But. . . but. . . GRAMMAR!

Grammar-based approaches to opinion mining: Part 5 (ESSLLI 2013)

Asad Sayeed

Uni-Saarland

On the menu

- Processing grammatical structure to detect fine-grained opinions.
- Our dystopian future.

Q: So, uh, where was the grammar?

A: Uh . . . uhmmm . . .

But your point is well-taken.

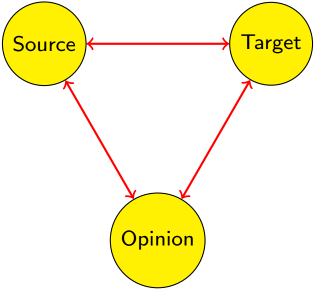
To recap. . .

1. We started with basic bag-of-words product review work $\rightarrow$ not much grammar.
2. Then we covered resource construction $\rightarrow$ sometimes intended *for* grammar work.
3. Next we covered a little bit of machine learning $\rightarrow$ could be *for* grammar work.
4. And then we covered a simple vector space model (not grammar) and CRF-based techniques (some grammar).

But what we want is *full grammar*.

Q: Why do we want it?

A: Because it's cool?



Well yeah, but, we need as much evidence as possible to identify the full sentiment triangle.

Remember this from part 2?

Example: information technology business press

Lloyd Hession, chief security officer at BT Radianz in New York, said that virtualization also opens up a slew of potential network access control issues.

- "slew" and "issues": convey negative sentiment about "virtualization".
- How do we know they're negative in this domain?
- What about words like "update"? Important in IT domain, not mentioned in major polarity lexicon.

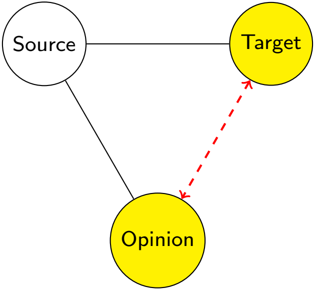
The "little" details of syntax/semantics and the "big" details of pragmatics actually intertwine.

So let's try this without much machine learning.

Remember we talked about Zhuang et al. (2006) in part 4? Only in passing.

- Movie review mining – author is source.
- Use grammatical templates and keyword lists from training data to identify candidate targets in test data.

So it sits about here on the sentiment triangle.



As we discussed in part 4, targets tend to need more grammar.

What does that look like overall?

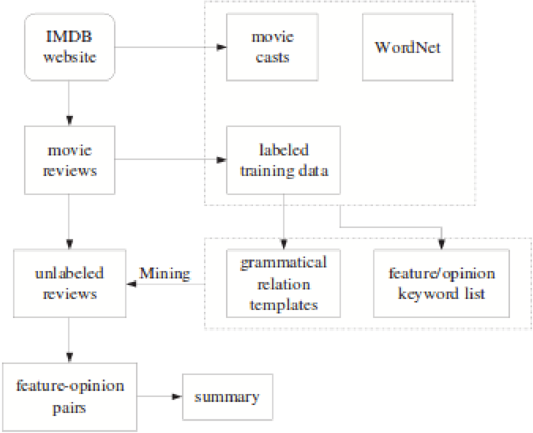


Figure 1: Architectural overview of our multi-knowledge based approach

And what do the extracted patterns look like?

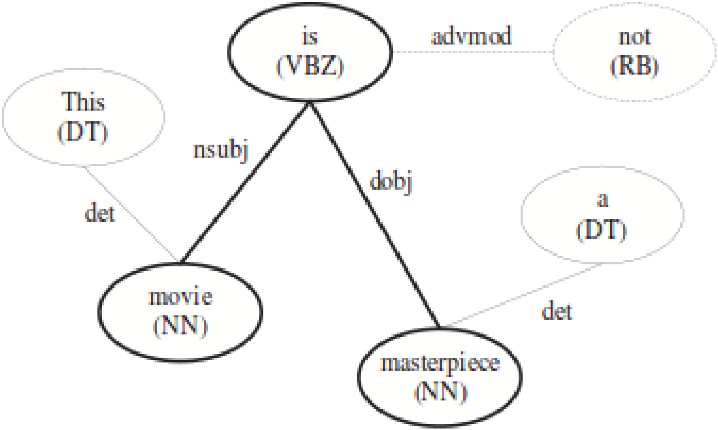


Figure 2: Dependency grammar graph

And how well does their overall approach work?

