



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

Information Extraction aus natürlichsprachlichen Phrasen am Beispiel von Hotelsuchanfragen

Timm Heuss

25.11.2012

Summary

- Schaffung einer **neuen Eingabemethode** von **natürlichsprachlichen Hotelsuchanfragen** für das Touristik-Portal HRS.
- Entwicklung einer schnellen und erweiterbaren Lösung für die **Information Extraction: Query Analysis**.
- Im **forward-chaining**-Verfahren werden **Parts-of-Speech** und **Konstituenten** durch eine **regelbasierte Named Entity Recognition** ausgewertet und **Constraints** erzeugt.
- Konzeption und Implementierung einer angepassten **Apache UIMA-Pipeline**, Integration von **JBoss Drools**.
- Auswertung mit den üblichen Methoden des **Information Retrievals** durch die Maße **F**, **Precision** und **Recall**.

Agenda

1

1. Motivation
2. Grundlagen
3. Architektur & Implementierung
4. Auswertung
5. Demo
6. Fazit & Ausblick

Motivation

HRS.com – Stand heute

The screenshot displays the HRS.com homepage. At the top, a red navigation bar contains the HRS logo, a home icon, and links to 'Angebote', 'Themenhotels', 'Geschäftsreise', 'Gruppen', 'My HRS', 'Ändern & Stornieren', and 'Kundenservice 24/7 +49 221 2077 600'. Below the navigation bar, a search form is positioned on the left, and a promotional banner for Hamburg is on the right.

Search Form:

- Ihr Ziel:**
- Anreise:**
- Abreise:**
- Einzelzimmer:**
- Doppelzimmer:**
- Erwachsene:**
- Kinder:**
- ▼ Weitere Suchkriterien**
- Hotel suchen** (button)

Promotional Banner:

- IM ZENTRUM HAMBURG** (red box)
- HRS DEALS** (clock icon)
- 50% RABATT**
- REGULÄR: ~~88,00 €~~
- DOPPELZIMMER (für 2 Pers.)**
- 44,00 €**
- ZUM ANGEBOT** (red button)

Footer:

- 40 Sterne Service
- HRS Deals** (highlighted)
- HRS Vorteile
- Hotel Specials

Motivation

HRS.com – Stand heute

The screenshot shows the HRS.com website interface. At the top, there is a red navigation bar with the HRS logo and links to 'Angebote', 'Themenhotels', 'Geschäftsreise', 'Gruppen', 'My HRS', 'Ändern & Stornieren', and 'Kundenservice 24/7 +49 221 2077 600'. Below the navigation bar, the main content area features a large background image of a Hamburg harbor scene. On the left, there is a search form with the following fields: 'Ihr Ziel' (with a placeholder 'Ort, Region, Insel, Adresse, PLZ'), 'Anreise' (21.10.12), 'Abreise' (22.10.12), 'Einzelzimmer' (with a placeholder 'Bitte geben Sie die gewünschte Zimmeranzahl ein.'), 'Doppelzimmer' (with a placeholder 'Bitte geben Sie die gewünschte Zimmeranzahl ein.'), 'Erwachsene' (with a placeholder), and 'Kinder' (0). Below these fields is a 'Weiter Suchkriterien' link and a 'Hotel suchen' button. On the right, there is a promotional banner for 'HRS DEALS' featuring a 50% discount. The banner includes the text 'IM ZENTRUM HAMBURG', 'REGULÄR: 88,00 €', 'DOPPELZIMMER (für 2 Pers.)', and a large red price tag '44,00 €' with a 'ZUM ANGEBOT' button. At the bottom of the banner, there are four buttons: '40 Sterne Service', 'HRS Deals', 'HRS Vorteile', and 'Hotel Specials'.

HRS Mobil Deutsch

HRS

Angebote Themenhotels Geschäftsreise Gruppen My HRS Ändern & Stornieren Kundenservice 24/7 +49 221 2077 600

Ihr Ziel
Ort, Region, Insel, Adresse, PLZ
Geben Sie bitte ein Reiseziel ein.

Anreise 21.10.12 Abreise 22.10.12

Einzelzimmer
Bitte geben Sie die gewünschte Zimmeranzahl ein.

Doppelzimmer
Bitte geben Sie die gewünschte Zimmeranzahl ein.

Erwachsene Kinder 0

▼ Weitere Suchkriterien

Hotel suchen

IM ZENTRUM HAMBURG

HRS DEALS
50% RABATT
REGULÄR: ~~88,00 €~~
DOPPELZIMMER (für 2 Pers.)
44,00 €
ZUM ANGEBOT

40 Sterne Service HRS Deals HRS Vorteile Hotel Specials

Motivation

Google setzt die Maßstäbe

Anmelden



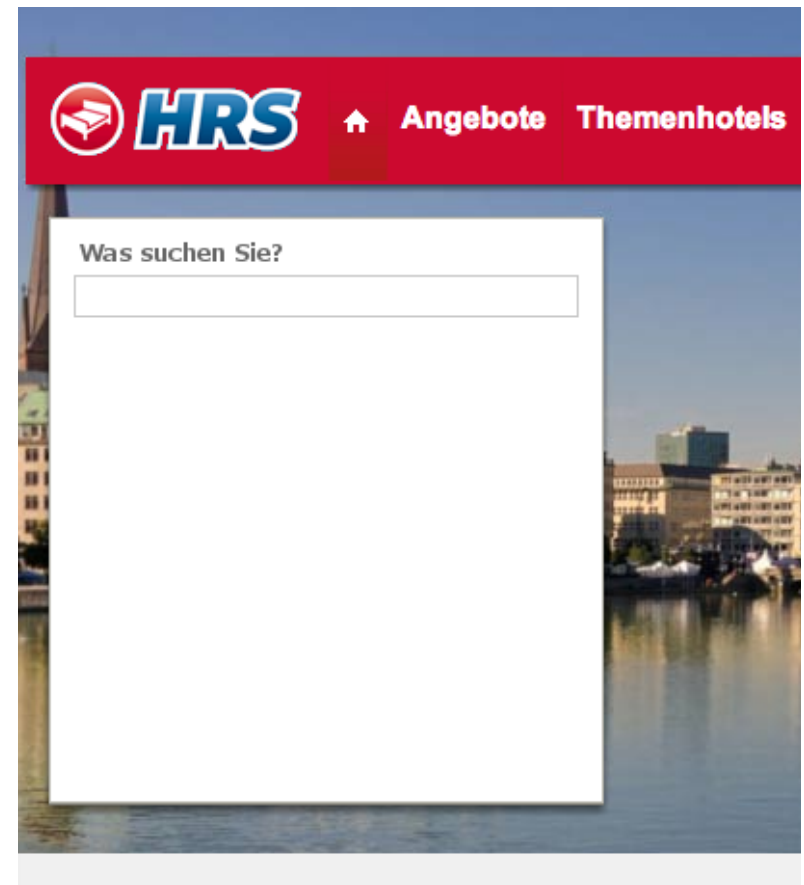
Google-Suche

Auf gut Glück!

