Patrick Lürwer

Data Wrangling mit KNIME: Screaming-Frog-Crawls für Analysen aufbereiten

Vor der Analyse kommt die Fleißarbeit: Daten müssen erhoben, bereinigt, verknüpft und angereichert werden. Excel stößt dabei sehr schnell an seine Grenzen, denn ein Tabellenkalkulationsprogramm ist kein ETL-Tool – auch nicht mit Add-ons wie SEOTools4Excel oder Analytics Edge. KNIME bietet hier eine geeignete Alternative für alle, die nicht direkt eine Programmiersprache wie R oder Python lernen möchten. Durch ein grafisches Interface können auch Anfänger komplexe Datenprozesse intuitiv und schnell umsetzen sowie dokumentieren. Auf die Weise lassen sich diese jederzeit nachvollziehen, wiederverwenden und gegebenenfalls erweitern.

Bei der täglichen Arbeit mit Daten ist Excel meistens das Werkzeug der Wahl, um „schnell mal“ tabellarische Daten zu bearbeiten, zu verdichten und Einsichten bzw. Erkenntnisse zu gewinnen. Aus „schnell mal“ kann aber auch sehr schnell „sehr lange“ werden, wenn Excel dann doch wieder nicht so will wie man selbst. Insbesondere Manipulationen großer Datenmengen – seien es Crawls, Logfiles, Google-Analytics- oder Google-Search-Appearance-Daten – sind mit Excel kaum noch performant, geschweige denn reproduzierbar durchzuführen. Nach der zehnten Hilfstabelle, gesetzten Filtern, String-Umschreibungen und Datenaggregationen mittels Pivot-Tabellen weiß man dann meistens nicht mehr so genau, was man fünf Schritte zuvor eigentlich getan hat. Hat man dann ein Ergebnis, das einen Fehler in einem der vorausgegangenen Schritte vermuten lässt, kann man den Verarbeitungsweg wieder zurückgehen. Die Zwischenergebnisse von Arbeitsschritten hält Excel nämlich nicht automatisch vor. Alle Datentransformationen werden auf ein und derselben Tabelle vorgenommen – es sei denn, man macht sich die Mühe, jeden Arbeitsschritt in einer neuen kopierten Tabelle durchzuführen. Effizient ist das alles nicht, und spätestens nach drei Monaten, wenn man „mal eben“ gucken will, wie man die Analyse damals gemacht hat, möchte das Zukunfts-Ich das Vergangenheits-Ich am liebsten ohrfeigen, weil nichts dokumentiert wurde.

DER AUTOR

Patrick Lürwer ist Senior Analyst bei get:traction GmbH. Dort ist er für die Datenerfassung, -aufbereitung und -analyse zuständig. Sein tägliches Handwerkszeug sind KNIME, R und Python.

Foto: Worawee Meepian / GettyImages

Vorteil. Insbesondere bei sehr komplexen Workflows gehen gerne mal einige Zeilen verloren, weil man einen Filter falsch gesetzt hat. Da ist es ungemein praktisch, wenn man sich schnell durch die Ergebnistabellen klicken kann, um zu schauen, an welcher Stelle die entsprechenden Zeilen abhandengekommen sind.

Einen weiteren wesentlichen Vorteil von KNIME kann man anhand des Screenshots nicht erkennen, nämlich dass KNIME ohne Probleme mit sehr großen Datenmengen zurechtkommt. Da KNIME ursprünglich aus dem Bereich des Machine Learnings kommt, ist dies auch nicht weiter verwunderlich. Auf die SEO-Welt übertragen profitiert man davon, dass bspw. eine Analyse des internen Linkgraphen, die mehrere Hundert Millionen Zeilen respektive mehrere Gigabyte umfasst, KNIME nicht in die Knie zwingt. Je nach Prozessor- und RAM-Ausstattung dauert die Verarbeitung mal kürzer, mal länger. Dass KNIME aber gänzlich abstürzt, kommt nahezu nie vor.

KNIME entfaltet also sein volles Potenzial, wenn es darum geht, (wiederkehrende) Datentransformationen und -analysen in einem strukturierten, dokumentierten und reproduzierbaren Workflow abzubilden. Ein klassischer Anwendungsfall ist hier die explorative Daten-Analyse (EDA) von (Screaming-Frog-)Crawls.

