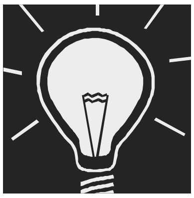
Lernen, Informationen im Internet zu verstehen



Seminar „Lernende Agenten im WWW“

SS 1997

Mirko Fröhlich

Übersicht

- Motivation und Problematik
- Extraktion:
  ShopBot
- Übersetzung:
  ILA (Internet Learning Agent)
- Lernen von Grammatiken:
  WIL (Web Information Learner)

Motivation und Problematik

- Zahlreiche Informationssysteme (IS) sind im Internet verfügbar
- Bisherige Suchmaschinen und Agenten unzureichend
- Ziel: automatisch lernen, mit IS zu interagieren
  - *Entdeckung* (neue IS aufspüren)
  - *Extraktion* (Zugriff auf IS)
  - *Übersetzung* (Antwort in eigene Konzepte übersetzen)
  - *Bewertung* (Genauigkeit, Umfang, Zuverlässigkeit, ...)

ShopBot

- Aufgabe: „Web Comparison Shopping“
- 2 Phasen:
  - *Lernphase*: Generierung von Händlerbeschreibungen (offline)
  - *Einkaufsphase*: Suche nach günstigsten Preisen für vorgegebene Produkte
- Domänenbeschreibung:
  - Beispielprodukte P1, P2, ..., Pn
  - zugehörige Attribute (z.B. manufacturer(P1)=Microsoft, name(P1)=Encarta, ...)

ShopBot Händlerbeschreibungen

- Vorgehensweise:
  - Ausgehend von Homepage entsprechendes Formular finden
  - Herausfinden, wie dieses auszufüllen ist
  - Format der zurückgegebenen Produktbeschreibungen erkennen
- Bias (Menge von Annamen):
  - *Navigation* (Suchindex)
  - *Einheitlichkeit* (einfaches, durchgängiges Format)
  - *Vertikale Trennung* (neue Beschreibung in neuer Zeile)

ShopBot Händlerbeschreibungen

- Standardalgorithmen zum Lernen von Grammatiken nicht anwendbar
- Grundidee:
  - für jedes in Frage kommende Formular F_i Schätzung E_i berechnen, wie erfolgreich Einkaufsphase mit F_i wäre
    - herausfinden, wie Formular auszufüllen ist
    - Testabfragen mit populären Produkten
  - Produktbeschreibungsformate erkennen

ShopBot Händlerbeschreibungen

- Formulare finden und ausfüllen:
  - ausgehend von Homepage Links folgen und Formulare suchen
  - einige durch Heuristiken sofort verwerfen, endgültige Auswahl erfolgt später
  - bestimmen, welches Attribut in welches Feld einzutragen ist (reguläre Ausdrücke mit Synonymen)
- Produktbeschreibungsformate erkennen:
  - Herausfinden, wie „Fehlerseite“ aussieht (product not found)

ShopBot Händlerbeschreibungen

- Populäre Produkte abfragen
  - Ergebnisse mit Fehlerschablone vergleichen
  - keine Übereinstimmung => Suche erfolgreich
- Header und Tailer entfernen
- im Body nach abstrakten Formaten suchen
  - **`<li>Click<a href="http://store.com/Encarta">here</a>for the price of Encarta.`**
  - **`<li>text<a>text</a>text`**
  - in logische Zeilen (beginnend mit <p> oder <li>) zerlegen

ShopBot Händlerbeschreibungen

- Auswahl des mit größter Wahrscheinlichkeit korrekten Formats mit Bewertungsfunktion: $B(F) = Z(F) + P(F) + A(F)$
  - F = Format
  - Z(F) = # Zeilen dieses Formats
  - P(F) = # Zeilen mit Preis
  - A(F) = # Zeilen mit Attribut(en)
- E_i = B(F) für das beste Format F
- Händlerbeschreibung beinhaltet:
  - URL der Seite mit Suchformular
  - Funktion zur Abbildung der Attribute auf Formularfelder
  - Funktion zur Extraktion der Produktdaten

