Kombiniertes Mining von strukturellen und relationalen Daten

Frank Eichinger Klemens Böhm

Institut für Programmstrukturen und Datenorganisation (IPD),
Universität Karlsruhe (TH), Deutschland, {eichinger, boehm}@ipd.uka.de

Zusammenfassung

Data Mining Techniken wie Klassifikation, Regression und Clusteranalyse finden heutzutage eine weite Verbreitung. Entsprechende relationale Daten liegen in vielen Anwendungsdomänen vor, und effiziente Data Mining Algorithmen sind in kommerzielle Werkzeuge sowie in Datenbank Management Systeme integriert. In den letzten Jahren wurden aber auch verschiedene strukturelle Data Mining Techniken entwickelt, die z. B. mit Graph-basierten Daten arbeiten. Solche Techniken erschließen neue Anwendungsgebiete, bieten aber auch das Potential, bisherige Techniken zu ergänzen. Oft können durch Kombination bisherige Ergebnisse verbessert werden. In diesem Beitrag präsentieren wir Arbeiten aus dem Bereich der Vorhersage von Kundenverhalten und der Fehlersuche in Software, in denen strukturelle und relationale Data Mining Techniken erfolgreich kombiniert wurden. Schließlich geben wir einen Ausblick auf weitere Anwendungsgebiete und zukünftige Herausforderungen.

1 Einleitung

Viele Data Mining Techniken wie Klassifikation und Regression zur Vorhersage von Zielgrößen sind inzwischen weit verbreitet. Entsprechende effiziente Algorithmen sind in kommerziellen Data Mining Anwendungen wie auch in Datenbank Management Systemen (DBMS) integriert. Auch Explorationstechniken wie die Cluster- und Assoziationsregelanalyse, die in der Lage sind, Zusammenhänge in Datensammlungen aufzudecken, finden weite Verbreitung. Solche Data Mining Techniken arbeiten üblicherweise mit Daten, die relational vorliegen. Jede Instanz (z. B. jeder Kunde im Fall von Kundendaten) wird mit einem Tupel von numerischen und kategorischen Werten beschrieben. Im Fall von Kundendaten sind dies z. B. „Alter“, „Umsatz im letzten Jahr“ und „Dauer des Kundenverhältnisses“ als numerische Bestandteile und „Kundengruppe“ und „Wohngegend“ als kategorische Größen. Entsprechende Daten liegen neben Kunden-Datenbanken in den unterschiedlichsten Anwendungsdomänen vor. Dies hat zu einer weiten Verwendung von Data Mining Techniken geführt. In bestimmten Domänen hingegen können Techniken, die auf relationalen Daten operieren, nicht angewendet werden. Dies ist beispielsweise bei der Analyse von Molekülstrukturen in der Pharmaforschung der Fall. In einigen Domänen, in denen Data Mining inzwischen etabliert ist (z. B. bei der weit verbreiteten Analyse von relationalen Kundendaten), wünschen sich Anwender jedoch oft deutlich bessere Genauigkeiten von Vorhersagen, als sie allein mit den bisherigen Techniken möglich sind. Neben der Weiterentwicklung von bisherigen Techniken zur Analyse und Datenvorverarbeitung, sowie der Integration solcher Techniken in verbreitete Produkte wie DBMS, gibt es einen Bedarf an fortgeschrittenen Technologien, die über das bisher übliche hinausgehen.

In den letzten Jahren wurde eine Reihe von strukturellen Data Mining Techniken entwickelt. Dazu gehört vor allem *Sequenz Mining* [1], *Tree Mining* [2] und *Graph Mining* [3]. Diese Techniken werden zunächst zur Exploration und Wissensentdeckung eingesetzt. Sie sind in der Lage, automatisch strukturelle Muster wie Teilsequenzen und Teilgraphen zu finden, die eine bestimmte Häufigkeit in einer Datenbank von Instanzen haben. Mit diesen strukturellen Techniken wird eine Reihe von neuen Analysemöglichkeiten erschlossen. Beispielsweise werden mit Se-

quenz Mining Einkaufsgewohnheiten von Kunden über längere Zeiträume analysiert und Graph Mining, wie bereits erwähnt, in der Pharmaforschung auf Moleküldaten angewendet. So kann genau entdeckt werden, welche Teilstruktur für die Wirkung eines bestimmten Medikaments verantwortlich ist, und nicht nur, dass Moleküle mit bestimmten gemessenen Eigenschaften diese Wirkung erzielen. Es wird also direkt mit Strukturen gearbeitet und nicht mit beispielsweise numerischen Daten, die eine bestimmte Struktur in aggregierter Form beschreiben.

