Automatisierte Bewertung bei MOOC

Seminar aus Informatik

Siegmar Alber, Luca Debiasi

24.05.2013

Einleitung

Aufgabenstellung

- Automatisierte Bewertung bei MOOC
  (Massive Open Online Courses)
- Ziel automatisierter Bewertung:
  - Viele Studenten
  - Betreuung durch Lehrenden ermöglichen und erleichtern
  - Feedback für Studenten
  - Möglichkeit zur selbstständigen Verbesserung von Arbeiten durch Studenten

Grundidee

- Verschiedene Aufgaben:
  - Unterstützung bei Korrektur von Aufgaben
  - Automatisierte Benotung
  - Überprüfung des Wissensstandes
- Betrachtet werden soll:
  - Korrektheit/Funktionalität
  - Qualität
  - Form/Design
  - Überprüfung auf Plagiate
- Unter Berücksichtigung von:
  - Unterschiedlichen Ausdrucksweisen
  - Bemühen des Studenten

Automatisierte Bewertung

Arten von Dokumenten

- Lückentexte und Multiple Choice
- Texte
  - freie Texte
  - offene Fragen
  - Übersetzungen (z. B. Englisch–Chinesisch)
- Mathematische Formeln
  - Statistik Übungen
- Mischformen
  - Dokumente mit technischen Texten, mathematischen Formeln, Diagrammen und Graphiken
- Programmieraufgaben
- Verwendung von Software
  - Tabellenkalkulation
  - UNIX (Shell)
  - Systemkonfiguration (z. B. virtuelle Maschinen)

UNIVERSITÄT SALZBURG

Texte

- Zu beachtende Merkmale:
  - Form
  - Inhalt
  - Stil
  - Aufbau und Argumentation
- Systeme zur automatisierten Bewertung:
  - Inhalt: Intelligent Essay Assessor (IEA)
  - Stil: Project Essay Grade (PEG)
  - Beides: Paperless School free-text Marking Engine (PS-ME)

Texte - Technische Umsetzung

- Inhalt:
  - Latent Semantic Analysis (LSA)
  - Matrix von Wörtern in verschiedenem Kontext
  - Vergleich mit Referenztext(en)
  - Ziel: Konzepte in Dokument finden, auch wenn gesuchtes Wort nicht explizit enthalten
- Stil:
  - Analyse von linguistischen Merkmalen
  - Länge des Textes (Anzahl der Wörter)
  - Worthäufigkeiten
  - Vorkommen von Relativpronomen, Präpositionen, ...
  - Ziel: Komplexität des Textaufbaus soll erfasst werden

UNIVERSITÄT SALZBURG

Texte - Technische Umsetzung

- Kombination aus Stil und Inhalt:
  - Natural Language Processing (NLP)
  - Vergleich mit „positiven“ und „negativen“ Referenztexten
  - Verschiedene Parameter beschreiben Wissenstand des Studenten und Korrektheit/Stil des Inhaltes
  - Feedback für Student anhand dieser Parameter

Programmieraufgaben

- Zu untersuchende Merkmale:
  - Syntax
  - Semantik
  - Korrektheit
  - Code-Qualität
  - Verwendung von Patterns

Programmieraufgaben - Frameworks/Methoden

- Generic Automated Marking Environment (GAME) oder Sakai:
  Einfache Codeanalyse (Anzahl Variablen, Kommentare, Schleifen. . .) und Output-Vergleich; für verschiedene Programmiersprachen geeignet
- AutoGrader oder Infandango:
  Statische Code-Analyse, Student implementiert Interace, dann Unit-Tests auf diesem Interface
- AnalyseC:
  Parst Code, baut AST (Abstract Syntax Tree), führt darauf normierende Transformationen aus und vergleicht ihn mit Referenz

Mathematische Formeln

- Zu untersuchende Merkmale:
  - Syntax
  - Korrektheit
- Technische Umsetzung:
  - MathML:
    Dokumentenformat zur Darstellung mathematischer Formeln und komplexer Ausdrücke
  - MathTree:
    Darstellung von mathematischen Gleichungen mit Hilfe einer Baum-Datenstruktur
  - Überprüfung:
    Mustererkennung durch überwachtes Lernen (Support Vector Machines (SVMs)), erkennt Vorkommen von MathTrees in Dokumenten

UNIVERSITÄT SALZBURG

Weitere Frameworks/Methoden

- BOSS: Unterstützt administrative Abläufe (Abgabe von Arbeiten, Bewerten, Feedback...), Bewertung selbst ist auswechselbar
- SAVE: Überprüfen von Wissen über Naturphänomene durch das Anpassen von Parametern in simulierten Welten
- Ontology-Driven Auto-evaluation for e-Learning Approach (ODALA): Allgemein verwendbares System basierend auf einer Ontologie

Weitere Frameworks/Methoden

- Spreadsheet marking: Überprüft in einer Tabellenkalkulation eingegebene Formeln
- VM system administration: Shell-Skripte, die in der VM ausgeführt werden, überprüfen ihre Funktion und Konfiguration
- Automated Predictive Assessment from Unstructured Student Writing: Prognose von Noten basierend auf Beteiligung in Foren

Diskussion und Ausblick

Allgemein

- Ziel: Konsistenz bei Bewertung und qualitativ hochwertiges Feedback
- Einsatz von künstlicher Intelligenz in verschiedenen Bereichen

Text

- Zu Beginn geringe Akzeptanz durch indirektes Messen von Schreibfähigkeiten
- Täuschen des Systems relativ einfach
- Außerdem kein Feedback möglich
- Genauere Analyse durch LSA und NLP nötig
- Leseverständlichkeit von Texten hängt mit Qualität zusammen, wie dieser geschrieben wurde
- Dadurch können auch Lernmaterialien verbessert werden

Programmieren

- Nicht nur Output vergleichen
- Engagement von Studenten messen
- Ausblick: Versionskontrolle, Teamwork, design patterns, unit testing, test-driven development, user stories, ...

Plagiat-Erkennung

- Großes Problem in MOOCs
- Einfache Informationsbeschaffung über WWW Segen und Fluch zugleich
- Erkennung durch automatische Systeme (Coursera) $\rightarrow$ Effizienz?
- Zusammenarbeit mit Test-Centern (edX, Udacity) $\rightarrow$ zusätzliche Kosten

Fazit

Nutzen

- Objektivere Bewertung
- Verarbeiten von großen Datenmengen
- Weltweiter Zugriff möglich
- Ermöglicht hochqualitative Lehre für breites Publikum
- Kosten- und Zeitersparnis
- Technologien können auch in anderen Bereichen eingesetzt werden

Probleme

- Es braucht für jede Art von Problem ein eigenes Verfahren und standardisierte Bewertungskriterien, (derzeit) keine allgemein gültige Methode
- Verschiedenste Probleme bei unterschiedlichen Dokumenten:
  - Textverständnis
  - Überprüfen von Interaktivität (z. B. Menüs) bei Programmieraufgaben
- Verarbeiten von großen Datenmengen
- Oft Überprüfung der automatisierten Bewertung nötig

Fragen, Diskussion