CS388: Natural Language Processing
Lecture 4: Sequence Models I

Parts of this lecture adapted from Dan Klein, UC Berkeley
and Vivek Srikumar, University of Utah
Mini 1 due today

Project 1 out today, due September 27

- Viterbi algorithm, CRF NER system, extension
- Extension should be substantial: don’t just try one additional feature (see syllabus/spec for discussion, samples on website)
- This class will cover what you need to get started on it, the next class will cover everything you need to complete it
Recall: Multiclass Classification

- Logistic regression: \( P(y|x) = \frac{\exp (w^\top f(x, y))}{\sum_{y' \in \mathcal{Y}} \exp (w^\top f(x, y'))} \)

  Gradient (unregularized): \( \frac{\partial}{\partial w_i} \mathcal{L}(x_j, y_j^*) = f_i(x_j, y_j^*) - \mathbb{E}_y [f_i(x_j, y)] \)

- SVM: defined by quadratic program (minimization, so gradients are flipped)

  Loss-augmented decode

  \( \xi_j = \max_{y \in \mathcal{Y}} w^\top f(x_j, y) + \ell(y, y_j^*) - w^\top f(x_j, y_j^*) \)

  Subgradient (unregularized) on \( j \)th example = \( f_i(x_j, y_{\text{max}}) - f_i(x_j, y_j^*) \)
Recall: Optimization

- Stochastic gradient *ascent*
  
  \[ w \leftarrow w + \alpha g, \quad g = \frac{\partial}{\partial w} \mathcal{L} \]

- Adagrad:

  \[ w_i \leftarrow w_i + \alpha \frac{1}{\sqrt{\epsilon + \sum_{\tau=1}^{t} g_{\tau,i}^2}} g_{t,i} \]

- SGD/AdaGrad have a batch size parameter
  - Large batches (>50 examples): can parallelize within batch
  - ...but bigger batches often means more epochs required because you make fewer parameter updates

- Shuffling: online methods are sensitive to dataset order, shuffling helps!
This Lecture

- Sequence modeling
- HMMs for POS tagging
- HMM parameter estimation
- Viterbi, forward-backward
Linguistic Structures

- Language is tree-structured

Understanding syntax fundamentally requires trees — the sentences have the same shallow analysis

I ate the spaghetti with chopsticks

I ate the spaghetti with meatballs

PRP VBZ DT NN IN NNS PRP VBZ DT NN IN NNS

I ate the spaghetti with chopsticks I ate the spaghetti with meatballs
Linguistic Structures

- Language is sequentially structured: interpreted in an online way

Tanenhaus et al. (1995)
What tags are out there?

Ghana’s ambassador should have set up the big meeting in DC yesterday.
POS Tagging

Open class (lexical) words

Nouns
- Proper: IBM, Italy
- Common: cat / cats, snow

Verbs
- Main: see, registered

Adjectives: yellow

Adverbs: slowly

Closed class (functional)

Determiners: the, some

Conjunctions: and, or

Auxiliary: can, had

Numbers: 122,312, one

Prepositions: to, with

Particles: off, up

Additional open class words: ... more

Additional closed class words: ... more

Slide credit: Dan Klein
Other paths are also plausible but even more semantically weird...

What governs the correct choice? Word + context

- Word identity: most words have <=2 tags, many have one (*percent, the*)
- Context: nouns start sentences, nouns follow verbs, etc.
What is this good for?

- Text-to-speech: record, lead
- Preprocessing step for syntactic parsers
- Domain-independent disambiguation for other tasks
- (Very) shallow information extraction
Sequence Models

- Input $x = (x_1, \ldots, x_n)$  Output $y = (y_1, \ldots, y_n)$

- POS tagging: $x$ is a sequence of words, $y$ is a sequence of tags

- Today: generative models $P(x, y)$; discriminative models next time
Hidden Markov Models

- Input $x = (x_1, \ldots, x_n)$  Output $y = (y_1, \ldots, y_n)$

- Model the sequence of $y$ as a Markov process (dynamics model)

- Markov property: future is conditionally independent of the past given the present

- Lots of mathematical theory about how Markov chains behave

- If $y$ are tags, this roughly corresponds to assuming that the next tag only depends on the current tag, not anything before
Hidden Markov Models