Jedoch muss man auch hier, wie bei fast jeder EDA, die Faustregel zugrunde legen, dass eine Analyse aus 80 % Datenerhebung und -aufbereitung besteht. Somit bleiben nur 20 % der Zeit für die eigentliche Analyse (siehe http://einfach.st/crowdf, dort S. 6). Gerade die Aufbereitung von Screaming-Frog-Crawls lässt sich jedoch gut standardisieren, sodass sie ein schönes Beispiel für die nachfolgende Einführung in KNIME bietet.

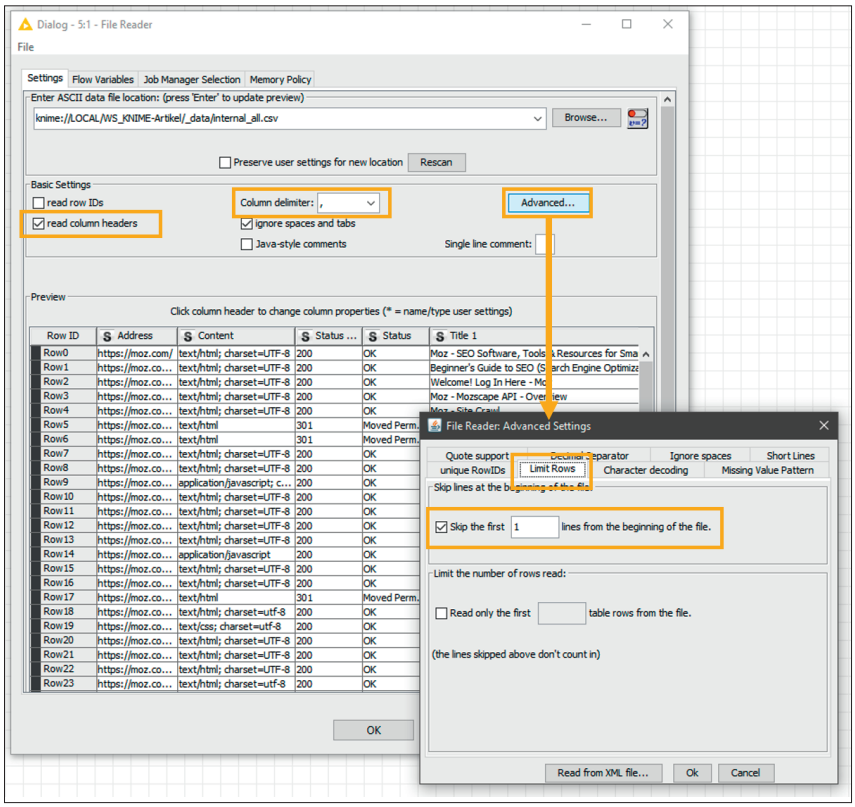


Abb.3: Konfiguration des File Readers zum Einlesen einer CSV

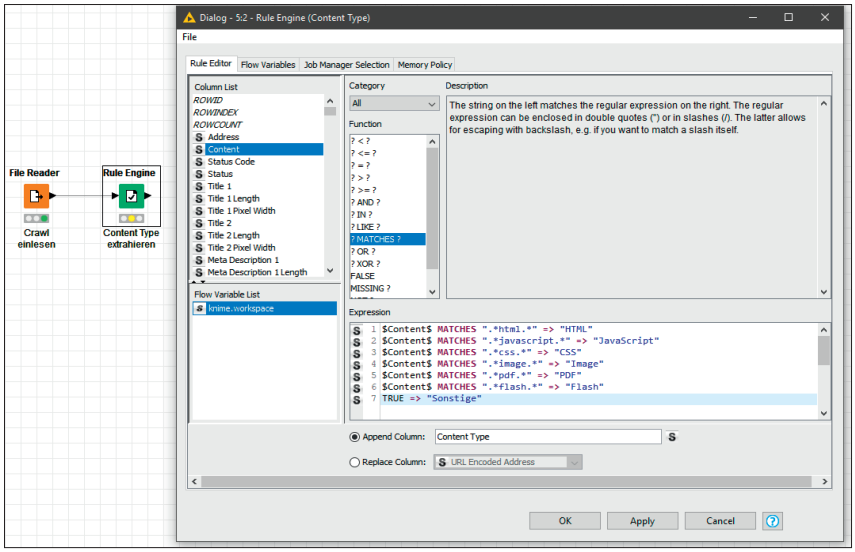


Abb.4: Definition des Regel-Sets in der Rule Engine zur Klassifizierung des Content Types

