Implementierung eines POS-Taggers mithilfe von Haskell

Andy Püschel
Till Köhler

Institut für Theoretische Informatik
Fakultät Informatik
Technische Universität Dresden

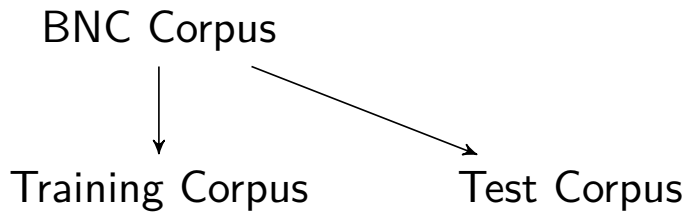
19.01.2018

Outline

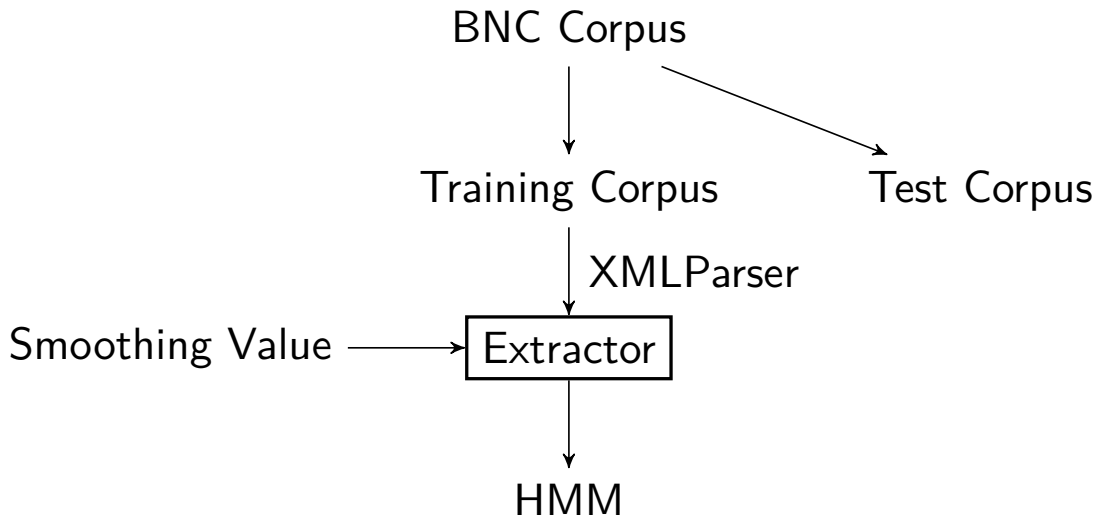
Aufgabenstellung

- Spezifikation einer Haskelldatenstruktur zur Darstellung von Hidden-Markov-Modellen
- Supervised Training eines Hidden-Markov-Modells aus einem mit POS-Tags annotierten Textkorpus
- Serialisierung von Hidden-Markov-Modells aus der Haskelldatenstruktur in menschenlesbares Format
- Deserialisierung zurück in die Haskelldatenstruktur
- Annotation eines beliebigen Textkorpus mit POS-Tags
- Evaluation des gewonnenen Hidden-Markov-Modells