Table 4: Results of feature-opinion pair mining

Movie	Hu and Liu's approach			The proposed approach		
	Precision	Recall	F-score	Precision	Recall	F-score
Gone with the Wind	0.462	0.651	0.551	0.556	0.564	0.560
The Wizard of OZ	0.475	0.705	0.568	0.589	0.648	0.618
Casablanca	0.431	0.661	0.522	0.452	0.521	0.484
The Godfather	0.400	0.654	0.496	0.476	0.619	0.538
The Shawshank Redemption	0.443	0.620	0.517	0.514	0.644	0.571
The Matrix	0.353	0.565	0.434	0.468	0.593	0.523
The Two Towers	0.338	0.583	0.428	0.404	0.577	0.476
American Beauty	0.375	0.576	0.454	0.393	0.527	0.450
Gladiator	0.405	0.619	0.489	0.505	0.632	0.562
Wo hu cang long	0.368	0.567	0.447	0.465	0.537	0.498
Spirited Away	0.388	0.583	0.466	0.493	0.567	0.527
Average	0.403	0.617	0.488	0.483	0.585	0.529

Q: How to make it more flexible?

A: By learning generalized characteristics of useful paths.

Another opportunity for self-aggrandizement ;-)

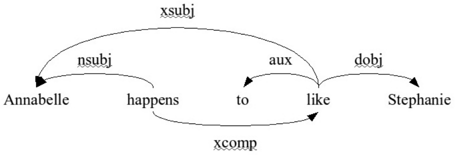
Sayeed et al. (2012) presents a data structure to facilitate learning grammatical connections.

- SRT – "syntactic relatedness trie", compress dependency trees? information to overcome data sparseness.
- Use graphical modelling technique to learn characteristics of grammatical connections.

How does this use dependency trees?

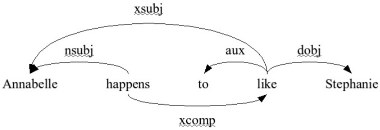
Anchor sentiment at words that apply to a target.

- Our approach: word-level annotations with links to domain concepts.
- What do we mean by "apply to a target"? Transitive links (paths) through dependency parse.
- Example: Stanford dependency parse for "Annabelle happens to like Stephanie":



- Ultimately: make grammatical info avail. for polarity classification.

This is a labelling problem.



We need to learn the difference. How?

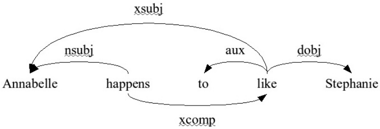
Flow path

Stephanie $\xrightarrow{\text{dobj}}$ like

Inert path

Annabelle $\xrightarrow{\text{nsubj}}$ happens $\xrightarrow{\text{xcomp}}$ like

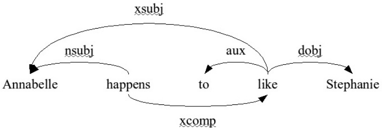
This is a labelling problem.



By labelling each element along the path.

- flow node: there is a node that follows that eventually leads to a opinion word.
- inert node: no node that follows leads to an opinion word.

This is a labelling problem.



By labelling each element along the path.

Flow path

Stephanie:flow $\xrightarrow{\text{dobj}}$ like:flow

Inert path

Annabelle:inert $\xrightarrow{\text{nsubj}}$ happens:inert $\xrightarrow{\text{xcomp}}$ like:inert

Let's have a more complicated example.

From the MPQA, with Pitt lexicon sentiment words

The **dominant** *role of the European climate protection policy has benefits for our economy.*

Let's say that "dominant" applies to "role", not "policy." Then paths from "policy" are the following:

Flow paths

policy $\xrightarrow{\text{nn}}$ protection

policy $\xrightarrow{\text{prep\_of}}$ role $\xrightarrow{\text{nsubj}}$ has $\xrightarrow{\text{dobj}}$ benefits

Let's have a more complicated example.

