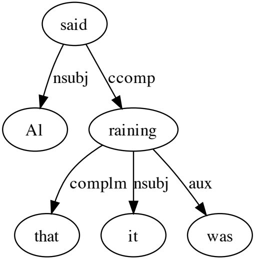
The Stanford parser is distributed with starter Java code for parsing your own data. It also has a flexible command-line interface. Some relevant commands:

```
# Map plain text to dependency structures:

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


# Map tagged data to dependency structures:

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):

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

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

Graphviz

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



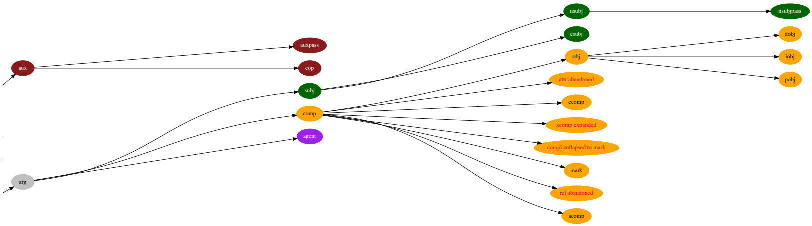

```
digraph g {
  /* Nodes */
  "Al-1" [label="Al"];
  "said-2" [label="said"];
  "that-3" [label="that"];
  "it-4" [label="it"];
  "was-5" [label="was"];
  "raining-6" [label="raining"];
  /* Edges */
  "said-2" -> "Al-1" [label="nsubj"];
  "raining-6" -> "that-3" [label="complm"];
  "raining-6" ->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

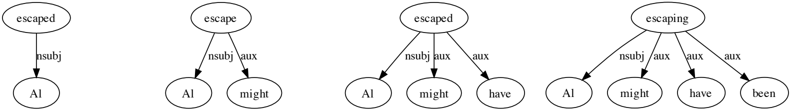
Verbal structures



Verbal structures: intransitive and transitive

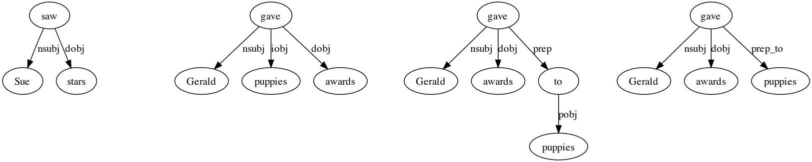
Intransitive

Al escaped. | Al might escape. | Al might have escaped. | Al might have been escaping.



Transitive

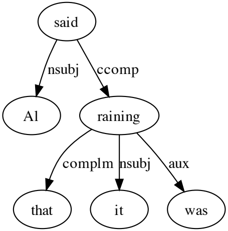
Sue saw stars. | Gerald gave puppies awards. | Gerald gave awards to puppies — basic | collapsed



Verbal structures: sentential complements

Tensed

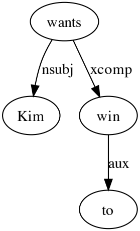
Al said that it was raining.



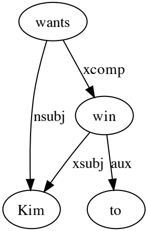
Infinitival

Kim wants to win.