Einlesen eines Screaming-Frog-Crawls

Zunächst wird die entsprechende Website mittels Screaming Frog gecrawlt. Für diesen Artikel wurde die Website moz.com ausgewählt, die mit einem Crawl-Umfang von ca. 145.000 URLs eine gute exemplarische Datenbasis bietet. Der Crawl wird als CSV (internal_all.csv) exportiert und an beliebiger Stelle gespeichert. In KNIME kann nun über die Suche des Node Repository (im Standardlayout links unten) der **File-Reader-**Knoten gesucht und in den Workflow-Editor gezogen werden. In diesem ist nun der Pfad zum Crawl-Export anzugeben. Außerdem sind noch weitere Einstellungen vorzunehmen. So haben Screaming-Frog-Exporte standardmäßig einen Kommentar am Anfang der CSV, der die Art des Exportes beschreibt. Dieser Kommentar

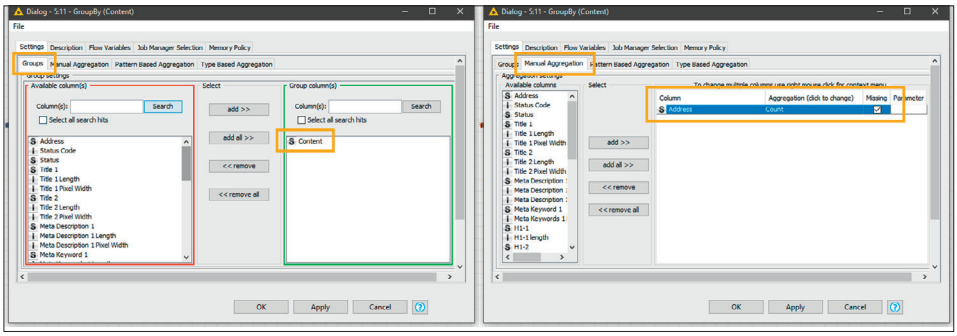


Abb.5: Aggregation des Content Types mittels GroupBy-Knoten

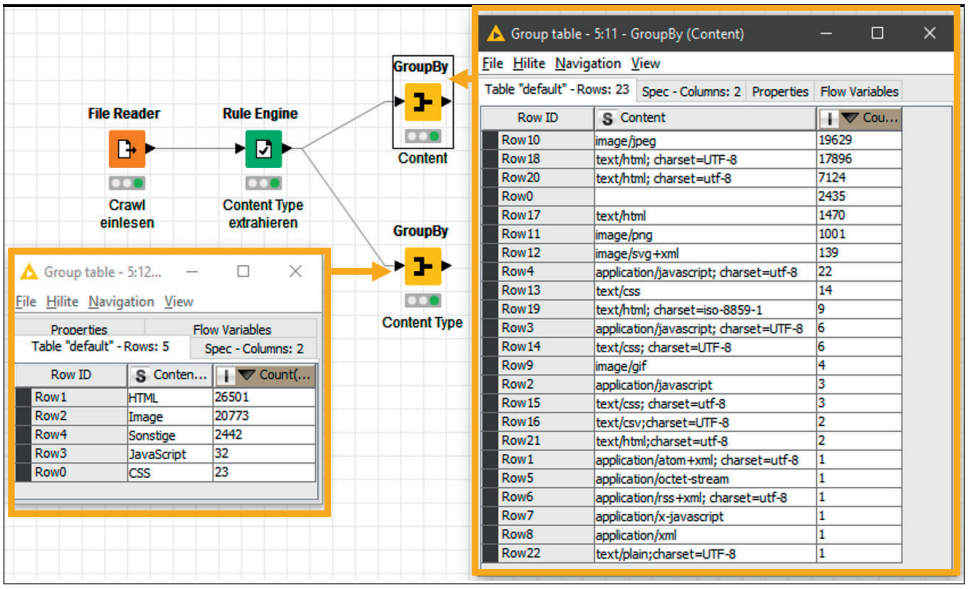


Abb.6: Gegenüberstellung der aggregierten Spalten Content Type und Content

soll natürlich nicht eingelesen werden. Um den **File Reader** zu konfigurieren, wird er mit der rechten Maustaste angeklickt und dann Configure ausgewählt. Im sich öffnenden Dialog sieht man bereits eine Vorschau auf die Daten, so wie sie KNIME als Tabelle einlesen würde. Man erkennt direkt, dass die Einstellungen noch nicht passen. Als Erstes wird daher über

Advanced ... → Limit Rows → Skip first ...

definiert, dass die erste Zeile, die den Kommentar enthält, übersprungen werden soll (Abbildung 3). Gegebenenfalls muss unter dem Reiter Quote support der Text Qualifier (beim Screaming Frog „) angegeben und unter Character decoding UTF-8 ausgewählt werden. Danach können die erweiterten Einstellungen geschlossen, der Haken bei read column headers gesetzt und als Column delimiter ein Komma definiert werden. Im unteren Bereich des Dialogs sollte die Tabelle jetzt korrekt geparst dargestellt werden. Der Dialog wird dann mittels OK geschlossen, der Knoten wieder mit der rechten Maustaste angeklickt und per Execute ausgeführt.

