Typing candidate answers using type coercion[7]

Vortrag von Jörg Meier

30. November 2012

[7]: By J. W. Murdock, A. Kalyanpur, C. Welty, J. Fan,
D. A. Ferrucci, D. C. Gondek, L. Zhang, H. Kanayama

Inhalt

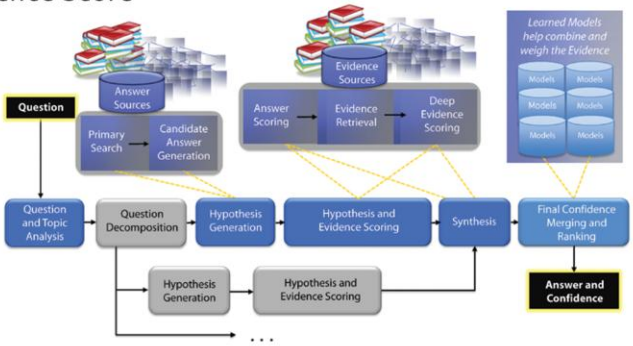
I. Überblick und Einordnung in Watson

- Hintergrund
  - Wie kann ich spezielle Antworttypen aus der Frage ableiten?
  - Ansatz 1: Listen mit gewöhnlichen Typen
- Type Coercion (TyCor) in DeepQA
  - Kombination vieler verschiedener Ansätze
  - Potentielle Antworttypen werden lediglich Bewertet, allerdings nie direkt verworfen

1) Ansatz 1
    1) Länder, Namen, Tiere, Essen, ..
    2) unpassend für Jeopardy!, da nur kleine Menge abgedeckt
2) TyCor
    1) Einige Ansätze mit großer Typ-Abdeckung, andere sehr speziell
    2) können durch Bewertung in **späteren** Schritten von **DeepQA wieder aufgegriffen** werden, **wenn genügend Hinweiskraft** vorhanden

I. Überblick und Einordnung in Watson (2)

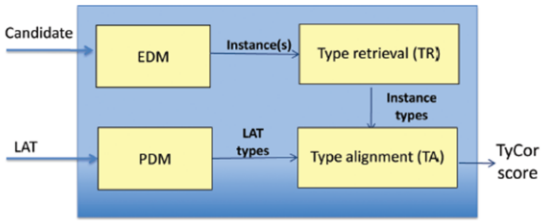
- Aufteilung
  - Question Analysis -> LAT (Lexical Answer Type)
  - Candidate Generation -> Candidate Answer
  - TyCor ∈ Hypothesis & Evidence Scoring
  - TyCor -> Evidence Score



1) LAT = Wort in der Frage welches einen Antworttypen andeutet
2) Candid. Answer = potentielle Antwort, z.B. durch Mehrdeutigkeitenauflösung von Wikipedia (letzter Talk)

II. Die Architektur von TyCor

- Entity Disambiguation Matching (EDM)
  - findet in *Quellen* Entität von Typen die der Cand. Answ. entsprechen
  - muss Polysemie und Synonymie handhaben können
- Type Retrieval (TR)
  - „versteht“ Typ-Instanzen des EDM
  - Unterschied zwischen strukturierten und nicht-strukturierten Quellen

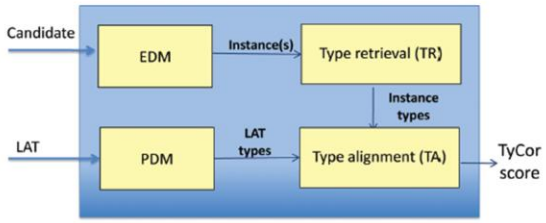


1) **Input**
   1) 1 Cand. Answ.
   2) ≥ 1 LATs
2) **Output Score für Cand.Answ.**: wie stark ist es möglich, dass ein Typ einer Cand.Answ. mit Typ von LAT übereinstimmt? **Hohe Score**, wenn Cand.Answ. **Unterklassen** oder **Instanzen** von LATs seinen könnten.