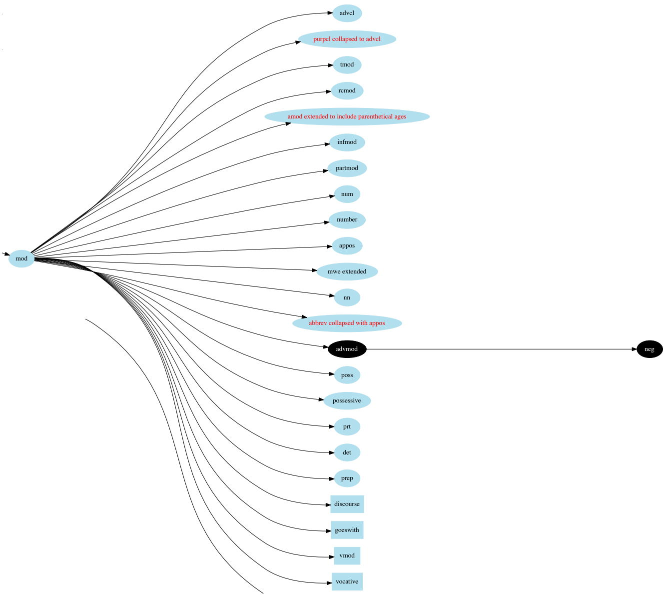
Basic



Collapsed



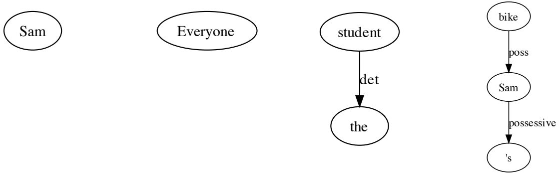
Nominals


mod
advcl
purpcl collapsed to advcl
tmod
rcmod
amod extended to include parenthetical ages
infmod
partmod
num
number
appos
mwe extended
nn
abbrev collapsed with appos
advmod
neg
poss
possessive
prt
det
prep
discourse
goeswith
vmod
vocative

Nominal structures

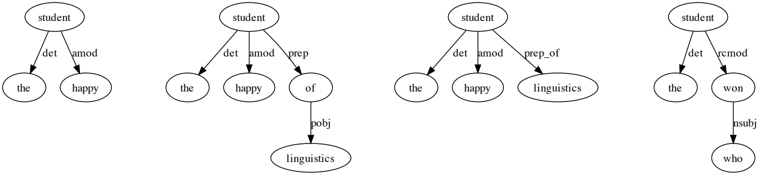
Basic

Proper name | Quantifier | Determiner | Possessive: basic | collapsed



Modified

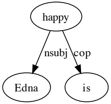
Adjective | Prepositional: basic | collapsed | Relative clause



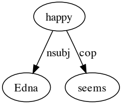
Modification

Predicative constructions

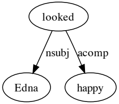
Basic



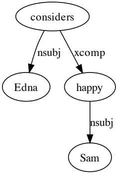
Lexical pred



Lexical

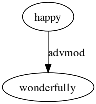


Small clause

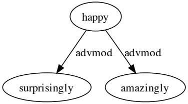


Adverbs

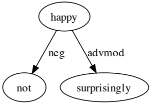
wonderfully happy



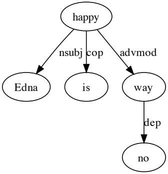
surprisingly amazingly happy



not surprisingly happy



in no way happy

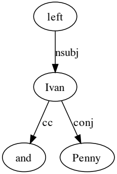


Coordination: conj and cc

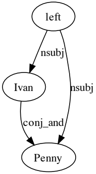
Nominals (here, nsubj)

Ivan and Penny left.

basic



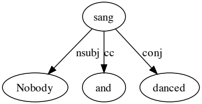
collapsed



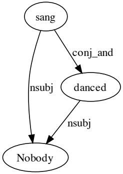
Verb phrases

Nobody sang and danced.

basic



collapsed



advmod dependencies

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*, . . .

Adverb	Totally open	Totally closed	Upper closed	Lower closed
Maximality	*	✓	✓	*
Proportion	*	✓	*	*
Minimality	*	✓	*	✓

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:

1	end	here	98434
2	well	as	84031
3	longer	no	74486
4	far	so	71853
5	much	so	71460
6	now	right	66373
7	much	too	66264
8	much	how	64794
9	said	also	62588
10	year	earlier	60290
	⋮		
3211133	scuff	how	1

Gigaword NYT (h/t to Nate Chambers for the parsing!)

dependent × parent matrix: raw counts

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

Dependent × parent matrix: positive PMI with contextual discounting

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

Some neighbors (cosine distance, PPMI+discounting matrix)

Adverbs

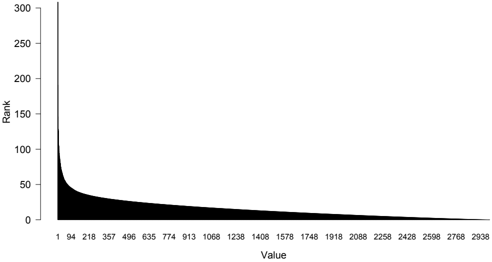
absolutely	certainly	never	recently	somewhat	quickly
utterly	definitely	not	subsequently	slightly	swiftly
totally	surely	maybe	ago	considerably	soon
truly	probably	either	since	decidedly	gradually
completely	obviously	ever	later	extremely	rapidly
equally	undoubtedly	yes	shortly	terribly	slowly
quite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