Die Daten werden nun importiert. Die gesamte importierte Tabelle kann wieder über die rechte Maustaste → File Table eingesehen werden.

Content Type ermitteln

Standardmäßig werden bestimmte Informationen im Crawl-Export nur implizit oder zu detailliert dargestellt, sodass im nächsten Schritt Spalten hinzugefügt werden, die die Informationen in gewünschter Form extrahieren oder verdichten.

Ein klassisches Beispiel für die Verdichtung von Informationen ist die Reduktion der Werte aus der Spalte Content. Für Analysen ist es meistens unerheblich, ob eine HTML-Seite mittels UTF-8 oder UTF-16 encodiert ist. Relevant ist nur,

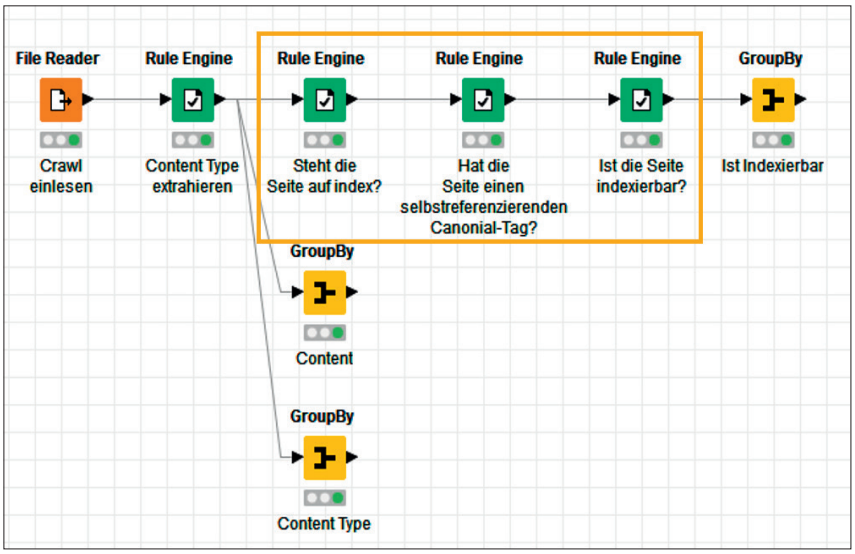


Abb.7: Rule Engines zur Ermittlung der Indexierbarkeit der URLs

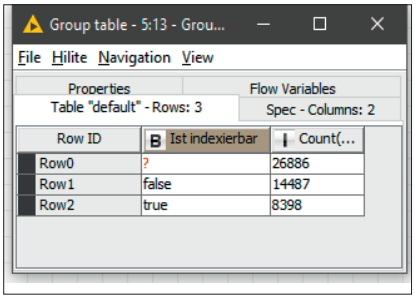


Abb.8: Anzahl der (nicht) indexierbaren Seiten

dass es sich um eine HTML-Ressource handelt. Gleiches gilt für Bilder, bei denen es irrelevant ist, ob sie im PNG- oder im JPEG-Format gespeichert sind. Im Grunde handelt es sich bei diesem Arbeitsschritt somit um eine Klassifizierung der URLs, basierend auf einem Regel-Set. Entsprechend wird der Knoten **Rule Engine** über das Node Repository gesucht, in den Editor gezogen und der Output-Port des **File Readers** mit dem Input-Port der Rule Engine verbunden.

In der **Rule Engine** können nun mittels einfacher Wenn-Dann-Bedingungen die Regeln definiert werden (s. Abbildung 4). Dabei stehen verschiedene Operatoren zur Verfügung. Über Klammerungen und **AND- /OR**-Bedingungen können auch sehr komplexe Regeln abgebildet werden. Die erste Zeile liest sich bspw. wie folgt: Wenn der Wert der Spalte Content den regulären Ausdruck **.*html.*** enthält, schreibe in die Spalte Content Type den Wert **HTML**. Dies wird für alle weiteren Content-Arten definiert. Sollte keine der vorherigen Regeln zutreffen, wird über **TRUE => „Sonstige“** ein Fallback angegeben.

