Web-scale IE: KnowItAll – KnowItNow - TextRunner

Iwona Kudlinska, Nino Butkhuzashvili,
Ekaterina Razzhivina

Ziel:

eine Überführung von Fakten aus dem Web in eine konventionelle relationale Datenbank.

Entwicklung:

KnowItAll

Klassen, domänenunabhängige Regeln, Anfrage an eine externe Suchmaschine geschickt.

KnowItNow

Klassen, domänenunabhängige Regeln, Anfrage an eine interne Suchmaschine geschickt, Geschwindigkeitsoptimierung

TextRunner

Keine Regeln oder Klassennamen benötigt

KnowItAll: allgemein

KnowItAll ist ein System zur Informationsextraktion und wurde 2004 in der Washington Universität vorgestellt [Etz04].

Es leistet eine Überführung von Fakten aus dem Web in eine konventionelle relationale Datenbank, wobei jedem dort gespeicherten Fakt eine Wahrscheinlichkeit für seine Korrektheit zugeordnet ist.

KnowItAll: Konzept

- Es wird mit Domänen (Klassen) gearbeitet.
- Es werden mehrere **domänenunabhängige** Regeln mit **Discriminator-Phrasen (discriminators)** definiert.

  Beispiel: NP1 such as NP2 – Cities such as Munich, Vienna and Paris. Books such as „Romeo and Julia“ etc.
- KnowItAll formuliert Anfragen mit Hilfe von Regeln, schickt sie an eine Suchmaschine und wertet die Ergebnisse statistisch aus.
- Falls die Ergebnisse hohe Relevanz haben, werden die Regeln weiterbenutzt um die nötigen Informationen aus dem Web zu extrahieren und die Instanzen der Klasse werden in einer Datenbank gespeichert.

Hauptmodule

1. Extractor: wählt aus allen domänenunabhängigen Regeln die Regeln aus, die für eine bestimmte Klasse relevant sind.
2. Search Engine Interface: formuliert Anfragen, die sich auf den Regeln basieren.
3. Assessor: wertet die Ergebnisse statistisch aus, um festzustellen, ob die Anfragen korrekt formuliert wurden.
4. Database: die Ergebnisse werden in einer Datenbank gespeichert.

Extractor

Sobald KnowItAll eine neue Klasse oder Relation als Eingabe bekommt, fängt der Extractor an aus allen domänenunabhängigen Regeln die Relevanten für diese Klasse herauszuziehen.

Extractor: Beispiele der domänenunabhängigen Regeln

- NP1 “such as” NPList2
- NP1 “and other” NP2
- NP1 “including” NPList2
- NP1 “is a” NP2
- NP1 “is the” NP2 “of” NP3
- “the” NP1 “of” NP2 “is” NP3

Extractor: Beispiel für eine Formale Beschreibung einer Regel.

NP1 "such as" NPList2

& head(NP1)= plural(name(Class1))

& properNoun(head(each(NPList2)))

=>

instanceOf(Class1,head(each(NPList2)))

Extractor: der Vorgang

Der Extractor generiert eine Regel für „city“ indem er „city“ für „Class 1“ einsetzt, baut Plural “cities” nach dem Muster. Dadurch wird folgender Regel produziert:

NP1 "such as" NPList2
& head(NP1)="cities"
& properNoun(head(each(NPList2)))
=>
instanceOf(City,head(each(NPList2)))
keywords: "cities such as"

Es wird eine Anfrage formuliert, die dann an eine Suchmaschine geschickt wird.

Web Bilder Groups News Products **Mehr »**

"cities such as" Suche Erweiterte Suche Einstellungen

Suche: ◉ Das Web ○ Seiten auf Deutsch ○ Seiten aus Deutschland

Web Ergebnisse **1** - **1**

Tipp: Suchen nur nach Ergebnissen auf **Deutsch** . Sie können Ihre bevorzugten Spracheinstellungen in Eins

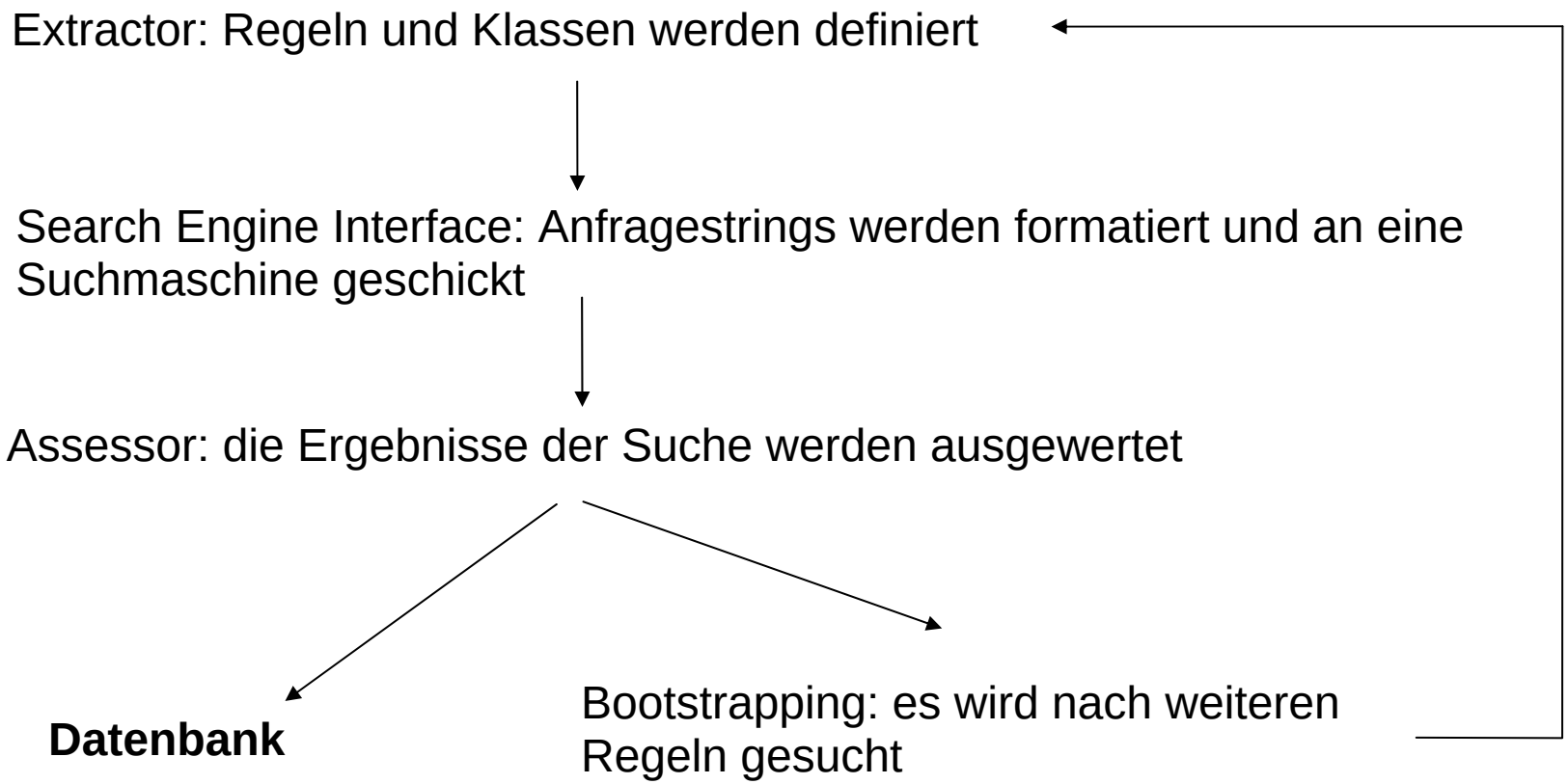
Amazon.de: Fifty Major Cities of the Bible: From Dan to Beersheba ...
Not only covering renowned **cities such as** Jerusalem and Jericho, the book also includes lesser known towns like Aroer, Beth-Zur and Gibeah, which have all ...
www.amazon.de/Fifty-Major-Cities-Bible-Beersheba/dp/0415223156 - 56k -
Im Cache - Ähnliche Seiten