Adjectives

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

Latent Semantic Analysis

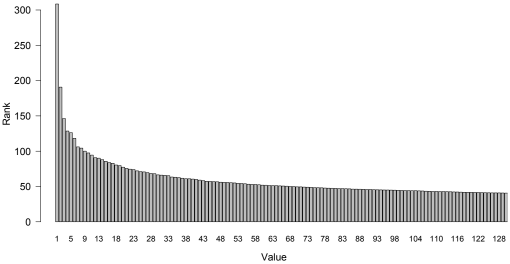
1. Apply singular value decomposition to the PPMI+discounting matrix.
2. Inspect singular values; settle on 25 dimensions:



3. For rows (dependents): $R[\,,1:25] \times S[1:25, 1:25]$
4. For columns (dependents): $S[1:25, 1:25] \times C[\,,1:25]^T$

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): $R[\,,1:25] \times S[1:25, 1:25]$
4. For columns (dependents): $S[1:25, 1:25] \times C[\,,1:25]^T$

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

Adverbs without LSA (repeated from earlier)

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

Adverbs with LSA (25 dimensions)

absolutely	certainly	never	recently	somewhat	quickly
utterly	surely	you	subsequently	palpably	swiftly
truly	definitely	maybe	later	decidedly	soon
totally	probably	just	d.calif	seeming	prematurely
manifestly	doubt	yes	ago	any	instantly
wholly	undoubtedly	ok	r.ohio	slightly	immediately
patently	necessarily	q	shortly	congenitally	speedily
hardly	importantly	pray	first	distinctly	eventually
indisputably	doubtless	hey	d.mo	visibly	gradually
flat.out	secondly	anyway	since	sufficiently	slowly

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

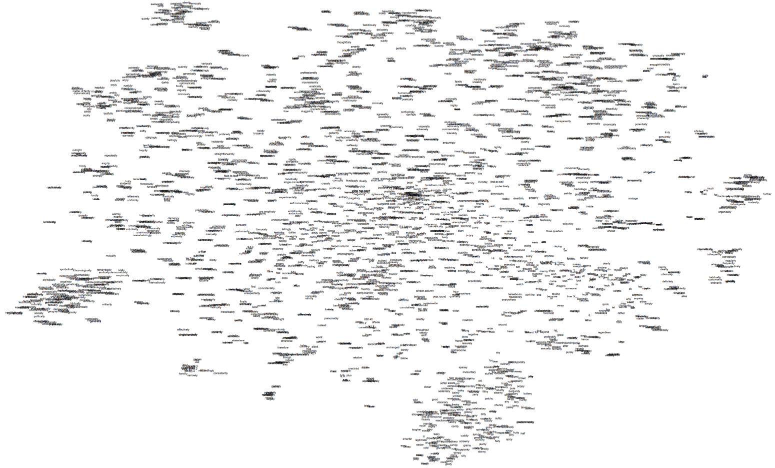
Adjectives without LSA (repeated from earlier)

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

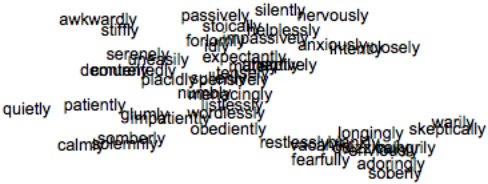
Adjectives with LSA (25 dimensions)

happy	sad	tall	full	straight	closed
nice	ugly	thick	light	normal	suspended
terrible	scary	deep	flat	free	shut
strange	weird	loud	calm	flat	retired
cute	strange	bright	dry	natural	halted
scary	tragic	cheap	smooth	certain	replaced
wild	nasty	tight	quiet	conventional	stopped
excited	dumb	fast	cool	routine	cleared
cool	boring	hot	soft	benign	locked
special	odd	quick	steady	reasonable	sealed