Das Resultat bzw. den Vorteil dieser Informationsverdichtung kann man sich direkt ansehen, wenn nun das Aufkommen der einzelnen Ausprägungen der Spalten Content bzw. der gerade neu hinzugefügten Spalte Content Type berechnet wird. Dazu wird der Knoten **GroupBy** verwendet, der basierend auf den Ausprägungen (Gruppen) einer Spalte A eine Funktion auf einer Spalte B (hier: zählen) ausführt. Die Konfiguration ist in Abbildung 5 zu sehen. Unter dem Reiter Groups wird die Spalte Content als zu gruppierende Spalte definiert. Unter dem Reiter Manual Aggregation werden die Spalte Address und die Aggregation-Methode count ausgewählt (weitere Aggregationsmethoden sind bei numerischen Werten bspw. Mean, Median, Min, Max je Gruppe). Zu Deutsch: Zähle innerhalb der Gruppen (HTML, Image etc.) in der Spalte Content die jeweils zugehörigen URLs aus der Spalte Address.

Das Gleiche wird mit einem weiteren **GroupBy**-Knoten für die neue Spalte Content Type durchgeführt. Schaut man sich die beiden Ergebnistabellen im Vergleich an, sieht man deutlich (s. Abbildung 6), dass die Verdichtung der Content-Ausprägungen wesentlich genauer die Frage beantwortet, welche und wie viele Ressourcen auf der Website vorkommen.

Darüber hinaus ist auf der Abbildung 6 ein weiterer Vorteil von KNIME ersichtlich. Der Datenfluss kann an jeder Stelle aufgeteilt werden. Ein Beispiel wäre, mittels **Row-Splitter**-Knoten die Eingangstabelle hinsichtlich der Status Codes aufzusplitten. Der **Row Splitter** ist vergleichbar zum **Row Filter** aus Abbildung 2, mit dem Unterschied, dass er zwei Output-Ports besitzt. Denkbar wäre also, alle **200 OK** URLs in den einen, alle restlichen URLs mit anderem Status Code in den anderen Output-Port zu leiten, um sie fortan getrennt voneinander weiterzuverarbeiten.