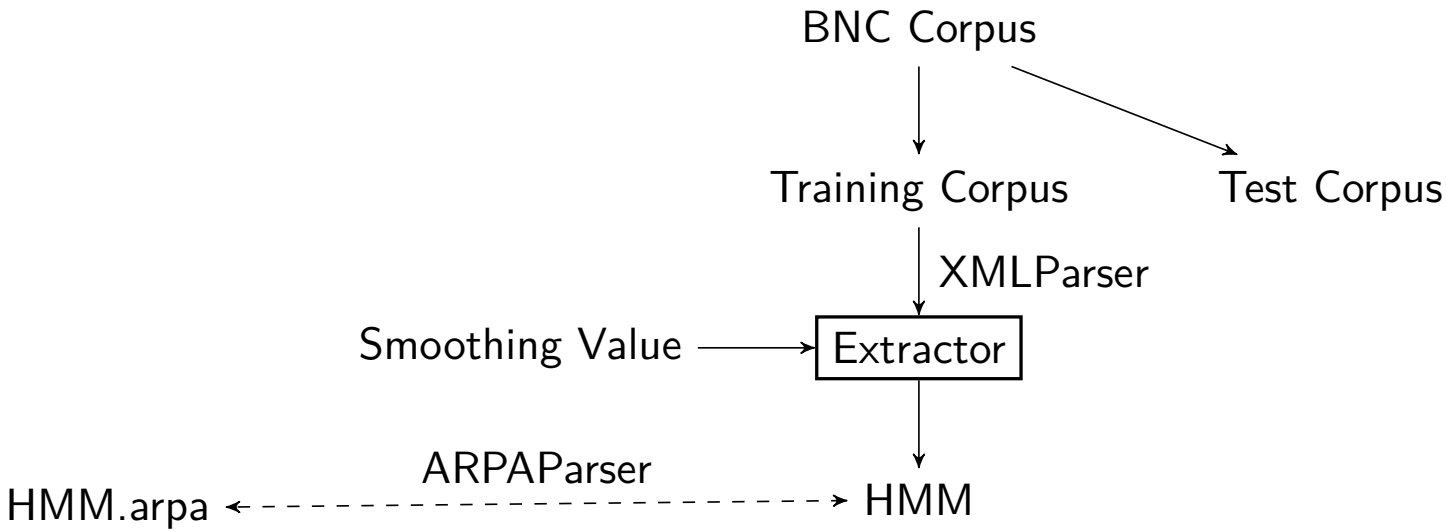
Schematische Darstellung



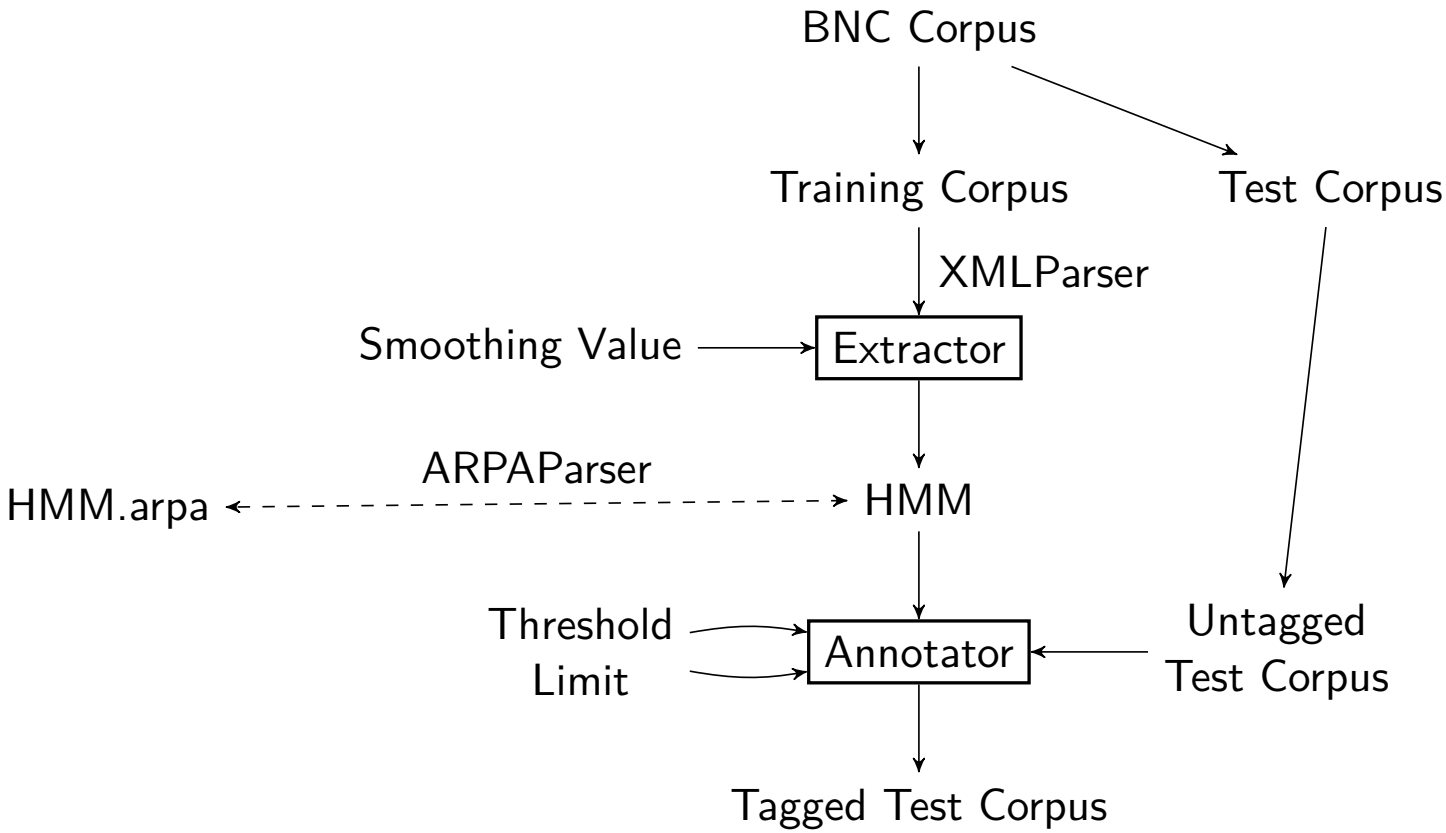
Schematische Darstellung



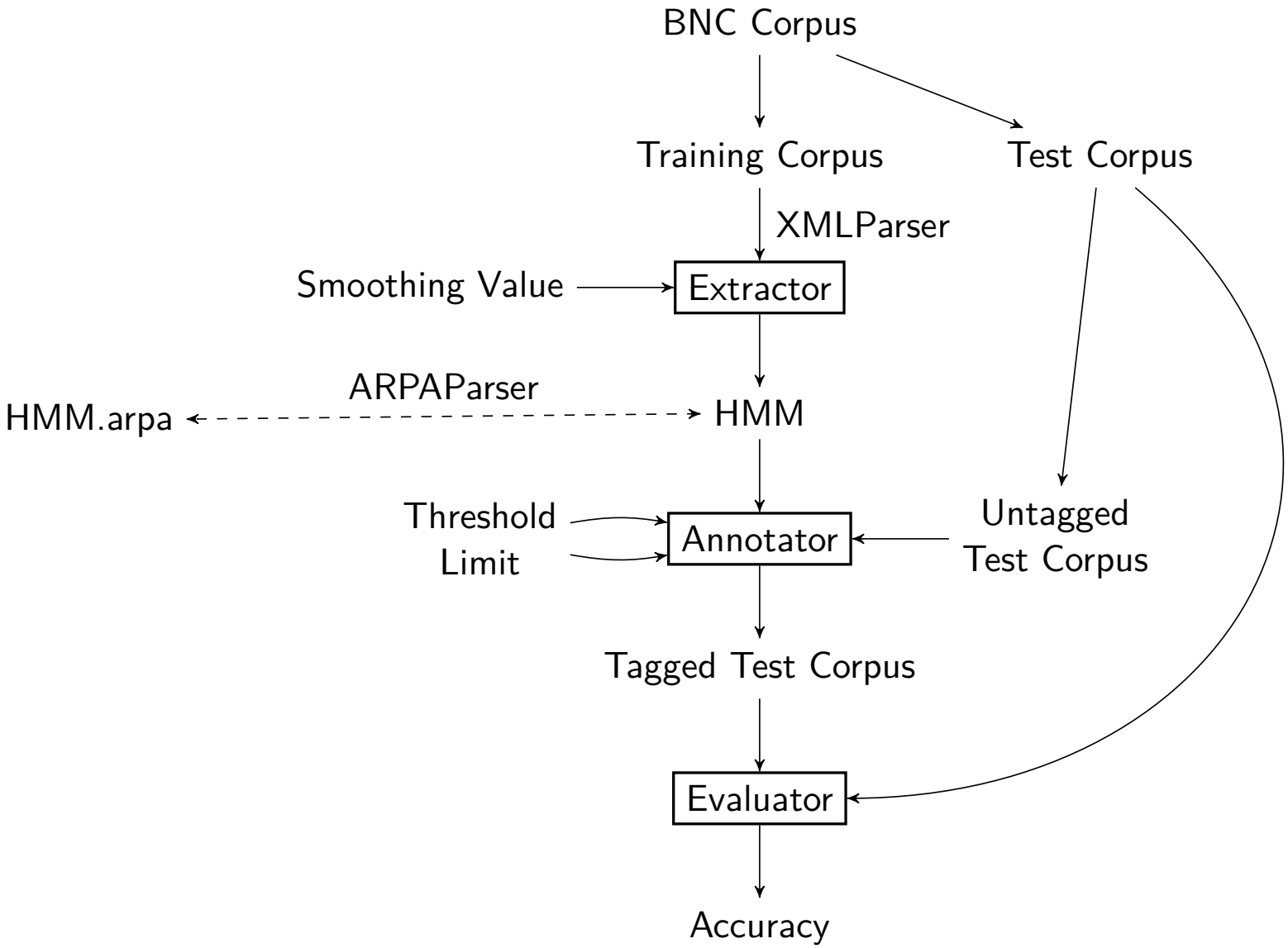
Schematische Darstellung



Schematische Darstellung



Schematische Darstellung



Outline

Modellierung von Hidden-Markov-Modellen

HMM $H = (Q, V, \#, \tau, \varepsilon)$, wobei:

- Q eine endliche Menge von Zuständen (POS-Tags),
- V eine endliche Menge von Beobachtungen (Wörter),
- $\#$ der Startzustand,
- $\tau\colon Q \to Q \to [0, 1]$ die bed. Wsk.-Verteilung der Transitionen,
- $\varepsilon\colon Q \to V \to [0, 1]$ die bed. Wsk.-Verteilung der Emissionen.

Modellierung von Hidden-Markov-Modellen

HMM $H = (Q, V, \#, \tau, \varepsilon)$

```haskell
data HMM =
