8 Wissensextraktion aus Texten mittels semantischer KI

8 Wissensextraktion aus Texten mittels semantischer KI

Bernd Geiger

8.1 Problemstellung

Im Zuge der digitalen Transformation von Geschäftsprozessen entsteht die Notwendigkeit, in natürlicher Sprache verfasste Handlungsanweisungen, Verträge und vertragsähnliche Dokumente in computerausführbaren Code zu wandeln, um z. B. Workflow Automation oder Smart Contracts umzusetzen. Mit herkömmlichen (statistischen) KI-Methoden ist die Transformation oft unsicher und nicht nachvollziehbar, da es bei den Ursprungsdokumenten meist auf jedes Wort ankommt und Sätze linguistisch (z. B. aufgrund von Auslassungen und Grammatik) mehrdeutig sein können. Mit dem hier vorgestellten alternativen Ansatz der »semantischen KI« werden mittels Higher-Order Logic (HOL) Algorithmen mit nur geringem Einrichtungsaufwand natürlich sprachliche Texte in semantisch eindeutige und ausführbare Computer-Modelle transformiert. Dabei bleibt die Kausalitätsbeziehung zwischen Transformationsergebnis und Textursprung immer exakt nachvollziehbar – das Verfahren ist deshalb inhärent in seiner Arbeitsweise nachvollziehbar und somit auch auditierbar. Es wurde bislang auf englische und deutsche Texte angewandt.

Das hier vorgestellte Verfahren SemanticMatcher wurde für die Übersetzung von PDF-basierten Wartungshandbüchern für Flugzeuge in ausführbare BPMN-Abläufe getestet und optimiert. Die Adaption für natürlichsprachliche Dokumente aus anderen Kontexten geschieht durch einfache Modifikation der semantischen HOL-Algorithmen und wurde auch schon bei Vertragstexten erfolgreich umgesetzt. Wie im Bereich der Wartungshandbücher ist der Einrichtungsaufwand auch für Vertragstexte gering.

8.2 Einleitung

Allgemeine Formalien (Instruktionen, Vorschriften, etc.), die in textlicher Beschreibung vorliegen, sollen computerverständlich gemacht werden. Das Anwendungspotenzial hierfür ist enorm: alle menschenlesbaren Formalien sind typischerweise in natürlicher Sprache niedergeschrieben (Montageanleitungen, Verträge, gesetzliche Regulierungen, etc.). Das Bestreben, Prozesse zu automatisieren (z. B. robotergestützte Wartung, Regelkonformität bei Banktransaktionen, etc.) macht es erforderlich, textlich gefasste Formalien computerisiert ausführbar zu machen. Der klassische Weg hierfür ist die manuelle »Übersetzung« der textlichen Formalien in Skripte bzw. prozedurale Programmiersprachen. Diese Übertragung ist nicht nur sehr aufwendig, sondern auch fehleranfällig und eine formale Verifikation mit dem Inhaltsursprung ist nicht möglich.

Grundsätzlich handelt es sich bei der Aufgabenstellung um eine semantische Erweiterung des sogenannten Natural Language Processing (NLP). Der SemanticMatcher extrahiert nicht nur Einzelfakten, sondern verteilte Faktenzusammenhänge und somit exaktes Wissen.

Exaktes Wissen ist typischerweise dort notwendig, wo es sich um operativ kritische oder juristische Anwendungsdomänen handelt. Um die benötigte Exaktheit zu gewährleisten, ist es essenziell, dass das NLP-Verfahren sowohl algorithmisch als auch im Erzielen des Extraktionsergebnisses nachvollziehbar ist und dokumentiert, aus welchen Worten im Textteil und unter Zuhilfenahme welcher zusätzlichen Wissenskomponenten die computerisierte Repräsentation generiert wurde.

Der hier vorgestellte SemanticMatcher ist fokussiert auf Texte, die auf Prägnanz und Vermeidung linguistischer Redundanzen optimiert sind, sogenannte Controlled Natural Languages (CNLs). Dies trifft auf Texte in operativ kritischen bzw. in juristischen Anwendungsdomänen üblicherweise per Design zu, um menschliche Fehlinterpretationen zu vermeiden. Herkömmliche, statistische NLP-Verfahren wurden meist zu einem anderen Zweck entwickelt – sie sollen möglichst universell unterschiedliche Ausdrucksformen »verstehen« können und haben daher eine hohe (linguistische) Ausdruckstoleranz. Die Toleranz bedingt aber in Folge eine Detailunschärfe und eignet sich daher nicht zur präzisen Wissensextraktion[1,2].