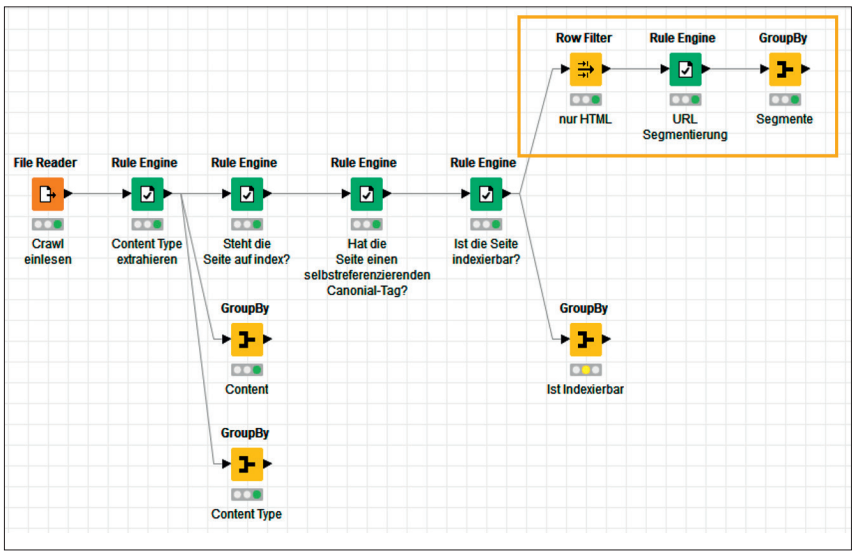


Abb.9: Segmentierung der HTML-URLs und anschließende Aggregation

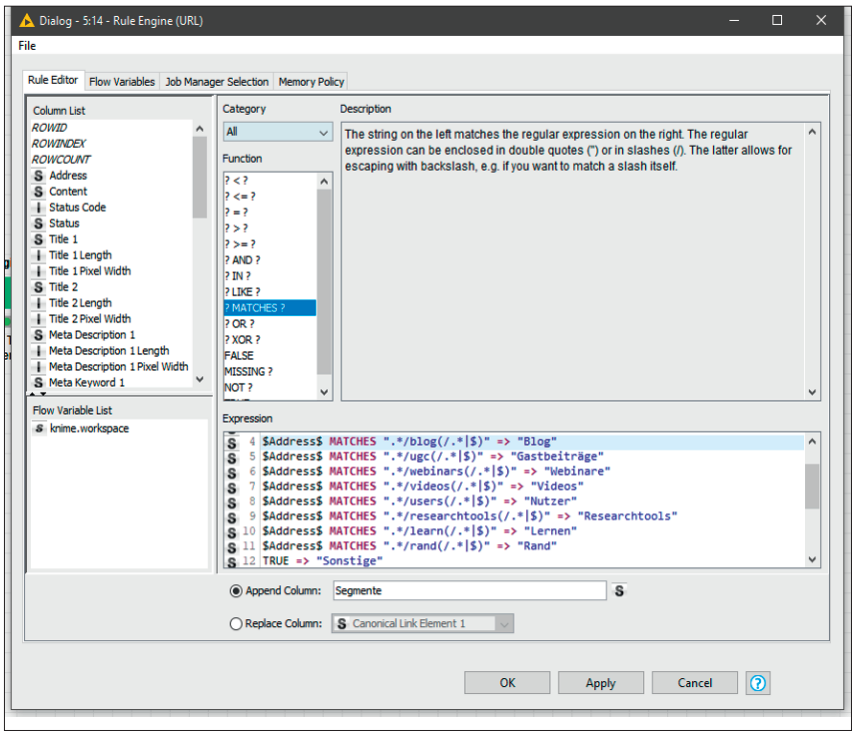


Abb.10: Segmentierung der URLs anhand der ersten Verzeichnisebene

Indexierbarkeit überprüfen

Eine weitere Information, die in den Crawl-Daten des Screaming Frogs nur implizit vorliegt, ist, ob eine Seite indexierbar ist oder nicht. Hier müssen die Informationen aus den Spalten Status Code, Meta Robots 1 und Canonical Link Element 1 extrahiert und zusammengeführt werden. Um dies zu ermitteln, werden drei **Rule Engines** benötigt (s. Abbildung 7).

Die erste **Rule Engine** kennzeichnet eine URL dann als nicht indexierbar, wenn in der Meta-Angabe noindex vorkommt. Andernfalls wird sie als indexierbar markiert. Das Regel-Set sieht wie folgt aus:

$Meta Robots 1$ MATCHES „.*(noindex|NOINDEX).*“ AND $Status Code$ = 200 AND $Content Type$ = „HTML“ => FALSE

$Status Code$ = 200 AND $Content Type$ = „HTML“ => TRUE

TRUE oder **FALSE** werden in die neue Spalte Meta Index geschrieben.

Die zweite **Rule Engine** überprüft analog dazu, ob eine URL keinen/einen selbstreferenzierenden oder einen fremdreferenzierenden Canonical-Tag aufweist:

$Status Code$ = 200 AND $Content Type$ = „HTML“ AND ($Address$ = $Canonical Link Element 1$ OR $Canonical Link Element 1$ = „“) => TRUE

$Status Code$ = 200 AND $Content Type$ = „HTML“ AND NOT $Address$ = $Canonical Link Element 1$ => FALSE

TRUE oder **FALSE** werden in die neue Spalte Canonical selbstreferenzierend geschrieben.

Zum Schluss werden in der letzten **Rule Engine** die Werte der beiden zuvor gebildeten Spalten zusammengenommen, um zu ermitteln, ob eine Seite indexierbar ist oder nicht.

$Meta Index$ = TRUE AND $Canonical selbstreferenzierend$ = TRUE => TRUE

($Meta Index$ = FALSE AND NOT MISSING $Meta Index$) OR ($Canonical selbstreferenzierend$ = FALSE AND NOT MISSING $Canonical selbstreferenzierend$) => FALSE

Anschließend kann erneut über einen **GroupBy**-Knoten nachgesehen werden, wie die Verteilung der Indexierbarkeit ist (s. Abbildung 8). Die ca. 27.000 NAs resultieren aus Weiterleitungen, 404er etc.

URLs segmentieren

Als letztes Beispiel soll hier die Segmentierung von URLs angeführt werden. Wie in der Abbildung 9 zu sehen, werden die URLs zunächst auf die HTML-Ressourcen gefiltert. Natürlich kann die Segmentierung auf allen URLs erfolgen, für das Beispiel soll aber nur ermittelt werden, wie viele HTML-URLs in den jeweiligen Segmenten liegen.

Die Segmentierung erfolgt wieder mittels **Rule Engine** und exemplarisch nur anhand des ersten Verzeichnisses (s. Abbildung 10). Grundsätzlich sind hier natürlich sehr komplexe Regel-Sets denkbar, um bspw. nicht nur nach Seitenbereichen (Blog, Shop etc.), sondern auch nach Seitentypen (Übersichtsseite, Produktdetailseite, Artikelseite etc.) zu differenzieren.

