Semantic Grammar

CSCI-GA.2590 – Lecture 5B

Ralph Grishman
Finishing the Pipeline

• name tagger

• semantic patterns for information extraction
Name Tagging

• names play an important role in IE, so we need to be able to identify and classify names

• a big name dictionary will help (e.g., from Wikipedia), but we can’t just use a dictionary ... there will be many names we have not seen before
  – Fred Motelybush
  – Association of Snow Shovelers
Patterns for Name Tagging

• person:
  – title cap+ (Mr. Ziperah Schindler)
  – common-first-name cap+ (Fred Schindler)
  – cap+ verb-with-human-subject (Ziperah Schindler believes)

• organization
  – cap+ corporate suffix (Amalgamated Nonesense Inc.)

• location
  – cap+ , state (Newark, New Jersey)
Corpus-trained Name Taggers

Name tagging is also a sequence tagging task ... another opportunity to use an HMM (or TBL)

(lots of training data available for many languages ... a widely used testbed for trying out new sequence taggers)
A simple HMM for name tagging with 2 name types

1 state per name type

context information must be provided by conditioning transition probabilities on tokens
Jet name tagger: HMM structure for each name type

(simplified) state configuration for each name type $T$
pre and post states capture context

```
pre-T ---- i-T ---- post-T
|      | \
|------| \
b-T   m-T   e-T
```

3/3/15

NYU
Semantic Patterns

• We cannot build patterns for larger constituents based on syntactic categories:
  • too much ambiguity
  • we must look for more specific patterns based on 
    *semantic* categories
Appointment Patterns (1)

Goal: extract information on executive hiring ... patterns like

- company "appointed" person "as" position
- company "named" person "as" position
- company "selected" person "as" position
Appointment Patterns (2)

Need to generalize over tenses:

• company ("appointed" | "appoint" | "appoints") person "as" position
Appointment Patterns (3)

More conveniently, we can make use of the \textit{pa} feature assigned by the Jet lexicon, which records the base form of verbs and nouns:

- \textit{company [constit cat=tv pa=[head=appoint]] person "as" position}
Appointment Patterns (4)

Want to also handle *verb groups* such as

- Enron *has appointed* Fred Smith as treasurer for the day.
- Enron *will appoint* Fred Smith as comptroller.
Appointment Patterns (5)

We can do this by defining a verb group for each verb:

\[
\begin{align*}
\text{vg-appoint} & := [\text{constit cat}=\text{tv} \ pa=[\text{head}=\text{appoint}] ] \mid \\
& \quad [\text{constit cat}=w] \ \text{vg-inf-appoint} \mid \text{tv-vbe} \ \text{vg-ving-appoint}; \\
\text{vg-inf-appoint} & := [\text{constit cat}=v \ pa=[\text{head}=\text{appoint}] ] \mid \\
& \quad "\text{be}" \ \text{vg-ving-appoint}; \\
\text{vg-ving-appoint} & := [\text{constit cat}=\text{ving} \ pa=[\text{head}=\text{appoint}] ]; \\
\text{when vg-appoint add} & \ [\text{constit cat}=\text{vgroup-appoint}];
\end{align*}
\]
Appointment Patterns (6)

It is much more efficient to define a single verb group with a variable feature value which is bound and later used

\[
vg := \begin{cases}
\text{[constit cat=tv pa=PA-verb]} \\
\text{[constit cat=w] vg-inf | tv-vbe vg-ving;}
\end{cases}
\]

\[
vg-inf := \begin{cases}
\text{[constit cat=v pa=PA-verb]} \\
"be" \ vg-ving;
\end{cases}
\]

\[
vg-ving := \begin{cases}
\text{[constit cat=ving pa=PA-verb]}
\end{cases}
\]

when vg add [constit cat=vgroup pa=PA-verb];
Appointment Patterns (7)

Still inconvenient to write a separate pattern for each word ...

• [constit cat=vgroup pa=[head=appoint]] | [constit cat=vgroup pa=[head=name]] | etc
Appointment Patterns (8)

We can define an appointment concept and associate it with a set of appointment words:

\[
[\text{constit cat=vgroup pa=[head?isa(cAppointment)]}]
\]
Patterns for Appointment Events (1)

appoint:= appoint-act | appoint-pass | appoint-nom;

// pattern for active verb phrase: appointed <person> as <position>
appoint-act:= [constit cat=vgroup pa=[head?isa(cAppoint)]]
  [constit cat=ngroup]:Person
  ("as" | "to" [constit cat=ngroup pa=[head=position]] "of" | "to"
    "become")
  [constit cat=ngroup]:Position;

// pattern for passive clause: <person> was appointed as <position>
appoint-pass:= [constit cat=ngroup]:Person
  [constit cat=vgroup-pass pa=[head?isa(cAppoint)]]
  ("as" | "to" [constit cat=ngroup pa=[head=position]] "of" | "to" "become")
  [constit cat=ngroup]:Position;
Patterns for Appointment Events (2)

// pattern for nominalization: appointment of <person> as <position>
appoint-nom:=[constit cat=ngroup pa=[head?isa(cAppointment)]] "of"
   [constit cat=ngroup]:Person
   ("as" | "to" [constit cat=ngroup pa=[head=position]] "of" | "to"
   "become" | "to") [constit cat=ngroup]:Position;

// write out person and position to standard output
when appoint  write "Appointed " + Person + " as " + Position;