Free Imperial City - Wikipedia, the free encyclopedia - [Diese Seite übersetzen]
Rather than legal matters, what mattered more was the difference in wealth: rich **cities such as** Lübeck or Augsburg for examples, were genuinely self-ruling ...
en.wikipedia.org/wiki/Imperial_Free_City - 60k - Im Cache - Ähnliche Seiten

The Lincoln Highway - [Diese Seite übersetzen]
Roads in some **cities, such as** Ames, Iowa, are still known as "Lincoln Way." A few of the 3000 Boy Scout markers can still be found along the old route. ...
www.fhwa.dot.gov/infrastructure/lincoln.htm - 36k - Im Cache - Ähnliche Seiten

[PDF] "In my vision of a better world, **cities such as** Mumbai, Nairobi ...
Dateiformat: PDF/Adobe Acrobat - HTML-Version
"In my vision of a better world, **cities such as** Mumbai,. Nairobi, and Lima would be as well developed as Frankfurt,. Tokyo, and New York. ...
www.db.com/en/downloads/company/CSR_Report_2005_en_microfinance.pdf -
Ähnliche Seiten

Der Vorgang


Extractor: Regeln und Klassen werden definiert
Search Engine Interface: Anfragestrings werden formatiert und an eine Suchmaschine geschickt
Assessor: die Ergebnisse der Suche werden ausgewertet
Datenbank
Bootstrapping: es wird nach weiteren Regeln gesucht

Assessor

Ziel: Präzisionswerte zu verbessern

Was wird gemacht: Assessor misst wie oft die Extraktionskandidaten mit verschiedenen Discriminators (Regelphrasen) zusammen vorkommen (PMI Werte werden benutzt). Die Mutual Information Statistik wird mit Naive Bayes kombiniert.

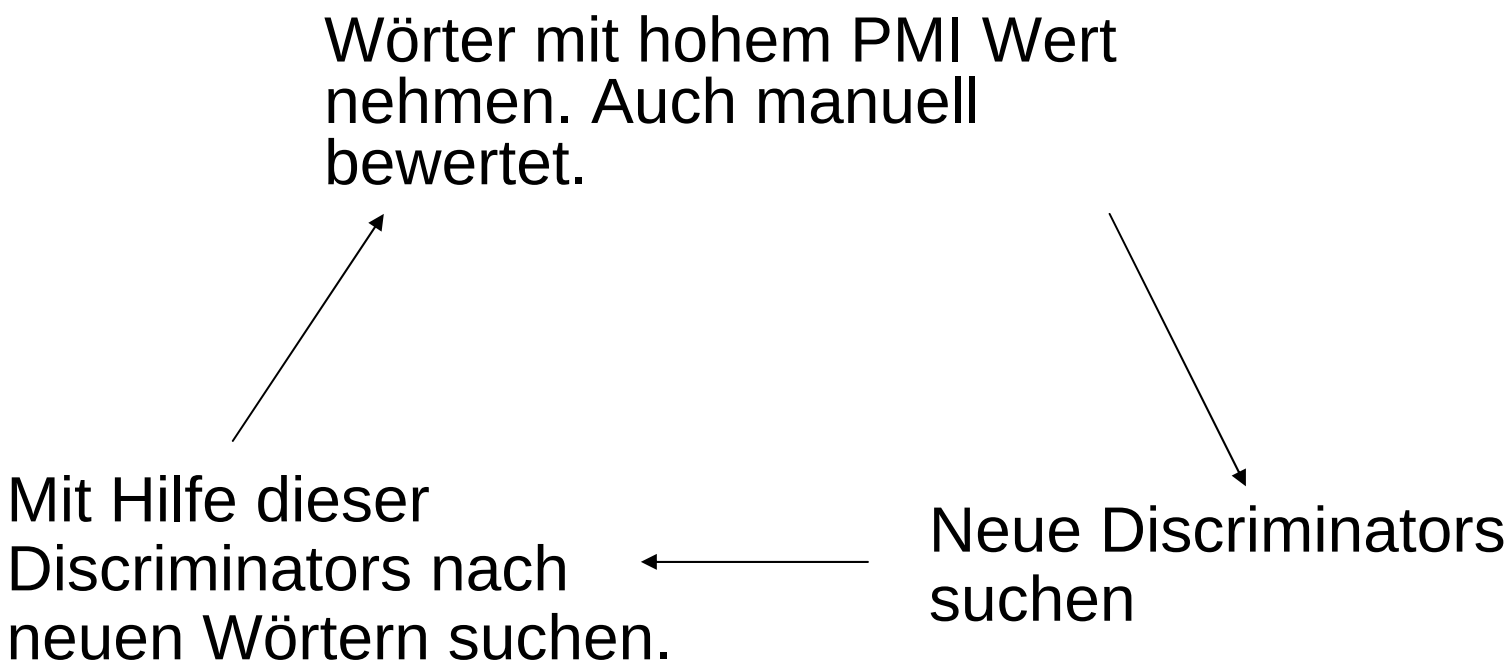
Beispiel: Extractor vermutet, dass Liege der Name einer Stadt ist. Falls PMI zwischen „Liege“ und der Phrase „city of Liege“ hoch ist, ist das ein Zeichen, dass „Liege“ tatsächlich eine Instanz der Klasse „City“ ist.

