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

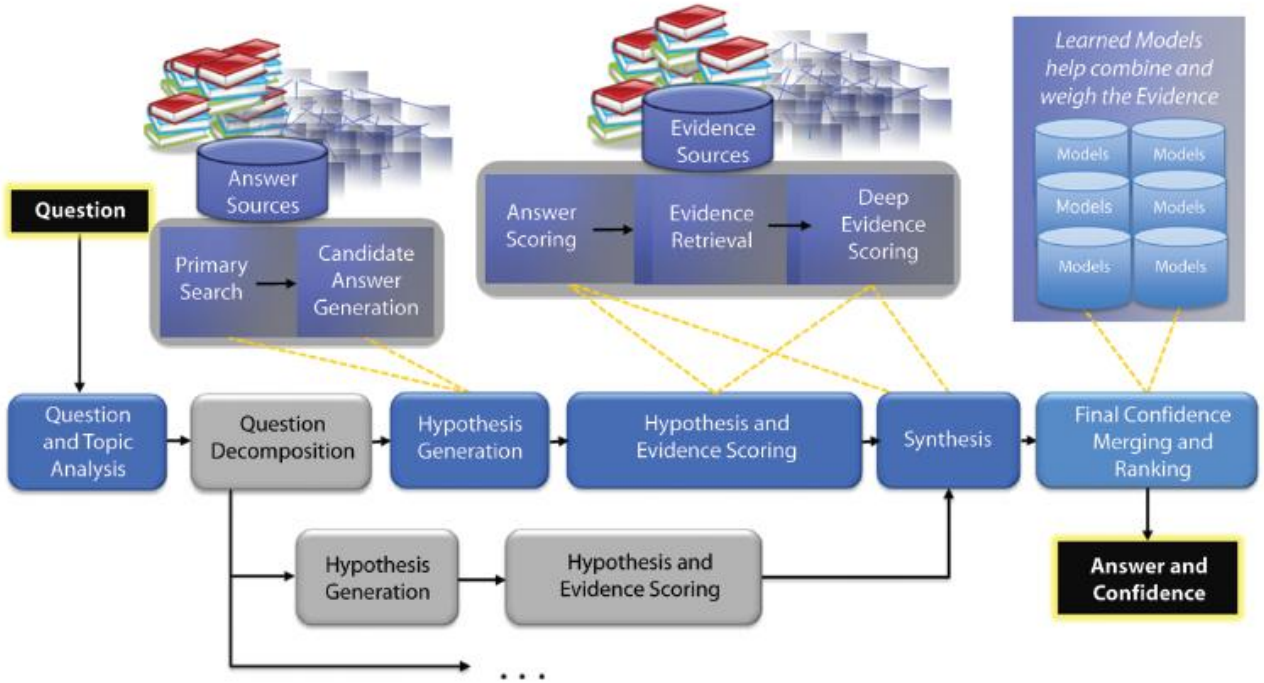
I. Überblick und Einordnung in Watson

- 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

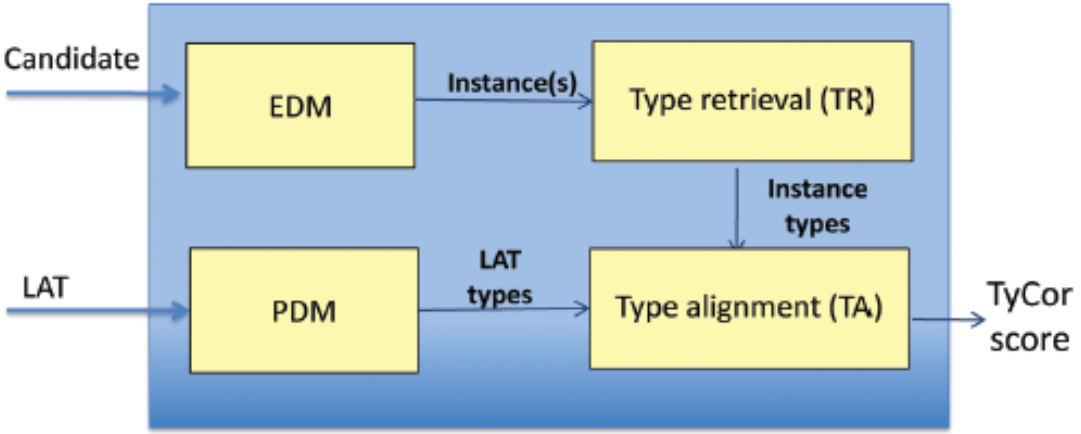
I. Überblick und Einordnung in Watson (2)