Im anschließenden **GroupBy**-Knoten kann nun das Ergebnis der Segmentierung eingesehen werden. Um ein Gefühl für die Mengengerüste der Website zu bekommen, können unter anderem die URLs je Segment, Status Code und Indexierbarkeit gezählt werden (s. Abbildung 11). Hier ist schnell ersichtlich, dass die Segmente Blog und Gastbeiträge nur zu ca. 50 % zur Indexierung freigegeben sind. Das Verzeichnis **/researchtools** weist eine hohe Anzahl an URLs auf, wird aber gänzlich ausgeschlossen.

Explorative Datenanalyse

Möchte man sich genauer ansehen, warum die Segmente Blog resp. Gastbeiträge zur Hälfte auf **noindex** stehen, kann die Tabelle mittels **Rule-based Row Filter** auf diese URLs eingegrenzt werden. Ähnlich wie bei der **Rule Engine** können in diesem Knoten Regeln definiert werden, die dann jedoch nicht klassifizieren, sondern nur darüber ent-

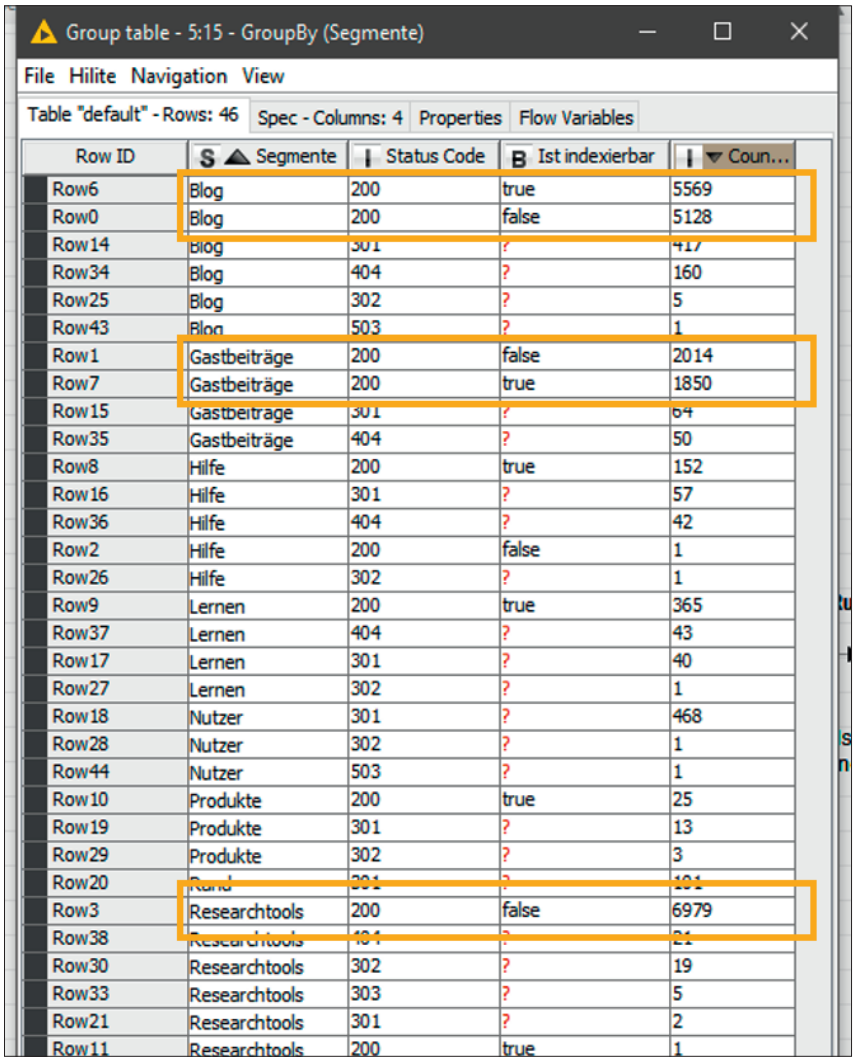
scheiden, ob eine Zeile in der Ausgabe enthalten ist oder nicht. Um die näher zu betrachtenden URLs zu ermitteln, wird folgende Regel definiert:

($Segmente$ = „Blog“ OR $Segmente$ = „Gastbeiträge“) AND $Status Code$ = 200 AND $Ist indexierbar$ = FALSE => TRUE

Beim Betrachten der Ausgabetabelle wird sehr schnell deutlich, dass die nicht indexierbaren URLs Kategorie- bzw. Paginationsseiten sowie Seiten mit reinen Statistiken des Whiteboard Fridays sind.