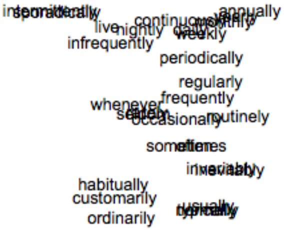
t-SNE (van der Maaten and Geoffrey 2008) 2d embedding of the PPMI+discounting matrix: adverbs



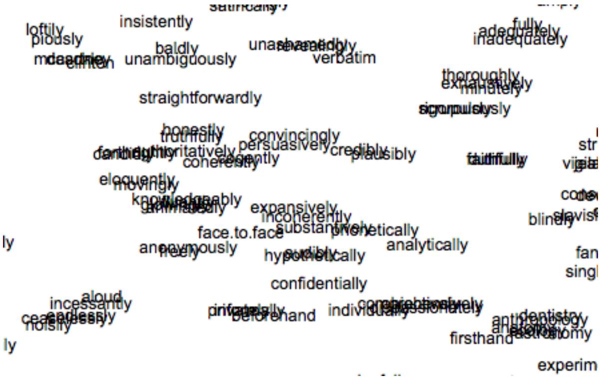
t-SNE (van der Maaten and Geoffrey 2008) 2d embedding of the PPMI+discounting matrix: adverbs



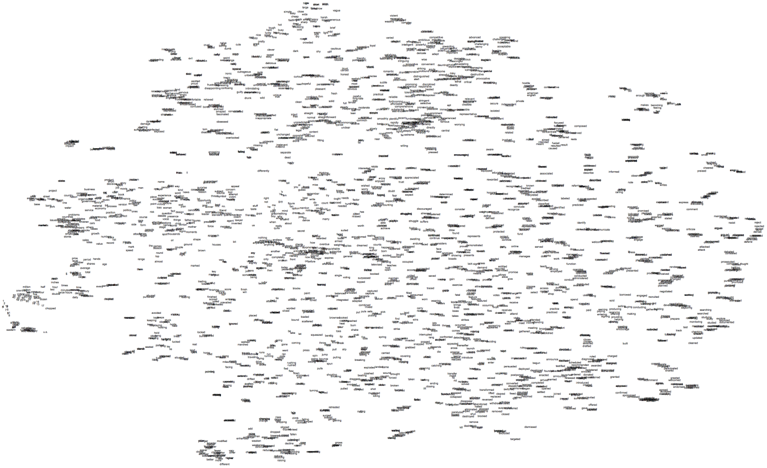
t-SNE (van der Maaten and Geoffrey 2008) 2d embedding of the PPMI+discounting matrix: adverbs



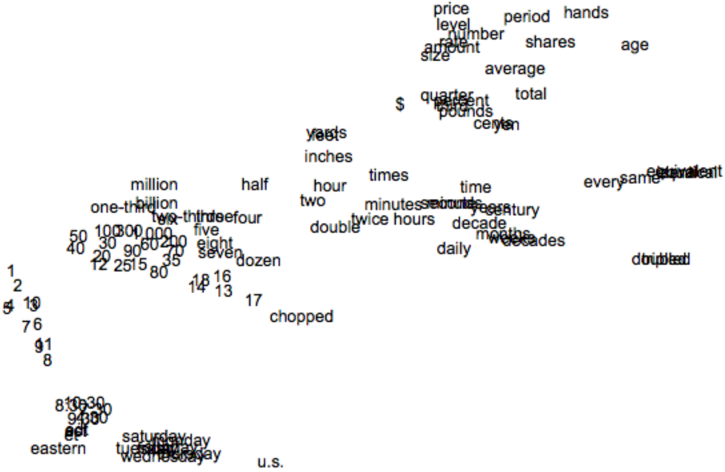
t-SNE (van der Maaten and Geoffrey 2008) 2d embedding of the PPMI+discounting matrix: adverbs



t-SNE (van der Maaten and Geoffrey 2008) 2d embedding of the PPMI+discounting matrix: dependents