8.3 Semantische KI

Was ist semantische KI und worin besteht der Unterschied zu KI basierend auf neuronalen Netzen?

Grundsätzlich kann man zwischen KI basierend auf statistischen Methoden (meist Neuronale Netze) und KI basierend auf logischen, mathematischen Modellen (logische Semantik) unterscheiden, wobei jede Methode auch mit der jeweils anderen erweitert werden kann. Allgemein betrachtet ist die Vorgehensweise die gleiche: ein unbekanntes Prüfmuster soll mit einem Standardmuster verifiziert bzw. falsifiziert werden. Bei statistischer KI ist das Standardmuster in der Regel im neuronalen Netz kodiert, bei semantischer KI in dem HOL-Modell.

1 Johnson (2009), How the statistical revolution changes (computational) linguistics, in Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?, S. 3–11.

2 Gao/Fodor/Kifer (2018), Knowledge authoring for rule-based reasoning, in OTM Confederated International Conferences »On the Move to Meaningful Internet Systems«, S. 461–480.

Wo wendet man die unterschiedlichen KI-Verfahren an?

Statistische KI-Verfahren werden bevorzugt dann angewendet, wenn Signale mit hohem statistischem Anteil vorhanden sind, d. h. wenn im Verhältnis zur Aussage der Erkennung (z. B. »das ist ein Stoppschild«) sehr viele Daten vorliegen. Die Komplexität in der Signalerkennung (z. B. Bild des geometrischen Verkehrsschildes) liegt im Wesentlichen darin begründet, dass das Signal aufgrund der Zerlegung in einzelne Pixel hochgradig redundant und verrauscht ist.[3]

Im Gegensatz dazu werden *semantische KI-Verfahren* primär dort angewendet, wo die Komplexität in den inhaltlichen Zusammenhängen begründet liegt. So müssen beispielsweise die über viele Textpassagen aufgeteilten Gesamtzusammenhänge eines Vertrages mit hoher Präzision verifizierbar erfasst werden. Dabei sind die Redundanz und das Rauschen des Signals (des zugrunde liegenden Textes) um Größenordnungen geringer als bei dem vorher genannten Bildbeispiel des Stoppschildes. Vergleichbares gilt für die knapp kommunizierten Instruktionen eines Wartungshandbuchs, die nur bei Kenntnis des detaillierten Kontextes Sinn ergeben, weil das Handbuch eben für den »Experten« geschrieben ist.

Semantische KI für hohe Präzision und Nachverfolgbarkeit der Ergebnisse

Je deterministischer das Datensignal ist und je geringer die Redundanzen, desto eher lassen sich handhabbare HOL-Modelle finden, mit denen sich die ursprüngliche Bedeutung des Signals rekonstruieren lässt. Ein Beispiel für eine sehr knappe und dadurch vermeintlich mehrdeutige (verrauschte) Nachrichtenübermittlung ist die folgende Kommunikation in deutscher Sprache: »Schließe noch die Waschmaschine an und dann machen wir Schluss für heute.«

Um diesen Zuruf des Meisters an den Gesellen roboterausführbar zu machen, muss man die ursprüngliche Intention/Semantik verstehen:

- Waschmaschine funktional mit der notwendigen Infrastruktur verbinden
- Es wird keine weitere Tätigkeit folgen

Das in dem Beispielsatz vorhandene »Restrauschen« – die linguistische Mehrdeutigkeit von »Anschließen« der Waschmaschine und »Schluss machen« – lässt sich durch a priori Wissen (das ebenso mittels HOL modelliert wird) eliminieren:

- Beim technischen Gerät Waschmaschine bedeutet die Tätigkeit »Anschließen« ein technisches Verbinden (im Gegensatz z. B. zum Anschließen eines Fahrrads),
- Die idiomische Phrase »Schluss machen«, bezogen auf ein zeitliches Intervall, bedeutet die Beendigung einer Tätigkeitsreihe.

3 Das Zeichen »Stoppschild« ist überlagert mit nicht auf das Stoppschild bezogenen Signalen im Sinne eines stochastischen Prozesses. Diese Signale sind z. B. Vegetation, Schmutz und sonstige Objekte, die von der Kamera mit erfasst werden.