Assessor: verwendet naive Bayes Classifier

- Berechnet den Wahrscheinlichkeitswert für einen Fakt abhängig von Merkmalen, wobei die Merkmale in diesem Fall gleich den Discriminators sind.
- Sobald die Wahrscheinlichkeit eines Faktes überhalb eines bestimmten Wertes liegt, wird dieser als gültiges Exemplar in die Datenbank aufgenommen, andernfalls verworfen.

Für mehr Information schau:
http://www.cs.washington.edu/research/knowitall/papers/www-paper.pdf

Bootstrapping



Bootstrapping: Beispiel

1. Wir haben schon die Phrasen nach der Regel *NP1 such as NP2* extrahiert und wissen, dass Munich, London und Vienna höchstwahrscheinlich Instanzen der Klasse „City“ sind.
2. Jetzt nehmen wir diese Namen und schauen, in welchen Kontext sie vorkommen.
3. Wir stellen fest, dass wir mehrere ähnlichen Phrasen finden: *„Cities including London“, „cities including Vienna“.*
4. Hier stellt KnowItAll fest, dass man eine neue Regel benutzen kann und zwar: *NP1 including NP2*. Diese Regel können wir also weiter verwenden um weitere Instanzen der Klasse City zu finden.
5. Jetzt haben wir noch einige Instanzen gefunden und können entweder wieder zum Punkt 2 gehen um noch mehr Regeln (Discriptors) zu finden oder Bootstrapping beenden und die Ergebnisse an den Extractor schicken (schau Folie 10 bzw. Ausgabeblatt).

Das Terminieren von KnowItAll

Wichtig: KnowItAll muss wissen, wenn man mit der Suche aufhört.

Beispiel: suche nach US Staaten, wenn das System nach ca. 50 Instanzen nicht aufhört zu suchen, dann sind die weiteren Staaten fast immer Fehler. Ohne geregelter Terminierung hat KnowItAll 3927 US Staaten gefunden.

Terminierung: Signal-to-noise ratio (STN)

- STN ist die Proportion der Extraktionen mit hoher Wahrscheinlichkeitswert über die Extraktionen mit niedriger Wahrscheinlichkeitswert.
- Falls STN der letzten Extraktionen unter 0,05 liegt, findet KnowItAll zwanzig mal soviele fehlerhaften Instanzen, als die richtigen (Also aus 105 Ergebnissen sind 5 Instanzen relevant, 100 falsch).
- Man legt also mit Hilfe von Präzision und Recall Werten den STN Wert fest, bei dem die KnowItAll die Suche aufhört.

Wie sieht das im Web aus

KnowItAll v.2

Database

Database: KnowItAll Connect Show summary

Search hide

Search

Extended Search hide

Predicates:

instanceOf (Corporation)
ceoOf (Corporation , Ceo)
presidentOf (Corporation , President)
vicePresidentOf (Corporation , VicePresident)
directorOf (Corporation , Director)
managerOf (Corporation , Manager)
cfoOf (Corporation , Cfo)

