Schwachstellenermittlung und Rückmeldungsprinzipen in einem intelligenten Tutorensystem für juristische Argumentation

Niels Pinkwart*, Vincent Aleven*, Kevin Ashley**, Collin Lynch**

**HCI Institute
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh PA 15213, USA
nielsp@cs.cmu.edu
aleven@cs.cmu.edu

**School of Law & Intelligent Systems
Program, University of Pittsburgh
3900 Forbes Ave
Pittsburgh PA 15260, USA
ashley@pitt.edu
collinl@cs.pitt.edu

Abstract: Dieser Artikel stellt ein intelligentes Tutorensystem für die Rechtswissenschaften vor. Das System, welches bisher im Rahmen von einigen Pilotstudien getestet wurde, soll Jurastudenten helfen, vor Gericht notwendige Argumentationsstrategien zu lernen. Im verwendeten Ansatz werden Gerichtsprotokolle als Lernmaterialien verwendet: Studenten annotieren diese und erstellen graphische Repräsentationen des Argumentationsverlaufs. Das System kann dabei zur Reflexion der von Anwälten vorgebrachten Argumente anregen und Lernende auf mögliche Schwächen in ihrer Analyse des Disputs hinweisen. Dies geschieht größtenteils in Form von Aufforderungen zur Selbsterklärung – eine für präzisere Rückmeldungen notwendige exakte Definition von Korrektheit ist in der betrachteten Domäne oft nicht möglich. Zur Erkennung von Schwächen verwendet das System Graphgrammatiken und kollaborative Filtermechanismen. Die Prioritätsbestimmung für Rückmeldungen, welche in diesem Artikel genauer dargestellt wird, berücksichtigt mehrere Faktoren des Benutzungskontextes.

1 Einleitung

E-Learning-Systeme haben im Bereich der Rechtswissenschaften eine lange Tradition und existieren für ganze eine Reihe juristischer Disziplinen [MB01, Ro92]. In der juristischen Praxis ist Argumentation von zentraler Bedeutung: die Fähigkeit, vor Gericht gute Argumente zu formulieren und angemessen im Kontext von Gegenargumenten zu verteidigen, kann oft über Erfolg oder Misserfolg entscheiden und ist daher für Anwälte essentiell. Daher ist juristische Rhetorik und Argumentation Gegenstand vieler Übungsveranstaltungen in Jura-Studiengängen. Es ist in diesem Zusammenhang überraschend, dass trotz der Vielfalt von e-Learning-Systemen für verschiedene juristische Bereiche nur wenige Anwendungen wie CATO [Al03] und ArguMed [Ve03] existieren, die Lernende speziell in der Argumentation unterstützen. CATO operiert auf Basis existierender Fälle und Urteile, ArguMed zielt auf strukturelle Aspekte in Argumenten ab und hilft Lernenden, visuelle Repräsentationen anfechtbarer

Aussagen zu erstellen. Teilweise kann die geringe Anzahl von e-Learning-Systemen für juristische Argumentation dadurch erklärt werden, dass die zugrunde liegende Domäne schlecht strukturiert ist. Im Gegensatz zu gut strukturierten Problemdomänen wie z.B. Mathematik gibt es im Bereich der juristischen Argumentation meist keine eindeutig definierbare „korrekte Lösung einer Aufgabe“, die als Modell in einem Tutorensystem verwendet werden könnte: obwohl Argumente typischerweise rational auf Basis von Gesetzestexten begründet werden, ist der Argumentationsprozess ein sprachlicher Diskurs, in dessen Verlauf juristische Prinzipien und Gesetzesartikel im Angesicht spezifischer Faktenlagen interpretiert werden. Die fehlende Lösungseindeutigkeit wird in zahlreichen Urteilsrevisionen in höheren Instanzen wie auch in der Schwierigkeit, Verfahrensausgänge zu prognostizieren, deutlich.

Dieser Artikel beschreibt ein Tutorensystem, das Studenten beim Lernen juristischer Argumentationsstrategien unterstützt. Hierbei nutzen die Studenten Protokolle von Gerichtsverfahren und beschreiben die Positionen und Beiträge der Akteure in Diagrammform. Das System kann dabei zur Reflexion der durch Anwälte vorgebrachten Argumente anregen und die Lernenden auf mögliche strukturelle und inhaltliche Schwächen in ihrer Analyse des Disputs hinweisen. Technisch ist dies durch Graphgrammatiken und kollaborative Filteralgorithmen realisiert. Die Systemrückmeldungen erfolgen in Form von Aufforderungen zur Selbsterklärung [Ch00]. Im Schwerpunkt beschreibt dieser Artikel die Prioritätsbestimmung für Rückmeldungen, welche den aktuellen Kontext der Systemnutzung durch den Lernenden über mehrere charakteristische Parameter ermittelt.