ShopBot
Resultate

- Laufzeit linear in Anzahl der Händler, Anzahl dort vorhandener Formulare, Anzahl der Testabfragen, Zeilenanzahl der Ergebnisseiten
- typischerweise 5-15 Minuten
- Einkauf z.Z. bei 12 Händlern
- Aufnahme neuer Domäne benötigt etwa einen Arbeitstag
- Grundidee allgemein auf Extraktionsproblem anwendbar
  - Standardabfragen möglich
  - strukturierte Ausgabe

ILA
Category Translation

- Lernen der Bedeutung externer Information durch Erklärung mit Hilfe eigener Kategorien
- *Gegeben*:
  - unvollst. internes Weltmodell
    - Objekte (z.B. Personen P1, ..., Pn)
    - Objektattribute (z.B. lastname(P1)=Etzioni, department(P1)=CS, userid(P1)=Etzioni, mail-stop(CS)=FR-35)
  - IS (z.B. *staffdir*)
    - *Abfrage*: Etzioni,
    - *Antwort*: Oren Etzioni 685-3035 FR-35

ILA
Category Translation

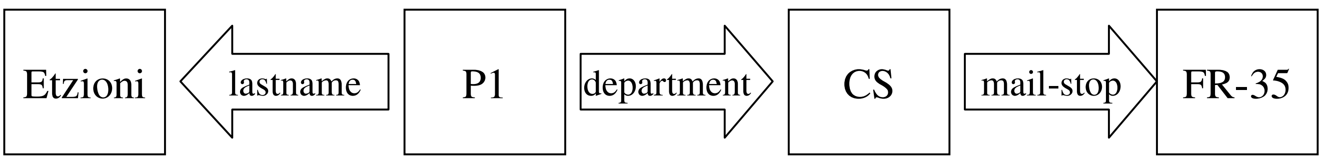
- *Gesucht*:
  logische Ausdrücke, die Abfrage/Antwort-Paare erklären
  - *Abfrage*: lastname(P)
  - *Antwort*:
    - 1. Feld = firstname(P)
    - 2. Feld = lastname(P)
    - 3. Feld = phone-number(P)
    - 4. Feld = mail-stop(department(P))
- *Benötigt:*
  - gleichartige Antworten
  - Überlappung zwischen Weltmodell und IS

ILA
Algorithmus

- Grundidee:
  - Abfrage nach bekannten Objekten
  - Übereinstimmungen zwischen Antwort und Weltmodell suchen
  - Hypothesen aufstellen und danach bewerten, wie oft sie auf Beobachtungen zutreffen
- Auswahl eines Objektes:
  - Ausnutzung Wissens über das IS
  - diskriminierende Abfrage
  - entkräftigende Abfrage
  - Abfrage nach Objekt, über das viele Informationen vorliegen

ILA Algorithmus

- Auswahl der zur Abfrage zu verwendenden Objektinformation:
  - zu Beginn alle Fakten verwenden
  - richtige Abfrage lernen
- Interpretation der Antwort:
  - relationale Pfadsuche
  - Fuzzy-Matcher, der Satz- und Leerzeichen ignoriert und Substring-Matches zuläßt



ILA
Algorithmus

- Bewertung der Hypothese:
  - drei Arten:
    - erklärend (richtige Voraussage)
    - inkonsistent (falsche Voraussage)
    - konsistent (keine Voraussage)
  - Entscheidung, ob Hypothese h1 besser als h2 ist:
    - nach Anzahl der Inkonsistenzen
    - falls gleich entscheidet Anzahl erklärender Hypothesen
- Ende des Lernvorgangs:
  - keine Abfrageobjekte mehr vorh.
  - Führende Hypothese ist „wesentlich“ besser als restliche

ILA
Resultate

- Bei vorhandenem Wissen über Mitarbeiter der UW lernt ILA Modell für *staffdir* in 16 Abfragen, 19s Interaktion und 24s Rechenzeit
- führende Hypothesen:
  - $\mathrm{staffdir}_2(x)=\mathrm{lastname}(x)$
    E=11,I=0
  - $\mathrm{staffdir}_2(x)=\mathrm{userid}(x)$
    E=8, I=3
- allg. Problem: Mehrdeutigkeiten
  - zufällige Mehrdeutigkeiten
  - semi-reguläre Mehrdeutigkeiten