From the MPQA, with Pitt lexicon sentiment words

The **dominant** *role of the European climate protection policy has benefits for our economy.*

Let's say that "dominant" applies to "role", not "policy." They share elements with:

Inert path

policy $\xrightarrow{\text{prep\_of}}$ role $\xrightarrow{\text{amod}}$ dominant

It leaves us with a data sparsity problem.

Overlapping paths with potentially overlapping labels

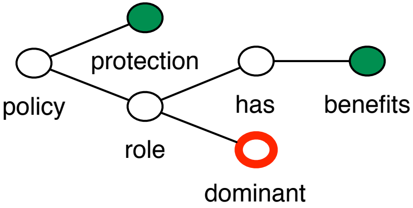
All flow: policy $\xrightarrow{\text{prep\_of}}$ role $\xrightarrow{\text{nsubj}}$ has $\xrightarrow{\text{dobj}}$ benefits

All inert: policy $\xrightarrow{\text{prep\_of}}$ role $\xrightarrow{\text{amod}}$ dominant

- When flow and inert paths coincide, this can cause a sparsity problem.
- Solution: partially mark inert paths with flow at any point where it coincides with flow.
  - We want to follow paths from target to opinion word.
  - flow means "continue following".

And our answer was the SRT

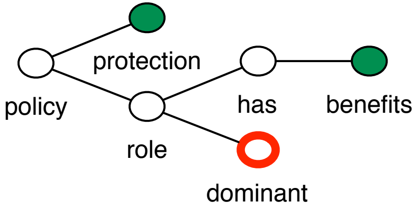
- Legend:
  - Unlabelled nodes are empty.
  - flow nodes are filled green.
  - inert nodes are red circles.
- (We omit dependency edge labels for space).



And our answer was the SRT

SRT construction:

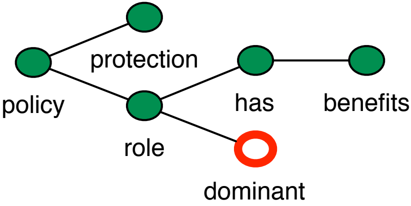
- Step 1:
  - Insert all paths into tree.
  - Label leaves as flow or inert.
- Step 2:
  - Propagate all flow up the tree.
  - (Anything left over is inert.)



And our answer was the SRT

SRT construction:

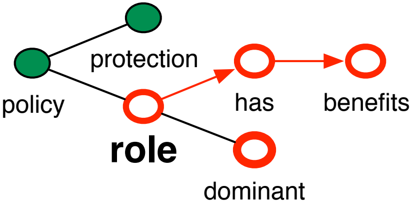
- Step 1:
  - Insert all paths into tree.
  - Label leaves as flow or inert.
- Step 2:
  - Propagate all flow up the tree.
  - (Anything left over is inert.)



And our answer was the SRT

During inference: node propagation scheme guarantees coherent paths.

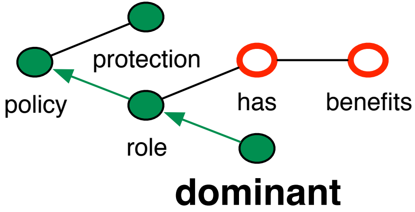
- Changing a node to inert makes all its **descendants** inert.
- Changing a node to flow makes all its **ancestors** flow.



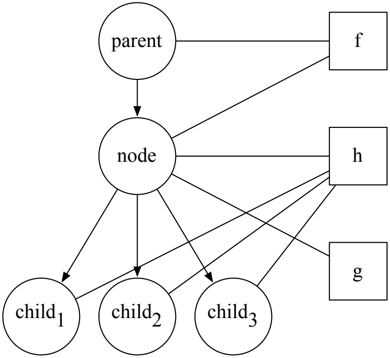
And our answer was the SRT

During inference: node propagation scheme guarantees coherent paths.

- Changing a node to inert makes all its **descendants** inert.
- Changing a node to flow makes all its **ancestors** flow.



Finally, we need a learning algorithm.