Strukturelles Data Mining stellt aber keineswegs eine Ablösung von etablierten Techniken dar, sondern eine Ergänzung. Interessant sind vor allem Kombinationen von herkömmlichen und strukturellen Techniken, da sich so die Stärken von beiden Herangehensweisen gegenseitig ergänzen können. So kann die Einbeziehung von struktureller Information in herkömmliche Verfahren zur Steigerung der Ergebnisqualität beitragen, bzw. können etablierte Techniken strukturelles Data Mining ergänzen, um so neue Anwendungsgebiete zu erschließen. Verbesserte Ergebnisse sind dabei vor allem deshalb möglich, weil etablierte Techniken oft mit numerischen Daten arbeiten, die Strukturen beschreiben. Diese können hingegen mit strukturellen Techniken unmittelbar und ohne Verluste betrachtet werden, die oft durch die Reduktion auf numerische Größen entstehen.

Im Folgenden stellen wir Arbeiten vor, bei denen wir durch Integration von Sequenz Mining und Klassifikation die Qualität von Voraussagen von Kundenverhalten verbessern konnten und es durch die Kombination mit Feature Selection Algorithmen ermöglichst haben, beim Graph Mining auch numerische Kantengewichte mit einzubeziehen. Des Weiteren werden wir einen Ausblick auf weitere Arbeiten geben, bei denen wir durch Kombination von herkömmlichen und strukturellen Techniken verbesserte Ergebnisse erwarten.

2 Voraussagen von Kundenverhalten

In der Telekommunikationsbranche gehört es zu einer der wichtigsten Fragen, Voraussagen darüber machen zu können, welche Kunden mit hoher Wahrscheinlichkeit kündigen werden, um den Anbieter zu wechseln. Mit verschiedenen Data Mining Techniken werden dazu Voraussagen getroffen [4], die eine wichtige Information für Marketing und Geschäftsführung darstellen. Die Analyse von Kundendaten ist in aller Regel schwierig, so dass Voraussagen meist nicht außerordentlich gut sind, allerdings oft besser als gar keine.

Zu den Kundendaten gehört neben den beschriebenen relationalen Daten auch die Interaktions-Historie mit dem Anbieter. Hierzu gehören z. B. Anrufe bei der Service-Line des Anbieters sowie Reparaturen, Störungsmeldungen, Bestellungen etc. Diese Daten müssen für die Anwendung von herkömmlichen Techniken aggregiert werden. Das führt zu einer Vielzahl von dünn-besetzten Attributen wie der Anzahl der Reparaturen im letzten Monat, Jahr und insgesamt. In [5] haben wir Sequenz Mining eingesetzt, um häufige Sequenzen von Kundenverhalten aufzudecken. Das Ergebnis ist, dass zwar interessante Sequenzen gefunden wurden, dass diese alleine aber nicht geeignet sind, das zukünftige Kundenverhalten vorauszusagen.

Aus diesem Grund haben wir ein kombiniertes Verfahren entwickelt: In einem ersten Schritt suchen wir häufige Sequenzen in einem Kundendatensatz. Davon ausgehend identifizieren wir für jede dieser Sequenzen die Kunden, die die jeweilige Sequenz von Interaktionen in ihrer Historie aufweisen. Ausgehend von der strukturellen Information, welche Sequenzen ein Kunde enthält, werden die Kunden also in Gruppen eingeteilt. Für jede dieser Kundengruppen lernen wir dann einen Entscheidungsbaum, der neben Daten zu den jeweiligen Interaktionen auch den allgemeinen Kundendatensatz mit einbezieht. Um dann neue Kunden zu klassifizieren und Voraussagen zu machen, überprüfen wir zunächst, welche Sequenzen in der Historie eines Kunden enthalten sind. Davon ausgehend wählen wir die spezialisierten Entscheidungsbäume aus, mit denen der Kunde mit seinem spezifischen Interaktionsverhalten klassifiziert werden kann. Liegen mehrere unterschiedliche Klassifikationen für einen Kunden vor, so kombinieren wir diese mit Ensemble-Techniken [6]. Der Vorteil von diesem Vorgehen liegt im Wesentlichen darin, dass jeder Entschei-

dungsbaum nicht einfach auf alle Kunden angewendet wird, sondern auf homogenere Gruppen. Das führt zu besseren Ergebnissen. Auf diesem Weg können auch die Informationen betrachtet werden, die zu den einzelnen Interaktions-Events gehören, beispielsweise die Kosten einer Reparatur. Nach herkömmlichem Vorgehen hätte solche Information nur in aggregierter Form analysiert werden können. Das bedeutet im Allgemeinen einen Verlust an potentiell nützlicher Information.