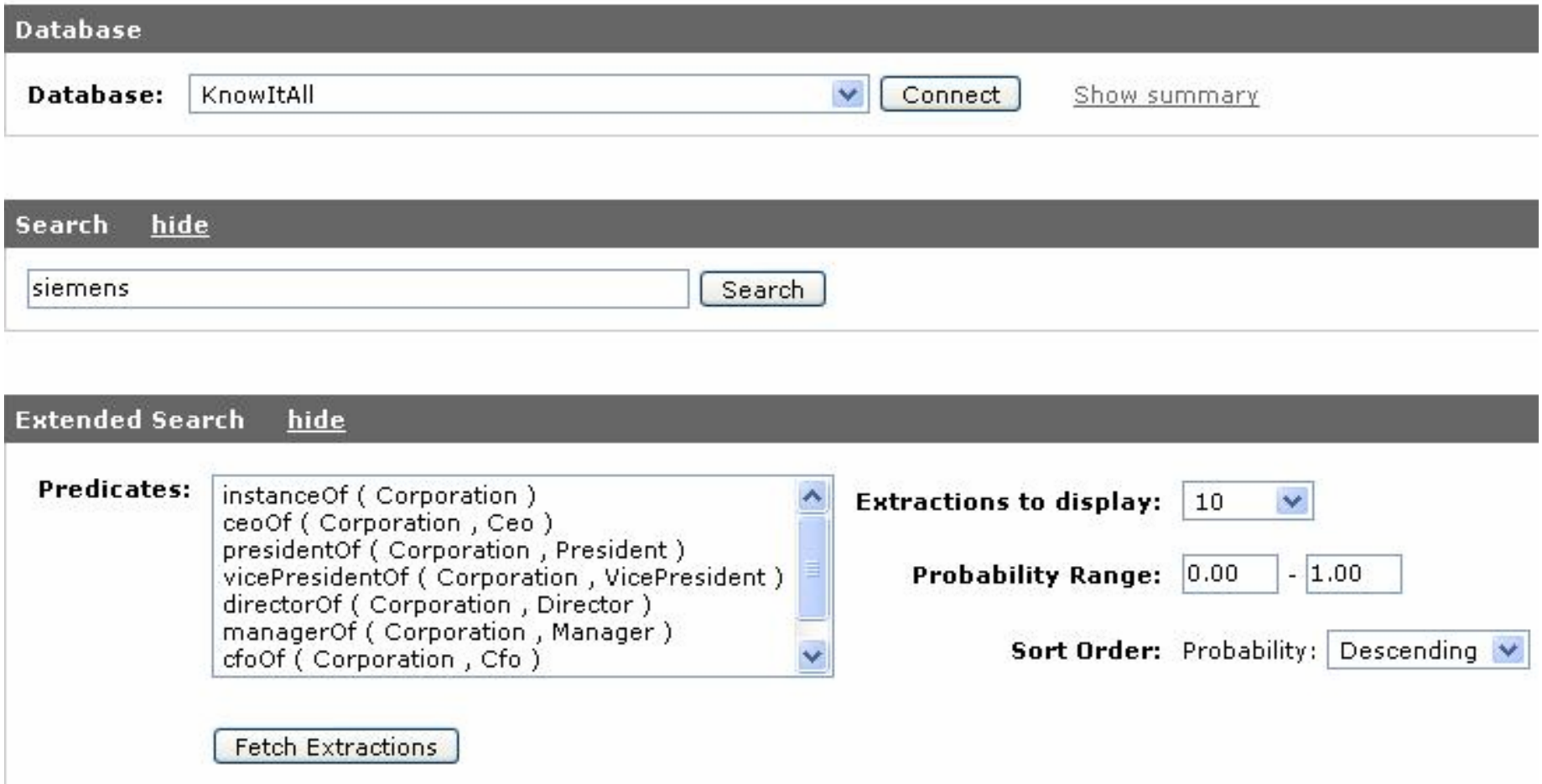
Extractions to display: 10

Probability Range: 0.00 - 1.00

Sort Order: Probability: Descending

Fetch Extractions

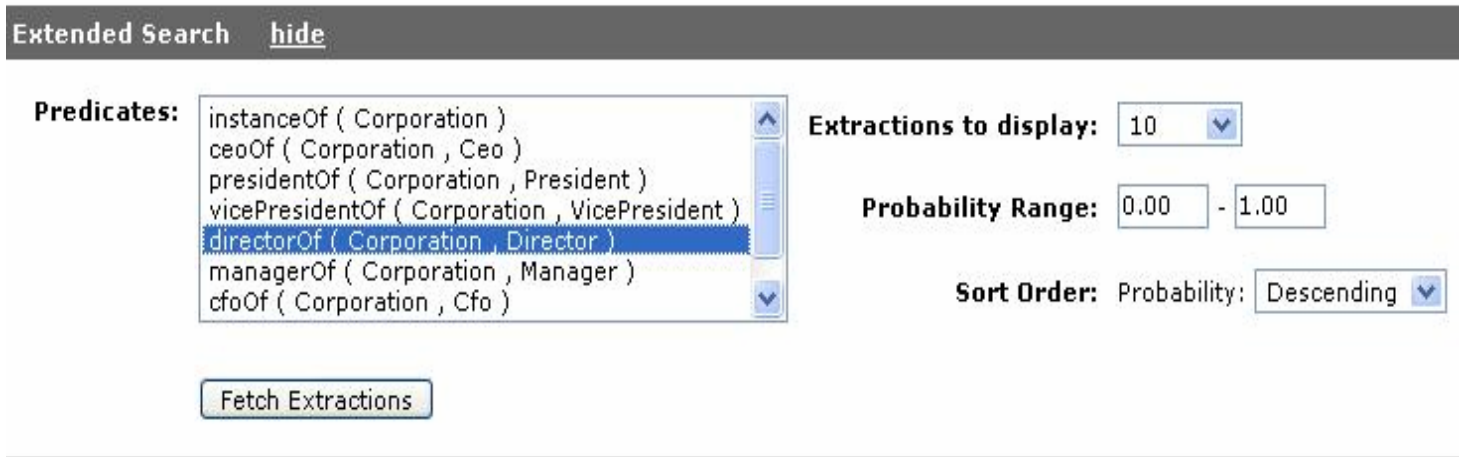
Im Web: Suche im Suchfeld



Im Web: Suchergebnisse für die Suche im Suchfeld

Predicate	Extraction	Probability
instanceOf (Corporation)	Bosch-Siemens	0.8882983000
instanceOf (Corporation)	BSH Bosch Siemens	0.1000000000
instanceOf (Corporation)	BSH Bosch und Siemens Hausgeraete	0.1000000000
instanceOf (Corporation)	BSH BOSCH UND SIEMENS HAUSGERÄTE	0.2500000000
instanceOf (Corporation)	CONTINENTAL AG AND SIEMENS VDO AUTOMOTIVE AND ZF FRIEDRICHSHAFEN AND SAP AND INA-HOLDING AND ROBERT BOSCH	0.2500000000
instanceOf (Corporation)	EFFICIENT SIEMENS BUSINESS SOLUTION	0.2500000000
instanceOf (Corporation)	EFFICIENT SIEMENS SPEEDSTREAM	0.2500000000
instanceOf (Corporation)	Fujitsu Siemens Computers S.p.A	0.6247858000
instanceOf (Corporation)	Fujitsu Siemens Computers	0.9317479000

Im Web: Suche im Prädikatfeld



Extraction	Probability
Intel, George Goodman	0.3938680000
General Motors, Byron McCormick	0.3494881000
IBM, Dave Snowden	0.3494881000
Intel, Gerald Marcyk	0.3494881000
Microsoft, Aubrey Edwards	0.3494881000
Microsoft, Craig Spiezle	0.3494881000
Sony, Sam Raimi	0.3494881000
Apple, Frank Casanova	0.3123929000
General Motors, Mark Hogan	0.3123929000
General Motors, Terry Connolly	0.3123929000

Im Web: Zusammenfassung

- Suche im Search-Feld: Ergebnisse ähnlich zu Ergebnissen von Suchmaschinen.
- Suche im Prädikat-Feld: Ergebnisse sind in Form einer Datenbank dargestellt.

Nachteile von KnowItAll

- Kombinieren von beiden Feldern leider nicht möglich.

- Sehr lange Wartezeiten bei der Suche.