2 Diagramme zur Argumentvisualisierung

In Gerichtsverhandlungen formulieren die Anwälte beider Seiten üblicherweise Vorschläge zur Interpretation der Gesetzestexte, auf Grund derer der behandelte Fall zu Ihren Gunsten entschieden werden sollte. Diese Vorschläge haben oft die strukturelle Form einer auch auf vergleichbare Faktenlagen anwendbaren allgemein gehaltenen Entscheidungsregel (Test). Um abzuwägen, welcher Argumentationslinie das Gericht in seiner Urteilssprechung folgt, konstruieren Richter oft hypothetische Sachverhalte mit Herausforderungscharakter und befragen die Anwälte, wie deren Argumente im Bezug auf die hypothetischen Sachverhalte zu bewerten seien. Diese rechtfertigen typischerweise ihre Positionen, z.B. durch Veränderung ihres Tests oder Unterscheidung der Faktenlage von der Hypothese. Diese Dispute vor Gericht sind die fundamentale Basis für das gesamte angloamerikanische „common law“-System, treten jedoch auch im kontinentaleuropäischen System mit römischer Tradition zunehmend auf. Hier spielen sie vor allem auf Verfassungsebene sowie in den EU-Gerichtshöfen eine wichtige Rolle [MS97]. Sie illustrieren hervorragend die wichtigen Prozesse der juristischen Argumentation und Begriffsbildung [As90]. Protokolle dieser Argumentationen stellen eine wertvolle Lernressource für Jurastudenten dar, welche jedoch auf Grund der enormen Komplexität der Argumente schwierig zu nutzen ist. Eine Möglichkeit, mit dieser Komplexität umzugehen, besteht darin, Diagramme zur Argumentrepräsentation einzusetzen.

Forschungsergebnisse zeigen, dass integrierte Text/Graphik-Repräsentationen eine Lernhilfe darstellen können [Ai99] und die Fähigkeit zum kritischen Denken schulen [Ge02]. Auch speziell im juristischen Bereich existieren einige e-Learning-Systeme, die verschiedene Arten von Diagrammen zur Darstellung von Argumenten verwenden. Carr [Ca03] verwendet hierzu Toulmin-Schemata, im System Araucaria [RR04] kommen Voraussetzungs/Schlussfolgerungsketten zum Einsatz. ArguMed [Ve03] beinhaltet zusätzlich zu visuellen Repräsentationen einen „Assistenten", der Diagrammstrukturen analysiert und den Lernenden auf dieser Basis Hilfe anbietet. TakeLaw[1], dessen Schwerpunkt eher auf Fallanalyse als auf Disput liegt, erkennt ebenfalls strukturelle Fehler in studentischen Lösungsdiagrammen und bietet multimedial unterstützte Korrektur von Aufgaben, jedoch keine freie Texteingabemöglichkeiten an. Dies ist einerseits ein Vorteil, da gesicherte Aussagen über Korrektheit von Diagrammen möglich werden, andererseits aber hinsichtlich der Ausbildung von Argumentationsfähigkeiten nicht unproblematisch. Zu keinem der genannten Systeme existiert jedoch eine empirische Studie zur Belegung der lernunterstützenden Wirkung. Auch bietet kein existierendes System zur Argumentationsunterstützung eine Hilfefunktion an, die über rein strukturelle Faktoren in Diagrammen hinausgeht. So bleibt der unzweifelhaft sehr wichtige vom Lernenden eingetragene Inhalt in Diagrammelemente typischerweise auf Grund von fehlenden geeigneten Verfahren uninterpretiert. Wie in Abschnitt 3 und [Pi06] beschrieben, versuchen wir dies im hier vorgeschlagenen Ansatz mit Hilfe von kollaborativen Filteralgorithmen zu ändern. Unser Ansatz verwendet „Tests", „Fakten" und „Hypothesen" als primäre ontologische Kategorien zur Analyse von juristischen Argumenten, und konsequenterweise auch als Elementtypen in Diagrammen. Ähnlich wie das Araucaria-System [RR04] erlaubt unser Ansatz, ein Textdokument (Verfahrensprotokoll) mittels eines Diagramms zu annotieren. Hierbei kann jedes Element des Diagramms mit einem speziellen Teil des Textes in Beziehung gesetzt werden. Abbildung 1 zeigt das von einem Studenten im Rahmen von Pilotversuchen mit dem System erstellte Diagramm[2]. Die linke Seite des Fensters enthält den aktuell sichtbaren Teil des Protokolls und eine Palette mit den Elementen und Relationen (Modifikation, Unterscheidung, Analogiebildung, kausale Beziehung, allgemeine Beziehung), welche zwischen den Elementen herstellbar sind. Auf der rechten Seite ist das vom Studenten erstellte Diagramm im scrollbaren Arbeitsbereich zu sehen.

3 Schwachstellenermittlung in Diagrammen

Die Lokalisierung von „Fehlern" in Argumentationsdiagrammen ist aus zwei Gründen problematisch: eine eindeutige „beste Lösung" existiert oft nicht, und die Interpretation des Textinhalts der Diagramme ist mit heutigen Verfahren in der hier betrachteten Domäne nicht möglich.

---

[1] http://www.take-law.de

[2] Das System wurde im englischsprachigen Raum getestet, daher wurde eine übersetzte Version des Programms mit Material des US Surpreme Courts (hier der Fall Lynch vs Donnelly, 1983) verwendet.