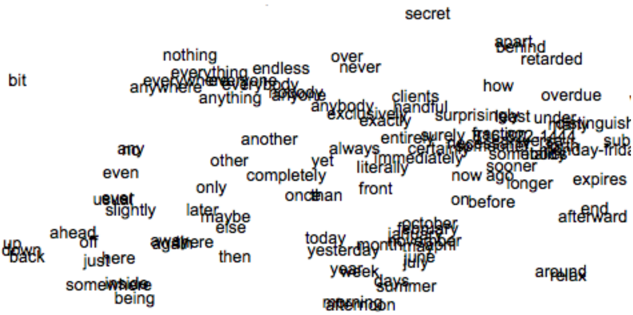


t-SNE (van der Maaten and Geoffrey 2008) 2d embedding of the PPMI+discounting matrix: dependents



t-SNE (van der Maaten and Geoffrey 2008) 2d embedding of the PPMI+discounting matrix: dependents

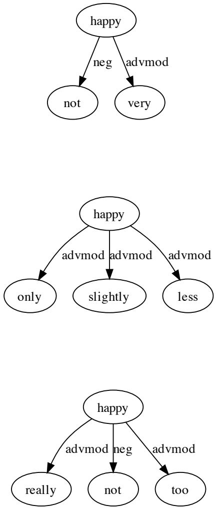
t-SNE (van der Maaten and Geoffrey 2008) 2d embedding of the PPMI+discounting matrix: dependents



Adverbial constructions

From a large collection of online product reviews:

Modifiers	Count
much more	4724
even more	4334
not very	2723
far more	2490
not too	2458
just plain	2117
just too	1938
very very	1819
not only	1771
way too	1594
little more	1508
not really	1422
⋮	
just not very	216
just too damn	89
really not very	82
not only very	79
only slightly less	66
still not very	65
actually not too	58
still pretty darn	49



Classifier hypothesis: dependency edges beat bigrams

$$\begin{bmatrix} \text{det(movie, This)} & \mapsto & 1 \\ \text{nsubj(good, movie)} & \mapsto & 1 \\ \text{aux(good, does)} & \mapsto & 1 \\ \text{neg(good, not)} & \mapsto & 1 \\ \text{cop(good, seem)} & \mapsto & 1 \end{bmatrix} \begin{bmatrix} \text{`<s> This'} & \mapsto & 1 \\ \text{`This movie'} & \mapsto & 1 \\ \text{`movie does'} & \mapsto & 1 \\ \text{`does not'} & \mapsto & 1 \\ \text{`not seem'} & \mapsto & 1 \\ \text{`seem good'} & \mapsto & 1 \\ \text{`good </s>'} & \mapsto & 1 \end{bmatrix}$$

Figure: This movie does not seem good

$$\begin{bmatrix} \text{det(scenery, the)} & \mapsto & 1 \\ \text{nsubj(spectacular, scenery)} & \mapsto & 1 \\ \text{cop(spectacular, was)} & \mapsto & 1 \\ \text{conj\_but(spectacular, distracting)} & \mapsto & 1 \end{bmatrix} \begin{bmatrix} \text{`<s> The'} & \mapsto & 1 \\ \text{`The scenery'} & \mapsto & 1 \\ \text{`scenery was'} & \mapsto & 1 \\ \text{`was spectacular'} & \mapsto & 1 \\ \text{`spectacular but'} & \mapsto & 1 \\ \text{`but distracting'} & \mapsto & 1 \\ \text{`distracting </s>'} & \mapsto & 1 \end{bmatrix}$$

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/depvsbigram.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 χ^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 χ^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 χ^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

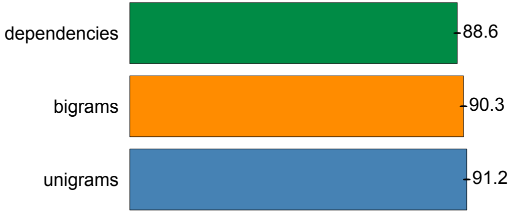


Figure: Results of 10-fold cross-validation. Error bars are standard errors. All pairs of models are statistically different ($p < 0.001$).