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



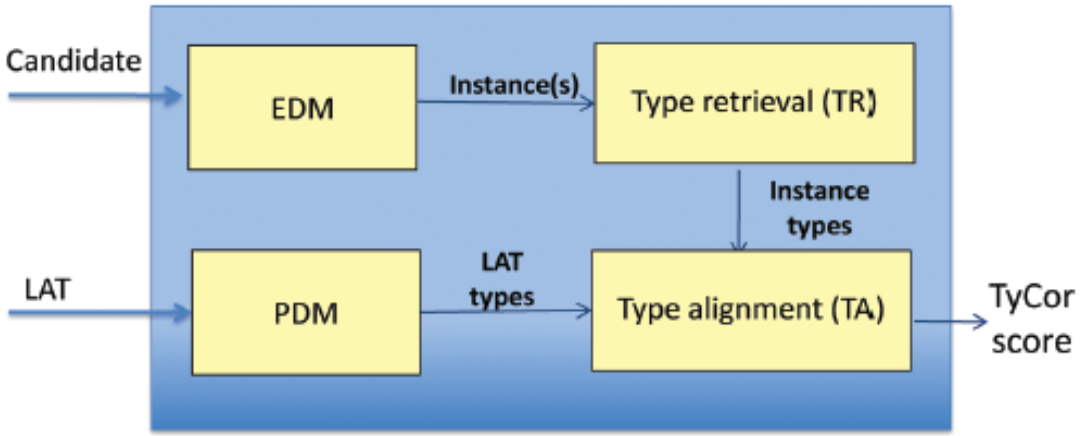
II. Die Architektur von TyCor

- 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



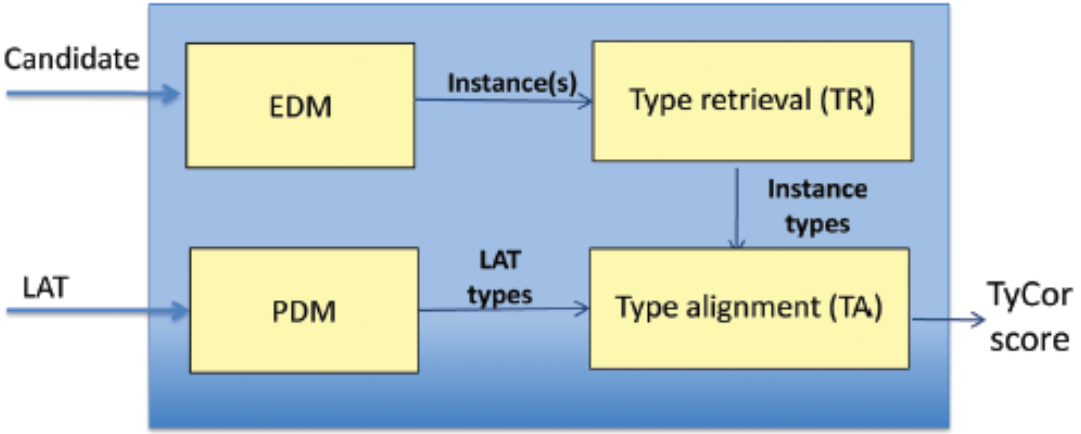
II. Die Architektur von TyCor (2)