1) EDM
   1) **Polysemie**: selber Name bezeichnet mehrere Objekte
   2) **Synonymie**: ein Objekt hat mehrere Namen
2) TR
   1) strukturierte Quellen -> easy
   2) unstrukturiert -> **Parsing** und andere **semantische Analyses** für natürliche Sprache

II. Die Architektur von TyCor (2)

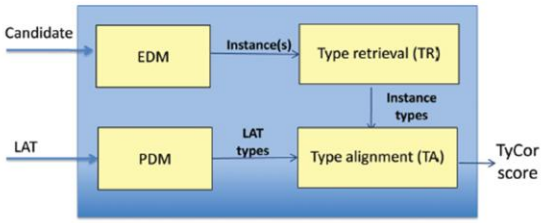
- Predicate Disambiguation and Matching (PDM)
  - Identifiziert Typen korrespondierend zu LATs
  - TyCors auf unstrukturierten Quellen geben oft LAT zurück
- Type Alignment (TA)
  - wie stark stimmen Ergebnisse von TR und PDM überein?
  - Output: Bewertungsmaß für beste Übereinstimmung



1) PDM
   1) = EDM für manche Quellen, manche Quellen kodieren Typen und Instanzen unterschiedlich
   2) PDM = „finde das richtige Wort für den Kontext“
2) TA
   1) in strukturierten Quellen = prüfe auf **Unterkategorie, Disjunktion**
   2) in unstrukturierten Quellen = **ist Text** von LAT **mit Text** vom gefundenen Typ **konsistent**? (durch Parsen, Synonyme finden..)
   3) **-> TA** vergleicht alle Typen miteinander und **zwingt die Typen** mit höchster Übereinstimmung **aufeinander**

II. Die Architektur von TyCor (2)

- Predicate Disambiguation and Matching (PDM)
  - Identifiziert Typen korrespondierend zu LATs
  - TyCors auf unstrukturierten Quellen geben oft LAT zurück
- Type Alignment (TA)
  - wie stark stimmen Ergebnisse von TR und PDM überein?
  - Output: Bewertungsmaß für beste Übereinstimmung
- Beispiel
  - CA: „Gone with the wind“
  - LAT: „book“



1) EDM, TR & PDM können **mehrere Ergebnisse** liefern -> nicht zu früh festlegen
2) EDM -> Film, Roman
3) PDM -> veröffentlichte Arbeit, Phänomen in Graphen Theorie, ..

III. Verschiedene TyCor Komponenten

- 12 verschiedene TyCor Komponenten
- Unterschied in Algorithmen und/oder Quellen
- zwei Ansätze
  - Komponenten mit wohl-definierten Mengen von strukturierten Typen
  - Komponenten mit Typen, die willkürlichem Text entsprechen (basiert auf natürlicher Sprache)
- Komponenten erzeugen Scores
  - positiv / negativ / neutral

4) Typisierung mit **negativer Score** (Hinweiskraft) **werden nicht gelöscht** oder nicht weiter betrachtet, sondern können, wenn genügend Hinweise vorhanden sind, im weiteren Prozess von DeepQA wieder aufgewertet werden.

III.I Komponenten mit strukt .Typen

- YAGO (Yet Another Great Ontology)
  - Candidate Answers sind oft Einträge in DBpedia
  - zusätzlich 200 manuelle Disjoint-Relationen
- Gender
  - Bewertet ausschließlich das Geschlecht von Personen
- Closed LAT
  - negative Score, wenn Liste für LAT Typ vorhanden und Candidate Answer nicht auf Liste

DBpedia = Wiki Infoboxes

1) **YAGO** negative Scores, wenn Kanditatentypen disjoint von LAT Typen
2) **Gender** spezielle Daten über Personen; Unterscheidung durch Pronomen; Score, ob Geschlecht mit LAT übereinstimmt
3) **Closed LAT** Gondor nicht auf Liste; Annahme: Listen sind vollständig

III.I Komponenten mit strukt .Typen (2)

- Lexical
  - benutzt viele spezielle Algorithmen zur lexikalischen Erkennung von z.B. Verben, Phrasen, Namen
- Named Entity Detection (NED)
  - Labelt LAT's und Cand. Answ. jeweils mit ≥ 0 strukturierten Typen und sucht Schnittmengen
- WordNet
  - enthält spezifische Informationen über Taxonomien von z.B. Wissenschaftlern, Geographen und Biologie

1) **Lexical** Candidate Answer ohne Leerzeichen kann kein LAT-Typ „Phrase“ sein
2) **NED** > 100 strukturierte Typen vorhanden; keine Eigenentwicklung, entwickelt für klassische QA Systeme
3) **WordNet** wird außerdem von anderen TyCor Komponenten genutzt zur Typenausrichtung; Scores für Unterordnungen/Instanzen von Cand.Answ. vom LAT-Typ