Features	Penalty	Prior
Unigrams	L2	0.1
Bigrams	L2	0.2
Dependencies	L2	0.2

Discussion

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

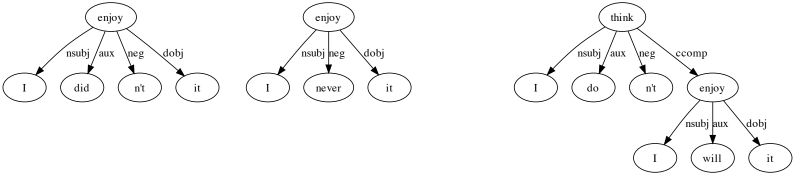
Negation

- 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

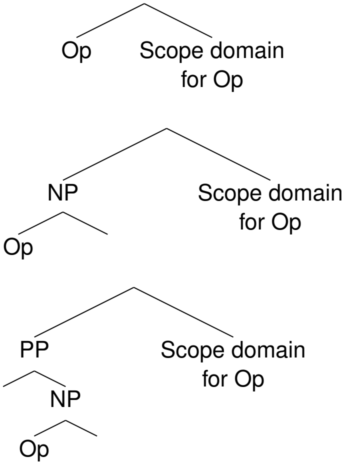
A few examples (of many):

I didn't enjoy it. I never enjoy it. I don't think I will enjoy it.

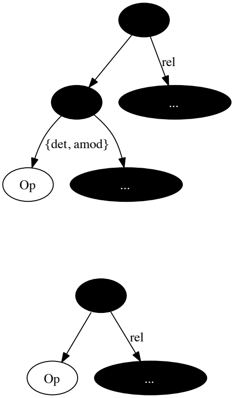


Scope domains

Parse trees



Dependencies. 'rel' should exclude certain non-scope relations.



(Danescu-Niculescu-Mizil et al. 2009; Danescu-Niculescu-Mizil and Lee 2010)

Negation generalized: downward monotonicity

Definition (Upward monotonicity)

An operator δ is upward monotone iff for all expressions α in the domain of δ:

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

Definition (Downard monotonicity)

An operator δ is downward monotone iff for all expressions α in the domain of δ:

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

Negation generalized: downward monotonicity

Definition (Upward monotonicity)

An operator δ is upward monotone iff for all expressions α in the domain of δ:

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

Definition (Downard monotonicity)

An operator δ is downward monotone iff for all expressions α in the domain of δ:

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

A student smoked.

A Swedish student smoked. A student smoked cigars.

Negation generalized: downward monotonicity

Definition (Upward monotonicity)

An operator δ is upward monotone iff for all expressions α in the domain of δ:

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

Definition (Downard monotonicity)

An operator δ is downward monotone iff for all expressions α in the domain of δ:

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

A student smoked.

$\Nearrow$ $\Nwarrow$

A Swedish student smoked. A student smoked cigars.

Negation generalized: downward monotonicity

Definition (Upward monotonicity)

An operator δ is upward monotone iff for all expressions α in the domain of δ:

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

Definition (Downard monotonicity)

An operator δ is downward monotone iff for all expressions α in the domain of δ:

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

A student smoked.

$\Nearrow$ $\Nwarrow$

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 δ is upward monotone iff for all expressions α in the domain of δ:

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

Definition (Downard monotonicity)

An operator δ is downward monotone iff for all expressions α in the domain of δ:

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

A student smoked.
$\Nearrow$ $\Nwarrow$
A Swedish student smoked. A student smoked cigars.

No student smoked.
$\Swarrow$ $\Searrow$
No Swedish student smoked. No student smoked cigars.

Negation generalized: downward monotonicity

Definition (Upward monotonicity)

An operator δ is upward monotone iff for all expressions α in the domain of δ:

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

Definition (Downard monotonicity)

An operator δ is downward monotone iff for all expressions α in the domain of δ:

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

A student smoked.

$\Nearrow \quad \Nwarrow$

A Swedish student smoked. A student smoked cigars.

No student smoked.

$\Swarrow \quad \Searrow$

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 δ is upward monotone iff for all expressions α in the domain of δ:

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

Definition (Downward monotonicity)

An operator δ is downward monotone iff for all expressions α in the domain of δ:

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

A student smoked.
⇗ ⇖
A Swedish student smoked. A student smoked cigars.

No student smoked.
⇙ ⇘
No Swedish student smoked. No student smoked cigars.

Every student smoked.
⇗ ⇘
Every Swedish student smoked. Every student smoked cigars.

Negation generalized: downward monotonicity

Definition (Upward monotonicity)

An operator δ is upward monotone iff for all expressions α in the domain of δ:

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

Definition (Downard monotonicity)

An operator δ is downward monotone iff for all expressions α in the domain of δ:

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

A student smoked.