Mit dem beschriebenen Verfahren konnten in einem Sample von einem echten Kundendatensatz 19,5% aller Kunden im Voraus entdeckt werden, die dann tatsächlich gekündigt haben. Die False-Positive-Rate lag dabei bei lediglich 2,6%. Diese Ergebnisse sind etwas besser als die Ergebnisse mit bisherigen nicht-strukturellen Verfahren auf derselben Datenbasis.

3 Fehlerlokalisierung in der Softwaretechnik

Auch in intensiv getesteter Software kommt es immer wieder zu Fehlfunktionen nach der Auslieferung, die mit hohen Kosten einhergehen. Aus diesem Grund spielen Werkzeuge, die in der Lage sind, Fehler zu entdecken, eine wichtige Rolle. Besonders schwer zu finden sind die so genannten *noncrashing bugs*, bei denen es nicht zum Abbruch des Programms kommt, sondern zu einem Fehlverhalten wie der Berechnung eines falschen Ergebnisses. Bei solchen Fehlern werden keine *stack traces* ausgegeben, die dem Entwickler einen Hinweis geben, wo sich ein Fehler versteckt.

In der Literatur gibt es vielfältige Ansätze, Fehler mit Hilfe von Data Mining zu finden. In [7] werden beispielsweise Quellcode-Metriken aus der Softwaretechnik mit Daten aus einer Bug-Datenbank zusammengeführt und mit Regressionstechniken analysiert. Für die Metrik *cyclomatic complexity* [8] wird hier z. B. ein starker Zusammenhang zur Fehleranfälligkeit von Software festgestellt. Diese basiert auf den Kontrollflussgraphen einzelner Funktionen und dem Grad der Verzweigungen dieser. Ein Kontrollflussgraph umfasst die sogenannten Basic Blocks mit mehreren Nicht-Sprung-Befehlen als Knoten und die verschiedenen Ausführungswege als Kanten. Die cyclomatic complexity ist aber dennoch nur eine Kenngröße, die aus Graphen abgeleitet wird, die genaueren Strukturen selbst können in ihr nicht abgebildet werden.

Es gibt aber auch Arbeiten [9, 10], die die *Call-Graphen* von fehlerfreien und fehlerhaften Programmausführungen mit Graph Mining Techniken analysieren und auf diesem Weg Fehler lokalisieren. Solche Call-Graphen spiegeln die Aufrufstruktur von Methoden bzw. Funktionen einer Programmausführung wieder. Ausgehend von der `main()`-Methode werden weitere Methoden aufgerufen, die als Kindsknoten dargestellt werden und selbst ggf. weitere Methoden aufrufen (s. Abb. 1(a)). Im Allgemeinen werden solche Call-Graphen sehr groß. Das liegt vor allem daran, dass durch Schleifen bestimmte Teilstrukturen sehr oft hintereinander ausgeführt werden. Da Graph Mining Algorithmen im Allgemeinen nicht gut skalieren, müssen die Call-Graphen zunächst reduziert werden. In [9] wird der Call-Graph derart reduziert, dass jede Methode nur einmal vorkommt (*totale Reduktion*). In [10] dagegen werden nur Iterationen betrachtet: Taucht eine Teilstruktur mehr als zweimal auf, werden alle darüber hinausgehenden Teilstrukturen aus dem Graphen entfernt. Beispiele für die verschiedenen Reduktionsstufen finden sich in Abb. 1.

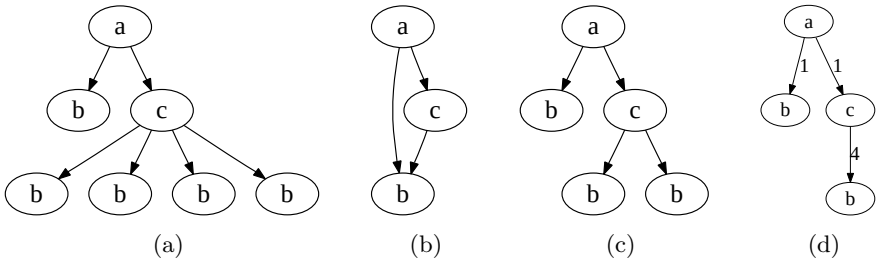

Abbildung 1: **(a)** Ein unbearbeiteter Call-Graph, **(b)** der gleiche Graph total reduziert wie in [9], **(c)** dargestellt wie in [10] und **(d)** reduziert mit Kantengewichten.