- 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



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“



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

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

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

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*.“

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

IV. TyCor Komponenten an einem Beispiel

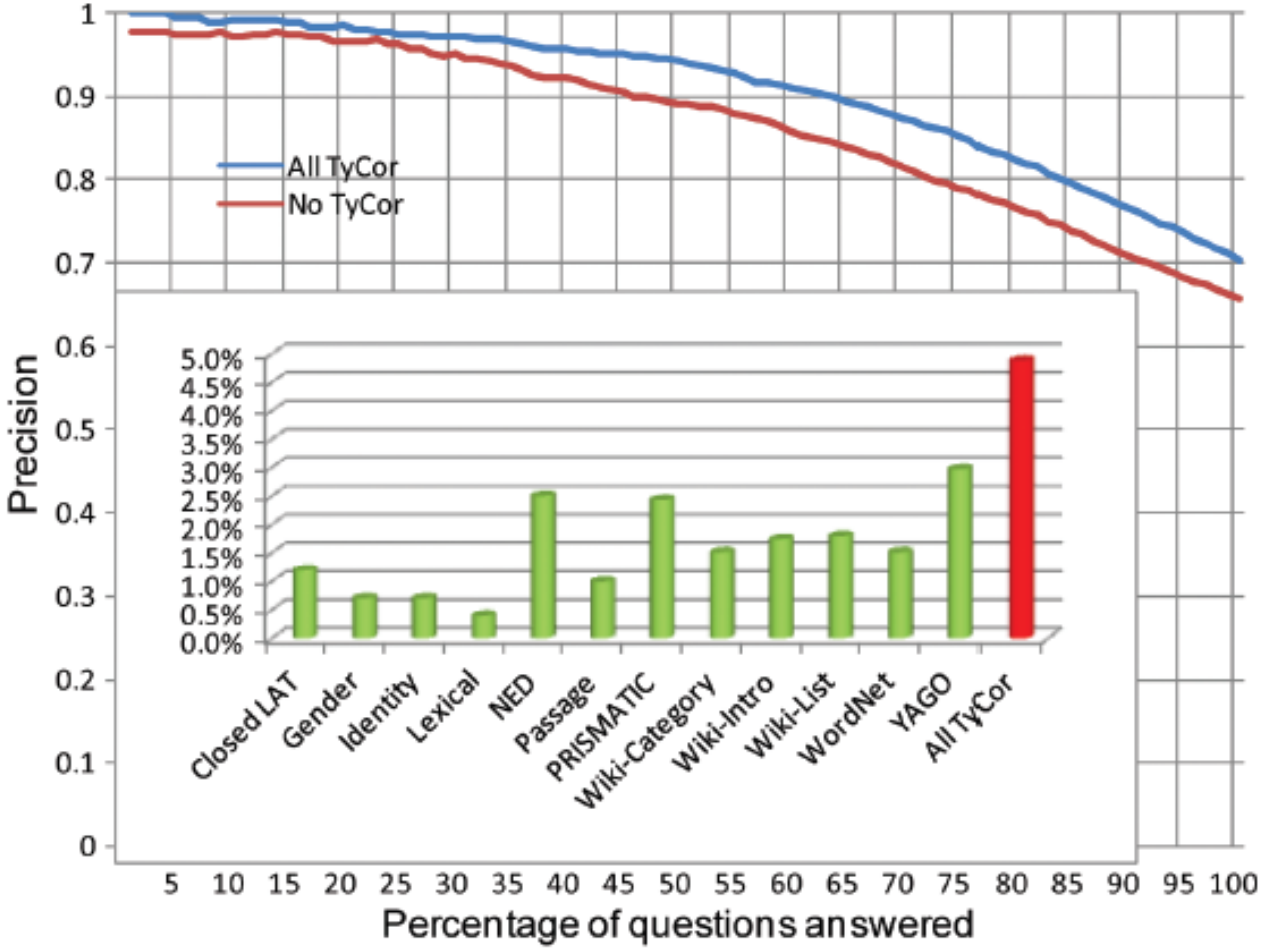
- Input: LAT: „emperor“ CA: „Napoleon“
- EDM
  - TyCor mit Wiki-Sources: *DBpedia:Napoleon*; *:Card_Game*
  - WordNet -> drei verschiedene Bedeutungen
  - NED, Identity -> Entitätstyp ist „Napoleon“
- Type Retrieval
  - YAGO: leitet formale YAGO Typen von EDM Entitäten ab
  - WordNet: „Napoleon“ ist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