ILA
Resultate

- Weitere Experimente:
  - nutzen vorhandener Informationen zum Lernen des Internet-Dienstes *whois*
  - nutzen von *whois* zum Lernen der Personalverzeichnisse einiger Universitäten in durchschnittlich weniger als 40 Abfragen und 15 Minuten (überwiegend Netzübertragung)
- Verbesserungsmöglichkeiten:
  - Category Mismatch
  - Token Mismatch
  - komplexere Domänen

WIL
Web Information Learner

- Problemstellung:
  - *Grammatik-Inferenz*: aus Beispielmenge Grammatik lernen
  - hier: nur positive Beispiele, stochastische reguläre Grammatiken (darstellbar als DFA)
  - *Eingabe*: Menge positiver Beispielstrings S+ aus L(M), L(M) = von DFA M erzeugte Sprache
  - *Ausgabe*: DFA M*, der M approximiert

WIL Problematik

- Stochastischer Automat:
  - jede von Knoten *n* ausgehende Kante *a* besitzt Wahrscheinlichkeit $P_n(a)$
  - jeder Knoten *n* besitzt Terminationswahrscheinl. $P_t(n)$
  - $P_t(n) + \sum P_n(a) = 1$ für $n \in M$ (≈ 1 für $n \in M^*$)
  - S+ ist stochastisch aus M erzeugt
- Bisherige Arbeiten:
  - Problem der Identifizierung einer beliebigen regulären Sprache aus pos. Beispielen nicht lösbar