- Input $x = (x_1, ..., x_n)$  
  Output $y = (y_1, ..., y_n)$

- Observation ($x$) depends only on current state ($y$)
- Multinomials: tag $x$ tag transitions, tag $x$ word emissions
- $P(x | y)$ is a distribution over all words in the vocabulary — not a distribution over features (but could be!)

\[ P(y, x) = P(y_1) \prod_{i=2}^{n} P(y_i | y_{i-1}) \prod_{i=1}^{n} P(x_i | y_i) \]

Initial distribution  
Transition probabilities  
Emission probabilities
Transitions in POS Tagging

- Dynamics model: \( P(y_1) \prod_{i=2}^{n} P(y_i|y_{i-1}) \)

Fed raises interest rates 0.5 percent

- \( P(y_1 = \text{NNP}) \) likely because start of sentence
- \( P(y_2 = \text{VBZ}|y_1 = \text{NNP}) \) likely because verb often follows noun
- \( P(y_3 = \text{NN}|y_2 = \text{VBZ}) \) direct object follows verb, other verb rarely follows past tense verb (main verbs can follow modals though!)
Estimating Transitions

Fed raises interest rates 0.5 percent.

- Similar to Naive Bayes estimation: maximum likelihood solution = normalized counts (with smoothing) read off supervised data

- \[ P(\text{tag} \mid \text{NN}) = (0.5, 0.5 \text{ NNS}) \]

- How to smooth?

- One method: smooth with unigram distribution over tags

\[ P(\text{tag} \mid \text{tag}_{-1}) = (1 - \lambda) \hat{P}(\text{tag} \mid \text{tag}_{-1}) + \lambda \hat{P}(\text{tag}) \]

\[ \hat{P} = \text{empirical distribution (read off from data)} \]
Emissions in POS Tagging

Emissions $P(x \mid y)$ capture the distribution of words occurring with a given tag

$P(\text{word} \mid \text{NN}) = (0.05 \text{ person}, 0.04 \text{ official}, 0.03 \text{ interest}, 0.03 \text{ percent} ...)$

When you compute the posterior for a given word’s tags, the distribution favors tags that are more likely to generate that word

How should we smooth this?
Inference in HMMs

- Input \( x = (x_1, \ldots, x_n) \)  
- Output \( y = (y_1, \ldots, y_n) \)

\[
P(y, x) = P(y_1) \prod_{i=2}^{n} P(y_i | y_{i-1}) \prod_{i=1}^{n} P(x_i | y_i)
\]

- Inference problem: \( \arg \max_y P(y | x) = \arg \max_y \frac{P(y, x)}{P(x)} \)

- Exponentially many possible \( y \) here!

- Solution: dynamic programming (possible because of Markov structure!)

- Many neural sequence models depend on entire previous tag sequence, need to use approximations like beam search
\[
P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots, y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)
\]

\[
\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)
\]

Transition probabilities  
Emission probabilities  
Initial probability
\[ P(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i) \]

\[ \max_{y_1, y_2, \ldots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1) \]

\[ = \max_{y_2, \ldots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1) \]

The only terms that depend on \( y_1 \)

---

Viterbi Algorithm

slide credit: Vivek Srikumar
The Viterbi Algorithm is used to find the best (partial) score for a sequence ending in state $s$. The score can be abstracted away for all decisions till here into $\text{score}_1(s)$.

The formula for the Viterbi Algorithm is:

$$P(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i)$$

$$\max_{y_1, y_2, \ldots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

$$= \max_{y_2, \ldots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)$$

$$= \max_{y_2, \ldots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)\text{score}_1(y_1)$$

Best (partial) score for a sequence ending in state $s$

$$\text{score}_1(s) = P(s)P(x_1|s)$$
\[ P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^{n} P(x_i|y_i) \]

\[
\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)
\]

\[
= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)
\]

\[
= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)^{\text{score}_1(y_1)}
\]

\[
= \max_{y_3, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2)P(x_3|y_3) \max_{y_1} P(y_2|y_1)P(x_2|y_2)^{\text{score}_1(y_1)}
\]