Natürlich lassen sich Einzelinstruktionen und idiomatische Phrasen auch mit statistischen KI-Verfahren erkennen. Die notwendigen Trainingsphasen für die Zuordnung der jeweiligen Semantik zu den Einzelinstruktionen sind aufgrund der Vielfalt (alle möglichen Instruktionen müssen berücksichtigt werden) extrem aufwendig und bedürfen großer (oft nicht verfügbarer) Referenzdaten zum Trainieren des Systems. Darüber hinaus lassen sich die Ergebnisse statistischer KI-Verfahren aufgrund der inhärenten Blackbox-Natur der Systeme nicht verifizieren.

8.4 Semantische KI und Natürliche Sprache

Klassifizierung von natürlicher Sprache und deren kontrollierte Varianten (CNLs)

Unterschiedliche Ausdrucksformen einer natürlichen Sprache (z. B. Englisch) können nach dem PENS Klassifikationsschema eingeteilt werden[4]. Hierbei bedeutet: P = Precision, E = Expressiveness, N = Naturalness und S = Simplicity. Den einzelnen Parametern werden üblicherweise Werte von 1 = schwach bis 5 = stark zugeordnet. Die natürliche Sprache Englisch in ihrer allgemeinen Form wird mit $P^1E^5N^5S^1$ klassifiziert. Die CNL-Varianten Simplified (Technical) English[5] sowie Legislative Drafting Language[6] mit $P^2E^5N^5S^1$, die »Semantics Of Business Vocabulary and Rules« (SBVR) Sprache[7] mit $P^3E^4N^4S^2$. Alle Ausdrucksformen, die im Vergleich zur natürlichen englischen Sprache eine höhere Präzision haben, erreichen dies durch geregelte Einschränkungen (»Controlled«) z. B. bei Wortumfang, Grammatik, Satzbau und Idiomatik. Das zuvor Beschriebene ist unabhängig von der Sprache und kann genauso für die deutsche Sprache angewendet werden.

Modelle zur Wissensrepräsentation

Da das Ziel der semantischen Wissensextraktion in der Transformation des relevanten Textes (z. B. Wartungsanweisungen) in eine computerausführbare Form besteht, muss für die Wissensrepräsentation auf der Computerseite eine adäquate Ausdrucksform gewählt werden. Die Repräsentation muss eindeutig (Präzision = 5) sein (was in der Regel bei formalen Sprachen der Fall ist) und gleichzeitig eine hohe Expressivität haben. Prozedurale Sprachen wie Java können unter anderem aufgrund der Anforderungen zur Expressivität nicht berücksichtigt werden.

4 Kuhn (2014), A survey and classification of controlled natural languages, Computational Linguistics, 40(1), S. 121–170.

5 Aerospace and Defence Industry Association of Europe (2017), SIMPLIFIED TECHNICAL ENGLISH, Specification ASD-STE100, Issue 7.

6 Massachusetts Senate (2003), Legislative, Drafting and Legal Manual, third edition.

7 Bollen (2008), SBVR: A fact-oriented OMG standard, in OTM Confederated International Conferences »On the Move to Meaningful Internet Systems«, S. 718–727.

Higher-Order Logic (HOL) zur Wissensrepräsentation zu verwenden ist naheliegend, da dafür effiziente computerbasierte Umsetzungen existieren. In unserem Fall wird die industriell erprobte HOL-Sprache ObjectLogic[8,9,10] verwendet. ObjectLogic wird mit $P^5E^5N^2S^3$ klassifiziert. Prinzipiell zeichnen sich HOL-Sprachen durch eine hohe Expressivität aus. Gleichzeitig erlaubt die vergleichsweise »gute« Einfachheit mit S = 3 die Anwendung von ObjectLogic auch für Nichtmathematiker

Um ein Beispiel zu geben, wie einfach ObjectLogic eingesetzt werden kann, müssen zunächst die notwendigen Grundelemente der Sprache benannt werden:

Elemente	Notation
Funktionen	Head :- body
Frames, Maps, Lists	I[P → R], [P → R], [a, b, c]
Instanzen von Klassen	I : C
Vererbung von Klassen und Properties	SC :: C, SP << P
Prädikate	$P(n_1, ..., n_m)$
Quantoren	FORALL, EXIST
Junktoren	AND, OR, NOT, ->, <-, <->

Tabelle 2: Grundelemente ObjectLogic

ObjectLogic beinhaltet beliebig verschachtelte Objekte, Frame-Atome (F-Atome), F-Moleküle, parametrisierbare Prädikate, Attribute und Mengen- bzw. Logik-Funktionen, mit diesen Elementen kann man beliebige Realitäten beschreiben. Die Auswertung erfolgt in unserem Fall über den Interpreter (der »Reasoner«) OntoBroker (von semafora systems GmbH).