  HMM { -- | the initial state
        initialState :: POSTag
        -- | the list of states
      , states       :: [POSTag]
        -- | the list of observations
      , observations :: [Word]
        -- | the map of all transitions
      , transitions  :: M.Map (POSTag, POSTag) Prob
        -- | the map of all emissions
      , emissions    :: M.Map (POSTag, Word) Prob
      } deriving (Show)
```

Modellierung von Hidden-Markov-Modellen
ARPA-Format

```
\data\

\initial:
START

\transitions:
4.647932276069019e-2 AJ0 AJ0
...

\emissions:
5.2954882440160984e-5 AJ0 Abdominal
...

\end\
```

Outline

Supervised Training

- Trainingsgegenstand ist ein mit POS-Tags annotierter Textkorpus C (in unserem Fall: BNC im XML-Format)
- Supervised Training mithilfe relativer Häufigkeitsschätzung:

$$\text{Unigramme:} \qquad \tau_1(t_i) = \frac{f(t_i)}{|C|}$$

$$\text{Transitionen (Bigramme):} \qquad \tau_2(t_i \mid t_{i-1}) = \frac{f(t_i, t_{i-1})}{f(t_{i-1})}$$

$$\text{Emissionen:} \qquad \varepsilon(w_i \mid t_i) = \frac{f(w_i, t_i)}{f(t_i)}$$

mit w_j ... Wörter, t_j ... POS-Tags

Lineare Interpolation (Jelinek und Mercer, 1980)

- Gefahr bei kleinen Trainingskorpora: zu geringes Vorkommen für jedes POS-Tag-Bigramm, um verlässiche Wahrscheinlichkeit zu bestimmen
- Bestimmung des Transitionswahrscheinlichkeiten durch lineare Interpolation der zugehörigen POS-Tag-Unigramme und -Bigramme:

$$\tau(t_i, t_{i-1}) = \lambda_1 \tau_1(t_i) + \lambda_2 \tau_2(t_i \mid t_{i-1})$$

mit $\lambda_1 + \lambda_2 = 1$.