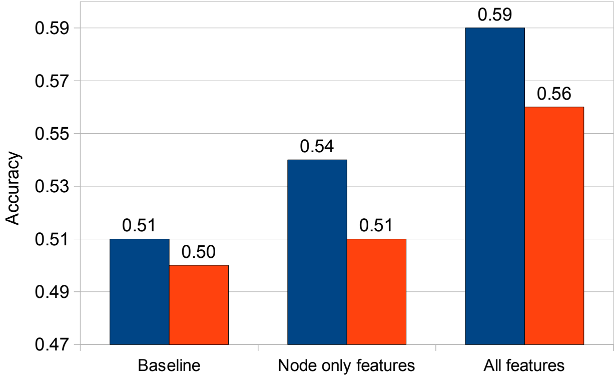
- Scoring function per node in-edge:

$$\text{score}(\text{label}) = \prod_{\phi \in \text{Feat}} f(\text{parent}_\phi, \text{node}_\phi) g(\text{node}_\phi) h(\text{node}_\phi \text{child1}_\phi \dots \text{child}n_\phi)$$

- Features include POS tag, role (in-edge dep. label), word.
- Gibbs sampling.
- Implemented in FACTORIE (UMass Amherst).

And, as usual, does it work?

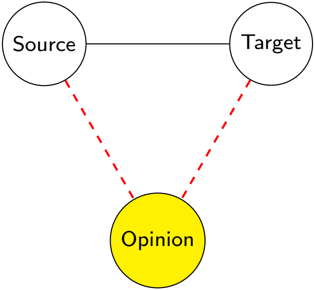
- Objective is retrieving flow labels: highest accuracy required for correct path classification.
- Some highlights of labelwise performance (mean avg 10 runs, many more results in paper):



(But, you'll notice, no targets were actually extracted.)

(This is what we call, in the business, “future work.”)

But there's at least one thing we need to come back to.



That's actually inferring polarity.

That requires some amount of semantic compositionality. . .

. . . if we want to do better than PMI/bag-of-words.
Compositional-distributional semantics is a major recent trend.

Distributional hypothesis

"If two words tend to occur in similar contexts, we can assume they are similar in meaning."

This can be implemented as vector space models.

- Words represented as vectors of statistically-induced contextual features.
- Semantic composition operations via matrix algebra.
    - **Big question**: what algebraic operations?

Whatever helps us calculate up the tree.

Socher et al. (2012)

- Matrix operations for composition need to preserve the dimensionality of the matrix.
- Otherwise, you run out of dimensions!
- Need a function to restore the dimensionality after composition:
  - They propose "Matrix-vector recursive neural networks" (MV-RNN).

What does it look like?

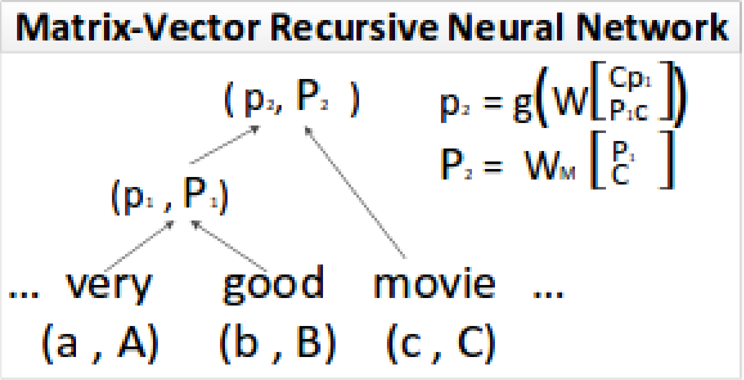


Figure 2: Example of how the MV-RNN merges a phrase with another word at a nonterminal node of a parse tree.

Does it work?

They evaluate on movie review ratings. 10,000 pos/neg sentence extracted from reviews.

Method	Acc.
Tree-CRF (Nakagawa et al., 2010)	77.3
RAE (Socher et al., 2011c)	77.7
Linear MVR	77.1
MV-RNN	**79.0**

Table 1: Accuracy of classification on full length movie review polarity (MR).

And I could literature-review onwards from there, but all good things come to an end. However. . .

Q: What does this have to do with our dystopian future?

A: Consider what we've been doing.

We're talking about increasingly rich formalisms and powerful systems . . .

. . . that infer subtle psychological features . . .

. . . from subtle linguistic cues . . .

. . . in a world where there are huge incentives . . .

. . . to make use of behavioural information.

You do the math.

(No seriously, you'll be doing the math.)