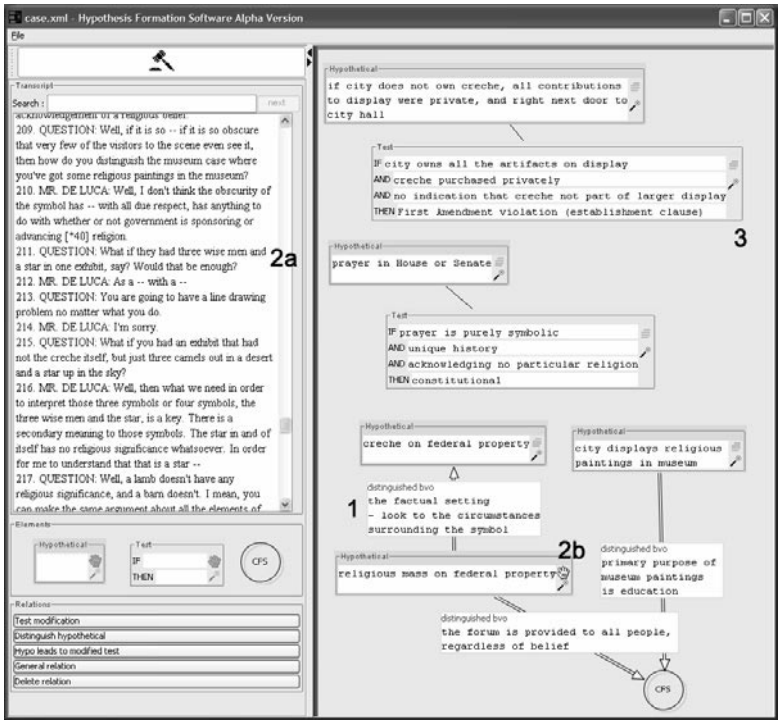

Abbildung 1. Beispiel eines Argumentationsdiagramms

Unser Verfahren beruht auf der Ermittlung von *Schwächen* (im Sinne einer Heuristik für *potenzielle* Schwachstellen in Argumenten) in Diagrammen. Hierbei werden strukturelle, kontextbezogene und inhaltliche Schwächen unterschieden. Unser Ansatz zur Ermittlung dieser drei Typen von Schwächen wird im Folgenden kurz beschrieben und ist in [Pi06] technisch detaillierter dargestellt. *Strukturelle* Schwächen sind allein auf Grund der im Diagramm auftretenden Typen (Elemente und Relationen) definiert. So werden in Abbildung 1 etwa zwei Hypothesen voneinander unterschieden (1). Dies kann durchaus im Verfahren geschehen, typischerweise sollten Hypothesen jedoch auch in Beziehung zu den Fakten des Falls gesetzt werden. Dies ist im Diagramm für die Hypothese „creche on federal property“ nicht erkennbar. Eine weitere strukturelle Schwäche des Diagramms besteht darin, dass drei der Hypothesen mit keinem Test in Verbindung stehen. Da Hypothesen zum Ziel haben, Tests zu überprüfen (auf Sinngehalt und Konsistenz mit den Artikeln des Gesetzes), stellt das Fehlen solcher Verbindungen eine Schwäche dar.

Kontextbezogene Schwächen beziehen sich auf die vom Lernenden erzeugten Verbindungen zwischen den Diagrammelementen und dem Verhandlungsprotokoll. Sind etwa wichtige Teile des Protokolls gar nicht oder mit einem unangemessenen Diagrammelement markiert worden (z.B. Auffassung einer Hypothese als Faktum), so stellt dies eine kontextbezogene Schwäche dar.

In Abbildung 1 wurde z.B. die im wichtige Hypothese (2a) nicht im Diagramm berücksichtigt, weiterhin ist das Element unter (2b) gar nicht mit dem Protokoll verlinkt. Kontextbezogene und strukturelle Schwächen werden im System mit Hilfe einer auf dem Diagramm operierenden Graphgrammatik erkannt. Diese Grammatik enthält sowohl Regeln, welche den Erstellungsprozess von Diagrammen modellieren, als auch „Fehlerregeln“ zur Lokalisierung von 44 konkreten Schwächenarten in Diagrammen. Strukturelle Fehlerregeln sind dabei unabhängig vom konkret verwendeten Gerichtsprotokoll, während kontextbezogene Fehlerregeln protokollabhängige Parameter haben. Durch die Grammatikregeln kann eine Ähnlichkeit von Diagrammen mit einer partiell definierbaren „Expertenlösung“ überprüft werden. Dies ist sinnvoll, da Argumentationsdiagramme zwar nicht eindeutig sind, lokale Lösungsbedingungen (z.B. eine Markierung der wichtigsten Passagen im Protokoll) jedoch durchaus möglich sind.

Auch wenn ein Argumentationsdiagramm keine strukturellen Schwächen aufweist und alle wichtigen Textpassagen sinnvoll mit Diagrammelementen verlinkt sind, können Verständnisschwierigkeiten beim Lernenden nicht ausgeschlossen werden – es ist z.B. möglich, dass ein Student die Hauptaussage eines vorgebrachten Arguments nicht verstanden hat und demzufolge keine qualitativ gute Beschreibung im entsprechenden Diagrammelement enthalten ist. Diese *inhaltlichen* Schwächen sind schwierig zu diagnostizieren. Selbst für einen menschlichen Tutor ist es nicht einfach, zu entscheiden, ob das Element (3) in Abbildung 1 eine adäquate Wiedergabe des durch den Anwalt im Verfahren vorgebrachten Tests ist – für ein Computersystem ist die Entscheidung nicht leichter. In unserem Ansatz werden inhaltliche Schwächen durch eine Variante eines kollaborativen Filteralgorithmus [KR02] erkannt. Hierbei wird ausgenutzt, dass durch die strukturellen und kontextbezogenen Regeln systemseitig bekannt ist, welches Diagrammelement welchen Teil des Protokolls beschreibt. Ist eine wichtige Passage des Protokolls vom Lernenden annotiert worden, bietet das System eine Auswahl von Beschreibungsalternativen der gleichen Textpassage an, die durch andere Studenten erstellt wurden (für die ersten Nutzer, bei denen noch keine Lösungen anderer vorliegen, werden Musterlösungen variierender Qualität verwendet). Der Lernende hat nun die Aufgabe, alle Alternativen zu bestimmen, die er für gut hält. Da dem System eine heuristische Bewertung der anderen Alternativen vorliegt, ist aus der Auswahl des Lernenden annäherungsweise bestimmbar, ob dieser das Argument in der entsprechenden Textpassage verstanden hat oder nicht. Hinzu kommt, dass ab diesem Zeitpunkt auch die eigene Lösung anderen Systemnutzern zur Bewertung vorgelegt wird. Diese entscheiden durch ihr Auswahlverhalten (Empfehlung/keine Empfehlung), wie sich die systeminterne Heuristik zum Verständnis der Textpassage durch den Lernenden verändert. Bei einer insgesamt geringen Qualitätseinschätzung wird eine inhaltliche Schwäche im System diagnostiziert.