8 Kifer/Lausen/Wu (1995), Logical foundations of object-oriented and frame-based languages, Journal of the ACM (JACM), 42(4), S. 741–843.

9 Angele/Kifer/Lausen (2009), Ontologies in F-logic, in Handbook on Ontologies, S. 45–70.

10 Maier et al. (2018), Datalog: Concepts, History, and Outlook, Declarative Logic Programming: Theory, Systems, and Applications, S. 3–100.

8.5 Das Wissensmodell

Ein Wissensmodell für technische Instruktionen hat angelehnt an das BPMN-Modell[11] folgende allgemeine Charakteristika (die man je nach Anwendung verfeinern oder weiter reduzieren kann):

- Aktivität
- Voraussetzung, unter der die Aktivität stattfinden darf
- Agent, der die Aktivität ausführt
- Beschreibung der Dynamik der Aktivität:
  - Objekt 1 wird in Relation zu Objekt 2 verändert
  - Verwendung von Hilfsmitteln bei der Umsetzung der Aktivität
  - Zustandsbeschreibung vor und nach der Aktivität von Objekt 1 und Objekt 2 (pre/post state)
- Reihenfolge der Aktivitäten

Das Wissensmodell hat eine grammatikalische Repräsentation (für die jeweils verwendete natürliche Sprache), die sogenannten semantischen N-Gramme[12], die im Folgenden für den Matching-Prozess eingesetzt werden.

Die konkreten Informationen, um das Modell zu füllen, befinden sich in den Texten der Wartungsanweisungen. Dieses zu extrahierende Wissen liegt in einer Aneinanderreihung von Grammatiksegmenten vor, die Text N-Gramme. Nun werden im Zuge des »semantic matching« die abstrakten Konzepte der semantischen N-Gramme des Modells mit den konkreten Text N-Grammen aus den Wartungsanweisungen gepaart. Bei erfolgreicher Paarung wird das Modell mit den konkreten Werten (in diesem Fall Worten) gefüllt. Auf diese Weise entsteht aus dem abstrakten Modell ein konkretes detailliertes Abbild der Anweisung.

11 Chinosi/Trombetta (2012), BPMN: An introduction to the standard, Computer Standards & Interfaces, 34(1), S. 124–134.

12 Wikipedia, (Retrieved July 21, 2019), ↗ **https://en.wikipedia.org/wiki/N-gram**.

In der HOL/ObjectLogic-Darstellung sieht das in seiner für den Anwendungsfall allgemeinsten Form wie nachfolgend aus (mit Schattierungen illustriert für die ObjectLogic-Elemente Konzepte, Instanzen, Properties und Einzelwerten). In dem Modell ist berücksichtigt, dass eine Instruktion aus mehreren Einzelschritten (»steps«) bestehen kann:

```
?InstructionID:Instruction[
        Act,
        next→?InstructionID:Instruction,
        prior→?InstructionID:Instruction,
        serial→?Serial,
        condition→?StateID:State[
                id→?ID,
        state→?State,
                stateSpecifier→?StateSpecifier,
                target→?Target,
                targetSpecifier→?TargetSpecifier
        ],
        step→?StepID:Step[ {AlternativeStep}
                id→?ID,
                serial→?Serial,
                next→?StepID:Step,
                prior→?StepID:Step,
                act→?Act,
        actor→?Actor,
                actZone→?ActZone,
                actSpecifier→?ActSpecifier,
                actDirection→?ActDirection,
                actObject→?ActObject,
        actObjectSpecifier→?ActObjectSpecifier,
                actTool→?ActTool,
                actToolSpecifier→?ActToolSpecifier,
                actConsumable→?ActConsumable,
         actConsumableSpecifier→?ActConsumableSpecifier,
        ]
        step→?StepID:Step[
                target→?Target,
                targetSpecifier→?TargetSpecifier,
                targetPostState→?StateID:State[...],
                purpose→?StepID:Step[...]
        ]
]
```

8.6 Der Matching-Prozess zur Wissensextraktion

Schritt 0: Initialisierung

Folgende Komponenten werden vorbereitend für die sechs im Wesentlichen automatischen Schritte des Matchings, also der Wissensextraktion benötigt:

- Die Textdatei mit der Satzsammlung. Optional: ein Layout-Parsing des Ursprungstextes (siehe Schritt 6).
- Eine WordNet[13]-Variante für die jeweilige Sprache zur Umsetzung der Named Entity Recognition[14] (NER – siehe Schritt 1).
- Setup des a priori Wissens bestehend aus:
  - Technische Taxonomie bzw. Wissensmodell (public domain oder käuflich zu erwerben).
  - Domänenspezifische Wörterbücher (public domain oder käuflich zu erwerben).
  - Partonomien, Designbäume (im Unternehmen vorhanden, können automatisch konvertiert werden).
  - Plausibilisierungsfunktionen bezogen auf das technische Wissensmodell, Partonomien, etc. (werden automatisch u. a. mithilfe der technischen Taxonomien erzeugt).
  - Optional: Stücklisten und Designreferenz (im Unternehmen vorhanden, können automatisch konvertiert werden).
- Semantische N-Gramme, die die Abfolge der Teil-Grammatik für die betrachtete Domäne in der CNL-Ausführung beschreiben (werden aus dem Wissensmodell abgeleitet).

Schritt 1: Grammatikbasierte NER

Jedes einzelne Wort (bzw. Funktionseinheit, wie z. B. Referenz auf eine Darstellung in Klammer oder Teilenummer) bekommt automatisch eine grammatikalische Funktion zugeordnet (grammar-tag) – siehe Spalte 2 der Tabelle 2.

Schritt 2: Auflösen von Referenzidentitäten

Hier erfolgt das einfache Auflösen bzw. »Bereinigen« von Sätzen:

- Stilistische Co-Referenzen (Pronomen) werden automatisch zur konkreten Referenz umbenannt.
- Aus dem a priori Wissen bekannte Wortgruppen mit fester Bedeutung, wie z. B. *»Emergency Escape Path Marking Systems«* werden als feste Begrifflichkeit erfasst, damit diese nicht weiter linguistisch ausdifferenziert werden müssen.

Spätestens auf dieser Stufe werden auch Sätze mit unbekannten Worten ausgesondert, um ggfs. die unbekannten Worte in ein Benutzerwörterbuch aufzunehmen.

13 Wikipedia, (Retrieved August 20, 2019), ↗ https://en.wikipedia.org/wiki/WordNet

14 Wikipedia, (Retrieved August 20, 2019), ↗ https://en.wikipedia.org/wiki/Named-entity_recognition

Schritt 3: Auflösen von Mehrdeutigkeiten

Die Instruktion der für einen bestimmten Arbeitsschritt notwendigen Vorbereitung:

»... in the cockpit, on the overhead panel, make sure that the dial is set to the release position ...«

ist für den Menschen einfach zu verstehen, aus der Linguistik ergibt sich aber nicht, wo sich das Einstellrad (»dial«) genau befindet. Ein Roboter könnte hier auch verstehen: *gehe auf das overhead panel und suche das Einstellrad im Cockpit* – eine Anweisung, die im Zweifel gefährlich ist. Die Auflösung ergibt sich erst aus dem Teile-Modell des Flugzeugs. In ObjectLogic ist die einfachste Form einer Partonomie für diesen Fall:

- Fuselage[hasPart → Cockpit[hasPart → OverheadPanel]]

Schritt 4: Matchen von semantischen N-Grammen des Modells mit den konkreten N-Grammen der Texte

In diesem Satz

- *»Detach the optical cable (3) and adhere it to the chair structure with BONDING AND ADHESIVE COMPOUNDS (Material No: xx-yyy).«*

sind die folgenden N-Gramme zutreffend (in Prädikatsform dargestellt):

- nGram(verb, adjectiv, noun)
- nGram(verb, noun, preposition, noun, noun, preposition, noun).