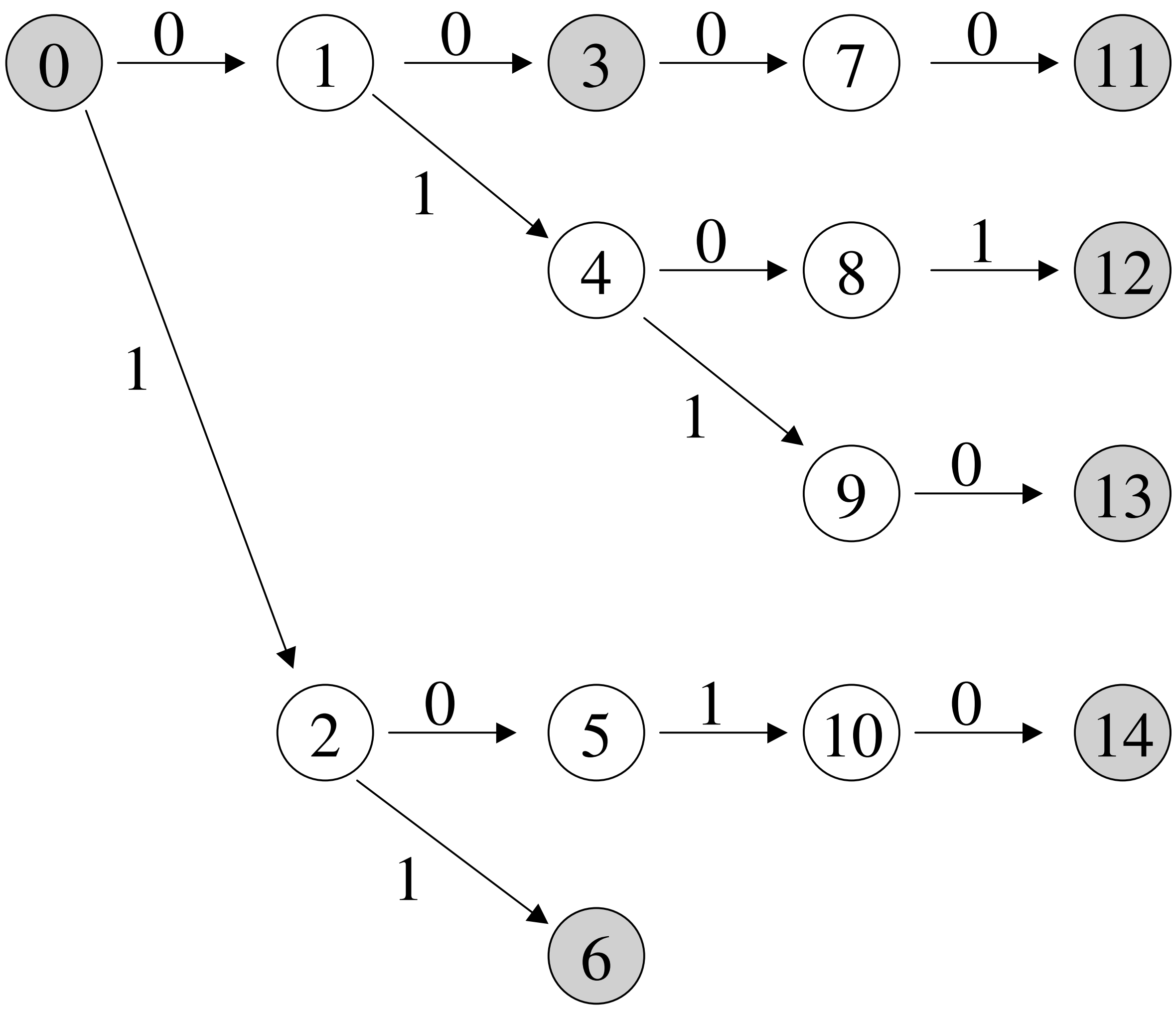
WIL
Bisherige Arbeiten

– Algorithmen für bestimmte eingeschränkte Sprachen (z.B. Angluin, k-reversible Sprachen)
  - nicht anwendbar, da vorab nicht bekannt ist, ob die Sprache zu solch einer Klasse gehört
  - Häufigkeit des Auftretens von Beispielstrings wird nicht beachtet: {aa, aaaa, aaa, aaaaa} => a*
    $\{aa^{100}, aaaa^{100}, aaa^{100}, aaaa^{100}\}$ => ?

– *Alergia* (Carrasco & Oncina):
  - Präfix-Baum erzeugen (DFA aus positiven Beispielen)
  - Knoten in lexikographischer Reihenfolge vergleichen
  - Vereinigung falls gleich bzgl. Maß

WIL

Lexikographisch sortierter Präfix-Baum zum Beispiel {λ, 00, 11, 0000, 0101, 0110, 1010}



WIL
Bisherige Arbeiten

- Bedingung für Äquivalenz:
  - Übergangshäufigkeiten für alle $a \in \Sigma$ bzgl. Hoeffding Bound äquivalent
  - von Knoten *i* und *j* durch Symbol *a* erreichte Knoten sind ebenfalls äquivalent (rekursiv)
  - *Hoeffding Bound*:

    $\frac{f}{n} \Leftrightarrow \frac{f'}{n'}$, falls

    $$\left|\frac{f}{n}-\frac{f'}{n'}\right|>\sqrt{\frac{1}{2}\log\frac{2}{\alpha}}\left(\frac{1}{\sqrt{n}}+\frac{1}{\sqrt{n'}}\right)$$