4 Rückmeldungen zu Benutzeraktionen

Eine wichtige Designentscheidung für ein Tutorensystem besteht darin, in welcher Form Lernern Rückmeldungen zu deren Aktionen gegeben werden.

Hierbei sind mehrere Faktoren zu berücksichtigen, die in unserer Anwendung auf Grund der verwendeten visuellen Repräsentation und der schlecht strukturierten zugrunde liegenden Domäne anders als in vielen Tutorensystemen gelagert sind. Diese werden in den folgenden Unterabschnitten näher diskutiert.

4.1 Inhalt der Rückmeldungen

Wichtigster Aspekt ist sicherlich der *Inhalt der Rückmeldungen*. In vielen intelligenten Tutorensystemen wird der Lernende in Rückmeldungen über Fehler in seinen Aufgabenbearbeitungen informiert, z.B. in Form von Tipps, die schrittweise bis hin zur Preisgabe von Lösungsteilen gehen können. Dies ist in unserer Anwendung nicht möglich, da das System weder über ein präzises Lösungsmodell noch über eine definitive Fehlererkennung verfügt und auch nicht verfügen kann: so ist es z.B. durchaus möglich, dass der Lernende den Sinnzusammenhang eines Arguments verstanden hat, aber nicht die „passende“ Stelle im Text markiert hat. Ebenso kann durch eine vermeintlich „falsche“ Verwendung der im System angebotenen Elemente und Relationen nicht automatisch darauf geschlossen werden, dass der Nutzer die dadurch repräsentierten Argumentationskonzepte nicht verstanden hat – obwohl dies durchaus möglich ist. Eine direkte Rückmeldung eines zu korrigierenden Fehlers ist in solchen unsicheren Diagnosesituationen nicht angemessen. Folgt man jedoch der Idee der „Schwäche“ in Argumentationsdiagrammen im Sinne der Lokalisierung von potenziellen Fehlern, so bietet sich es an, erkannte Schwächen dazu zu nutzen, den Lerner zur Selbsterklärung des betreffenden Argumentationsteils aufzufordern. Selbsterklärungen haben sich als effektive Lehrmethode in schlecht strukturierten Domänen erwiesen [SR02], und die Nutzung von erkannten Schwächen in Diagrammen vermeidet „unnötige“ Aufforderungen zur Selbsterklärung von Problemteilen, in denen sich der Lernende gut auskennt. Hier ein Beispiel einer Systemrückmeldung zu einer erkannten Kontextschwäche „wichtiger Teil des Textes wurde nicht markiert“:

> In Deiner Analyse des Textes hast Du einen wichtigen Teil übersehen. Bitte lies den nun gezeigten Teil nochmals – findest Du hier einen durch einen Anwalt vorgeschlagene Entscheidungsregel, die nahe legen soll, wie der Fall zu entscheiden ist?
> Erkläre Dir selbst, welche Bedeutung der gezeigte Textausschnitt im Verfahren hat. Wenn Du zur Ansicht kommst, dass hier ein Test geäußert wird, dann füge ein entsprechendes Element zum Diagramm hinzu und verlinke es mit dem Text. Notiere Deine Gedanken im Textfeld.

4.2 Prioritätsbestimmung für Rückmeldungen

Die große Mehrheit der existierenden intelligenten Tutorensysteme arbeitet dialogorientiert. Hierdurch ist zu jedem Zeitpunkt dem System genau bekannt, an welchem Teil einer Aufgabe der Benutzer arbeitet bzw. eine Eingabe macht. Auf einer eindeutigen Zuordnung von Domänenmodell und Systemeingaben basieren z.B. die kognitiven Tutoren [An95]. Bei Benutzerschnittstellen, die eine solche Zuordnung nicht gewährleisten, wird Mehrdeutigkeit typischerweise durch (teils störende) direkte Nachfragen aufgelöst [ACC89]. Selbst eine solche Strategie scheidet jedoch in der hier betrachteten Anwendung auf Grund des fehlenden eindeutigen Domänenmodells aus.