Motivation

Ziel dieser Arbeit

- Entwicklung einer alternativen Eingabe für natürlichsprachliche Hotelsuchen
- Anforderungen
 - Gute Erkennungsleistung
 - Fehlertoleranz bei der Eingabe
 - Intuitive Nutzung, keine Vorgabe an den Benutzer
 - Verarbeitungszeit im Sekundenbereich
 - Gute Erweiterbarkeit
 - Erkennung von Hotel- und Zimmeraustattungen



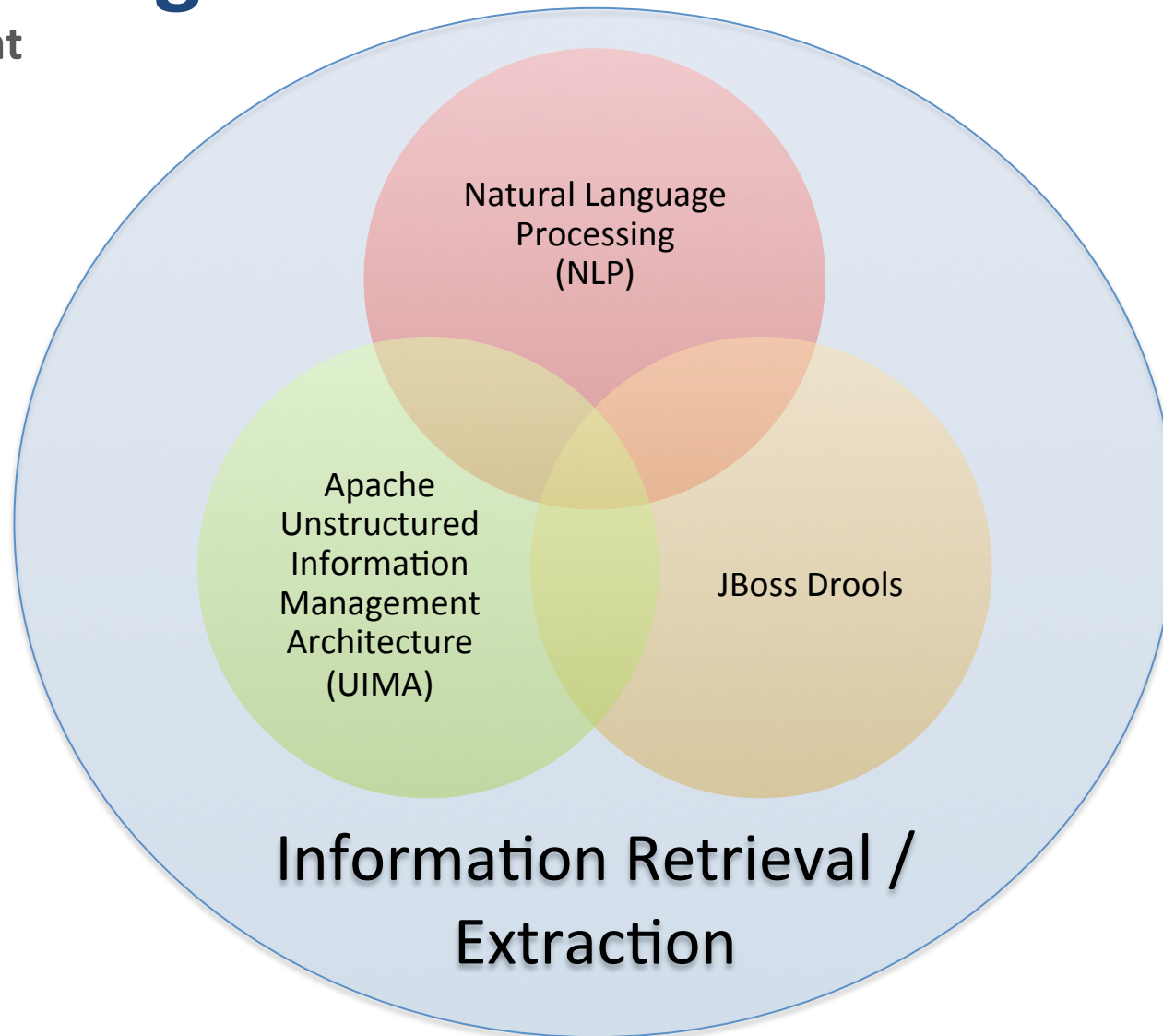
Agenda

2

1. Motivation
2. Grundlagen
3. Architektur & Implementierung
4. Auswertung
5. Demo
6. Fazit & Ausblick

Grundlagen

Übersicht



Grundlagen

Natural Language Processing

- Tokenization / Segmentierung
 - Herunterbrechen eines Textes in (sinnvolle) Teilstücke: **Tokens**
 - Basisoperation für viele anspruchsvolle Textverarbeitungsoperationen
 - Nicht trivial:

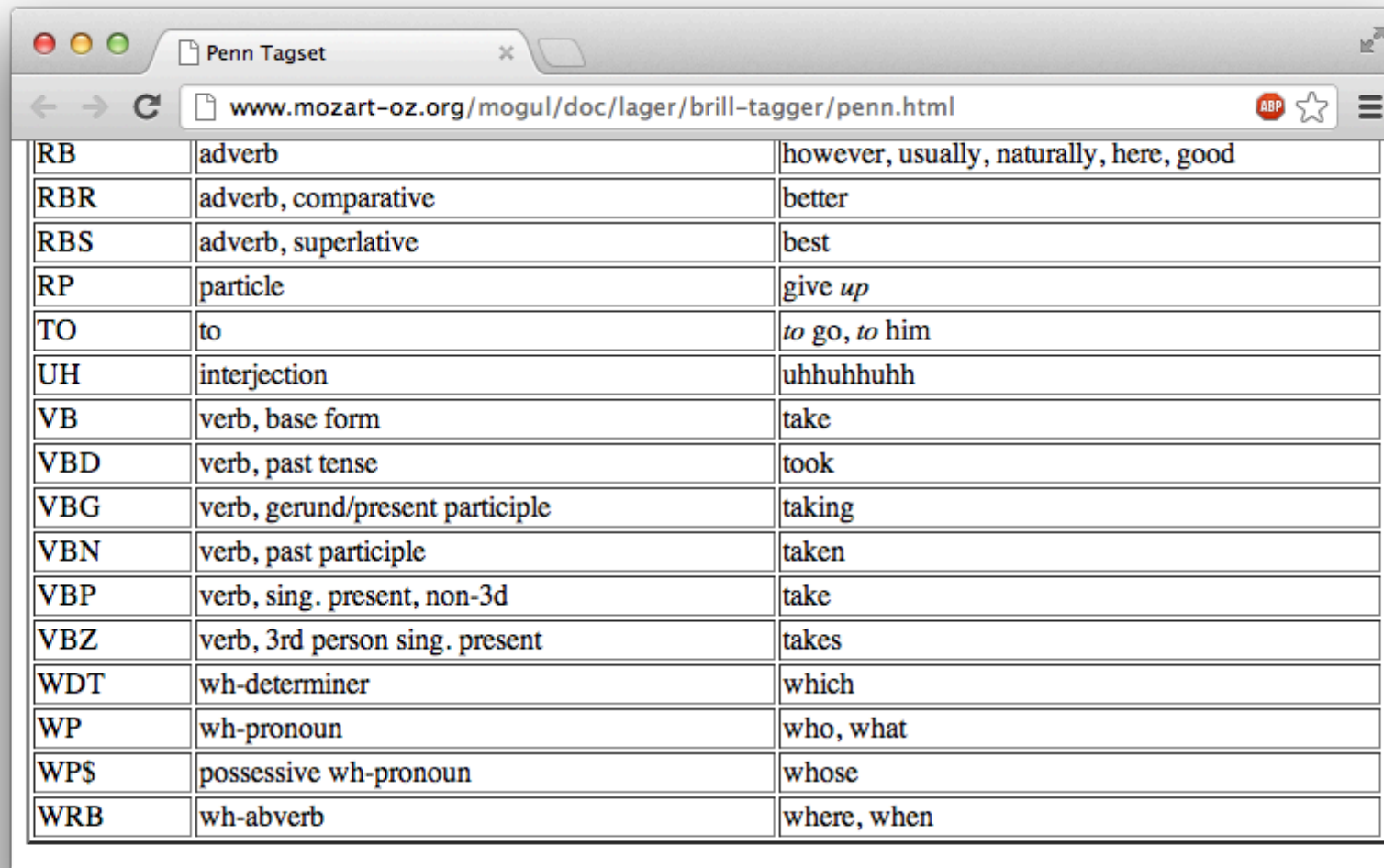
that 's a problem	4 Tokens
Daniel O'Brian	2 Tokens

- Parts-Of-Speech-Tagging
 - Beim Parts-Of-Speech-Tagging (PoS-Tagging) werden Worte zu **lexikalischen Wortklassen** zugewiesen:

Haus	->	Nomen
spielen	->	Verben
 - Die Wortklassen können auch feiner gewählt werden (je nach Sprache, Anwendungsgebiet) -> **Tagset**