Verbesserung :

KnowItNow

Wiederholung

Die Abfrage einer standard Suchmaschine besteht aus 2 Phasen :

1. Angeben eines Terms in der Abfragefeld um alle URLs zu finden, die den angegebenen Term enthalten (z.B.”cities such as”)

2. Arbeitsschritte für jede gefundene URL:

a) Beschaffen des Dokumenteninhalts
b) Finden angegebenen Terms
   in Dokumententext (z.B.“cities such as”)
c) Bestimmen durch einen Satzerkenner, ob
   der Text nach dem Term den linguistischen
   Anforderungen entspricht
d) wenn dies der Fall ist, den String
   zurückgeben

Einführung der Bindings Engine (BE)

BE beinhaltet:

- *typed variables* (wie z.B. *NounPhrase*)
- *string-processing functions*
  (wie z.B. *head(X)* oder *ProperNoun(X)*)
- standard Anfrageterme (z.B. *cities such as*)

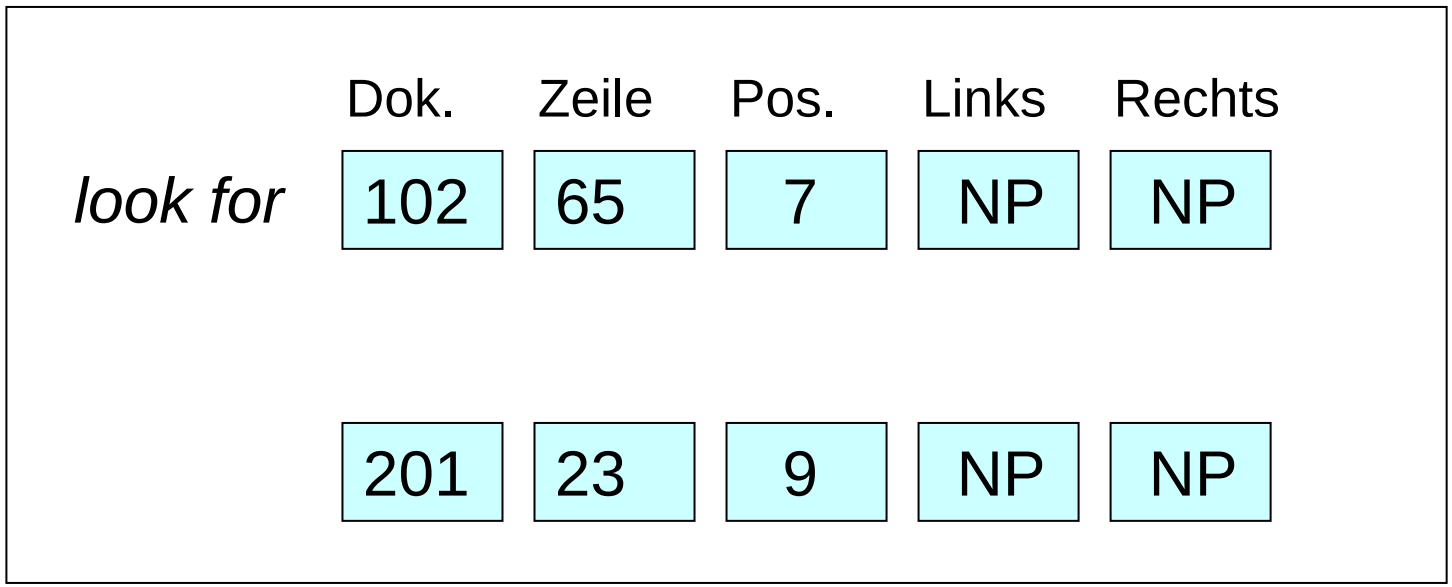
Abfragebeispiel:

president Bush <*Verb*>
cities such as *ProperNoun(Head(<NounPhrase>))*
<*NounPhrase*> is the capital of <*NounPhrase*>

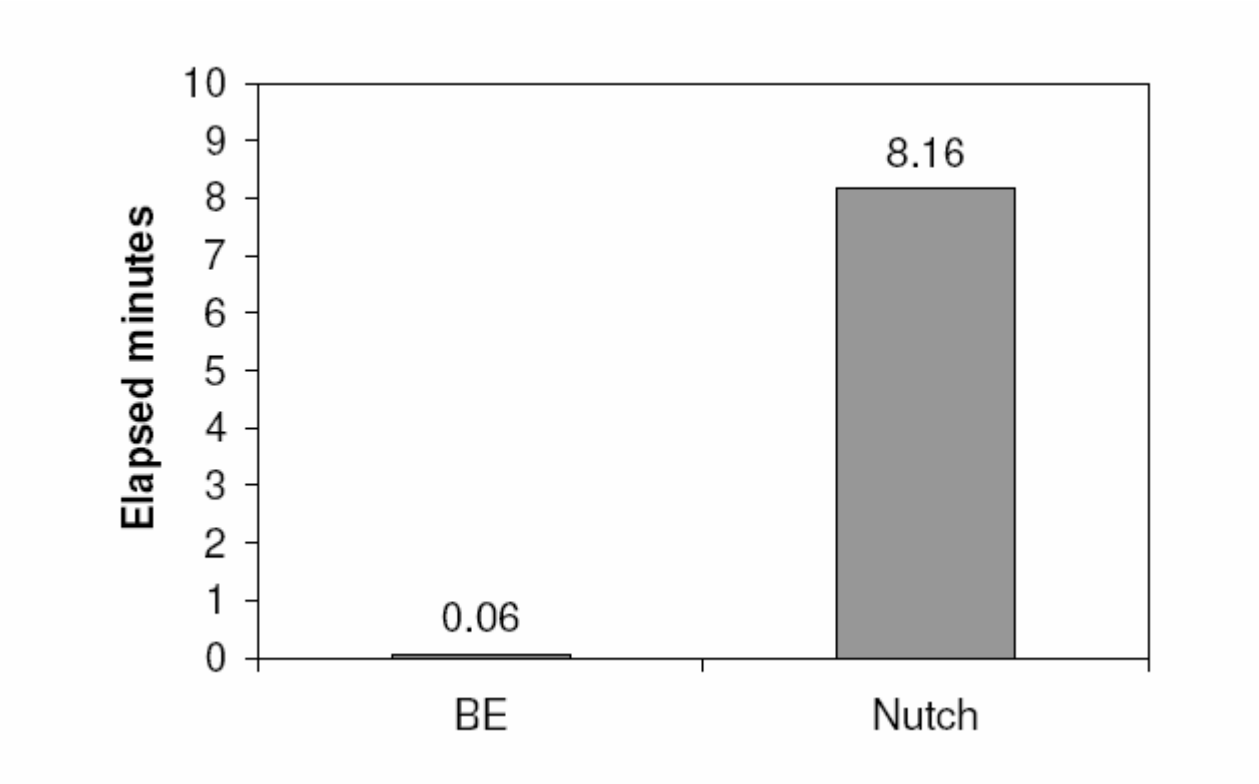
Suchanfragen, die *typed variables* und *string processing functions* enthalten, erlauben es den NLP Anwendungen effizient zu arbeiten ohne das original Dokument während des Anfrageprozesses downzuloaden.