Only terms that depend on \( y_2 \)

slide credit: Vivek Srikumar
\[ P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1} | y_i) \prod_{i=1}^{n} P(x_i | y_i) \]

\[
\max_{y_1, y_2, \cdots, y_n} P(y_n | y_{n-1}) P(x_n | y_n) \cdots P(y_2 | y_1) P(x_2 | y_2) P(y_1) P(x_1 | y_1) \\
= \max_{y_2, \cdots, y_n} P(y_n | y_{n-1}) P(x_n | y_n) \cdots \max_{y_1} P(y_2 | y_1) P(x_2 | y_2) P(y_1) P(x_1 | y_1) \\
= \max_{y_2, \cdots, y_n} P(y_n | y_{n-1}) P(x_n | y_n) \cdots \max_{y_2} P(y_3 | y_2) P(x_3 | y_3) \max_{y_1} P(y_2 | y_1) P(x_2 | y_2) P(y_1) P(x_1 | y_1) \\
= \max_{y_2, \cdots, y_n} P(y_n | y_{n-1}) P(x_n | y_n) \cdots \max_{y_2} P(y_3 | y_2) P(x_3 | y_3) \text{score}_1(y_1) \\
\text{score}_i(s) = \max_{y_{i-1}} P(s | y_{i-1}) P(x_i | s) \text{score}_{i-1}(y_{i-1})
\]

Abstract away the score for all decisions till here into score

slide credit: Vivek Srikumar
“Think about” all possible immediate prior state values. Everything before that has already been accounted for by earlier stages.
\[ P(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1}|y_i) \prod_{i=1}^n P(x_i|y_i) \]

\[
\max_{y_1, y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)
\]

\[
= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_2|y_1)P(x_2|y_2)P(y_1)P(x_1|y_1)
\]

\[
= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_1} P(y_3|y_2)P(x_3|y_3) \max_{y_2} P(y_2|y_1)P(x_2|y_2)\text{score}_1(y_1)
\]

\[
= \max_{y_2, \cdots, y_n} P(y_n|y_{n-1})P(x_n|y_n) \cdots \max_{y_2} P(y_3|y_2)P(x_3|y_3)\text{score}_2(y_2)
\]

\[
\vdots
\]

\[
= \max_{y_n} \text{score}_n(y_n)
\]

Abstract away the score for all decisions till here into \text{score}
\[ P(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots y_n) = P(y_1) \prod_{i=1}^{n-1} P(y_{i+1} | y_i) \prod_{i=1}^{n} P(x_i | y_i) \]

\[
\begin{align*}
\max_{y_1, y_2, \ldots, y_n} & \quad P(y_n | y_{n-1}) P(x_n | y_n) \cdots \max_{y_1} P(y_2 | y_1) P(x_2 | y_2) P(y_1) P(x_1 | y_1) \\
= & \quad \max_{y_2, \ldots, y_n} P(y_n | y_{n-1}) P(x_n | y_n) \cdots \max_{y_1} P(y_2 | y_1) P(x_2 | y_2) \text{score}_1(y_1) \\
= & \quad \max_{y_3, \ldots, y_n} P(y_n | y_{n-1}) P(x_n | y_n) \cdots \max_{y_2} P(y_3 | y_2) P(x_3 | y_3) \max_{y_1} P(y_2 | y_1) P(x_2 | y_2) \text{score}_1(y_1) \\
= & \quad \max_{y_3, \ldots, y_n} P(y_n | y_{n-1}) P(x_n | y_n) \cdots \max_{y_2} P(y_3 | y_2) P(x_3 | y_3) \text{score}_2(y_2) \\
\vdots \\
= & \quad \max_{y_n} \text{score}_n(y_n)
\end{align*}
\]

\[
\text{score}_1(s) = P(s) P(x_1 | s)
\]

\[
\text{score}_i(s) = \max_{y_{i-1}} P(s | y_{i-1}) P(x_i | s) \text{score}_{i-1}(y_{i-1})
\]

slide credit: Vivek Srikumar
1. **Initial**: For each state $s$, calculate

$$score_1(s) = P(s)P(x_1|s) = \pi_s B_{x_1,s}$$

2. **Recurrence**: For $i = 2$ to $n$, for every state $s$, calculate

$$score_i(s) = \max_{y_{i-1}} P(s|y_{i-1})P(x_i|s) score_{i-1}(y_{i-1})$$

$$= \max_{y_{i-1}} A_{y_{i-1},s} B_{s,x_i} score_{i-1}(y_{i-1})$$

3. **Final state**: calculate

$$\max_y P(y, x|\pi, A, B) = \max_s score_n(s)$$

This only calculates the max. To get final answer (argmax),
- keep track of which state corresponds to the max at each step
- build the answer using these back pointers
Forward-Backward Algorithm