Grundlagen

Natural Language Processing - Penn Tagset



RB	adverb	however, usually, naturally, here, good
RBR	adverb, comparative	better
RBS	adverb, superlative	best
RP	particle	give <i>up</i>
TO	to	<i>to</i> go, <i>to</i> him
UH	interjection	uhhuhhuhh
VB	verb, base form	take
VBD	verb, past tense	took
VBG	verb, gerund/present participle	taking
VBN	verb, past participle	taken
VBP	verb, sing. present, non-3d	take
VBZ	verb, 3rd person sing. present	takes
WDT	wh-determiner	which
WP	wh-pronoun	who, what
WP\$	possessive wh-pronoun	whose
WRB	wh-abverb	where, when

Grundlagen

Natural Language Processing - Parsing

- Beim **Parsing** werden Sätze syntaktisch analysiert und gemäß einer bestimmten **Grammatik** einer Struktur zugewiesen
- Bei **Konstituentengrammatiken** werden Sätze durch ineinander verschachtelte Teilausdrücke (**Konstituenten**) **vollständig** zusammengesetzt:

My dog also likes eating sausage.

```
(ROOT
  (S
    (NP (PRP$ My) (NN dog))
    (ADVP (RB also))
    (VP (VBZ likes)
      (S
        (VP (VBG eating)
          (NP (NN sausage))))
      (. .)))
```

NP -> PRP + NN

S -> VP + NP

VP -> VBG + NP

NP -> NN

Grundlagen

Apache UIMA - Commons Analysis Structure (CAS)



- Speicher- und Austauschformat für Analyseergebnisse
- Aufbau:

- Typsystem – Schema des Analyseergebnisses

- Standard view** {
 - Subject of Analysis (**Sofa**) – Analysegegenstand **.wav**
 - Feature Structures – Analyseergebnisse (**Fourier-Analysis**)

- Transcript view** {
 - Subject of Analysis (**Sofa**) – Analysegegenstand **.txt**
 - Feature Structures – Analyseergebnisse (**Text-Tokens**)

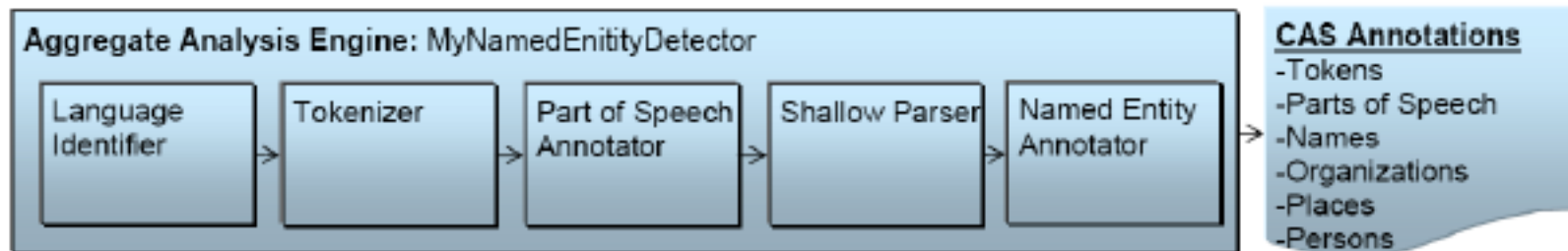
- **CAS-Views** ermöglichen den Umgang mit mehreren Modalitäten (**Beispiel: Sound-Datei**)

Grundlagen

Apache UIMA - Analysis Engine



- Analyse-Instanz innerhalb von UIMA
- Mehrere Analysis Engines können beliebig in Reihe geschaltet oder verschachtelt werden (Primitive Analysis Engine, Aggregate Analysis Engine)
- Eingabe und Ausgabe: CAS
- Erzeugt stand-off-Annotations für eine Sofa und speichert diese innerhalb der gegebenen CAS



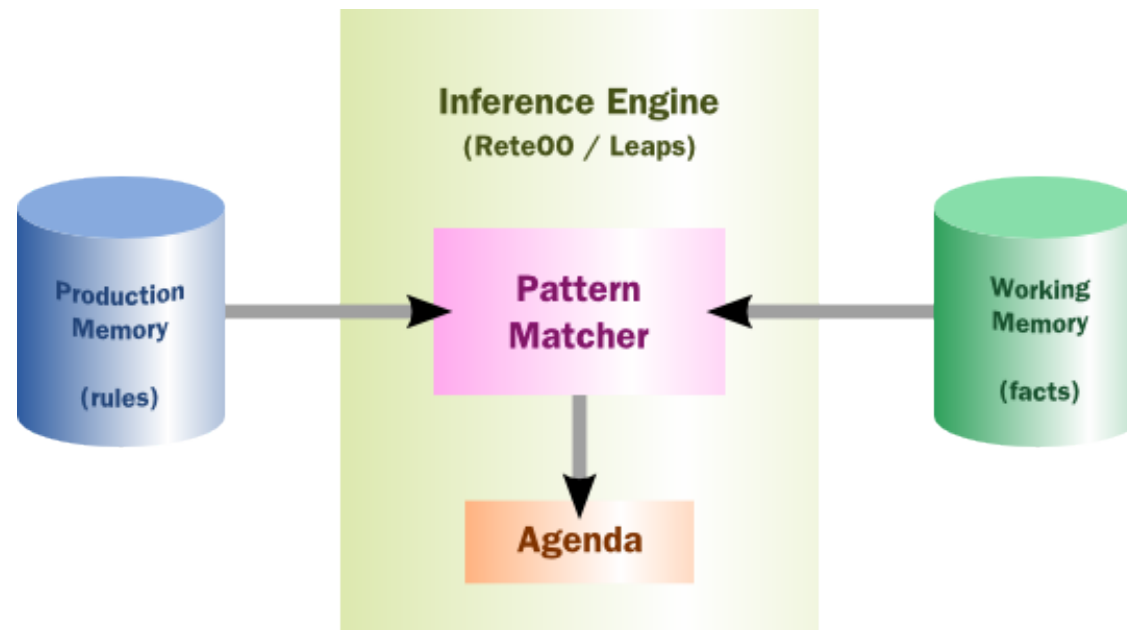
Source: http://uima.apache.org/d/uimaj-2.4.0/overview_and_setup.pdf

Grundlagen

JBoss Drools



- **Rule Engine**, liefert die technische Antwort auf die Frage, wie Wissen in einem System repräsentiert wird



Source: http://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html_single/

Grundlagen

JBoss Drools - Regel

- **Rule Engine**, liefert die technische Antwort auf die Frage, wie Wissen in einem System repräsentiert wird
- Schematischer Aufbau einer Regel:

```
rule "<name>"  
when  
    <conditional element>*  
then  
    <action>*  
end
```