$\Nearrow$ $\Nwarrow$

A Swedish student smoked. A student smoked cigars.

No student smoked.

$\Swarrow$ $\Searrow$

No Swedish student smoked. No student smoked cigars.

Every student smoked.

$\Nearrow$ $\Searrow$

Every Swedish student smoked. Every student smoked cigars.

Few students smoked.

Few Swedish students smoked. Few students smoked cigars.

Negation generalized: downward monotonicity

Definition (Upward monotonicity)

An operator δ is upward monotone iff for all expressions α in the domain of δ:

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

Definition (Downard monotonicity)

An operator δ is downward monotone iff for all expressions α in the domain of δ:

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

A student smoked.
⇗ ⇖
A Swedish student smoked. A student smoked cigars.

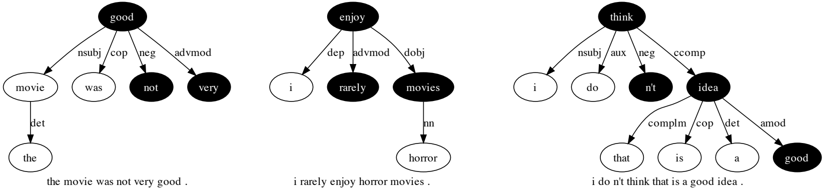
No student smoked.
⇙ ⇘
No Swedish student smoked. No student smoked cigars.

Every student smoked.
⇗ ⇘
Every Swedish student smoked. Every student smoked cigars.

Few students smoked.
⇙ ⇘
Few Swedish students smoked. Few students smoked cigars.

Marking the scope of negation

A few examples (of many):



Approximation with tokenized strings

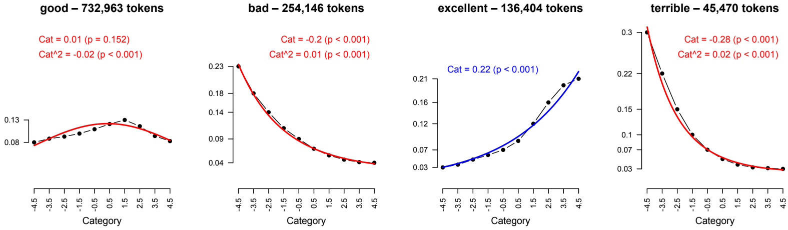
I'd be remiss if I didn't point out that the effects of negation can be nicely approximated by a string-level operation (Das and Chen 2001; Pang et al. 2002).

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:

```
(?:
    ^(?:never|no|nothing|nowhere|noone|none|not|
        havent|hasnt|hadnt|cant|couldnt|shouldnt|
        wont|wouldnt|dont|doesnt|didnt|isnt|arent|aint
    )$
)
|
n't
```

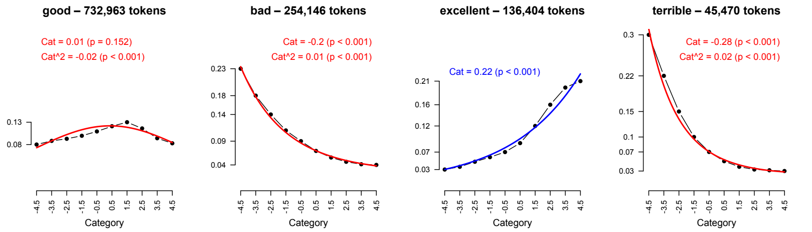
Predicting the effects of negation using IMDB user-supplied reviews

Outside the scope of negation

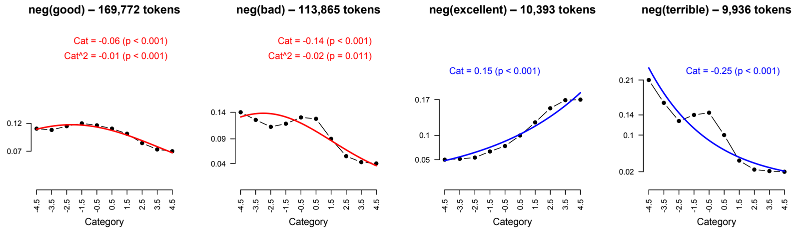


Predicting the effects of negation using IMDB user-supplied reviews

Outside the scope of negation



In the scope of negation



Generalizing further still: commitment and perspective

Overview

- Whereas **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

Balahur, Alexandra; Zornitsa Kozareva; and Andrés Montoyo. 2009. Determining the polarity and source of opinions expressed in political debates. In Alexander Gelbukh, ed., *Proceedings of the 10th International Conference on Computational Linguistics and Intelligent Text Processing*, 468–480. Berlin: Springer. doi:\bibinfo{doi}{10.1007/978-3-642-00382-0_38}.