Nun muss in einem zweiten Schritt die konkrete Bedeutung (die Semantik) der Bezüge ausgewertet werden, insbesondere die Präpositionen »to« und »with«. Hier im Gesamtüberblick:

Satzteil (PoS)	Grammatik	Strukturelle Bedeutung	Fluss	HOL Teil-Modell
Detach	verb	act, implies agent who does it	step 1	i:Instruction[Act, s1:Step[agent → ?Agent, act → detach]]
the	article	specifier of the object	step 1	–
optical	adjective	specifier to act on	step 1	i:Instruction[s1:Step[objectSpecifier → optical]]
cable	noun	target to act on	step 1	i:Instruction[s1:Step[object → cable]]
(3)	parenthesized reference	reference to drawing	step 1	i:Instruction[s1:Step[refDrawing → 3]]
and	conjunction	divides step 1 and step 2	–	–
adhere	verb	act, implies agent who does it	step 2	i:Instruction[s2:Step[agent → ?Agent, act → adhere]]
it	pronoun	co-reference to previous object optical cable	step 2	i:Instruction[s2:Step[objectSpecifier → optical, object → cable]]
to	preposition	direction of the act	step 2	i:Instruction[s2:Step[actDirection → torwards]]
the	article	–	step 2	–
chair	noun	specifier of the 2nd target	step 2	i:Instruction[s2:Step[targetSpecifier → chair]]
structure	Noun	2nd »argument« i. e. target of the verb adhere	step 2	i:Instruction[s2:Step[target → structure]]
with	preposition	reference to the »helper« of verb	step 2	–
BONDING AND ADHESIVE COMPOUNDS	compound noun (capitalized as defined term)	the »helper« which is a consumable as known from the BoM	step 2	i:Instruction[s2:Step[actConsumable → »BONDING AND ADHESIVE COMPOUNDS«]]
(Material No: xx-yyy).	parenthesized reference	Reference to BoM	step 2	i:Instruction[s2:Step[refBoM → xx-yyy]]

Tabelle 3: Transformation eines Satzes in eine HOL-Repräsentation

Eine besondere Teil-Aufgabe kann die Differenzierung von Aktivitäten und Status sein, wenn der Status idiomatisch beschrieben wird. Im Satz: *»... loosen the screws to take the chair off the rails ...«* ist der zweite Teilsatz *»... to take the chair off the rails ...«* eine weitere Aktivität. Im Satz: *»... remove the circuit breaker to cut off the power ...«* ist der zweite Teilsatz *»... cut off the power ...«* der Status der elektrischen Versorgung nach der Aktivität, die im ersten Teilsatz beschrieben wird. Solche idiomatischen Ausnahmen sind Teil des a priori Wissens und werden automatisch vom SemanticMatcher als »post state« berücksichtigt.

Schritt 5: Diskriminieren von Falsch-Positiven und Erkennen von Falsch-Negativen Ergebnissen

Falsch-Positive treten in zwei unterschiedlichen Formen auf:

- Es gibt zu einem Teilsatz der Wartungsanweisung mehr als eine Modellrepräsentation (weil mehr als ein N-Gramm auf den Teilsatz passt). Somit gibt es in dem Teilsatz eine (nicht intendierte, aber linguistisch vorhandene) Mehrdeutigkeit, die vom Modell nicht automatisch aufgelöst werden kann. Für eine korrekte Abbildung der Wartungsanweisung ist ein eindeutiger Bezug erforderlich, was in der Trainingsphase zu berücksichtigen ist.

- Es gibt zwar nur eine Modellrepräsentation, diese macht aber in der Anwendung keinen Sinn. Um dies festzustellen, gibt es zwei Stufen der Plausibilitätsprüfung. Die Erste erfolgt bei der N-Gramm-Auswertung: das Verb *»adhere«* kann zum Beispiel nicht mit der Präposition *»away«* verbunden sein. Die zweite Plausibilitätsprüfung findet auf der Anwendungsebene statt: *»Loosen anti-rattle nuts with a ladder«* würde inhaltlich kein Sinn machen. Dies wird mit dem technischen Wissensmodell des a priori Wissens abgefangen. Daraus ergibt sich, dass man Muttern nur mit bestimmten Werkzeugen bestimmungsgemäß bearbeiten kann. Diese Plausibilitätsprüfungen erfolgen auf Basis des a priori Wissens automatisch.

Textteile, die von den N-Grammen nicht zugeordnet wurden, werden zur Verifikation möglicher Falsch-Negativer genutzt.

Die Berücksichtigung von Falsch-Positiven und Falsch-Negativen in einer initialen Trainingsphase ist essenziell, um die Korrektheit der Standardmuster und die Vollständigkeit des a priori Wissens zu gewährleisten. Aber auch in der Produktionsphase können und müssen Falsch-Positive und Falsch-Negative zur Qualitätssicherung gesammelt werden.

