Transformation-Based Error-Driven Learning: Eine Fallstudie in Part of Speech Tagging

Malte Helmert

23. Februar 2000

Seminar Autonome Mobile Systeme/Machine Learning

Übersicht

1. Part of Speech Tagging
2. Transformation-Based Error-Driven Learning
3. Die Algorithmen
4. Empirische Ergebnisse
5. Bewertung

Part of Speech Tagging

Gegeben

Ein Text (Folge von Wörtern).

Gesucht

Die zugehörigen Wortarten (*parts of speech*).

Beispiel

This	is	a	simple	example
D.-Pron.	Verb	Artikel	Adjektiv	Substantiv

Parts of Speech

CC	Coordinating Conjunction
CD	Cardinal Number
DT	Determiner
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
MD	Modal
NN	Noun, singular or mass
NNP	Proper noun, singular
NNS	Noun, plural
NNPS	Proper noun, plural
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
. . .	. . .

Schwierigkeit

Mehrdeutige Wörter

- "It was a calm night."
- "He tried to calm the mob."
- "On the third day, the ship came into a calm."

Adjektiv, Verb oder Substantiv?

$\longrightarrow$ Hinweise aus dem lokalen Kontext!

Lösungsansätze

- **Unigram-Tagger**
  Weise jedem Wort das Tag zu, das für dieses Wort am häufigsten auftritt.
- **n-gram-Tagger [1976]**
  Erweiterung auf Gruppen von n aufeinanderfolgenden Wörtern (*n-gramme*):
  Maximiere für jedes Wort w_i das Produkt der Wahrscheinlichkeiten
  $P(w_i|tag_i) \cdot P(tag_i|tag_{i-1} \ldots tag_{i-n})$.
- **Transformation-Based Error-Driven Learning [1995]**
  Lerne an einem (bereits annotierten) Beispieltext eine Liste von *Transformationsregeln* und benutze diese zum Annotieren unbekannter Texte.
- **Maximum Entropy [1996]**
- **Kombinierte Methoden [1998]**

Transformationen

Bestandteile einer *Transformation*:

- **Modifikationsregel**
  $\longrightarrow$ "Ändere das Tag von 'Verb' zu 'Substantiv'."
- **Auslösende Umgebung**
  $\longrightarrow$ "Das Tag des vorhergehenden Wortes ist 'Artikel'."

Anwendung einer Transformation

Text von links nach rechts durchgehen und Modifikationsregel ausführen, wann immer auslösende Umgebung angetroffen wird.

Definition

$W[t]$ ist der annotierte Text, der sich durch Anwendung der Transformation t auf den annotierten Text W ergibt.

Transformationen: Beispiel

- **Modifikationsregel**
  $\longrightarrow$ "Ändere das Tag von 'Verb' zu 'Substantiv'."
- **Auslösende Umgebung**
  $\longrightarrow$ "Das Tag des vorhergehenden Wortes ist 'Artikel'."

Vor der Anwendung (W)

John/Eigenname caught/Verb a/Artikel fly/*Verb*

Nach der Anwendung ($W[t]$)

John/Eigenname caught/Verb a/Artikel fly/*Substantiv*

Tagging-Algorithmus

Eingabe

- Eine Folge von Wörtern W (ohne Tags)
- Eine Folge von Transformationen T

(Abstrakter) Algorithmus

1. Annotiere W mit dem *Startannotator*
   $\longrightarrow$ z.B. trivialer Annotator oder Unigramm-Algorithmus
2. Für jede Transformation $t \in T$:
   $W := W[t]$

Wie findet man eine gute Transformationsfolge?