Source: http://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html_single/

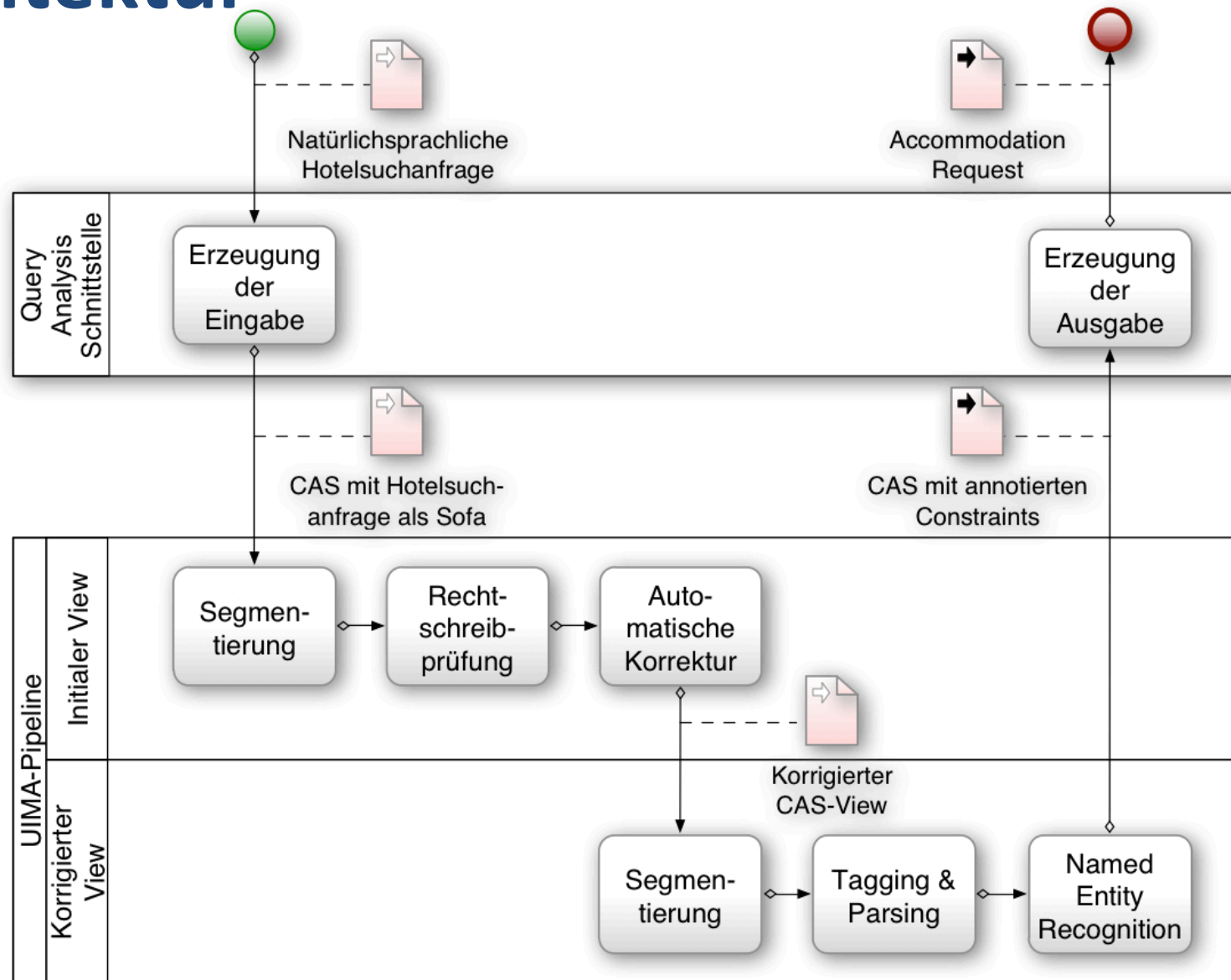
Agenda

3

1. Motivation
2. Grundlagen
3. Architektur & Implementierung
4. Auswertung
5. Demo
6. Fazit & Ausblick

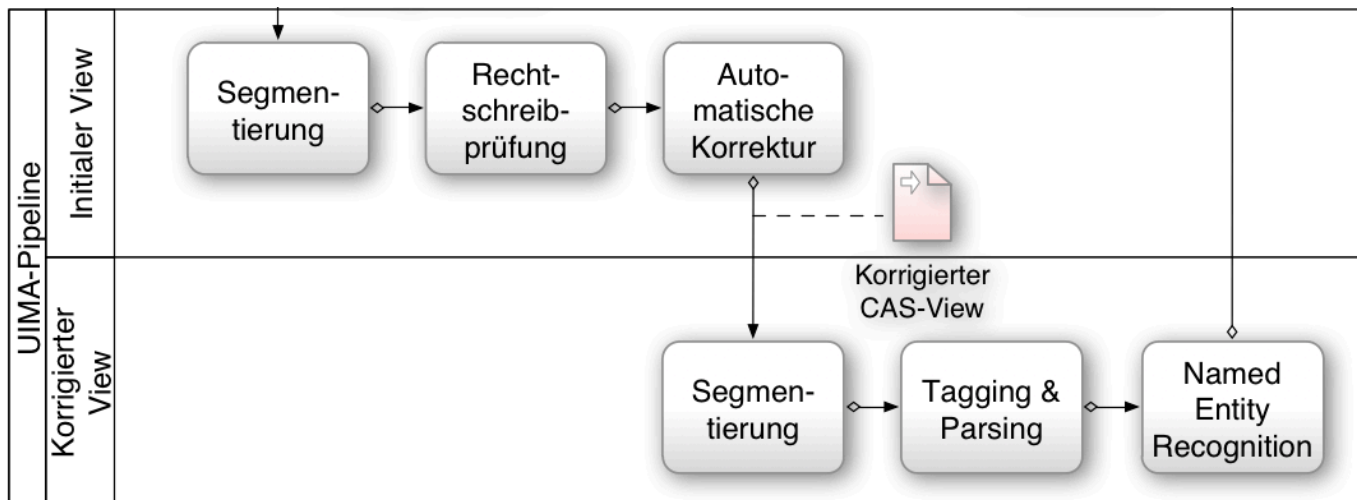
Architektur

Pipeline



Implementierung

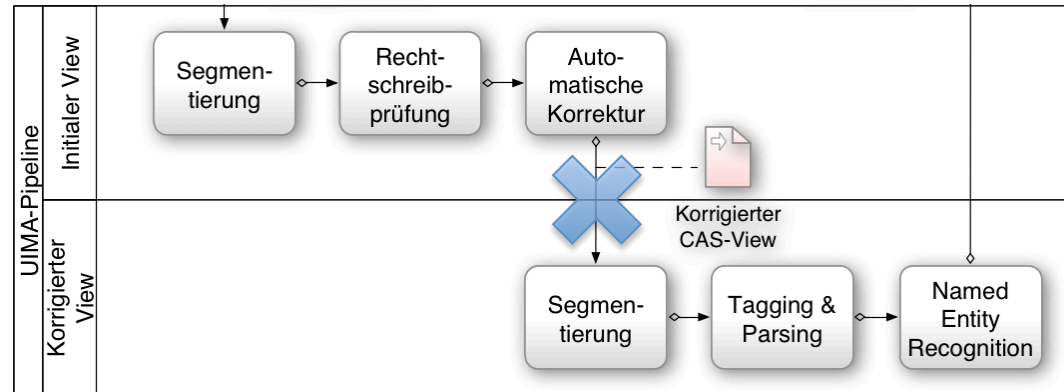
Zusammensetzung der Pipeline



Analyseschritt	Implementierung
Segmentierung	de.tudarmstadt.ukp.dkpro.core.stanfordnlp.StanfordSegmenter
Rechtschreibprüfung	de.tudarmstadt.ukp.dkpro.core.jazzy.SpellChecker
Autom. Korrektur	com.hrs.tourindi.queryanalysis.uima.AutoCorrectionAnnotator
Tagging & Parsing	de.tudarmstadt.ukp.dkpro.core.stanfordnlp.StanfordParser
Named Entity Recognition	com.hrs.tourindi.queryanalysis.uima.DroolsRuntimeAnnotator

Implementierung

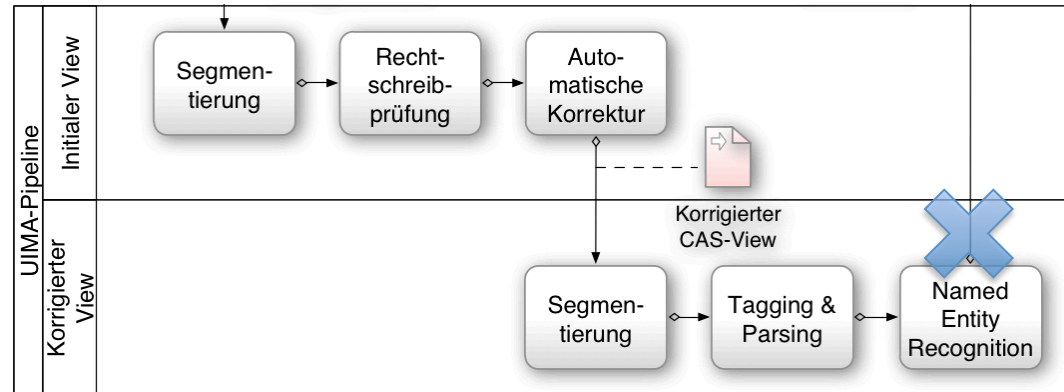
Pipeline-Konstruktion



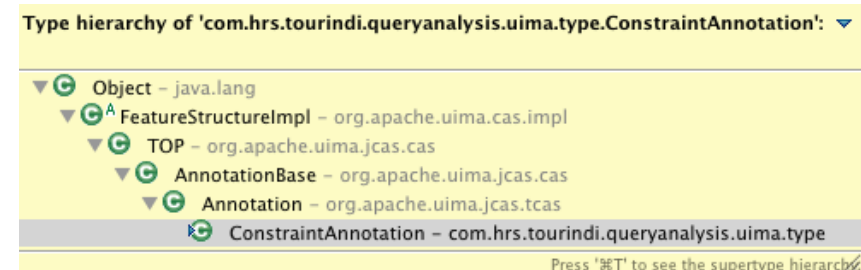
- Erzeugung einer „view-unaware“-Pipeline:
Der verwendete **Standard-View** einer Komponente wird **von außen vorgegeben**
- Flußsteuerung an zentraler Stelle,
Analysis Engines bleiben unberührt