Danescu-Niculescu-Mizil, Cristian and Lillian Lee. 2010. Don't 'have a clue'? Unsupervised co-learning of downward-entailing operators. In *Proceedings of the ACL 2010 Conference Short Papers*, 247–252. Uppsala, Sweden: Association for Computational Linguistics.

Danescu-Niculescu-Mizil, Cristian; Lillian Lee; and Richard Ducott. 2009. Without a 'doubt'? Unsupervised discovery of downward-entailing operators. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 137–145. Association for Computational Linguistics.

Das, Sanjiv and Mike Chen. 2001. Yahoo! for Amazon: Extracting market sentiment from stock message boards. In *Proceedings of the 8th Asia Pacific Finance Association Annual Conference*.

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.

Johansson, Richard and Pierre Nugues. 2008. Dependency-based semantic role labeling of PropBank. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 69–78. Honolulu, Hawaii: Association for Computational Linguistics.

Kennedy, Christopher. 2007. Vagueness and grammar: The semantics of relative and absolute gradable adjective. *Linguistics and Philosophy* 30(1):1–45.

Kennedy, Christopher and Louise McNally. 2005. Scale structure and the semantic typology of gradable predicates. *Language* 81(2):345–381.

Lerman, Kevin; Ari Gilder; Mark Dredze; and Fernando Pereira. 2008. Reading the markets: Forecasting public opinion of political candidates by news analysis. In *Proceedings of the 22nd International Conference on Computational Linguistics*, 473–480. Manchester, UK: Association for Computational Linguistics.

Liang, Percy; Michael I. Jordan; and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics* 39(2):389–446. doi:\bibinfo{doi}{10.1162/COLI_a_00127}.

Lin, Dekang. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL*, 768–774. Montreal: ACl.

van der Maaten, Laurens and Hinton Geoffrey. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9:2579–2605.

de Marneffe, Marie-Catherine; Miriam Connor; Natalia Silveira; Samuel R. Bowman; Timothy Dozat; and Christopher D. Manning. 2013. More constructions, more genres: Extending Stanford Dependencies. In Eva Hajičová; Kim Gerdes; and Leo Wanner, eds., *Proceedings of the Second International Conference on Dependency Linguistics*, 187–196. Prague.

de Marneffe, Marie-Catherine and Christopher D. Manning. 2008a. *Stanford Typed Dependencies Manual*. Stanford University.

de Marneffe, Marie-Catherine and Christopher D. Manning. 2008b. The Stanford typed dependencies representation. In *Proceedings of the COLING 2008 Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, 1–8. ACL.

de Marneffe, Marie-Catherine; Christopher D. Manning; and Christopher Potts. 2012. Did it happen? The pragmatic complexity of veridicality assessment. *Computational Linguistics* 38(2):301–333.

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.

References II

Pang, Bo; Lillian Lee; and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 79–86. Philadelphia: Association for Computational Linguistics.

Percha, Bethany; Yael Garten; and Russ B. Altman. 2012. Discovering and explanation of drug–drug interactions via text mining. *Pacific Symposium on Biocomputing* 410–421.

Snow, Rion; Daniel Jurafsky; and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In Lawrence K. Saul; Yair Weiss; and Léon Bottou, eds., *Advances in Neural Information Processing Systems 17*, 1297–1304. Cambridge, MA: MIT Press.

Surdeanu, Mihai; Richard Johansson; Adam Meyers; Lluís Màrquez; and Joakim Nivre. 2008. The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, 159–177. Manchester: Coling 2008 Organizing Committee.

Syrett, Kristen and Jeffrey Lidz. 2010. 30-month-olds use the distribution and meaning of adverbs to interpret novel adjectives. *Language Learning and Development* 6(4):258–282.