- In addition to finding the best path, we may want to compute marginal probabilities of paths $P(y_i = s | x)$

$$P(y_i = s | x) = \sum_{y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_n} P(y | x)$$

- What did Viterbi compute? $P(y_{\text{max}} | x) = \max_{y_1, \ldots, y_n} P(y | x)$

- Can compute marginals with dynamic programming as well using an algorithm called forward-backward
Forward-Backward Algorithm

\[ P(y_3 = 2|x) = \frac{\text{sum of all paths through state 2 at time 3}}{\text{sum of all paths}} \]
Forward-Backward Algorithm

\[ P(y_3 = 2|x) = \]

\[ \frac{\text{sum of all paths through state 2 at time 3}}{\text{sum of all paths}} = \]

\[ \begin{array}{c}
\text{Easiest and most flexible to do one pass to compute and one to compute}
\end{array} \]
Forward-Backward Algorithm

- Initial:
  \[ \alpha_1(s) = P(s)P(x_1|s) \]

- Recurrence:
  \[ \alpha_t(s_t) = \sum_{s_{t-1}} \alpha_{t-1}(s_{t-1})P(s_t|s_{t-1})P(x_t|s_t) \]

- Same as Viterbi but summing instead of maxing!

- These quantities get very small!
  Store everything as log probabilities
Forward-Backward Algorithm

- Initial:
  \[ \beta_n(s) = 1 \]

- Recurrence:
  \[ \beta_t(s_t) = \sum_{s_{t+1}} \beta_{t+1}(s_{t+1}) P(s_{t+1}|s_t) P(x_{t+1}|s_{t+1}) \]

- Big differences: count emission for the next timestep (not current one)
Forward-Backward Algorithm

\(\alpha_1(s) = P(s)P(x_1|s)\)

\(\alpha_t(s_t) = \sum_{s_{t-1}} \alpha_{t-1}(s_{t-1})P(s_t|s_{t-1})P(x_t|s_t)\)

\(\beta_n(s) = 1\)

\(\beta_t(s_t) = \sum_{s_{t+1}} \beta_{t+1}(s_{t+1})P(s_{t+1}|s_t)P(x_{t+1}|s_{t+1})\)

\[P(s_3 = 2|\mathbf{x}) = \frac{\alpha_3(2)\beta_3(2)}{\sum_i \alpha_3(i)\beta_3(i)}\]

- Does this explain why \(\beta\) is what it is?
- What is the denominator here? \(P(\mathbf{x})\)
HMM POS Tagging

- Baseline: assign each word its most frequent tag: ~90% accuracy
- Trigram HMM: ~95% accuracy / 55% on unknown words
Trigram Taggers

NNP  VBZ  NN  NNS  CD  NN
Fed raises interest rates 0.5 percent

- Trigram model: $y_1 = (<S>, \text{NNP}), \ y_2 = (\text{NNP}, \text{VBZ}), \ldots$
- $P((\text{VBZ}, \text{NN}) \mid (\text{NNP}, \text{VBZ}))$ — more context! Noun-verb-noun S-V-O
- Tradeoff between model capacity and data size — trigrams are a “sweet spot” for POS tagging
HMM POS Tagging

- Baseline: assign each word its most frequent tag: ~90% accuracy
- Trigram HMM: ~95% accuracy / 55% on unknown words
- TnT tagger (Brants 1998, tuned HMM): 96.2% accuracy / 86.0% on unks
- State-of-the-art (BiLSTM-CRFs): 97.5% / 89%+