Implementierung

Erweiterung des Typsystems



- ConstraintAnnotation
 - Repräsentiert gefundene Constraints **innerhalb einer CAS**
 - Spezialisierung des UIMA-Typs **org.apache.uima.jcas.cas.Annotation**



Implementierung

Erweiterung des Typsystems

▼ Types (or Classes)

The following types (classes) are defined in this analysis engine descriptor.

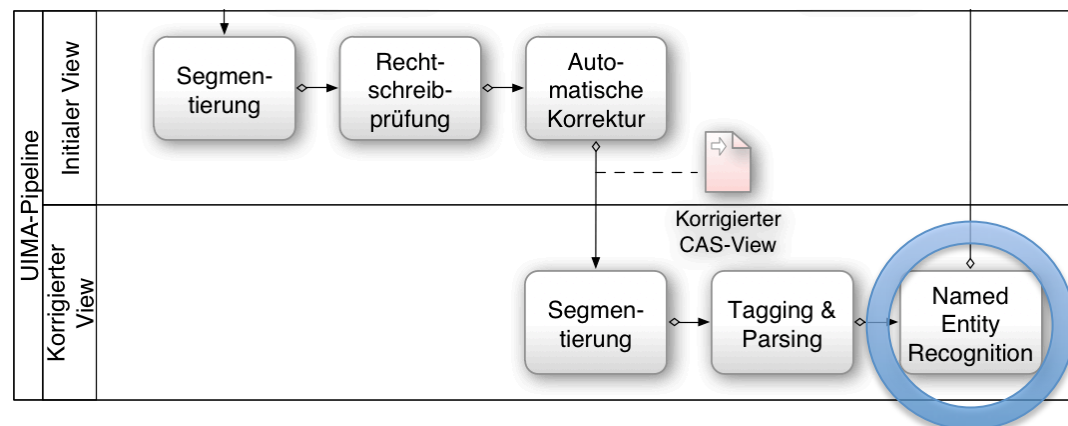
The grayed out items are imported or merged from other descriptors, and cannot be edited here. (To edit them, edit their source files).

Type Name or Feature Name	SuperType or Range	Element Type	
com.hrs.tourindi.queryanalysis.uima.type.ConstraintAnnotation	uima.tcas.Annotation		Add Type
key	uima.cas.String		Add...
value	uima.cas.String		Edit...
parent	uima.tcas.Annotation		Remove
attribute	uima.cas.String		Export...
			JCasGen

- Erzeugung durch einen Wizard (UIMA-Eclipse-Plugin)
- XML-Beschreibung, erzeugt Java-Klassen
- Enthält: Angabe des **Supertypes** im UIMA-Typsystem, **Typname** und dessen **Feature Structures** (hier: key, value, parent, attribute)

Implementierung

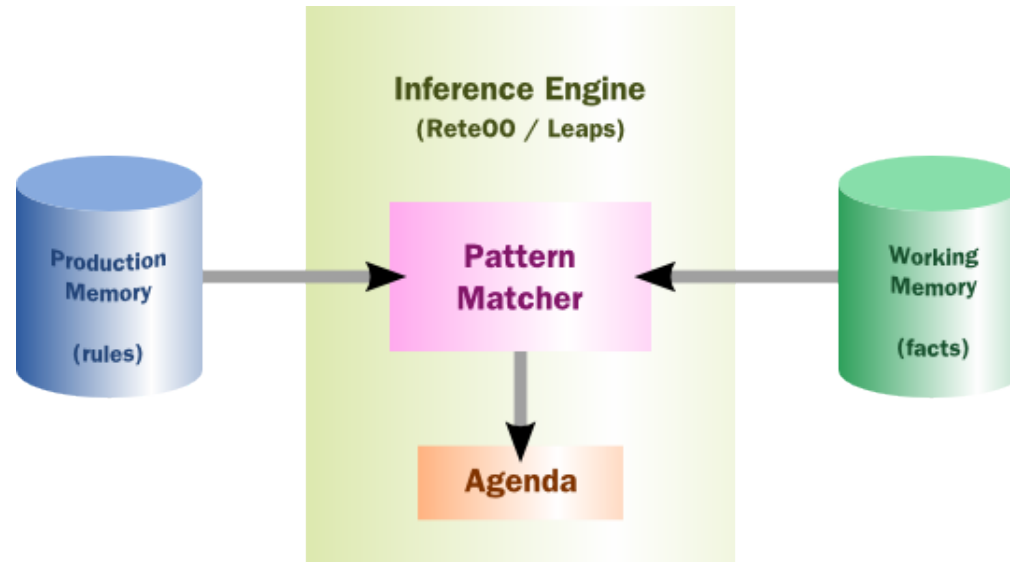
Named Entity Recognition



- Herzstück von Query Analysis:
regelbasierte Named Entity Recognition
- Arbeitet auf der **korrigierten Eingabe**, auf **Token-Level**, erzeugt Constraints mit Hilfe der **Parts-of-Speech**, **Parsing-Ergebnissen** und **Drools-Regeln**

Implementierung

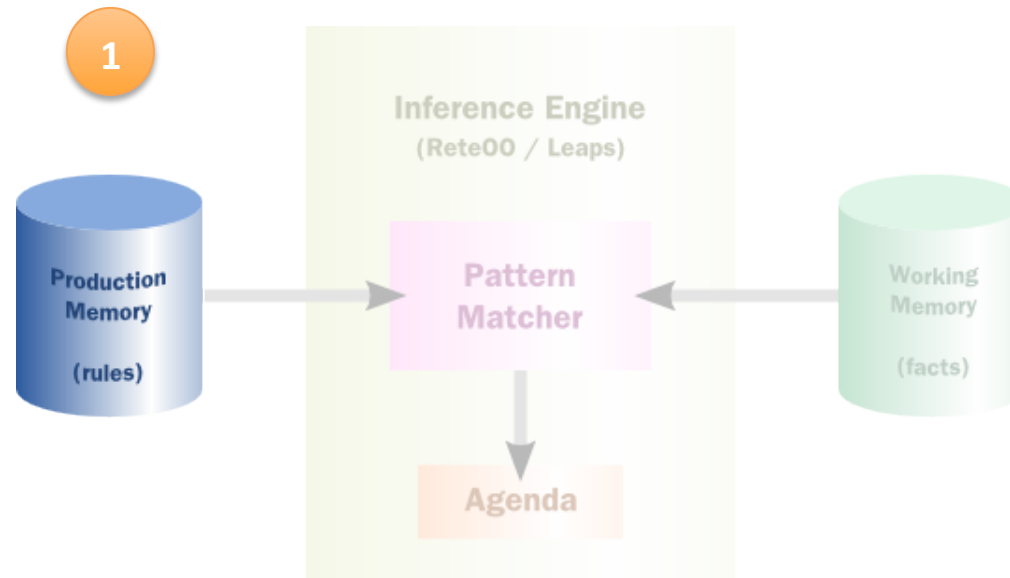
Integrationskomponente Drools Runtime Annotator