Schritt 6: Serialisierung

In einem an das BPMN-Modell angelehnten Verfahrensfluss müssen die Einzelaktivitäten im abschließenden Schritt serialisiert werden. Ein Abfolge-Ordnungskriterium ist durch die Satzabfolge der Instruktionen und Teilinstruktionen innerhalb eines Satzes gegeben. Gegebenenfalls muss zusätzlich die Semantik der Formatierung analysiert werden, da nicht jeder Satz eine Instruktion beinhaltet, z. B. weil manche Sätze Überschriften sind. Im nachfolgend abgebildeten Layout der Wartungsinstruktionen liegen die Instruktionen immer auf der zu den nächsten Nachbarn untersten Einrückungsebene. In den Überschriftsätzen sind dabei gegebenenfalls Konditionale formuliert, da die nachfolgenden Instruktionen nur dann ausgeführt werden dürfen, wenn die Konditionale erfüllt sind. Wenn Layout-Semantik eine Rolle spielt, werden die Einrückungen bei der Extraktion der Texte den Sätzen als Attribute mitgegeben.

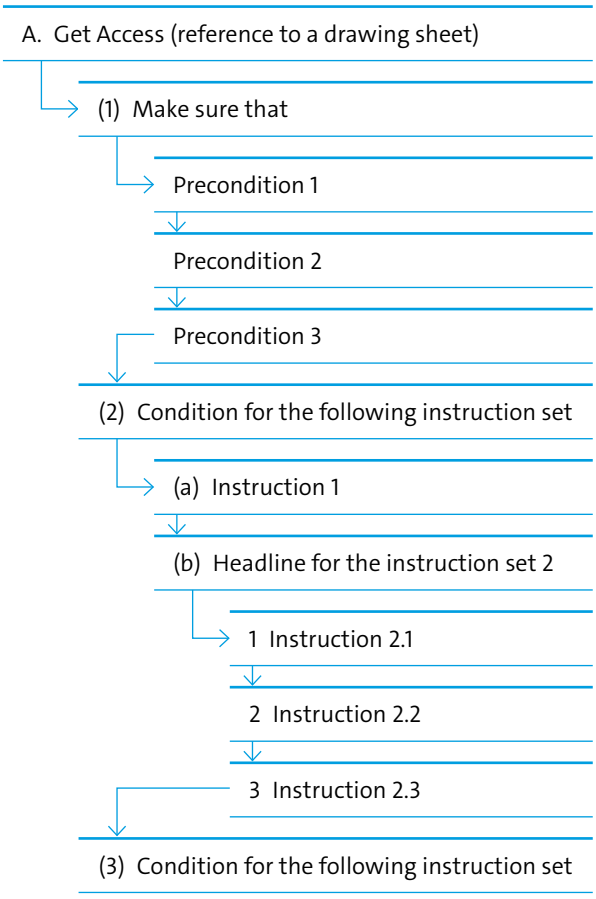

Abbildung 29: Beispiel-Schema einer Instruktions-Layout-Semantik

8.7 Erklär- und Nachvollziehbarkeit

Erklärbarkeit ist existenziell und Teil der Qualitätskontrolle bei der Bestimmung von Falsch-Positiven und Falsch-Negativen. Jeder der einzelnen Wissensextraktionsschritte baut aufeinander auf. D. h., wenn man sich einen bestimmten Schritt in der HOL-Wissensrepräsentation anschauen will, müssen zuvor alle anderen Schritte in einer deterministischen Abfolge abgearbeitet werden. Das bedeutet im Umkehrschluss, dass sich jedes Detail einer Instruktion auf seinen Ursprung im Text aufgrund der logischen Verkettung der Bezüge zurückverfolgen lässt.
Dieser deterministische Bezug zwischen Ursache und Wirkung ist eine der großen Stärken des Verfahrens und der Grund, weswegen sich die semantische KI insbesondere für präzise Erkennung in sensitiven Arbeitskontexten eignet (operativ kritische oder juristische Anwendungsdomänen). Siehe auch Annex 1 für eine beispielhafte Umsetzung.

8.8 Die IT-technische Umsetzung der Wissensextraktion

Die IT-technische Umsetzung der Wissensextraktion kommt aufgrund der Effizienz und Tragfähigkeit der verwendeten KI-Verfahren ohne umfangreiche manuelle Modellierung aus.
Sie greift stattdessen in automatisierter Form auf vorhandene Vorwissensstrukturen (z. B. public domain Wissensnetze) zu und wird auf etablierten Systemen prozessiert. OntoBroker ist ein kommerzieller HOL-Reasoner, der seit 1999 kontinuierlich erweitert und verbessert wurde.
Seit der Version 6.0 wird F-Logic 2 (ObjectLogic) unterstützt und derzeit liegt OntoBroker in der Version 6.3 vor. OntoBroker ist vollständig in Java implementiert und kann logische Auflösungen unter Ausnutzung von Multicore Prozessoren massiv parallel durchführen.