Wir sind der Überzeugung, dass bei der totalen Reduktion [9] zu viel potentiell wichtige Information verloren geht. Auch die Reduktionstechnik in [10] ist problembehaftet. Es kann z. B. passieren, dass bei Fehlern in Schleifenbedingungen eine Teilstruktur in fehlerfreien Ausführungen einige wenige Male ausgeführt wird, in fehlerhaften Fall aber einige hunderte Male. In beiden genannten Reduktionsverfahren resultiert das in identischen Graphen im fehlerfreien und fehlerhaften Fall. Aus diesem Grund schlagen wir vor, Call-Graphen derart zu reduzieren, dass Iterationen strukturell zu einer zusammengefasst werden, die Ausführungshäufigkeit jedoch als Kantengewicht notiert wird. Auf diese Art können wir die Größe der Graphen signifikant verringern und verlieren nur ein Minimum an Information (s. Abb. 1(d) für ein Beispiel).

Nachdem wir Programme ausgeführt haben, entscheiden wir durch Vergleich des Ergebnisses mit einem Referenz-Ergebnis, ob die Ausführung als fehlerfrei oder fehlerhaft zu klassifizieren ist. Dann reduzieren wir die Call-Graphen wie oben beschrieben, wobei Kantengewichte entstehen. Die so gewonnene Datenbank von Call-Graphen analysieren wir nun mit dem Graph Mining Algorithmus *CloseGraph* [11] – die Kantengewichte werden dabei zunächst außen vor gelassen. Dies resultiert in einer Menge von häufigen Teilgraphen. Da wir uns auf die Analyse von Kantengewichten konzentrieren wollen, betrachten wir nur die Teilgraphen, die sowohl in fehlerfreien als auch in fehlerhaften Ausführungen auftreten. Ausgehend von solchen Graphen betrachten wir die Kantengewichte, mit dem Ziel herauszufinden, welche Gewichte einen besonders großen Beitrag leisten, fehlerfreie von fehlerhaften Ausführungen zu unterscheiden. Dazu verwenden wir etablierte Feature Selection Algorithmen, die auf Entropie (Informationsgewinn) basieren. Auf diesem Weg gelangen wir zu einem Ranking, welche Methodenaufrufe eines Programms am wahrscheinlichsten für fehlerhafte Ausführungen verantwortlich sind.

Da unterschiedliche Arten von Fehlern unterschiedlich gut mit verschiedenen Techniken entdeckt werden können, kombinieren wir die Ergebnisse unseres auf Kantengewichten basierenden Verfahrens mit dem rein strukturell arbeitenden Verfahren aus [10]. Als Ergebnis wird eine kombinierte Liste von Methoden ausgegeben, die nach der Reihenfolge der Wahrscheinlichkeiten sortiert ist, einen Fehler zu enthalten. Eine solche Liste kann dann einem Software-Entwickler gegeben werden, der so ganz gezielt einige Methoden noch einmal überprüfen kann. Erste Experimente zeigen, dass auf diesem Weg einerseits Fehler lokalisiert werden können, die mit dem Verfahren aus [10] nicht gefunden werden konnten. Hierbei handelt es sich z. B. um Fehler in Schleifenbedingungen, die sich in unterschiedlichen Kantengewichten niederschlagen. Andererseits tritt im Mittel aller anderen Fälle, durch die Kombination beider Verfahren, eine Verdoppelung der Lokalisierungsgenauigkeit gegenüber der rein strukturellen Analyse in [10] ein.

4 Auswahl von Hardware-Architekturen

Ein interessantes Problem in der Rechnerarchitektur ist es, die Architektur vorherzusagen, die am Besten für ein bestimmtes Programm geeignet ist, auf dem dieses also am schnellsten ausgeführt wird. Dieses Auswahlproblem lässt sich als Klassifikationsproblem formulieren und findet seine Motivation in der zukünftigen Verfügbarkeit von rekonfigurierbarer Hardware und Multi-Core-Prozessoren mit mehreren spezialisierten Kernen. Hier möchte man gerne möglichst zur Laufzeit Aussagen darüber machen können, auf welcher Hardware(-Konfiguration) ein gegebenes Programm am Besten ausgeführt werden kann. Als Eingabedaten können prinzipiell alle Informationen dienen, die ein Programm beschreiben, beispielsweise verschiedene Code-Metriken aus der Softwaretechnik. Eine vielversprechende Herangehensweise ist auch die Analyse der Kontrollflussgraphen der einzelnen Funktionen, die aus einem Programm statisch generiert werden können.