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)

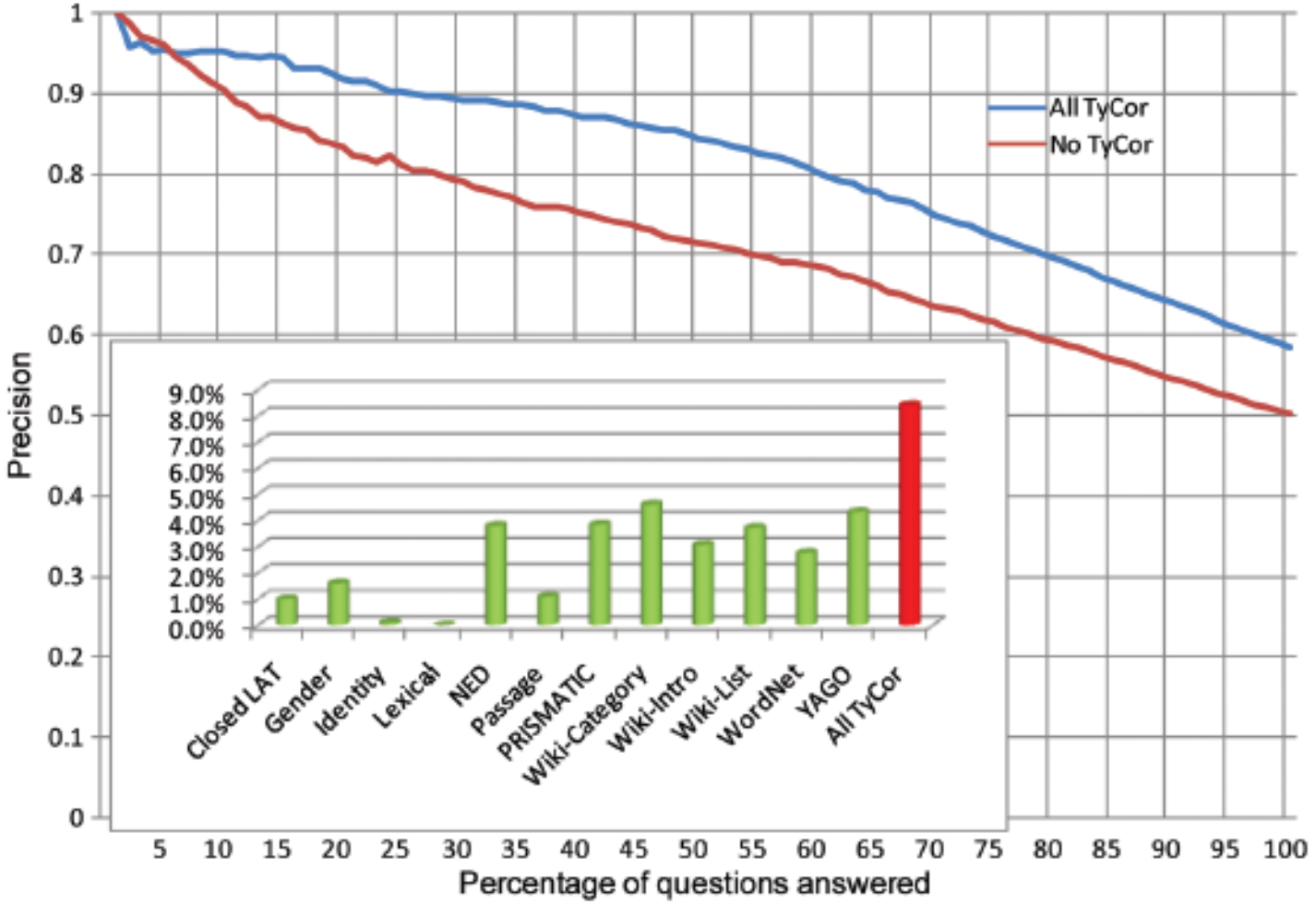
V. Evaluation

- DeepQA mit und ohne TyCor-Komponenten



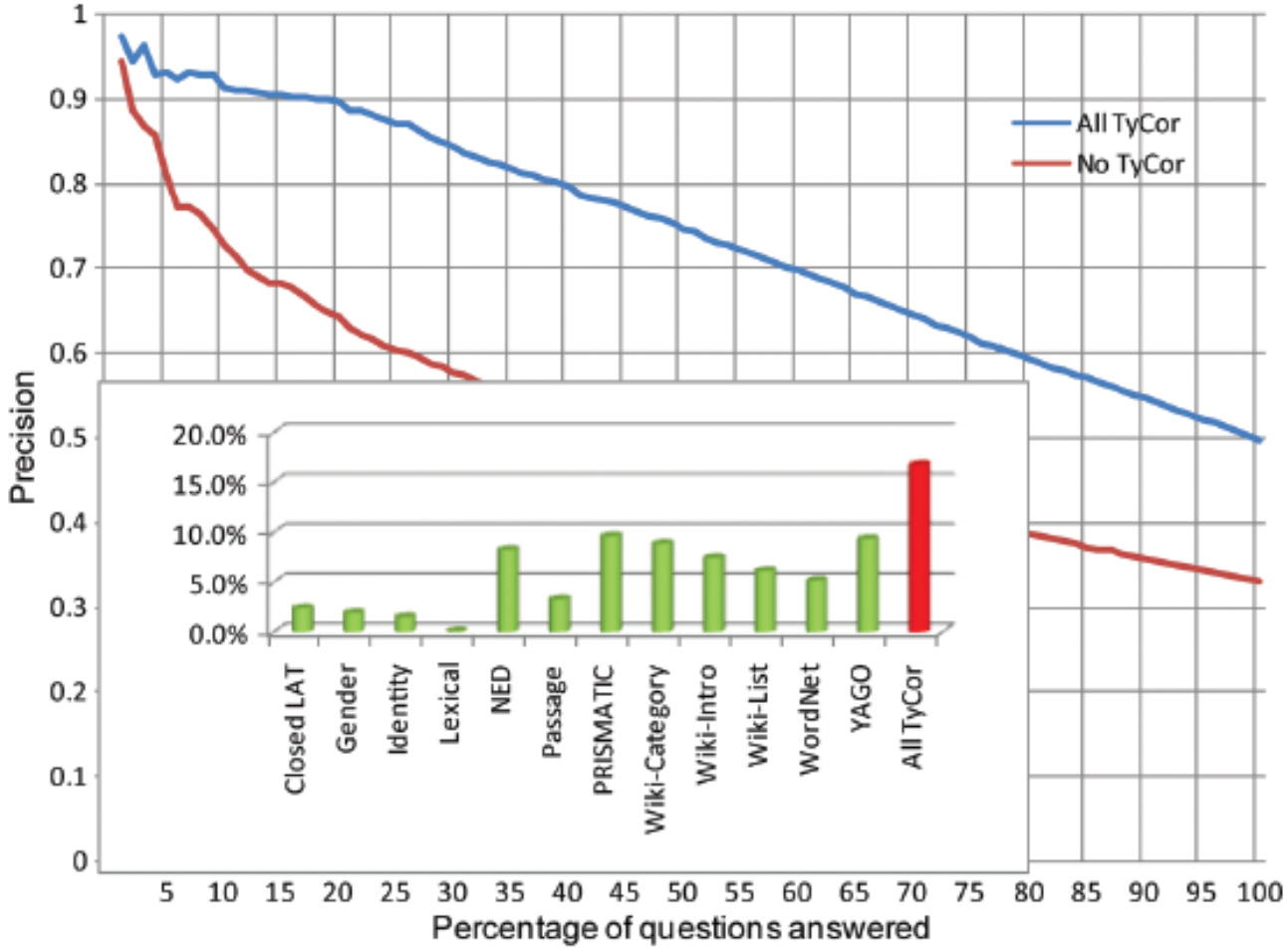
V. Evaluation (2)

- DeepQA ohne Evidence-Features mit/ohne TyCor



V. Evaluation (3)

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



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 %**