Kommen freiere Interaktionsformen und ein fehlendes eindeutiges Domänenmodell zusammen, so ist für ein Tutorensystem nicht mehr exakt bestimmbar, an welchem Teil der Aufgabe ein Lerner arbeitet und wie genau der Stand seiner Aufgabenbearbeitung zu bewerten ist – d.h., eine *optimale* Rückmeldung oder Hilfestellung ist nicht genau bestimmbar. Gleichzeitig kann es aber durchaus möglich sein, dass das System nach Analyse des aktuellen Standes der Aufgabenbearbeitung eine Vielzahl von *möglichen* Rückmeldungen ermittelt. In der von uns betrachteten Anwendung ist dies der Fall: in den Pilotstudien traten Diagramme auf, die über 100 verschiedene Rückmeldungsmöglichkeiten aktivierten. Da eine Überfrachtung des Lernenden mit Feedback wenig sinnvoll ist, ergibt sich das Problem der *Selektion von Rückmeldungsmöglichkeiten*: wie kann das Tutorensystem aus der Vielzahl von zum Diagramm passenden Rückmeldungen die für den Lernenden beste auswählen?

Der von uns hierzu verwendete Algorithmus basiert auf dem Prinzip, möglichst genau den aktuellen Arbeitskontext des Lerners zu bestimmen. Hierzu wird zunächst für jede mögliche Rückmeldung eine Priorität berechnet. Dann wird pro Kategorie (d.h. pro rückmeldungserzeugender Regel in der Graphgrammatik) jeweils die Rückmeldung mit der höchsten Priorität bestimmt und schließlich hieraus diejenigen fünf mit der höchsten Priorität ausgewählt. Diese werden dann dem Benutzer zur Auswahl angeboten (siehe 4.3). Dieses Verfahren garantiert, dass typgleiche Rückmeldungen, welche sich nur im Kontext im Diagramm unterscheiden, ausgeschlossen sind. Eine Priorität wird dabei auf Basis von vier Parametern berechnet:

1. Visuelle Information: Rückmeldungen zu aktuell sichtbaren Teilen des Textes bekommen eine hohe Priorität, nicht sichtbare Teile eine geringere abhängig von der Entfernung zum sichtbaren Textausschnitt. Auch der aktuell sichtbare Ausschnitt des Arbeitsbereichs wird berücksichtigt: sind alle zu einer Rückmeldung gehörenden Diagrammelemente im Moment sichtbar (und damit evtl. im Fokus des Lernenden), bekommt diese eine hohe Priorität.

2. Zeitliche Information: Da davon ausgegangen werden kann, dass kürzlich editierte Elemente des Diagramms wahrscheinlicher im Arbeitskontext des Benutzers liegen, bekommen Regeln, die Rückmeldungen zu diesen anbieten, eine hohe Priorität. Analog wird beim betrachteten Teil des Textes berücksichtigt, wann eine Passage das letzte Mal angezeigt wurde – Rückmeldungen, die sich auf Textteile beziehen, die vor langer Zeit gelesen wurden, erhalten nur eine geringe Priorität.

3. Typbasierte Information: Jede Rückmeldung ist zu einer Regel in der Graphgrammatik assoziiert, wobei jede Regel auf viele Stellen des Diagramms passen kann. Um den Benutzer nicht mit zu vielen ähnlichen Rückmeldungen nacheinander zu langweilen, merkt sich das System, wann eine Regel der Grammatik das letzte Mal Basis für eine Rückmeldung war. Die kürzlich zurückliegenden erhalten dann eine niedrigere Priorität.

4. Phaseninformation: Im Rahmen erster Pilotstudien mit dem System konnten wir typische wiederkehrende Verwendungsphasen (die nicht notwendig in einer festgelegten Reihenfolge auftreten) ausmachen: Orientierung im System (A), Markierung und Verlinkung des Textes (B), Verbindung von Diagrammelementen (C), Analyse und Korrektur des Diagramms (D) und Reflexion (E). Ist es möglich, diese Phasen im System zumindest heuristisch zu bestimmen, so kann die Rückmeldung auch hierauf eingehen, indem zur Phase passende Meldungen eine höhere Priorität erhalten.

Die Bestimmung der aktuellen Benutzungsphase ist der algorithmisch anspruchsvollste Teil der Kontextbestimmung. Wir stellen im Folgenden zwei Möglichkeiten dar, diese annäherungsweise zu bestimmen – eine exakte Berechnung ist in unserem Verfahren weder notwendig noch möglich, weil Phasen nicht scharf voneinander zu trennen sind und sich teilweise nicht in Aktionen im System manifestieren (z.B. kurzzeitige Reflexion über das Diagramm ohne Eingaben).

Grammatikbasierter Ansatz

Die erste von uns betrachtete Möglichkeit zur Bestimmung der Benutzungsphase basiert auf den in der Graphgrammatik enthaltenen Produktionsregeln. Jede der Fehlerregeln (und nur diese werden für Rückmeldungen genutzt) lässt sich relativ eindeutig einer der fünf Benutzungsphasen zuordnen. Tabelle 1 illustriert diesen Zusammenhang.