Slide credit: Dan Klein
Errors

<table>
<thead>
<tr>
<th>JJ</th>
<th>NN</th>
<th>NNP</th>
<th>NNPS</th>
<th>RB</th>
<th>RP</th>
<th>IN</th>
<th>VB</th>
<th>VBD</th>
<th>VBN</th>
<th>VBP</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>JJ</td>
<td>0</td>
<td>177</td>
<td>56</td>
<td>0</td>
<td>61</td>
<td>2</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>108</td>
<td>0</td>
</tr>
<tr>
<td>NN</td>
<td>244</td>
<td>0</td>
<td>103</td>
<td>0</td>
<td>12</td>
<td>1</td>
<td>1</td>
<td>29</td>
<td>5</td>
<td>6</td>
<td>19</td>
</tr>
<tr>
<td>NNP</td>
<td>107</td>
<td>106</td>
<td>0</td>
<td>132</td>
<td>5</td>
<td>0</td>
<td>7</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>NNPS</td>
<td>1</td>
<td>0</td>
<td>110</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RB</td>
<td>72</td>
<td>21</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>138</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>39</td>
<td>0</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IN</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>169</td>
<td>103</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VB</td>
<td>17</td>
<td>64</td>
<td>9</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>7</td>
<td>85</td>
</tr>
<tr>
<td>VBD</td>
<td>10</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>143</td>
<td>2</td>
</tr>
<tr>
<td>VBN</td>
<td>101</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>108</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>VBP</td>
<td>5</td>
<td>34</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>49</td>
<td>6</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>626</td>
<td>536</td>
<td>348</td>
<td>144</td>
<td>317</td>
<td>122</td>
<td>279</td>
<td>102</td>
<td>140</td>
<td>269</td>
<td>108</td>
</tr>
</tbody>
</table>

**Slide credit:** Dan Klein / Toutanova + Manning (2000)
Remaining Errors

- Lexicon gap (word not seen with that tag in training) 4.5%
- Unknown word: 4.5%
- Could get right: 16% (many of these involve parsing!)
- Difficult linguistics: 20%

VBD / VBP? (past or present?)

*They* *set* *up* *absurd* *situations*, *detached* *from* *reality*

Underspecified / unclear, gold standard inconsistent / wrong: 58%

adjective or verbal participle? JJ / VBN?

*a* *$10* *million* *fourth-quarter* *charge* *against* *discontinued* *operations*

Manning 2011 “Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics?”
Other Languages

- Universal POS tagset (~12 tags), cross-lingual model works as well as tuned CRF using external resources

<table>
<thead>
<tr>
<th>Language</th>
<th>CRF+</th>
<th>CRF</th>
<th>BTS</th>
<th>BTS*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulgarian</td>
<td>97.97</td>
<td>97.00</td>
<td>97.84</td>
<td>97.02</td>
</tr>
<tr>
<td>Czech</td>
<td>98.38</td>
<td>98.00</td>
<td>98.50</td>
<td>98.44</td>
</tr>
<tr>
<td>Danish</td>
<td>95.93</td>
<td>95.06</td>
<td>95.52</td>
<td>92.45</td>
</tr>
<tr>
<td>German</td>
<td>93.08</td>
<td>91.99</td>
<td>92.87</td>
<td>92.34</td>
</tr>
<tr>
<td>Greek</td>
<td>97.72</td>
<td>97.21</td>
<td>97.39</td>
<td>96.64</td>
</tr>
<tr>
<td>English</td>
<td>95.11</td>
<td>94.51</td>
<td>93.87</td>
<td>94.00</td>
</tr>
<tr>
<td>Spanish</td>
<td>96.08</td>
<td>95.03</td>
<td>95.80</td>
<td>95.26</td>
</tr>
<tr>
<td>Farsi</td>
<td>96.59</td>
<td>96.25</td>
<td>96.82</td>
<td>96.76</td>
</tr>
<tr>
<td>Finnish</td>
<td>94.34</td>
<td>92.82</td>
<td>95.48</td>
<td>96.05</td>
</tr>
<tr>
<td>French</td>
<td>96.00</td>
<td>95.93</td>
<td>95.75</td>
<td>95.17</td>
</tr>
<tr>
<td>Indonesian</td>
<td>92.84</td>
<td>92.71</td>
<td>92.85</td>
<td>91.03</td>
</tr>
<tr>
<td>Italian</td>
<td>97.70</td>
<td>97.61</td>
<td>97.56</td>
<td>97.40</td>
</tr>
<tr>
<td>Swedish</td>
<td>96.81</td>
<td>96.15</td>
<td>95.57</td>
<td>93.17</td>
</tr>
<tr>
<td>AVERAGE</td>
<td>96.04</td>
<td>95.41</td>
<td>95.85</td>
<td>95.06</td>
</tr>
</tbody>
</table>

Gillick et al. 2016

Óscar Romero was born in El Salvador.
CRFs: feature-based discriminative models

Structured SVM for sequences

Named entity recognition