Durch Experimente mit der SPEC-Benchmark-Suite und einer Reihe von Quellcode-basierten Metriken aus der Softwaretechnik haben wir bisher herausgefunden, dass durch die Verwendung von herkömmlichen Data Mining Techniken bereits gute Vorhersageergebnisse werden können. Diese haben bisher eine Genauigkeit von 72%. In ersten Tests mit einer Reihe von Metriken, die

auf Kontrollflussgraphen basieren, haben wir zudem gezeigt, dass diese die bisherigen Ergebnisse stak verbessern können. In Zukunft wollen wir auch in dieser Anwendungsdomäne mit Graph Mining Techniken häufige Substrukturen entdecken, mit denen Software-Artefakte noch präziser beschrieben werden können. So wollen wir Strukturen extrahieren und Informationen über ihr Auftreten in Kombination mit weiteren Metriken mit Standard-Techniken analysieren.

5 Ausblick

In den vorgestellten Arbeiten haben wir das Potential gezeigt, mit unterschiedlichen Kombinationstechniken von strukturellem und relationalem Data Mining zu verbesserten Ergebnissen zu gelangen und dass weitere Forschung in diese Richtung vielversprechend ist. In Zukunft wollen wir weitere Studien in diese Richtung unternehmen. Bei der Fehlerlokalisierung im Gebiet der Softwaretechnik analysieren wir die Kantengewichte zur Zeit in einem dem Graph Mining nachgelagertem Schritt. Diesem nachgelagertem Vorgehen wollen wir einen Ansatz in der Vorverarbeitung (z. B. Diskretisierung) gegenüberzustellen und Möglichkeiten der Behandlung von numerischer Information innerhalb des Graph Mining Algorithmus erforschen.

Vor allem aber wollen wir eine systematische Klassifikation von Kombinationsmöglichkeiten von strukturellen und herkömmlichen Techniken entwickeln, in die sowohl bisherige als auch zukünftige Arbeiten eingeordnet werden können. Daraus kann ein Rahmen entstehen, mit dem die identifizierten Kombinationsmöglichkeiten leicht realisiert werden können.

Durch weitere Studien, auch in anderen Gebieten wie der Transportlogistik, wollen wir einerseits durch kombinierte Methoden zu guten Ergebnissen kommen. Andererseits wollen wir aber auch an neuen Kombinationstechniken sowie an Herausforderungen im Bereich des Graph Minings selbst arbeiten.

Literatur

[1] Rakesh Agrawal and Ramakrishnan Srikant. Mining Sequential Patterns. In *Proceedings of the 11th International Conference on Data Engineering (ICDE)*, 1995.

[2] Y. Chi, R. Muntz, S. Nijssen, and J. Kok. Frequent Subtree Mining – An Overview. *Fundamenta Informaticae*, 66(1–2):161–198, 2005.

[3] D.J. Cook and L.B. Holder, editors. *Mining Graph Data.* John Wiley & Sons, 2006.

[4] Scott A. Neslin, Sunil Gupta, Wagner Kamakura, Junxiang Lu, and Charlotte H. Mason. Defection Detection: Measuring and Understanding the Predictive Accuracy of Customer Churn Models. *Journal of Marketing Research*, 43(2):204–211, 2006.

[5] Frank Eichinger, Detlef D. Nauck, and Frank Klawonn. Sequence Mining for Customer Behaviour Predictions in Telecommunications. In *Proceedings of the Workshop on Practical Data Mining at ECML/PKDD*, 2006.

[6] L.I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms.* John Wiley & Sons, 2004.

[7] Nachiappan Nagappan, Thomas Ball, and Andreas Zeller. Mining Metrics to Predict Component Failures. In *Proceedings of the 28th International Conference on Software Engineering (ICSE)*, 2006.

[8] T.J. McCabe. A Complexity Measure. *Transactions on Software Engineering*, 2(4):308–320, 1976.

[9] Chao Liu, Xifeng Yan, Hwanjo Yu, Jiawei Han, and Philip S. Yu. Mining Behavior Graphs for "Backtrace" of Noncrashing Bugs. In *Proceedings of the 5th SIAM International Conference on Data Mining (SDM)*, 2005.

[10] Giuseppe Di Fatta, Stefan Leue, and Evghenia Stegantova. Discriminative Pattern Mining in Software Fault Detection. In *Proceedings of the 3rd International Workshop on Software Quality Assurance (SOQUA)*, 2006.

[11] Xifeng Yan and Jiawei Han. CloseGraph: Mining Closed Frequent Graph Patterns. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining (KDD)*, 2003.