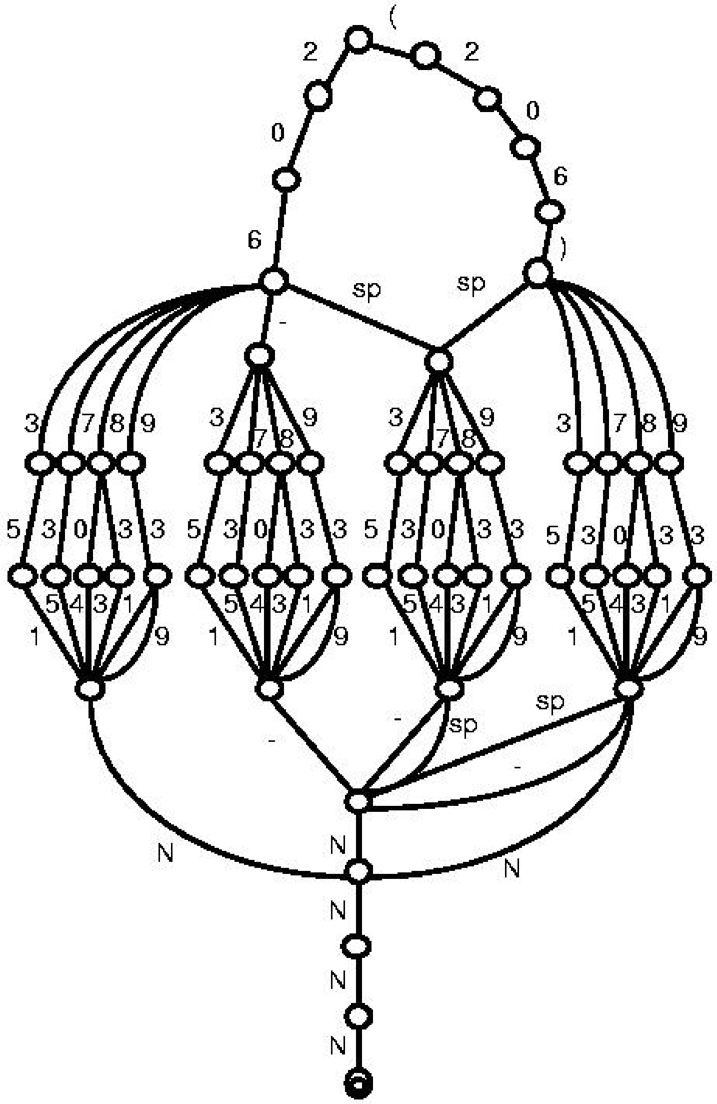
    » n, n´: # Strings durch Knoten

    » f, f´: # Strings, die *a* folgen

    » 1 - α: Confidence
- für uns ungeeignet wegen „buschiger“ Präfix-Bäume mit vielen Möglichkeiten für falsche Verallgemeinerungen

WIL

DFA zulässiger Telefonnummern in Auburn, Washington (sp = Leerzeichen, N = 10 Kanten markiert von 0-9)



WIL
Algorithmus

- Zwei wesentliche Verbesserungen gegenüber Alergia:
  – Strukturanalyse
  – More-Data-First Suche
- *Strukturanalyse*:
  – Alphabet in Typen einteilen (Ziffern, Buchstaben, Trennz., ...)
  – Verhindern falscher Verschmelz.

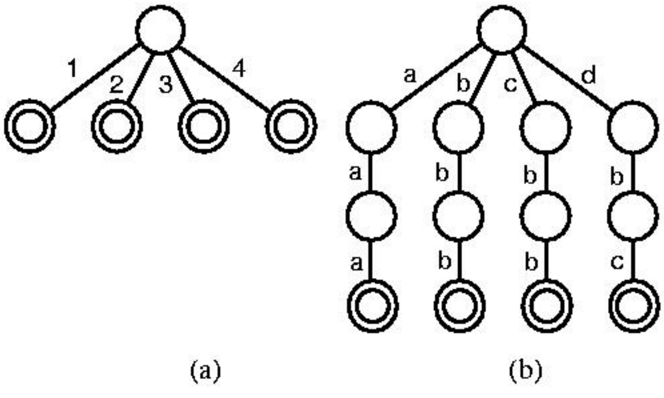


Figure 2. Two prefix trees which can be mistakenly recognized as equivalent by Algeria (Assume equal frequencies for strings).

WIL
Algorithmus

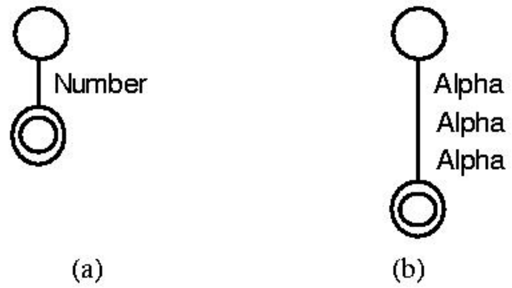


Figure 3. A structural view of the trees in Figure 2. Based on this view, the trees appear much less similar.