Neighborhood index - erweiterte invertierte Index Struktur

Beispiel:

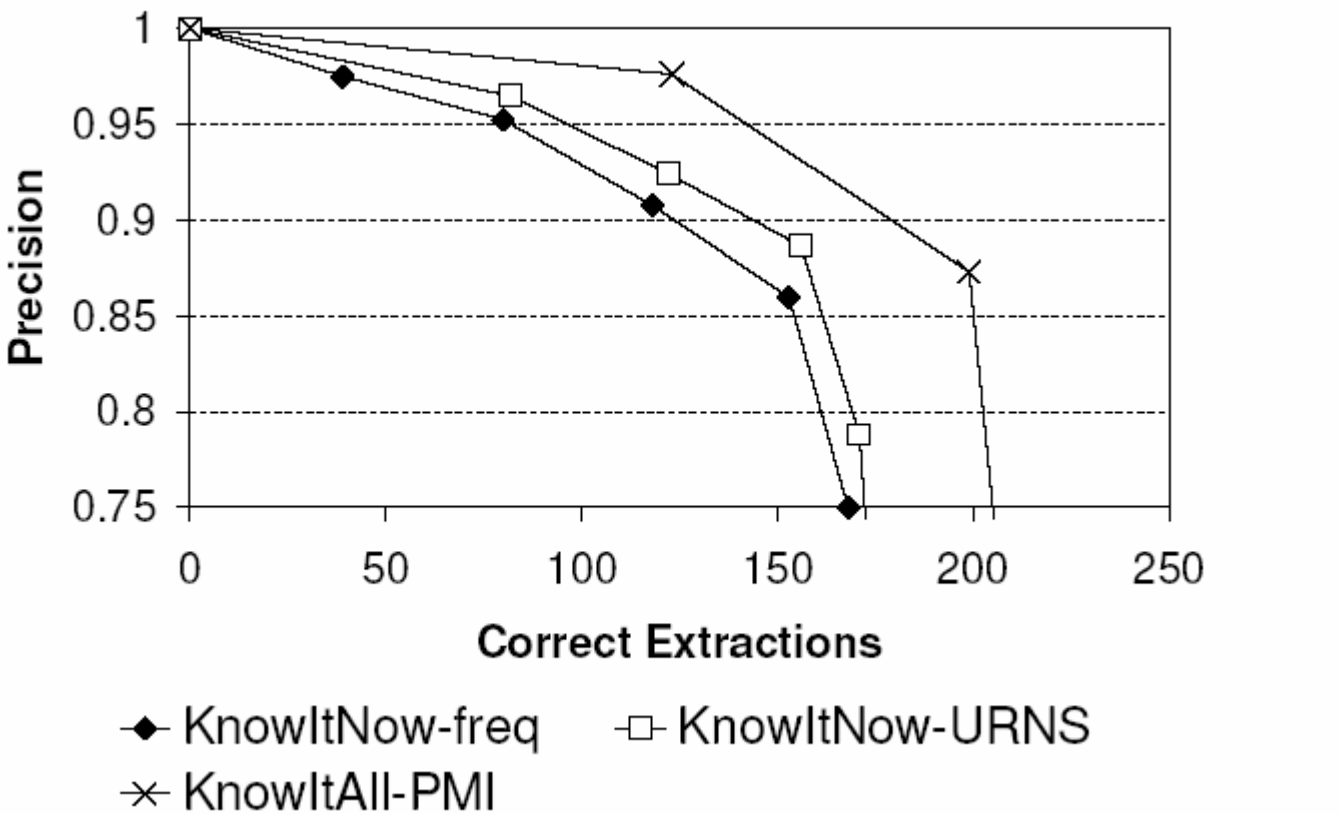


Vergleich der Durchschnittszeiten für die Rückgabe der relevanten *bindings*

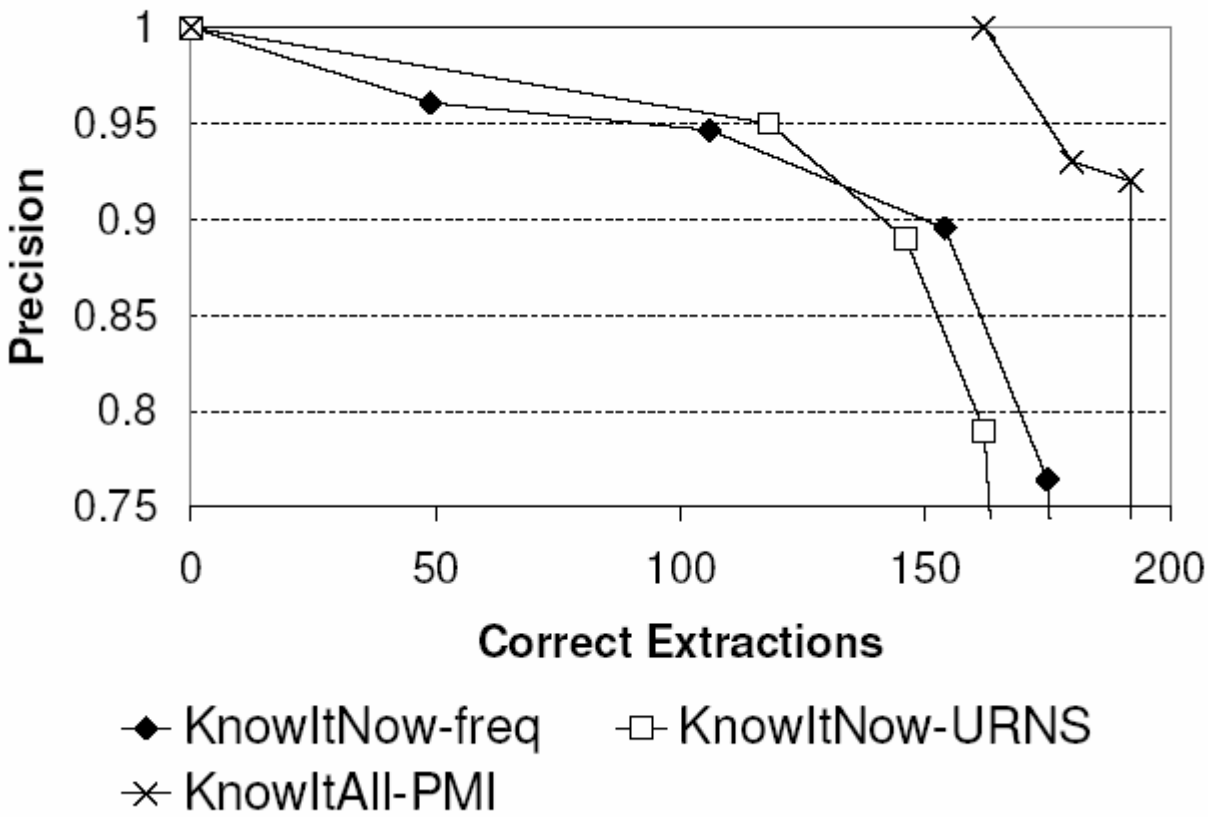


Vergleich der precision/recall Kurven von KnowItAll und KnowItNow

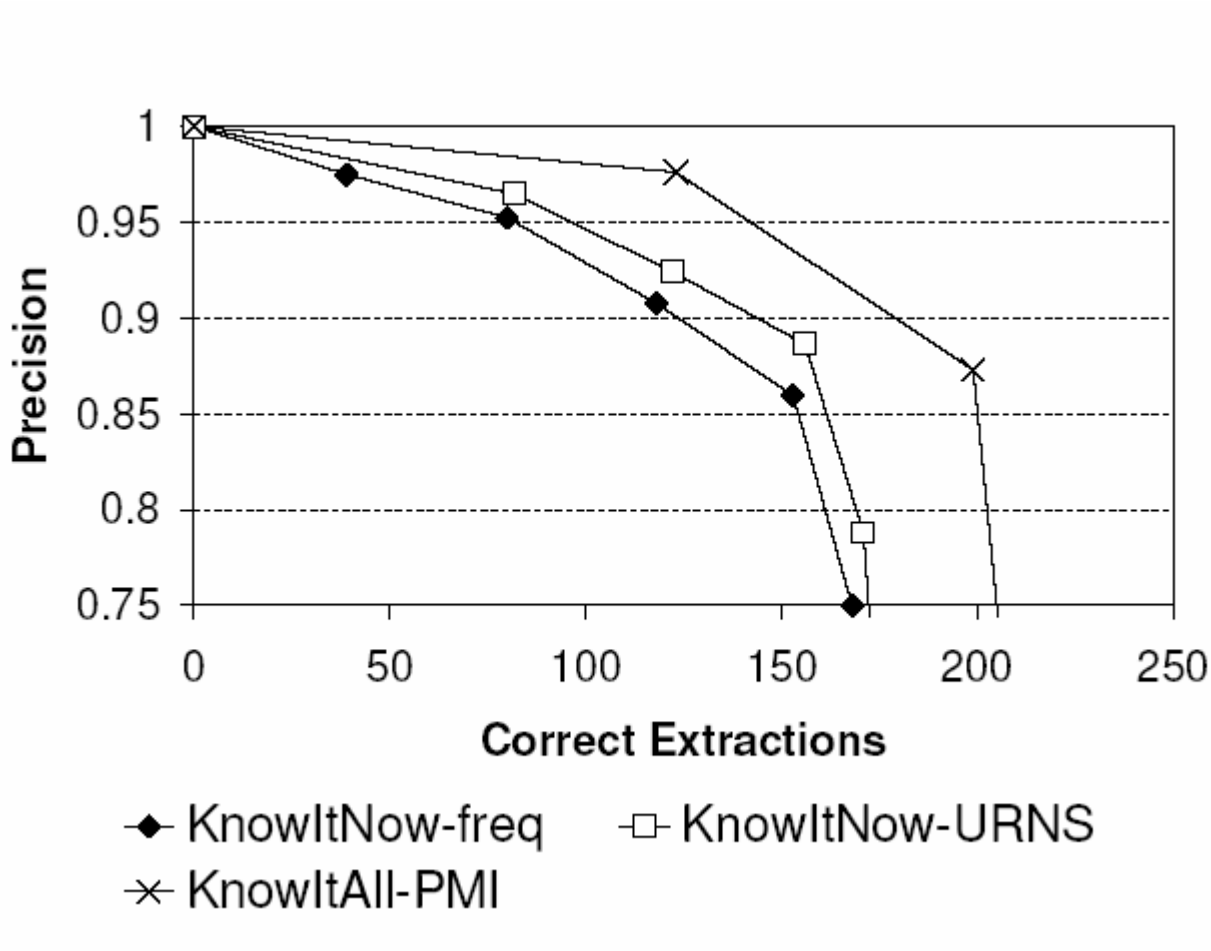
Relation: Country



Relation: CapitalOf



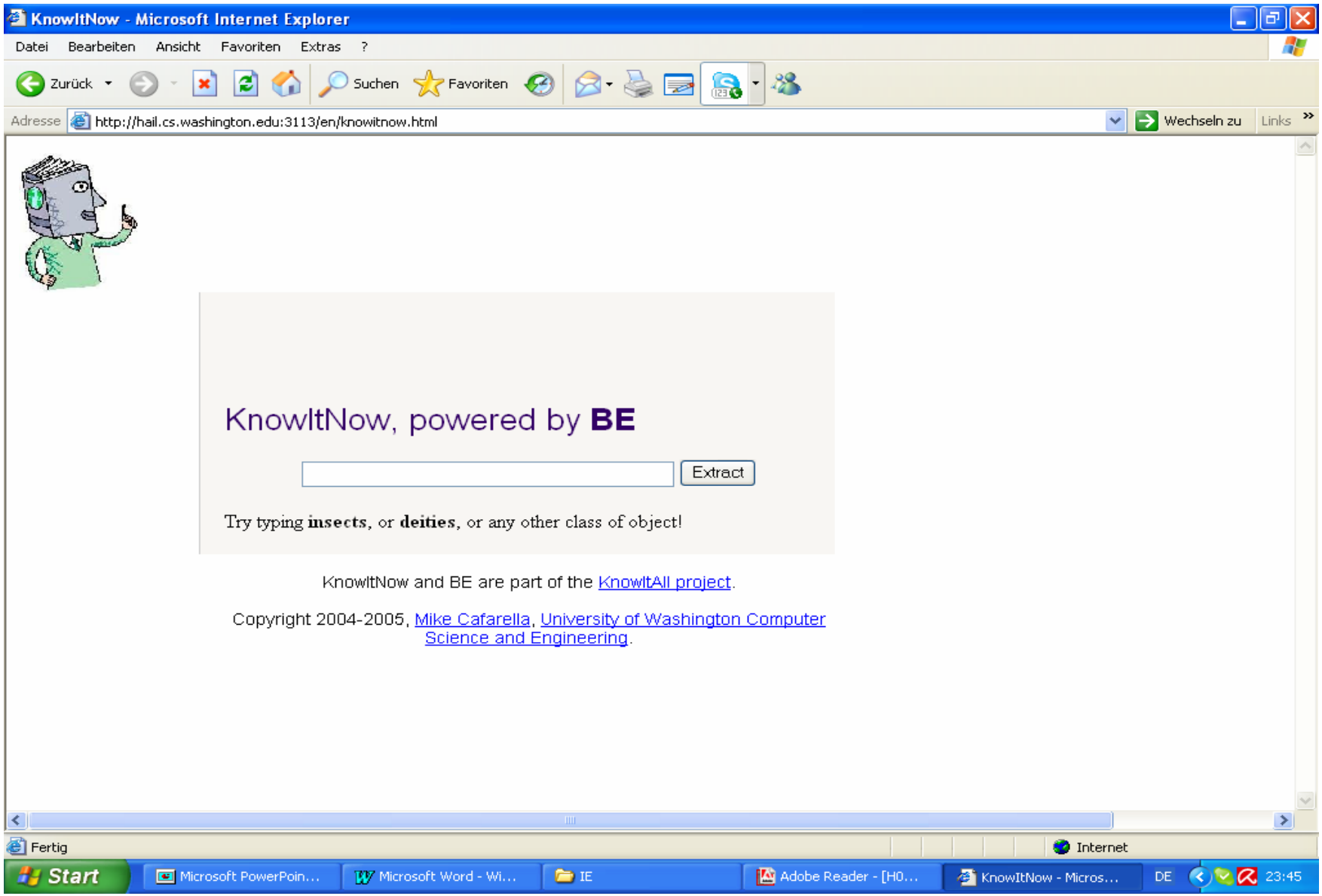
Relation: Corp



Vergleichswerte eines Abfrageergebnises für vier Relationen bei KnowItAll vs KnowItNow

	Google Queries		Time		Extractions		Extractions per minute		
	NOW	ALL	NOW (sec)	ALL (hrs)	NOW	ALL	NOW	ALL	ratio
Corp	0 (16)	201,878	42	56.1	23,128	23,617	33,040	7.02	4,707
Country	0 (16)	35,480	42	9.9	161	203	230	0.34	672
CeoOf	0 (6)	263,646	51	73.2	2,402	5,823	2,836	1.33	2,132
CapitalOf	0 (6)	17,216	55	4.8	169	192	184	0.67	275

Wie sieht das im Web aus



Suchergebnis

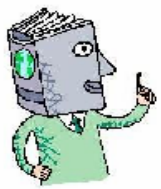
KnowItNow: search results - Microsoft Internet Explorer

Datei Bearbeiten Ansicht Favoriten Extras ?

Zurück Suchen Favoriten

Adresse http://hail.cs.washington.edu:3113/knowitnowsearch.jsp?query=cities

Wechseln zu Links



Über FAQ Ent

cities Extract

Found 6825 unique instances of 'cities' in 0.0 seconds. (Query was cached.)