Phase	*Beispiel für Aktivierungsbedingung einer zugehörigen Produktionsregel*
A: Orientierung im System	Diagramm enthält keine „Hypothesen"-Elemente
B: Markierung und Verlinkung des Textes	Keines der „Test"-Elemente im Diagramm annotiert eine wichtige Stelle im Textdokument, die die Formulierung eines Tests beinhaltet
C: Verbindung von Diagrammelementen	Ein „Test"-Element im Diagramm ist mit keinem „Hypothesen"-Element verbunden
D: Analyse und Korrektur des Diagramms	Das Diagramm enthält ein „Hypothesen"-Element, das in kausaler Beziehung („führt zu") zu mehreren Tests steht
E: Reflexion	Im Diagramm sind zwei weder direkt noch indirekt verbundene „Test"-Elemente enthalten (Rückmeldung: Anregung zur Diskussion dieses Teils des Arguments)

Tabelle 1. Regelzuordnung zu Benutzungsphasen

Diese Zuordnung von einzelnen Produktionsregeln zu Phasen liefert die Basis für eine Heuristik zur Bestimmung der aktuellen Benutzungsphase, in der sich der Lernende befindet. Hierzu werden zunächst alle anwendbaren Regeln bestimmt und die Summe der anwendbaren Regeln pro Phase berechnet. Da die unterschiedlichen Phasen verschieden viele zugeordnete Regeln haben (in unserem Fall: A:4, B:11, C:11, D:13, E:5), werden diese absoluten Anzahlen noch hierzu ins Verhältnis gesetzt und resultieren in einer Wahrscheinlichkeitsverteilung für die aktuelle Benutzungsphase. Dieses Vorgehen stellt sicher, dass z.B. drei passende „Phase A-Regeln" ein größeres Gewicht haben als drei passende „Phase B-Regeln". Ein Beispiel zu diesem Algorithmus: im in Abbildung 1 gezeigten Argumentdiagramm sind insgesamt neun Regeln anwendbar.

Davon gehören fünf zu Phase B (z.B.: „Wichtige Textstelle nicht beachtet“), eine zu Phase C und drei zu Phase E (z.B.: „Reflektiere über isoliere Hypothese“). Damit ergibt sich folgende Heuristik für die aktuelle Benutzungsphase: P(A)=0%, P(B)=40%, P(C)=8%, P(D)=0% und P(E)=52%. Der Lernende ist also mit hoher Wahrscheinlichkeit entweder noch in der Phase der Studie und Markierung des Textes oder schon in der Reflexionsphase. Diese Wahrscheinlichkeiten sind direkt als Prioritäten verwendbar: jede Regel erhält im Parameter „Phaseninformation“ die ihrer Phase entsprechende Wahrscheinlichkeit zugeordnet.

Aktionsbasierter Ansatz

Eine weitere Datenquelle, die zur heuristischen Bestimmung der aktuellen Benutzungsphase verwendet werden kann, sind die vom Benutzer im System durchgeführten Aktionen. So ist das Erzeugen einer Kante im Graph eher der Phase C zuzurechnen als der Phase B, und das Editieren eines bereits bestehenden Elements kann auf Phase D eher als auf Phase A hinweisen. Eine eindeutige Zuordnung von Aktionen zu Phasen ist jedoch nicht möglich. Die beobachtbaren Benutzeraktionen, welche in einem heuristischen Zusammenhang zu nicht beobachtbaren Benutungsphasen stehen, legen die Verwendung eines Hidden Markov Modells (HMM) zur Bestimmung der wahrscheinlichsten aktuellen Benutzungsphase nahe. Die in Tabelle 2 dargestellte Matrix enthält die von uns im Modell verwendeten Übergangswahrscheinlichkeiten zwischen den Phasen. Diese beruhen auf unseren Beobachtungen in den Pilotversuchen und weiteren Annahmen, wie z.B. dass ein direkter Übergang von der „Startphase“ A in die „Revisionsphase“ D nicht möglich ist.

Nach → Von ↓	A	B	C	D	E
A: Orientierung im System	0,4	0,4	0,2	0	0
B: Markierung und Verlinkung des Textes	0,1	0,3	0,3	0,1	0,2
C: Verbindung von Diagrammelementen	0,1	0,2	0,2	0,3	0,2
D: Analyse und Korrektur des Diagramms	0,1	0,2	0,2	0,3	0,2
E: Reflexion	0	0,1	0,1	0,2	0,6

Tabelle 2. Übergangswahrscheinlichkeiten zwischen Benutzungsphasen im HMM

Diese Werte stellen eine Heuristik für typische Phasenabfolgen dar. Analog verhält es sich mit dem zweiten für die Spezifikation des HMM notwendigen Parameter, der Beobachtungswahrscheinlichkeit einzelner Aktionen in den jeweiligen Phasen. Eine allgemeine Zuordnung ist relativ leicht möglich (so weist z.B. das Löschen einer Relation im Diagramm auf Phase D oder E hin, während die Markierung eines Teils des Protokolls den Phasen A oder B zugeordnet werden kann), konkrete für das HMM benötigte Wahrscheinlichkeiten sind ohne empirische Nutzungsdaten zunächst nur schätzbar. Wir haben uns in den durchgeführten Pilotversuchen zunächst für die in Tabelle 3 angegebenen Wahrscheinlichkeiten (Gleichverteilung aller innerhalb einer Phase beobachtbaren Aktionen) entschieden. Die konkreten Wahrscheinlichkeiten für die aktuelle Benutzungsphase können mittels des Viterbi-Algorithmus [Vi67] bestimmt werden. Dieser gibt direkt die Wahrscheinlichkeiten für spezifische Phasenfolgen aus, über Summierung erhält man die Wahrscheinlichkeit einer spezifischen Endphase.