Diesen exemplarischen Drilldown in die Daten wird man im Rahmen einer Analyse immer wieder an den unterschiedlichsten Stellen durchführen. Das grundsätzliche Vorgehen ist, den Crawl um bestimmte Metriken (wie bspw. Content Type, Indexierbarkeit, Segment) zu erweitern und sich mittels GroupBy-Knoten die Aggregationstabellen anzusehen. Treten in diesen Tabellen Auffälligkeiten auf, wird mittels **Row Filter/Rule-based Row Filter** das jeweilige URL-Set in einem „Seitenarm“ des Workflows aus der Gesamttabelle herausgefiltert und näher betrachtet.

Diese Drilldowns lassen sich kaum standardisieren, denn wie der Name Explorative Datenanalyse schon anzeigt, ist im Vorhinein nicht klar, was man in den Daten findet. Die Arbeitsschritte zur Aufbereitung des Crawls, die der eigentlichen EDA vorgelagert sind, lassen sich aber sehr gut als standardisierter Prozess abbilden. Beim nächsten Crawl müssen dann nicht mehr alle Knoten einzeln konfiguriert werden. Stattdessen muss nur der Pfad im initialen **File Reader** auf den neuen Crawl angepasst werden. Die nachfolgenden Schritte bleiben weitestgehend gleich, die Knoten müssen nur noch ausgeführt werden – ausgenommen ist hier natürlich die Segmentierung, die immer individuell anhand der URL-Struktur erfolgen muss.



Row ID	S Segmente	Status Code	B Ist indexierbar	Coun...
Row6	Blog	200	true	5569
Row0	Blog	200	false	5128
Row14	Blog	301	?	417
Row34	Blog	404	?	160
Row25	Blog	302	?	5
Row43	Blog	503	?	1
Row1	Gastbeiträge	200	false	2014
Row7	Gastbeiträge	200	true	1850
Row15	Gastbeiträge	301	?	64
Row35	Gastbeiträge	404	?	50
Row8	Hilfe	200	true	152
Row16	Hilfe	301	?	57
Row36	Hilfe	404	?	42
Row2	Hilfe	200	false	1
Row26	Hilfe	302	?	1
Row9	Lernen	200	true	365
Row37	Lernen	404	?	43
Row17	Lernen	301	?	40
Row27	Lernen	302	?	1
Row18	Nutzer	301	?	468
Row28	Nutzer	302	?	1
Row44	Nutzer	503	?	1
Row10	Produkte	200	true	25
Row19	Produkte	301	?	13
Row29	Produkte	302	?	3
Row20	Rand	301	?	101
Row3	Researchtools	200	false	6979
Row38	Researchtools	404	?	21
Row30	Researchtools	302	?	19
Row33	Researchtools	303	?	5
Row21	Researchtools	301	?	2
Row11	Researchtools	200	true	1

Abb.11: Aggregation der segmentierten URLs nach Status Code und Indexierbarkeit

Fazit

Die obigen Beispiele sind nur ein kleiner Ausschnitt der Arbeitsschritte, die bei einer Crawlaufbereitung und -analyse durchgeführt werden können. An ihnen werden aber bereits sehr gut die Vorteile von KNIME ersichtlich. Insbesondere wiederkehrende Prozesse lassen sich mit KNIME einfach und individuell abbilden, dokumentieren und reproduzieren. In Kombination mit Hunderten von Knoten können somit fast alle Aufgaben der Datenverarbeitung durchgeführt werden. Daten können aus Dateien importiert, via API geladen, aus XML-Sitemaps extrahiert oder direkt aus dem Web gescrapt werden. Geladene Daten können mit anderen Datenquellen bspw. aus der GSC oder GA (für Letzteres gibt es bereits einen eigenen Knoten zur Abfrage der API) mittels Joins verknüpft und anhand individueller Regel-Sets klassifiziert werden. Liegt das Daten-Set in der gewünschten Form vor, steht eine Reihe von Knoten zur Verfügung, um die Daten zu filtern und zu aggregieren. Schließlich können die Ergebnisse mittels Grafiken visualisiert und kommuniziert werden.

KNIME bietet damit einen guten Mittelweg für all jene, deren Ansprüche über Excel hinausgewachsen sind, die aber nicht die Zeit oder den Bedarf haben, eine Programmiersprache wie R oder Python zu lernen. ¶