(Only showing the first 1000)

#	'cities'	Score
1.	New York	1767
2.	Chicago	1390
3.	Paris	1281
4.	Los Angeles	1068
5.	London	898
6.	San Francisco	782
7.	Boston	769
8.	Milan	636
9.	Baghdad	537
10.	Philadelphia	501
11.	Shanghai	473
12.	Beijing	473

Fertig Internet

Start Microsoft PowerPoin... Microsoft Word - Wi... IE Adobe Reader - [H0... KnowItNow: search ... DE 23:47

Zusammenfassung *KnowItNow*:

- IE in KnowItNow ist unabhängig von Abfragen in Internet Suchmaschinen.
- erstes System, das die Fähigkeit besitzt, zehntausende Fakten in Minuten statt Tagen zu extrahieren.
- zwei bis dreimal höhere Extraktionsrate im vergleich zum KnowItAll.

TextRunner

Basiert auf der Idee der Open Information Extraction (OIE)

Ist eine Paradigma in der Form eines Tupels: $t=(e_1, r_{1,2}, e_2)$ wo e_1 und e_2 Nominalgruppen und $r_{1,2}$ Relationen wischen denen darstellt

Besteht aus drei Modulen:

- Self-Supervised Learner
- Single-Pass Extractor
- Redundancy-Based Assesor

Learner

- Korrespondiert mit dem Bootstrap bei KnowItAll.
- Benötigt keine Regelvorlagen und Klassennamen.
- Eingangsparameter ist ein kleiner Auszug aus dem Korpus.

Abläufe bei Learner

- Der Auszug aus dem korpus wird geparst.