In der verwendeten Ausführung ist der Reasoner OntoBroker eng in Microsoft Excel integriert (MSO365, Excel 2019), wobei der Reasoner sowohl lokal (Windows 10) als auch in beliebiger Cloud-Installation laufen kann (Azure, AWS, nativer Linux-/Windows-Server oder als Docker):

Die Deklarationen der HOL-Modelle sind in Excel Zellen erfasst und dadurch leicht handhabbar (N-Gramm Matcher, a priori Wissen, etc.).

- Textdaten werden von externen Dateien in Excel eingelesen oder schlicht hineinkopiert und dort in für den OntoBroker verarbeitbare ObjectLogic-Repräsentation konvertiert.
- Externe Massendaten wie Wörterbücher (z. B. WordNet) oder technische Taxonomien werden von Excel gesteuert in OntoBroker eingelesen.
- Das transformierte Wissen wird in Excel zur weiteren computerbasierten automatisierten Verwendung ausgegeben.
- Die Verwaltung von fehlerhaften (bzw. nicht vorhandenen) Zuordnungen wird in Excel durchgeführt (Fortsetzung des Trainingsprozesses aus Schritt 5 zur Erweiterung der N-Gramm Sammlung, der benutzerdefinierten Wörterbücher, etc.).

8.9 Zusammenfassung und Ausblick

HOL-Matcher erscheinen als das einfachste und effizienteste Verfahren um präzise Wissensextraktion aus Texten vorzunehmen. Präzision ist bei operativ kritischen bzw. in juristischen Anwendungsdomänen existenziell. Jedes inhaltlich relevante Detail eines Textes muss berücksichtigt werden und man benötigt die Rückverfolgungsmöglichkeit bzw. die Erklärbarkeit der Erkennung zur Verifikation der Ergebnisse.

Der HOL-Matcher zur Erkennung von Textmustern kann auch bei anspruchsvolleren HOL-Modellen eingesetzt werden. Das von uns auf dieser Basis entwickelte Vertragsmodell OntoLegal benötigt eine Anzahl von N-Grammen im oberen zweistelligen Bereich, um die Vielfalt der Ausdrucksformen abzudecken. Zukünftig wird hier ein semantischer Multi-Grid-Ansatz die Anzahl der N-Gramme weiter reduzieren.

8.10 Annex 1: Beispiel (simplifiziert) zur Rückverfolgung von Ergebnissen

Es gibt drei Aussagen A1–A3, in einem Satzprädikat P: P(A1[a, b, c]), P(A2[d, e, f]), P(A3[i, j, k]). Diese Aussagen habe jeweils unterschiedliche Attribute anhand derer die Aussagen gematched werden sollen. Ein Matcher dazu ist m(S1):M[d, e, f] (lies: der Matcher m für die Semantik S1 aus der Klasse der Matcher mit den Matching-Attributen d, e, f), der also immer Aussagen matched, die die Attribute d, e, f haben. Die HOL-Funktion @{f1} T1[?S] :- P(?A[?p1, ?p2, ?p3]), m(?S):M[?p1, ?p2, ?p3] »erzeugt« ein neues F-Atom, nämlich T1[S1], da der Matcher m(S1) einen Match gefunden hat. Falls eine weitere Funktion hinzu kommt, die bestimmte Semantiken modifiziert, z. B. Begriffe mithilfe einer Taxonomie normiert (aus S1 mach S11): @{f2} T2[?SMod] :- T1[?S], SException(?S, ?SMod), hat man zwei hintereinander durchgeführte Transformationen und der Ausgangspunkt ist nur durch die Rückverfolgung der Transformationsschritte möglich. Mit der Funktion TraceFunc() geht diese wie folgt: ?- TraceFunc(T2(S11), ?ID, ?RootBody). Das Ergebnis ist dann: ?ID = [f2, f1], also eine Liste von hintereinander durchlaufende Funktionen (diese kann auch verschachtelt sein bei komplexen Abhängigkeiten) und ?RootBody = [[P(A1[d, e, f]), m(S1):M[d, e, f]]], also der Ausgangspunkt des Matching-Prozesses.