In Phase → Aktion ↓	A	B	C	D	E
Hinzufügen eines Diagrammelements	0,33	0,25	0	0,125	0,125
Löschen eines Diagrammelements	0	0	0	0,125	0,125
Verlinken eines Elements mit dem Verfahrensprotokoll	0	0,25	0	0	0
Markieren eines Teils des Verfahrensprotokolls	0,33	0,25	0	0	0
Beschreibung zu einem Diagrammelement hinzufügen	0,34	0,25	0	0	0
Beschreibung in einem Diagrammelement ändern	0	0	0	0,125	0,125
Hinzufügen einer Relation zum Diagramm	0	0	0,5	0,125	0,125
Löschen einer Relation im Diagramm	0	0	0	0,125	0,125
Beschriftung einer Relation erstellen	0	0	0,5	0,125	0,125
Beschriftung einer Relation ändern	0	0	0	0,125	0,125
Ändern des Typs einer Relation	0	0	0	0,125	0,125

Tabelle 3. Beobachtungswahrscheinlichkeiten im HMM

Diese nutzen wir (analog zum grammatikbasierten Ansatz) für die Prioritätsbestimmung von einzelnen Regeln. Im Gegensatz zum grammatikbasierten Ansatz, bei dem die Phasenbestimmung alleine auf Basis des aktuellen durchgeführt wird, berücksichtigt der aktionsbasierte Ansatz die Entstehungsgeschichte des Diagramms. Damit wird der Kontext des Lernenden eher prozess- als produktorientiert definiert. Eine eindeutige zum Beispiel in Abbildung 1 passende Heuristik einer Benutzungsphase ist daher ohne Angabe der Historie des Diagramms nicht möglich. Für einen sehr einfachen, der aus den minimal zur Konstruktion des Diagramms notwendigen 36 Schritten besteht, ergeben sich folgende Wahrscheinlichkeiten: P(A)=7%, P(B)=18%, P(C)=18%, P(D)=28% und P(E)=29%. Es wird ersichtlich, wie diese von den Ergebnissen des grammatikbasierten Ansatzes abweichen – auch wenn in beiden Verfahren Phase E als die Wahrscheinlichste ausgegeben wird. Insbesondere sind auf Grund der notwendigerweise nicht exakten Zuordnung von Beobachtungen zu Phasen die Ausgabewahrscheinlichkeiten weniger scharf voneinander abgegrenzt als im grammatikbasierten Verfahren. In ersten Pilotversuchen haben sich beide Verfahren als geeignet (im Sinne der Auswahl von „passenden“ Rückmeldungen) erwiesen.

4.3 Zeitpunkt für Rückmeldungen

Ein weiterer wichtiger Faktor beim Design eines Rückmeldungsmechanismus ist, *zu welchem Zeitpunkt* vom System Feedback angeboten wird. Forschungsergebnisse lassen hier zwar keine allgemeingültigen Aussagen über einen besten Zeitpunkt zu, eine Vielzahl von Studien belegen jedoch, dass sofortige Rückmeldungen nicht weniger effektiv als verzögerte Rückmeldungen sind [Ba91]. In unserer Anwendung sind sofortige Rückmeldungen jedoch nur wenig hilfreich: sind Fehler nicht exakt zu diagnostizieren, würden häufige Systeminterventionen (insbesondere wenn sie zu kognitiv aufwändigen Selbsterklärungen führen) den Benutzer eher stören und in seinem Gedankengang unterbrechen als hilfreich sein.

Zusätzlich ist es bei einer konstruktiv orientieren Umgebung wie der von uns verwendeten nicht immer erkennbar, wann eine „Eingabe“ beendet ist: ein im Diagramm ausgedrückter Gedanke kann durchaus mehrere Elemente und Relationen umfassen. Es ist unangemessen, wenn hier während der Erstellung des Diagramms das System mit „Hilfestellungen“, die etwa ausdrücken, dass noch wichtige Teile im Diagramm fehlen, interveniert. Aus diesen Gründen haben wir uns dazu entschlossen, den Benutzer aktiv entscheiden zu lassen, wann er Rückmeldungen bzw. Hilfe bekommen möchte, selbst wenn dies das Risiko birgt dass ggf. verfügbare Hilfe nicht genutzt wird [AK00]. Der Benutzer kann sich über einen „Hilfe“-Button jederzeit einzeilige Kurzversionen von fünf Rückmeldungen anzeigen lassen. Dies trägt der nicht eindeutig bestimmbaren „besten“ Rückmeldung Rechnung und reduziert dabei gleichzeitig die kognitive Belastung (Lesen zu vieler kompletter Rückmeldungstexte) auf ein Minimum. Wählt der Lernende eine dieser Kurzversionen aus, so bekommt er die entsprechende Meldung angezeigt und wird ggf. zu einer Selbsterklärung aufgefordert. Der Zusammenhang einer erkannten Schwäche im Argumentationsdiagramm wird dabei visuell hervorgehoben.

5 Zusammenfassung und Ausblick

Das in diesem Artikel beschriebene System dient dazu, Jurastudenten im Bereich des gerichtlichen Argumentierens auszubilden. Der Ansatz beruht auf der Annotierung von Gerichtsprotokollen mittels graphischer Modelle, welche eine Domänenontologie repräsentieren. Die durch die Studenten erstellten und mit dem Protokoll verlinkten Modelle werden durch ein Verfahren, welches technisch auf Graphgrammatiken und kollaborativem Filtern beruht, auf Schwächen überprüft. Das System liefert Rückmeldungen zu Schwächen in Form von Aufforderungen zu Selbsterklärungen. Dies erscheint im betrachteten Problembereich, in dem oft viele Lösungen akzeptierbar sind und eine klare Entscheidung zwischen richtig und falsch teils unmöglich ist, sinnvoller als Fehlermeldungen, wie sie in traditionellen Tutorensystemen verwendet werden.