Supervised Training mit Smoothing

```haskell
extract content smoothingValue =
  ...
  where hmm = flip compute smoothingValue
              . flip count emptyCounts
              $ XP.parseXML content
```

```haskell
smooth :: Prob   -- ^ the bigram probability
       -> Prob   -- ^ the unigram probability
       -> Double -- ^ the smoothing value
       -> Prob   -- ^ the resulting probability
smooth biprob uniprob sv
  = (1 - sv) * biprob + sv * uniprob
```

Outline

Viterbi-Algorithmus

Input: HMM $H = (Q, V, \#, \tau, \varepsilon)$ und Satz $w = w_1 \ldots w_n$.

Output: Tag-Sequenz $\hat{t} = t_1 \ldots t_n$, sodass:

$$\hat{t} = \text{argmax}_{t_1 \ldots t_n \in T^n} P(\mathbb{W} = w_1 \ldots w_n, \mathbb{T} = t_1 \ldots t_n)$$

mit

$$P(\mathbb{W} = w_1 \ldots w_n, \mathbb{T} = t_1 \ldots t_n) =$$
$$\varepsilon(w_1 \mid t_1) \cdot \tau(t_1 \mid \#) \cdot \prod_{i=2}^{n} \varepsilon(w_i \mid t_i) \cdot \tau(t_i \mid t_{i-1}) \cdot \tau(\# \mid t_n)$$

Beam Search (Rabiner, 1989)

- Grenzwert θ ... gibt den Anteil des Maximalwertes an, der überschritten werden muss
- Limit l ... gibt die maximale Anzahl der Elemente (Beam-Breite) an
- Score δ ... gibt den derzeitigen Score eines Elements an

$$\delta_{new} \geq \frac{\delta_{max}}{\theta}$$

Annotation mit Beam Search

```haskell
annotate :: String        -- ^ the text corpus
         -> HMM           -- ^ the HMM
         -> Maybe Double  -- ^ the threshold
         -> Maybe Int     -- ^ the limit
         -> String        -- ^ the annotated corpus
annotate [] _ _ _     = []
annotate tc hmm th l =
  let (s : ss) = lines tc
      pq       = vInit th l
      rawTags  = vIterate (words s) hmm pq
      fillUp   = map (const "-") ([1 ..] :: [Int])
      tags     = L.drop 1 rawTags ++ fillUp
  in unwords (zipWith (\w t -> w ++ "/" ++ t)
                      (words s) tags)
     ++ "\n" ++ go (unlines ss) hmm th l
```

Unknown Words (Samuelsson, 1993)

Problem: Trainingskorpus enthält unbekannte Wörter

Lösung: Suffix-Analyse $\rightarrow$ Schätzung der Emissionen für das unbekannte Wort:

$$\varepsilon'(t_i \mid u_{l,m}) = \begin{cases} \tau_1(t) & \text{, wenn } |u_{l,m}| = 0 \\ \frac{\varepsilon(t_i|u_{l,m}) + \sigma\varepsilon'(t_i|u_{l+1,m})}{1+\sigma} & \text{, sonst} \end{cases}$$

mit σ ... Standardabweichung von τ_1, t_i ... POS-Tag und $w_j = u_1 \ldots u_m$ ... unbekanntes Wort.

Outline

Evaluation des Modells

- Evaluation des Hidden-Markov-Modells anhand eines Testkorpus mit entfernten POS-Tags
- Vergleich des getaggten Korpus und des originalen Korpus (vor dem Entfernen der POS-Tags):

$$accuracy = \frac{\text{Anzahl übereinstimmender POS-Tags}}{\text{Anzahl aller getaggten Wörter}}$$

Outline

Literaturverzeichnis