- Bei jedem gefundenen Satz wird Identifikation von NominalGruppen und Relationen versucht (anhand von syntaktischen Vorgaben).
- wird entweder verworfen oder als „positives Exemplar“ zum trainieren des Bayes-Klassifikators verwendet.

Extractor

- Korrespondiert mit den Extraktionsläufen bei KnowItAll.
- Der Korpus wird einmal komplett durchlaufen.
- NominalGruppen werden identifiziert um später als Klassennamen und Exemplare isoliert zu werden.

Assesor

- Jedes gewonnene Tupel,bestehend aus Klassenname,Relation und Exemplare,wird von Bayes-Klassifikator bewertet.
- Dabei wird ihm eine Wahrscheinlichkeit zugeordnet.
- Aufgrund der wird er verworfen oder in die Datenbank aufgenommen.

Zusammenfassung

- Ähnlichkeit vom TextRunner mit seinem Vorgängersystem KnowItAll bei der Vorbereitung des Klassifikators.
- Die Regelvorlagen sind bei TextRunner in Parser eingebaut.
- Bei beiden Systemen Klassifikation über naive Bayes-Klassifikator.
- Unterschied: bei TextRunner sind Tupel Domänen-unabhängig.

Literaturangabe:

- http://turing.cs.washington.edu/papers/www-paper.pdf
- http://acl.ldc.upenn.edu/H/H05/H05-1071.pdf (01.05.2007)
- http://turing.cs.washington.edu/papers/ijcai07.pdf
- http://page.mi.fu-berlin.de/~best/uni/UnscharfeDaten/IE_Systems.pdf