CS 4650/7650, Lecture 22
Information Extraction

Jacob Eisenstein

November 12, 2013

1 The Information Extraction pipeline

- Unstructured source: At a meeting of the Thirteen, Pyat Pree tells Daenerys he has her dragons in the House of the Undying.
- Annotated entities: At a meeting of <ORG>the Thirteen</ORG>, <PER>Pyat Pree</PER> tells <PER>Daenerys</PER> that he has <OBJ>her dragons</OBJ> in the <PER>House of the Undying</PER>.
- Linked entities:
  - <PER>Pyat Pree</PER> → Pyat Pree
  - <PER>Daenerys</PER> → Daenerys Targaryen
- Relations:
  - Pyat Pree <HAS> Dragons
  - Dragons <LOCATED-IN> House of the Undying
- Events:
  Possession: [Object: dragons; Location: House of the Undying; Possessor: Pyat Pree]

2 Entity labeling

Pyat/B-PER Pree/I-PER tells/O Daenerys/B-PER that/O he/O has/O her/B-OBJ dragons/I-OBJ ...

- **Tags**: B,I,O for each entity type

- **Features**: bag-of-words, word shape (characters), dictionary (list of known names), part-of-speech...
- **Method**: sequence labeling

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} \sum_i \boldsymbol{w}^\top \boldsymbol{f}(\boldsymbol{x}, y_i, y_{i-1}, i)$$

  - Hidden Markov Model: $\boldsymbol{w} = \arg\max_{\boldsymbol{w}} P(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w})$
  - Conditional Random Field: $\boldsymbol{w} = \arg\max_{\boldsymbol{w}} P(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{w})$
  - Structured Perceptron: $\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} + \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}) - \boldsymbol{f}(\boldsymbol{x}, \hat{\boldsymbol{y}})$

Dictionaries may contain **multitoken spans** (e.g. *The House of the Undying*)

- So we want features that can fire when a span matches a dictionary entry. ($\boldsymbol{f}(\boldsymbol{x}, y_i, y_{i'}, i', i)$: set of features for the span from $i' + 1$ to i)
- Can we still use Viterbi?
- Can we still use dynamic programming?

$$V(i|y) = \begin{cases} \max_{y'} \max_{i' \in i-L,\ldots,i-1} V(i'|y') + \boldsymbol{w}^\top \boldsymbol{f}(\boldsymbol{x}, y_i, y_{i'}, i', i), & i > 0 \\ 0, & i = 0 \\ -\infty, & i < 0 \end{cases}$$

- What is the complxity? $\mathcal{O}(nLm^2)$, with $n = \#|\boldsymbol{x}|, m = \#|\mathcal{Y}|, L =$ max span

3 Entity linking

Entity linking is typically performed as a process of

1. Identifying **candidate** entities for each name, e.g. for *Washington*, the set George Washington, Washington, DC, ...
2. Ranking the candidates and selecting one...
3. ... or, selecting NIL, indicating a mention that is not represented in the KB.

Rao, McNamee, and Dredze (2010) propose an SVM ranking approach,

$$
\begin{aligned}
\boldsymbol{w} =& \arg\min_{\boldsymbol{w}} ||\boldsymbol{w}||_2^2 \\
& s.t. \boldsymbol{w}^\top f(\boldsymbol{x}_i, y_i) > \max_{\hat{y} \neq y_i} \boldsymbol{w}^\top f(\boldsymbol{x}_i, \hat{y}),
\end{aligned}
$$

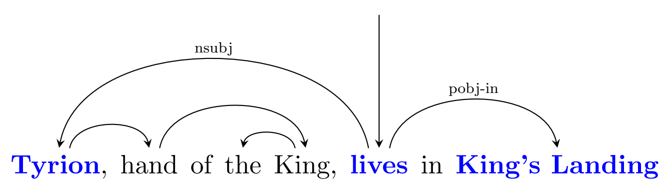
ensuring that the correct entity y_i gets higher rank than all alternative candidates $\hat{y} \in \mathcal{Y}(\boldsymbol{x}, i)$.

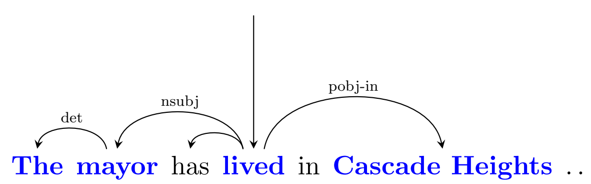
Features (from Dredze et al COLING 2010)

- **String match**: head match, edit distance, alias lists, finite state matchers
- **Wikipedia features**: in- and out-degrees in wikipedia graph,
- **Popularity**: pagerank of entity's wikipedia page
- **Entity type**: does the span type (PER, GPE, LOC) match the entity?

We can also attempt **collective** entity linking. In a document that is certain to mention Boston (City), the string *Washington* is likely to refer to the city and not the person. But in a document that mentions Hamilton (Person), *Washington* would likely refer to the person.

4 Kernels for relation classification





- These trees are intuitively similar, but defining features that capture all their similarities and differences would be a nuisance.
- Instead, we will define a **kernel function** $K((\boldsymbol{x}_i, \boldsymbol{x}_j))$ that quantifies their similarity.

- A Convolutional Tree Kernel (Moschitti 2006) scores pairs of examples by their number of shared substructures.
- $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is large if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are similar.
- The matrix $\mathbf{K}$ of kernel scores between all training instances is called the Gram matrix.
- We can then predict $\hat{y}(\boldsymbol{x}) = b + \sum_i \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i)$, with $\alpha_i \geq 0$.
- The parameters α_i are **dual variables** that result from an alternative (but equivalent) formulation of hinge-loss minimization!
- We only require that the Gram matrix be positive semidefinite.
- In practice, you can define your own kernel functions and use them in SVM-Light (`svmlight.joachims.org`); for Tree kernels, see `disi.unitn.it/moschitti/Tree-Kernel.htm`

5 Event detection

- **Relations** are predications involving two arguments.
- **Events** are predications involving arbitrary numbers of arguments.

Event semantics can be represented in FOL using neo-Davidsonian event representation (see Jurafsky and Martin page 566):

$$\begin{aligned}\exists e.\textsc{Possession}(e), &\textsc{Object}(e, \textsc{Dragons}),\\ &\textsc{Possessor}(e, \textsc{Pyat Pree}),\\ &\textsc{Location}(e, \textsc{House of the Undying})\end{aligned}$$

As shown in the example, events can involve relations between other events, such as the reporting of a bombing.