III.II Komponenten mit unstrukt .Typen

- Wiki-Category
  - Kategorienamen als Entitätentyp ⇒ Überspringe EDM-Step
  - keine Strukturinformationen, da zu viel Rauschen
- Wiki-List
  - Bspl. „List of *Argentinean Nobel Prize Winners*“
- Wiki-Intro
  - Erster Satz einer Wiki-Artikels enthält viele Typen
  - Bspl: „*Tom Hanks* is an *American actor*, *producer*, *writer*, and *director*.“

1) **Wiki-Category** in DBpedia, Artikel sind getagged mit Kategorien
   1) **kann** EDM überspringen, wenn Cand. Answ. bereits **spezielle ID** besitzt (**assoziiert mit Wikipedia Einträgen**)
3) **Wiki-Intro** spezielle Datenbank, entstanden durch spezielle syntaktische Analysen

III.II Komponenten mit unstrukt .Typen (2)

- Identity
  - testet auf Übereinstimmung von Strings
  - Bspl. „the Chu *River*“
- Passage
  - sucht Typenhinweise in Passagen aus Primary Search und Supporting Evidence Retrieval
- PRISMATIC
  - statistische Bewertung, wie häufig eine Candidate Answer als direkte Instanz eines LATs vorkommt

1) **Identity** Candidate Answer als Quelle für Typ-informationen
2) **Passage** sucht im Primary Search + Supporting Evidence Retrieval nach Passagen, die Candidate Answers enthalten und vergleicht die Typen dort mit dem LAT
3) **PRISMATIC** benutzt ein Repository aus Statistiken über Korpora

IV. TyCor Komponenten an einem Beispiel

- Input: LAT: „emperor“ CA: „Napoleon“
- EDM
  - TyCor mit Wiki-Sources: *DBpedia:Napoleon*; *:Card_Game*
  - WordNet -> drei verschiedene Bedeutungen
  - NED, Identity -> Entitätstyp ist „Napoleon“
- Type Retrieval
  - YAGO: leitet formale YAGO Typen von EDM Entitäten ab
  - WordNet: „Napoleon“ ist Instanz von „emperor“
  - NED: „Napoleon“ ist NAME Referenz zu *PoliticalLeader*
  - Wiki-List: „List of *French monarch*s“

1) *EDM*
    1) *Napoleon* hohe Score, geringere Score für *CardGame*
2) TR
    1) Wiki-List: *List* **zeigt** auf *NapoleonMonarch* **leitet weiter** auf *Napoleon*

IV. TyCor Komponenten an einem Beispiel (2)

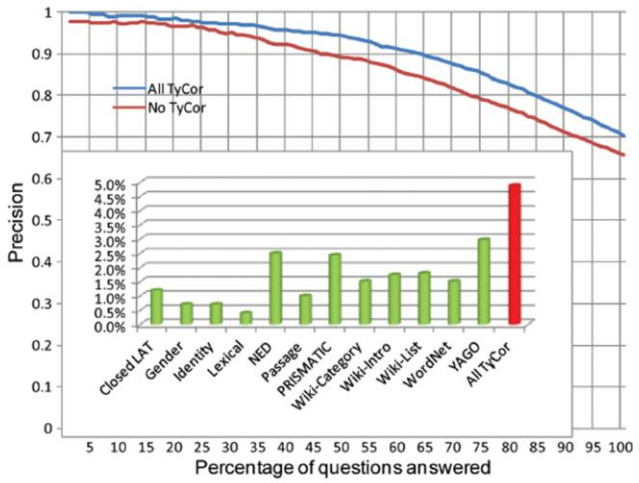
- PDM
  - WordNet: identifiziert vier Synsets für „emperor“
  - NED: LAT ist NOMINALE Referenz zu *PoliticalLeader*
  - YAGO: findet formale Typen in der YAGO Ontology
  - Andere: geben LAT einfach als String weiter
- Type Alignment
  - WordNet, YAGO, NED: prüfe Unterordnungen, Disjunktheit
  - Wiki-List: ist „French Monarch“ ein „emperor“?
    - findet durch parsen „monarch“
    - vergleicht Ausdrücke auf Quellen (WordNet, Wikipedia Weiterleitungen)