Lern-Algorithmus

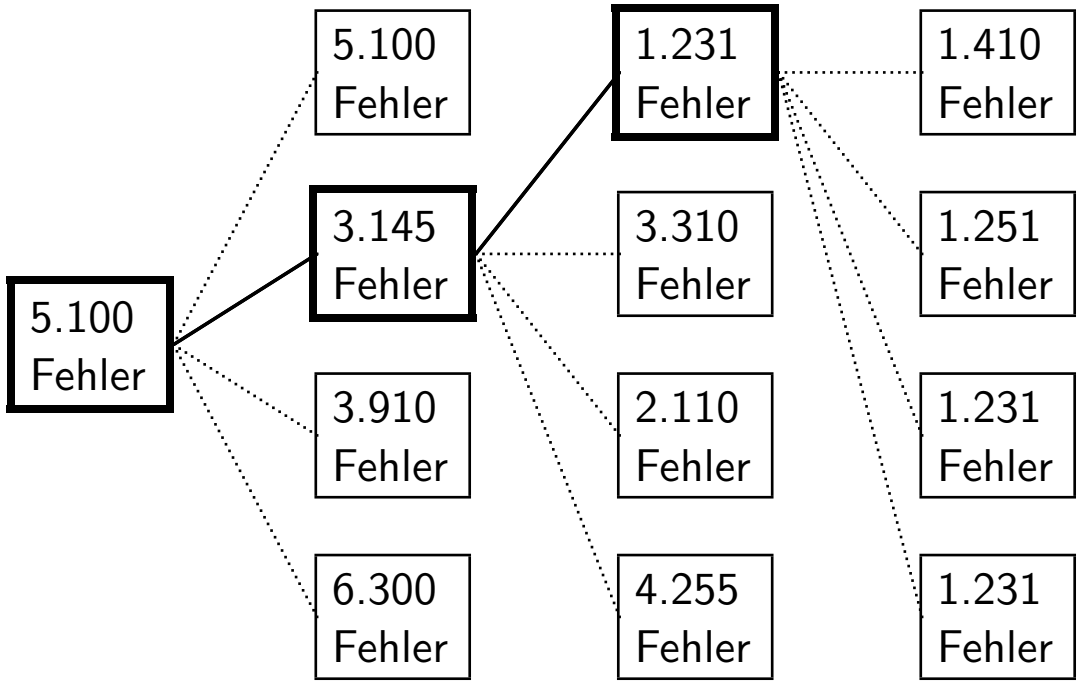
Eingabe

- Eine Folge von Wörtern *mit* Tags W (*Trainingstext*).
- Eine Menge von erlaubten Transformationen T.

Algorithmus

1. $W' := W$ ohne Tags
2. Annotiere W' mit dem Startannotator.
3. $t_0 :=$ ein $t \in T$, das die Fehlerzahl von $W'[t]$ minimiert
   (Fehlerzahl: Anzahl der von W abweichenden Tags).
4. Erweitere die Folge der gelernten Transformationen um t_0.
5. $W' = W'[t_0]$
6. Zurück zu 3., bis keine (wesentliche) Verbesserung mehr eintritt.

Lernen von Transformationen: Beispiel



Nicht lexikalisierte Variante

Erlaubte Transformationsschemata

Ändere beim aktuellen Wort das Tag von A nach B, falls B ein gültiges Tag für das aktuelle Wort ist und. . .

- . . . das vorhergehende (folgende) Wort das Tag Z besitzt.
- . . . das zweite Wort davor (dahinter) das Tag Z besitzt.
- . . . eines der beiden vorhergehenden (folgenden) Wörter das Tag Z besitzt.
- . . . eines der drei vorhergehenden (folgenden) Wörter das Tag Z besitzt.
- . . . das vorhergehende Wort das Tag Z, das folgende Wort das Tag W besitzt.
- . . . das vorhergehende (folgende) Wort das Tag Z und das zweite Wort davor (dahinter) das Tag W besitzt.

Potentiell Millionen von Transformationen
$\longrightarrow$ handhabbar durch datengetriebenen Ansatz.

Gelernte Transformationen

1.	Substantiv, singular → Verb, infinitiv falls Tag davor: to
2.	Verb, 1./2. Ps. sing. präs. → Verb, infinitiv falls unter den drei Tags davor: Modalverb
3.	Substantiv, singular → Verb, infinitiv falls unter den zwei Tags davor: Modalverb
4.	Verb, infinitiv → Substantiv, singular falls unter den zwei Tags davor: Bestimmungswort
5.	Verb, präteritum → Verb, Partizip II falls unter den drei Tags davor: Verb, 3. Ps. sing. präs.
6.	Verb, Partizip II → Verb, präteritum falls Tag davor: Personalpronomen
7.	Verb, Partizip II → Verb, präteritum falls Tag davor: Eigenname, singular
8.	Verb, präteritum → Verb, Partizip II falls Tag davor: Verb, präteritum

Lexikalisierte Variante

Zusätzliche Transformationsschemata

Ändere beim aktuellen Wort das Tag von A nach B, falls B ein gültiges Tag für das aktuelle Wort ist und. . .

- . . . das vorhergehende (folgende) Wort w ist.
- . . . das zweite Wort davor (dahinter) w ist.
- . . . eines der beiden vorhergehenden (folgenden) Wörter w ist.
- . . . das aktuelle Wort w und das vorhergehende (folgende) Wort x ist.
- . . . das aktuelle Wort w ist und das vorhergehende (folgende) Wort das Tag Z besitzt.
- . . . das aktuelle Wort w ist.
- . . . das vorhergende (folgende) Wort w ist und das vorhergehende (folgende) Wort das Tag Z besitzt.
- . . . das aktuelle Wort w ist, das vorhergehende (folgende) Wort w_2 ist und das vorhergehene (folgende) Wort das Tag Z besitzt.

Unbekannte Wörter

Was tun bei **unbekannten Wörtern**?
$\longrightarrow$ Eine weitere Variante des Transformation-Based Error-Driven Learning.

Startannotator für unbekannte Wörter

Eigenname sg. bei Großschreibung; Substantiv sg. sonst.

Transformationsschemata

Ändere beim aktuellen Wort das Tag (von A) nach B, falls. . .