Source: http://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html_single/

Implementierung

Integrationskomponente Drools Runtime Annotator

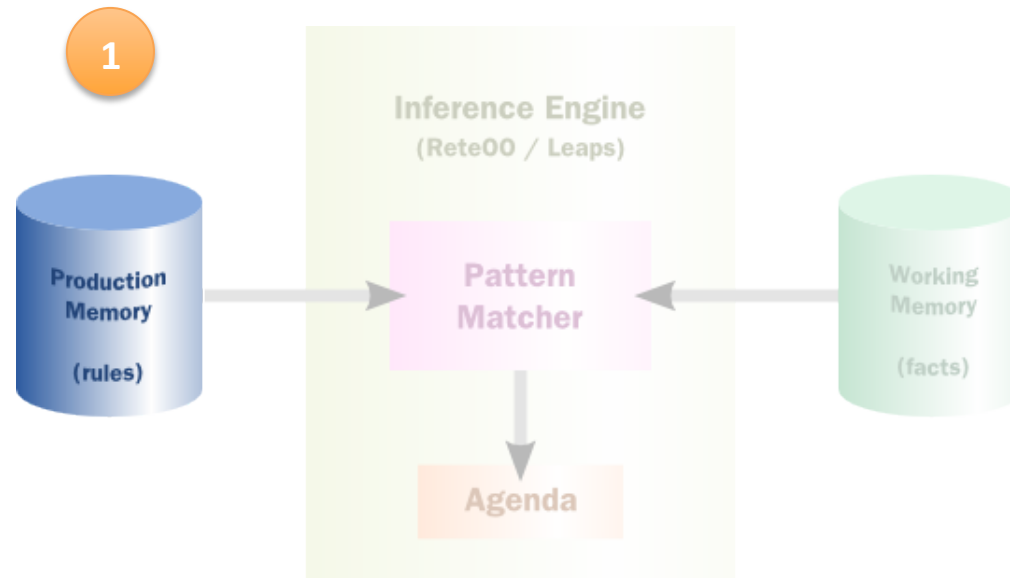


Source: http://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html_single/

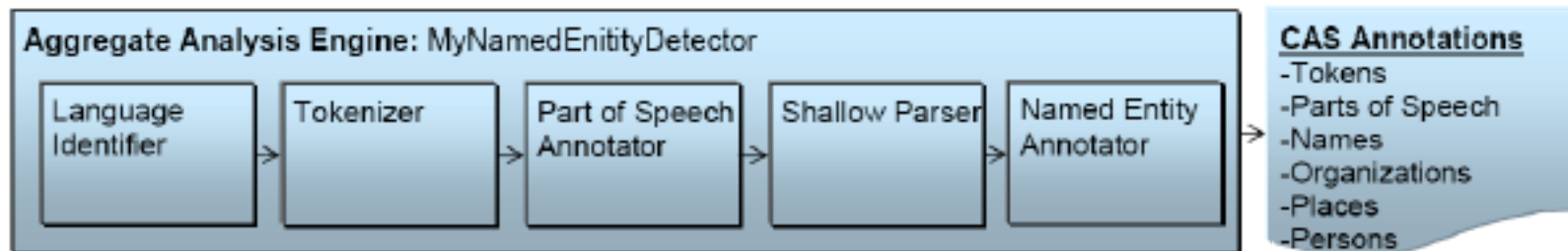
- ① Initialisierungsphase (bei **Anwendungsstart**)
Laden der Drools-Ressourcen in den Production Memory

Implementierung

Integrationskomponente Drools Runtime Annotator



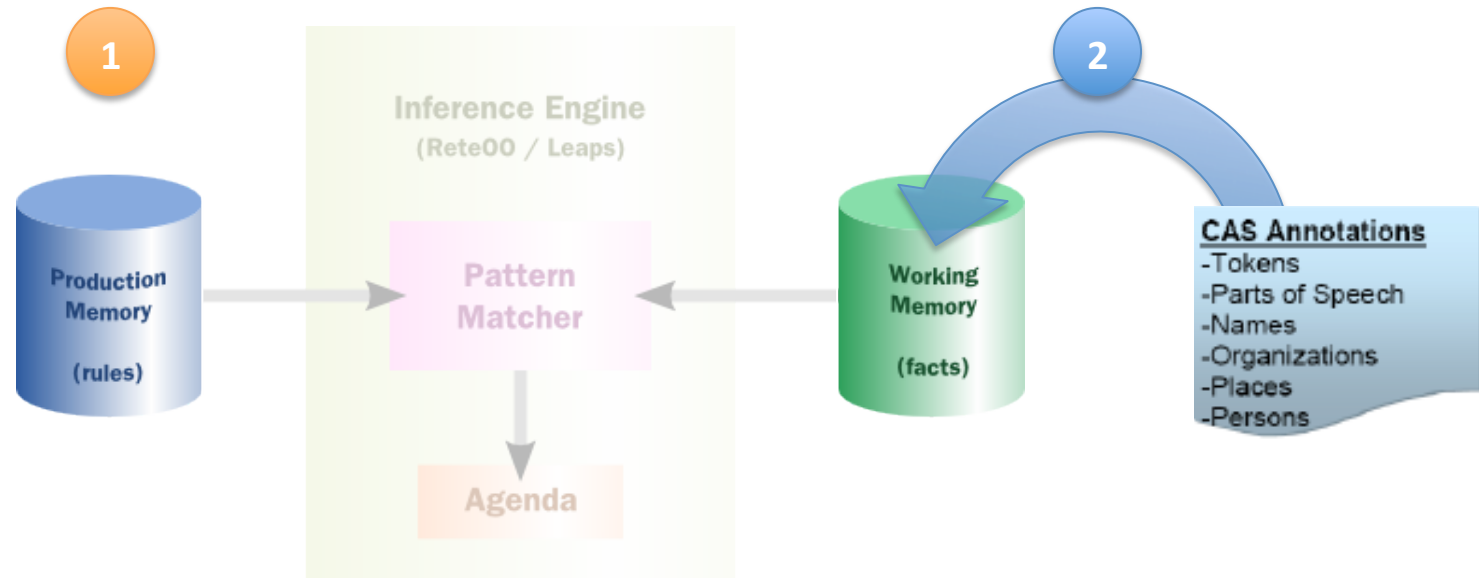
Source: http://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html_single/



Source: http://uima.apache.org/d/uimaj-2.4.0/overview_and_setup.pdf

Implementierung

Integrationskomponente Drools Runtime Annotator

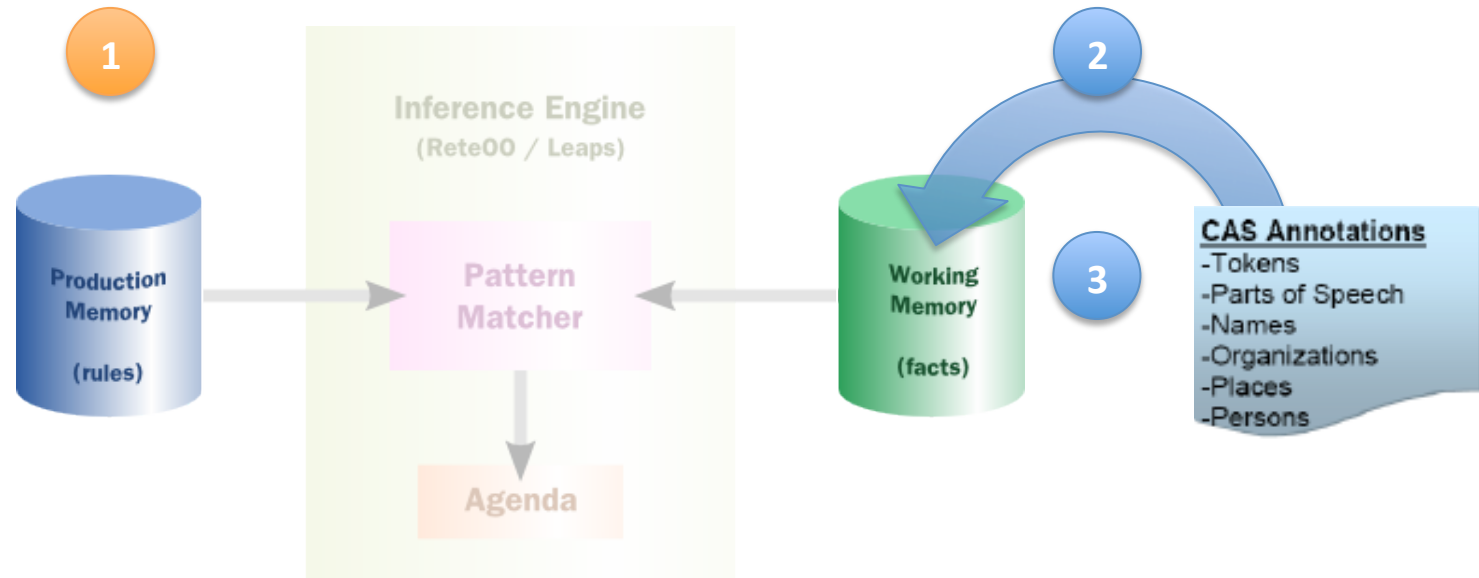


Source: http://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html_single/

- ① Initialisierungsphase (bei **Anwendungsstart**)
Laden der Drools-Ressourcen in den Production Memory
- ② Überführen aller **Feature Structures** einer CAS in den Working Memory