Synset = Verhältnis von Ober- zu Unterbegriffen
Ontology = sprachlich gefasste und **formal geordnete** Darstellung einer Menge von Begrifflichkeiten

1) TA
    1) TyCors, die auf formalen Typen arbeiten

V. Evaluation

- DeepQA mit und ohne TyCor-Komponenten

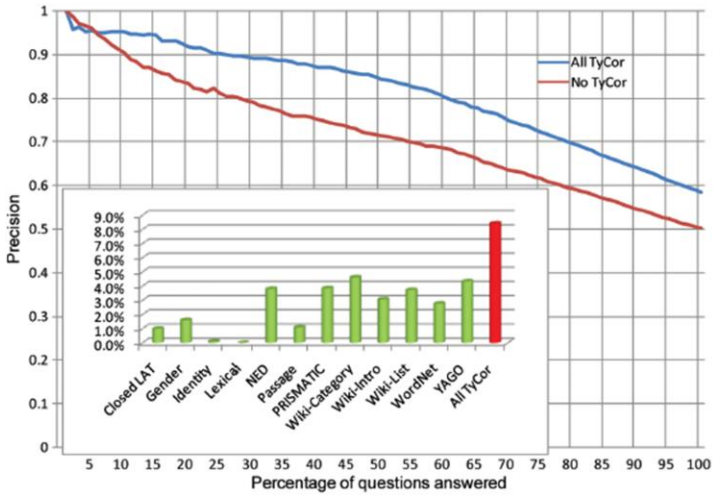


1) Vergleich der Effizienz des DeepQA **mit und ohne TyCor** Komponenten; evaluiert auf **3.508** vorher ungesehenen Jeopardy! Fragen
2) Precision@70: 87,5 - 81,5 = 6 % Unterschied
3) Precision@100: 4,9 % Unterschied = All TyCor

=> TyCor ist gut um Vertrauen in Antworten zu erhöhen, und nicht um Antworten vorzuschlagen

V. Evaluation (2)

- DeepQA ohne Evidence-Features mit/ohne TyCor

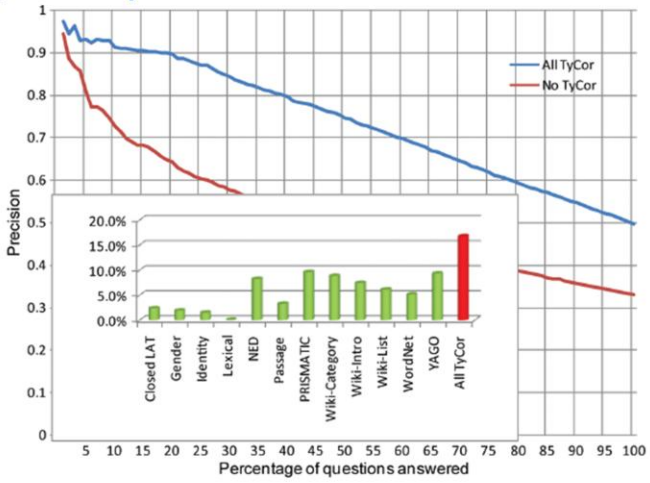


1) Precision@100: 8 % Unterschied

=> TyCor hat noch viel größeren Einfluss auf DeepQA

V. Evaluation (3)

- DeepQA ohne Evidence- und Answer-Features mit/ohne TyCor



1) Precision@100: > 15 % Unterschied
2) Rolle von **NED** als State-of-the-Art Baseline für QA Systeme

VI. Diskussion

- Funktionsweise
  - bewerte, ob Candidate Answer bestimmten LATs entspricht
  - Antwortengenerierung -> TyCor -> Ranking
- Unterschied zu bestehenden Ansätzen
  - vielfältige Ressourcen, daher große Abdeckung und hohe Konfidenz
  - ALLE Antworten werden verarbeitet, keine wird gestrichen
- Bewertung
  - verschiedene Komponenten ergänzen sich

    ⇒ **Verbesserung von DeepQA durch TyCor um 5 %**

1) TyCor als **Evidence Scoring**
2) **Ressourcen**
   1) **manuelle**, mit hoher Precision aber kleinem Gültigkeitsbereich (NED, closedLat)
   2) große, von der **Community** erstellte (Wiki-Category, YAGO)
   3) **automatisch** extrahierte von Sprachtexten (Wiki-Intro, PRISMATIC)
3) alle zusammen besser als eine einzelne