- . . . das Entfernen des Präfix (Suffix) x mit $1 \leq |x| \leq 4$ ein Wort ergibt.
- . . . das Hinzufügen des Präfixes (Suffixes) x mit $1 \leq |x| \leq 4$ ein Wort ergibt.
- . . . x mit $1 \leq |x| \leq 4$ ein Präfix (Suffix) des Wortes ist.
- . . . das Wort w jemals direkt davor (dahinter) auftritt.
- . . . das Zeichen z in dem Wort auftritt.

Gelernte Transformationen

1.	Substantiv, singular $\longrightarrow$ Substantiv, plural falls Suffix: "-s"
2.	Substantiv, singular $\longrightarrow$ Kardinalzahl falls Zeichen '.' enthalten
3.	Substantiv, singular $\longrightarrow$ Adjektiv falls Zeichen '-' enthalten
4.	Substantiv, singular $\longrightarrow$ Verb, Partizip II falls Suffix: "-ed"
5.	Substantiv, singular $\longrightarrow$ Verb, Partizip I falls Suffix: "-ing"
6.	(beliebig) $\longrightarrow$ Adverb falls Suffix: "-ly"
7.	(beliebig) $\longrightarrow$ Adjektiv falls Hinzufügen von "-ly" ein Wort ergibt
8.	Substantiv, singular $\longrightarrow$ Kardinalzahl falls das Wort "$" davor auftreten kann

Effektivität

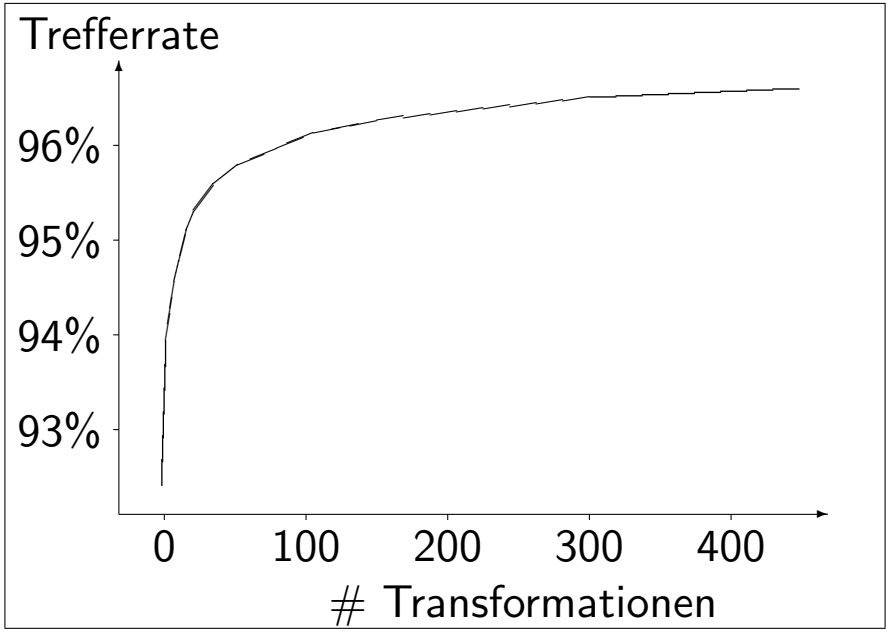
Methode	Trainings-text	# Regeln	Treffer-rate
Trigramm (CVA)	64.000	6.170	96,3%
Trigramm (CVA)	1.000.000	10.000	96,7%
TBEDL, lex. (CVA)	64.000	215	96,7%
TBEDL, lex. (CVA)	600.000	447	97,2%
TBEDL, n. lex. (CVA)	600.000	378	97,0%
TBEDL, lex.	950.000	690	96,6%
TBEDL, n. lex.	950.000	621	96,2%

CVA = *closed vocabulary assumption*

Grundlage: *Penn Treebank Wall Street Journal Corpus*

Korrekt annotierte unbekannte Wörter (lexikalisierte Version): 82,2%.

Transformationszahl vs. Effektivität



lexikalisierte Version, closed vocabulary assumption, 600.000 Wörter Trainingstext

Bewertung

Transformation-Based Error-Driven Learning vs. n-gram-Tagger

+ (leicht) höhere Trefferrate insgesamt
– (leicht) niedrigere Trefferrate bei unbekannten Wörtern
+ (weitgehend) sprachunabhängig
+ kompakte Repräsentation des linguistischen Wissens
+ leicht verständliche, menschenlesbare Regeln
  $\longrightarrow$ auch als Analysewerkzeug verwendbar
+ Tagging-Algorithmus kann wesentlich schneller laufen

Referenzen

Literatur

[1] Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics* 21(4), 1995.

[2] Mitchell Marcus, Beatrice Santorini, and Maryann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2), 1993.

[3] Ralph Weischedel, Marie Meteer, Richard Schwartz, Lance Ramshaw, and Jeff Palmucci. Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics* 19(2), 1993.