Implementierung

Integrationskomponente Drools Runtime Annotator

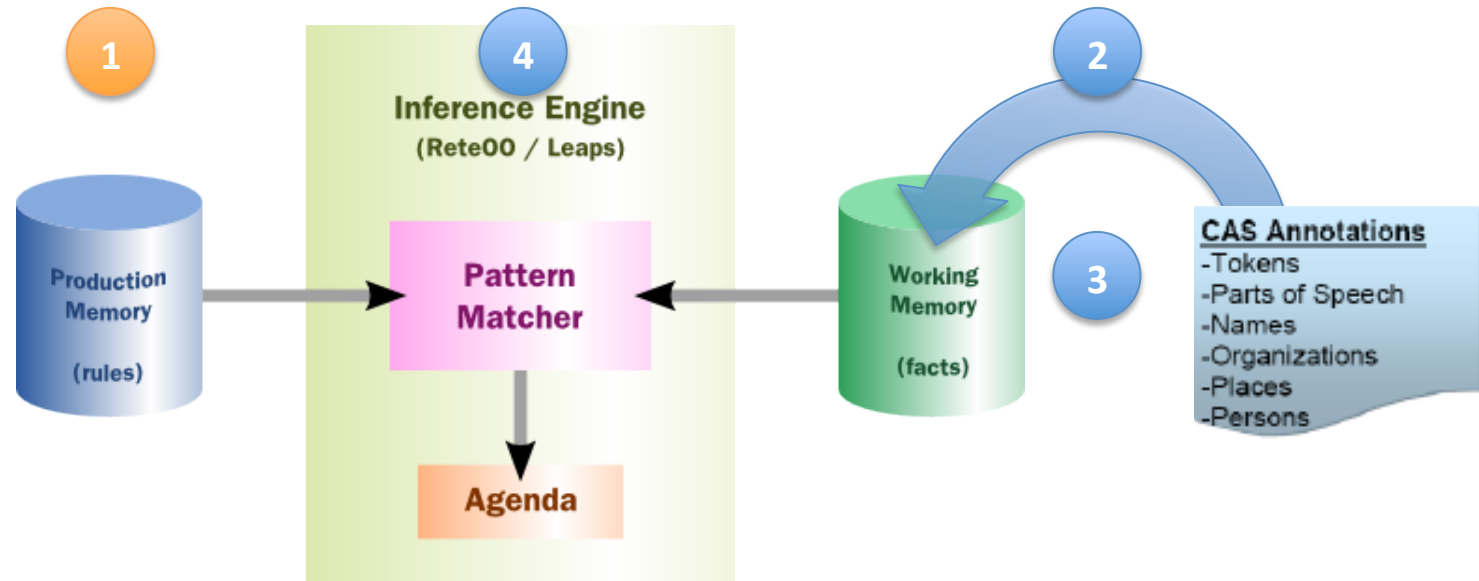


Source: http://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html_single/

- ① Initialisierungsphase (bei **Anwendungsstart**)
Laden der Drools-Ressourcen in den Production Memory
- ② Überführen aller **Feature Structures** einer CAS in den Working Memory
- ③ Registrierung eines **Listeners für Änderungen** im Working-Memory

Implementierung

Integrationskomponente Drools Runtime Annotator

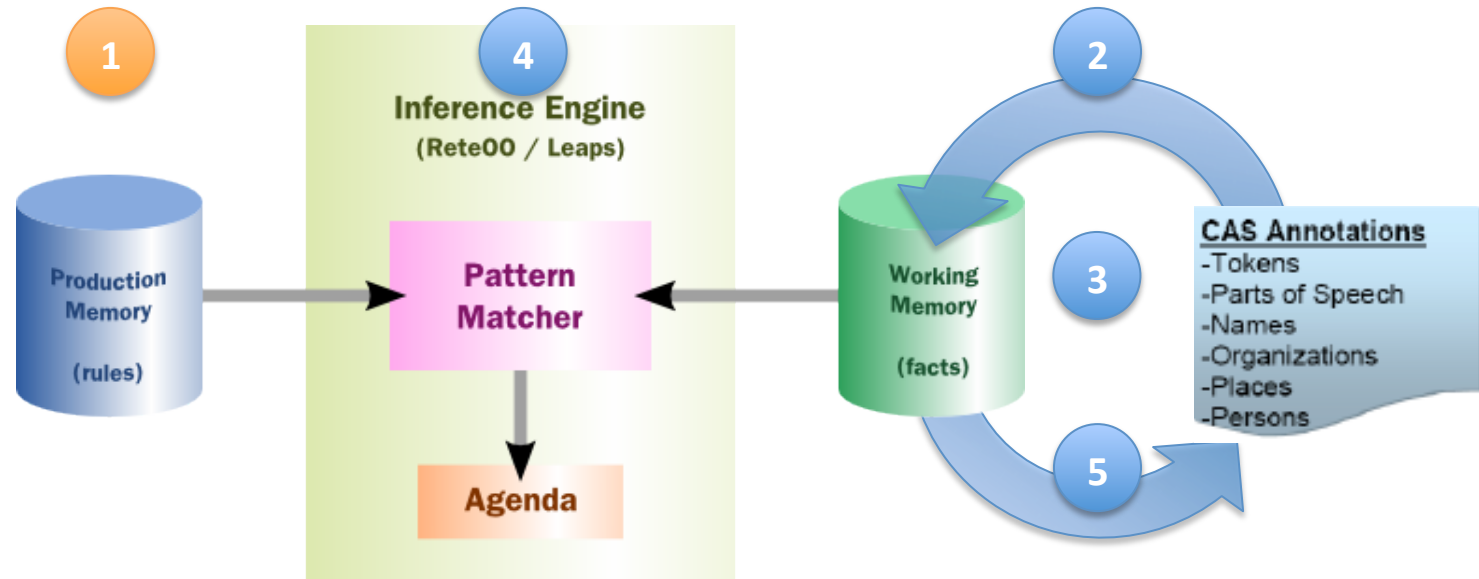


Source: http://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html_single/

- ① Initialisierungsphase (bei **Anwendungsstart**)
Laden der Drools-Ressourcen in den Production Memory
- ② Überführen aller **Feature Structures** einer CAS in den Working Memory
- ③ Registrierung eines **Listeners für Änderungen** im Working-Memory
- ④ Regelausführung, **Regeln ändern Fakten** im den Working-Memory

Implementierung

Integrationskomponente Drools Runtime Annotator



Source: http://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html_single/

- ① Initialisierungsphase (bei **Anwendungsstart**)
Laden der Drools-Ressourcen in den Production Memory
- ② Überführen aller **Feature Structures** einer CAS in den Working Memory
- ③ Registrierung eines **Listeners für Änderungen** im Working-Memory
- ④ Regelausführung, **Regeln ändern Fakten** im den Working-Memory
- ⑤ Änderungen werden durch den **Listener** in der CAS nachgezogen

Implementierung

Ablauf regelbasierte Named Entity Recognition

- Beispiel-Eingabe:
„Hotel with heated pool in London“
- In der CAS vorliegende Analyseergebnisse vor Ausführung der Regeln:

PoS-Tagging	
hotel/NN with/IN heated/ADJ pool/NN in/IN london/NNP	

Implementierung

Ablauf regelbasierte Named Entity Recognition

- Beispiel-Eingabe:
„Hotel with heated pool in London“
- In der CAS vorliegende Analyseergebnisse vor Ausführung der Regeln:

PoS-Tagging	Parsing
hotel/NN with/IN heated/ADJ pool/NN in/IN london/NNP	(NP (NN hotel)) (PP (IN with) (NP (NP (ADJ heated) (NN pool)) (PP (IN in) (NP (NNP london))))

Implementierung

Ablauf regelbasierte Named Entity Recognition

```
(NP (NN hotel))
  (PP (IN with)
    (NP (NP (ADJ heated) (NN pool))
      (PP (IN in)
        (NP (NNP london))))))
```

```
rule "NN Single Word Constraints"
  when
    nn : NN()
    token : Token ( entity : coveredText, pos == nn )
    eval ( hasEntity(entity) )
  then
    insertLogical( createConstraint(token, entity) );
end
```

Implementierung

Ablauf regelbasierte Named Entity Recognition

```
(NP (NN hotel))  
  (PP (IN with)  
    (NP (NP (ADJ heated) (NN pool) ,  
      (PP (IN in)  
        (NP (NNP london))))
```

Constraint
Annotation

Forward
chaining

```
rule "Assign ADJ Attributes"  
  when  
    adj : ADJ( attribute : coveredText )  
    Token ( pos == adj, parent : parent )  
    constraint : Constraint ( parent == parent,  
      attribute != attribute)  
    eval ( isDatatypeAttribute(constraint.getKey()) );  
    eval ( isAttribute(attribute) );  
  then  
    insert(createAttributedConstraint(constraint, attribute));  
    retract(constraint);  
end
```

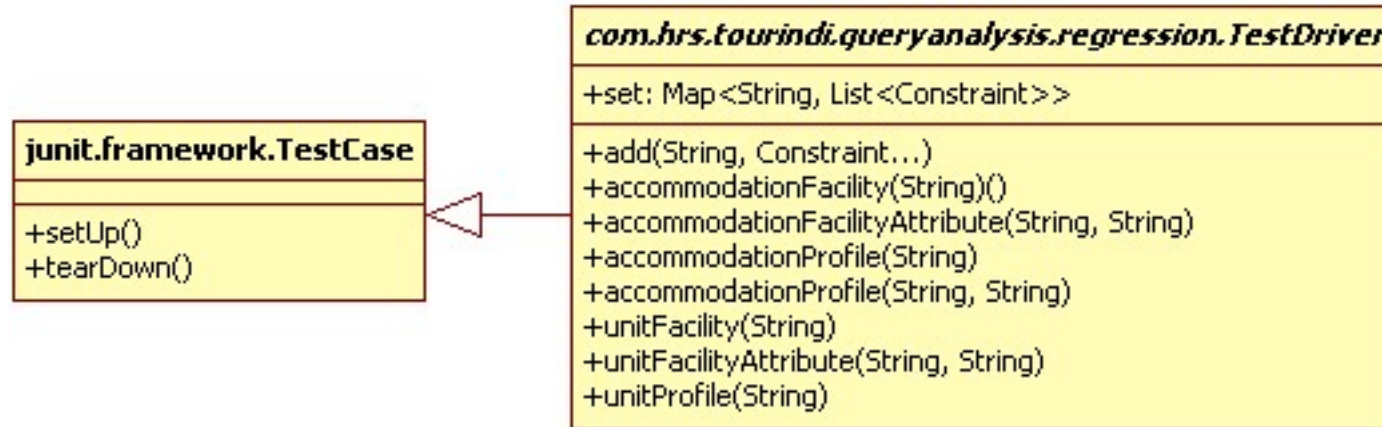
Agenda

4

1. Motivation
2. Grundlagen
3. Architektur & Implementierung
- 4. Auswertung**
5. Demo
6. Fazit & Ausblick

Auswertung

Testframework



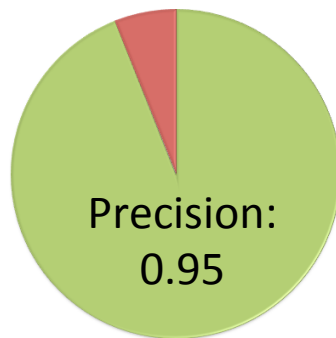
- Adressiert den **turnaround loop** eines TourInDi-Entwicklers
 - Wählt den passenden **Pipeline-Modus** und die **richtigen Logfiles** für die (drei) involvierten Logging-Techniken
 - Ergänzt **das UIMA-/ und Drools-Tooling**
 - Integriert sich nahtlos in die **TourInDi-Buildinfrastruktur**
- Erhebt Statistiken, berechnet wichtige Kennzahlen (Precision, Recall, f)

Auswertung

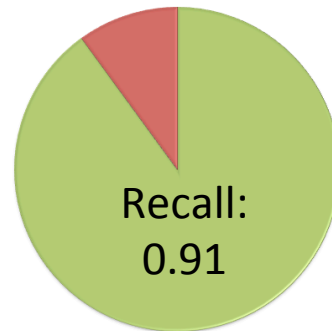
Ergebnisse - Erkennungsleistung



- Zugrundeliegend für die Bewertung:
50 beispielhafte Hotelsuchanfragen
21 zusätzlich mit Tippfehlern
- Messergebnisse:



in 47 von 50 Fällen
sind die
gefundenen Fakten
relevant



in 45 von 50 Fällen
werden relevante
Fakten auch
ausgebeutet

= F:
92.56 %

Auswertung

Ergebnisse - Erkennungsleistung



- Zugrundeliegend für die Bewertung:
50 beispielhafte Hotelsuchanfragen
21 zusätzlich mit Tippfehlern
- Messergebnisse:
In 3 von 4 Fällen hat das Hinzufügen von
Buchstabendrehern keine Auswirkungen auf die
Erkennungsleistung

Auswertung

Ergebnisse - Performance



- Zugrundeliegend für die Bewertung:
50 beispielhafte Hotelsuchanfragen
21 zusätzlich mit Tippfehlern
- Messergebnisse:
Durchschnittliche Analysezeit:

273 ms

Auswertung

Ergebnisse - Erweiterbarkeit

- Natives Drools als Wissensgrundlage
 - Single Point of truth
 - Deskriptive Logik, leicht zu lesen und zu erweitern
- Architektur-Konzept berücksichtigt Paradigmen der UIMA
 - Flexibler, standardkonformer Ansatz
 - Wechselbare Komponenten

Agenda

5

1. Motivation
2. Grundlagen
3. Architektur & Implementierung
4. Auswertung
5. Demo
6. Fazit & Ausblick

Agenda

6

1. Motivation
2. Grundlagen
3. Architektur & Implementierung
4. Auswertung
5. Demo
6. Fazit & Ausblick

Fazit

- Das entwickelte Query Analysis...
 - ...erfüllt alle Anforderungen 😊
 - ...stellt als Konzept auch die Weichen für eine künftige professionelle Weiterentwicklung.
 - ...erkennt bereits in Ansätzen eine (Verneinungs-)Semantik.
 - ...hat bereits ein praktisches Lessons Learned 😊
- Die entwickelte Drools-Integrationskomponente...
 - ...ist bisher einzigartig und daher auch für die Community wertvoll.
 - ...ist ein typsystemunabhängiger und deshalb sehr mächtiger Ansatz.

Ausblick

- Datenmodell-Issues wurden bereits adressiert:
 - schwache Typisierung
(Welche Entity-Attribut-Beziehungen sind sinnvoll?)
 - Buggy-CSV-Importer
- Vorgängerimplementierung könnte als Fallback weiterhin genutzt werden
- Weiterentwicklungen:
 - Erkennung aller TourInDi-Constraints
Unterschiedliche Constrainttypen im UIMA-Typsystem repräsentieren
 - Einführung der Mehrsprachigkeit
Umstellung der gesamten Pipeline auf Mehrsprachigkeit?
 - Aufgerufene Hilfsmethoden als native Drools-Operatoren

Danke für Ihre Aufmerksamkeit!