– einfachere Identifikation zulässiger Verschmelzungen
– Betrachtung von Symbolwert- und Typ, Vereinigung wenn ähnlich bzgl. des einen Maßes und nicht zu unähnlich bzgl. des anderen
– *Level-Checking* wenn alle Nachfolgeknoten vom gleichen Typ sind

WIL Algorithmus

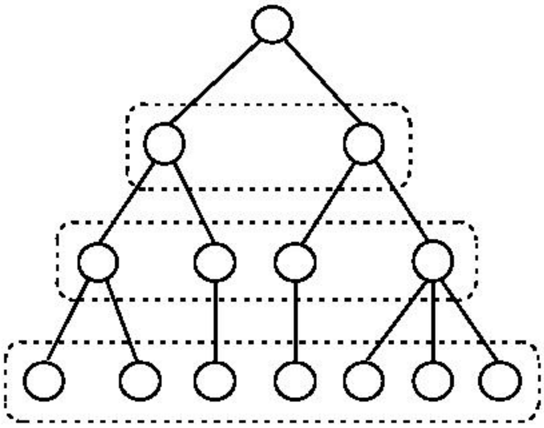


Figure 4. The groups of nodes compared in "level-checking"

- *Wildcards*: sind ausgehende Kanten eines Knotens etwa gleichmäßig über Wertebereich eines Typs verteilt, werden fehlende Kanten ergänzt

• *More-Data-First* Suche:
  - anstelle lexikographischer Sortierung nach Anzahl der Beispiele, in denen Knoten liegen, sortieren

WIL
Algorithmus

– Vermeidung schlecht begründeter Vereinigungen
– gleiche rechnerische Vorteile wie lexikographische Sortierung

```
Algorithmus WIL (S alpha beta gamma)
; S     -> Beispielmenge
; alpha -> 1 - Confidence-Grad (eins)
; beta  -> 1 - Confidence-Grad (zwei)
; gamma -> 1 - Confidence-Grad (Wildcard)

begin
T = stochastischer Praefix-Baum, sortiert nach More-Data-First()

for (k = 0 to letzte-Id)
  if (einzelner-Typ (knoten k))
    level-check (knoten k)
  end-if
end-for
```

WIL
Algorithmus, Resultate

```
for (j = 1 to letzte-Id)
  for (i = 0 to j - 1)
    if (or aequivalent ((knoten i) (knoten j) alpha beta)
           aequivalent ((knoten i) (knoten j) beta alpha))
      merge ((knoten i) (knoten j))
      exit (i-Schleife)
    end-if
  end-for
end-for

for (x = 0 to letzte-Id)
  if (wildcard? ((knoten x) gamma))
    make-wildcard (knoten x)
end-for
end
```

- *Resultate*:
  - Laufzeit $O(|S|^4)$
  - getestete Sprachen:
    - Telefonnummern

WIL
Resultate

    - Tech-Report-Codes
    - Reber-Grammatik
    - Busfahrpläne
  - abgesehen von Reber-Grammatik bessere Ergebnisse als Alergia
  - additive Wirkung von Strukturvergleich, Level-Checking, Wildcard-Verallgemeinerung wird deutlich
- Zukünftige Pläne:
  - Typ- und Strukturinformation noch weiter ausnutzen

Literatur

- Perkowitz, Doorenbos, Etzioni, Weld, „*Learning to Understand Information on the Internet: An Example-Based Approach*“
- Perkowitz, Etzioni, „*Category-Translation: Learning to understand information on the Internet*“
- Goan, Benson, Etzioni, „*A Grammar Inference Algorithm for the World Wide Web*“
- Angluin, 1982, „*Inference of reversible languages*“, in „Journal of the ACM, 29(3):741-765“
- Carrasco, Oncinam 1993, „*Learning stochastic regular grammars by means of a state merging method*“, in „Second International Colloqium, ICGI-94 Proceedings“, S.139-152