Im Unterschied zu eher dialogorientierten Systemen, bei denen der aktuelle Fokus des Lernenden leicht durch das aktive Eingabefeld zu bestimmen ist, tritt in unserer Anwendung das Problem auf, dass zu einem Diagramm typischerweise viele Rückmeldungen möglich sind ohne dass dabei a priori klar wäre, welche zur aktuellen Aktivität des Benutzers am besten passt. Die Beantwortung dieser Frage ist, insbesondere auf Grund der von uns verwendeten kognitiv sehr aufwändigen Rückmeldungsart, essentiell. Dieser Artikel stellt ein Verfahren zur Prioritätsbestimmung für Rückmeldungen vor, welches mehrere Faktoren (lokaler und zeitlicher Kontext, Rückmeldungshistorie und Benutzungsphase) des Benutzungskontexts berücksichtigt. Für die heuristische Bestimmung der Benutzungsphase haben wir zwei Verfahren (prozessorientiert vs. produktorientiert) näher diskutiert. Im Rahmen von Pilotstudien haben sich beide Ansätze als grundsätzlich praktikabel erwiesen. Wir planen, vor geplanten größeren empirischen Studien mit dem System zunächst genauere Vergleiche der Ansätze durchzuführen. Hierzu werden wir die von den Studenten aus den Kurzmitteilungen ausgewählten Rückmeldungen mit den systemseitigen Phasenheuristiken vergleichen.

Literaturverzeichnis

[ACC89] Anderson, J.; Conrad, F.; Corbett, A.: Skill acquisition and the LISP tutor. Cognitive Science 13. 1989; S. 467-505

[AK00] Aleven, V.; Koedinger, K.: Limitations of Student Control: Do Students Know when they need help? In Proceedings of the International Conference on Intelligent Tutoring Systems. Springer Verlag, Berlin, 2000; S. 292-303

[Ai99] Ainsworth, S.: The functions of multiple representations. Computers and Education 33. 1999; S. 131-152

[Al03] Aleven, V.: Using Background Knowledge in Case-Based Legal Reasoning: A Computational Model and an Intelligent Learning Environment. Artificial Intelligence 150. 2003; S. 183-238

[An95] Anderson, J.; Corbett, A.; Koedinger, K.; Pelletier, R.: Cognitive tutors: Lessons learned. The Journal of Learning Sciences 4. 1995; S. 167-207

[As90] Ashley, K.: Modeling Legal Argument: Reasoning with Cases and Hypotheticals. MIT Press/Bradford Books, Cambridge, 1990

[Ba91] Bangert-Drowns, R. et al.: The instructional effect of feedback in test-like events. Review of Educational Research 61(2). 1991; S. 213-238

[Ca03] Carr, C.: Using Computer Supported Argument Visualization to Teach Legal Argumentation. In Visualizing Argumentation. Springer Verlag, London, 2003; S. 75-96

[Ch00] Chi, M.: Self-explaining expository texts: The dual process of generating inferences and repairing mental models. In (Glaser, R. Hrsg.): Advances in Instructional Psychology. Lawrence Erlbaum, Hillsdale, 2000; S. 161-238

[Ge02] van Gelder, T: Argument Mapping with Reason!Able. The American Philosophical Association Newsletter on Philosophy and Computers. 2002; S. 85-90

[KR02] Konstan, J.; Riedl, J.: Collaborative Filtering: Supporting social navigation in large, crowded infospaces. In Designing Information Spaces: The Social Navigation Approach. Springer Verlag, Berlin, 2002; S. 43-81

[MS97] MacCormick, D.; Summers, R.: Interpreting Precedents: A Comparative Study. Dartmouth Publishing, Aldershot, 1997

[MB01] Muntjewerff, J.; Breuker, J.: Evaluating PROSA, a system to train solving legal cases. In Proceedings of the International Conference on Artificial Intelligence in Education. IOS Press, Amsterdam, 2001; S. 278–285

[Pi06] Pinkwart, N. et al.: Toward Legal Argument Instruction with Graph Grammars and Collaborative Filtering Techniques. In Proceedings of the International Conference on Intelligent Tutoring Systems. Springer Verlag, Berlin, 2006; S. 227–236

[Ro92] Routen, T.: Reusing formalisations of legislation in a tutoring system. Artificial Intelligence Review 6. 1992; S. 145–159

[RR04] Reed, C.; Rowe, G.: Araucaria: Software for Argument Analysis, Diagramming and Representation. International Journal of AI Tools 14. 2004; S. 961-980

[SR02] Schworm, S.; Renkl, A.: Learning by solved example problems: Instructional explanations reduce selfexplanation activity. In Proceedings of the Annual Conference of the Cognitive Science Society. Lawrence Erlbaum, Mahwah, 2002; S. 816-821

[Ve03] Verheij, B.: Artificial argument assistants for defeasible argumentation. Artificial Intelligence 150. 2003; S. 291-324

[Vi67] Viterbi, A.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory 13(2). 1967; S. 260–267