Von null bis unendlich

Starten wir mit ein paar einfacheren Metazeichen, den simplen Platzhaltern, Positionsmarkern und Wiederholungsoperatoren.

Nehmen wir an, wir würden folgender (nicht ganz schlüssiger) Auflistung ausschließlich die Pronomen der zweiten Person Singular entnehmen wollen:

d
ich
ihr
dich
mir
ihn
er
uns
wir
sie
euch
du
Ihnen
Sie
dir
sie
ihm
andere
dorthin

Was hätten die gewünschten Suchergebnisse (*du*, *dir*, *dich*) gemeinsam? Zweifelsfrei das *d* am Anfang des Wortes. Die Gemeinsamkeiten hören bereits hier auf, denn darüber hinaus existieren keine geteilten Buchstabenfolgen und auch die Länge der einzelnen Pronomen ist unterschiedlich. Kennen wir das genaue Zeichen, welches auf das *d* folgt, nicht, so können wir es durch einen Platzhalter ersetzen, am einfachsten durch einen Punkt (.).
Der reguläre Ausdruck *d.* findet alle Vorkommen von *d* gefolgt von einem beliebigen (konkreten) Zeichen, also (v.o.n.u.) *di*, *du*, *di*, *de* und *do*.
d.. sucht entsprechend nach einem *d*, das von zwei beliebigen Zeichen gefolgt wird, d. h. *dic*, *dir*, *der* und *dor*. *du* gehört nicht zu den Suchergebnissen von *d..*, da der Ausdruck eine dreistellige Zeichenkette beschreibt, *du* hingegen aus nur zwei Zeichen besteht.
d... führt entsprechend zu den Ergebnissen *dich*, *dere* und dort. *d...* schließt neben *du* auch *dir* aus den Ergebnissen aus, da der Ausdruck drei beliebige Zeichen nach *d* fordert und sowohl *du* als auch *dir* somit zu kurz sind.
Wenn das nun alles wäre, dann würden uns reguläre Ausdrücke keinen Vorteil verschaffen, da *d.*, *d..* und *d...* die gleiche Anzahl an Suchanfragen erfordern wie die Suche nach jedem einzelnen der Pronomen. Sie liefern uns zudem auch die nicht brauchbaren Suchergebnisse *dort* und *dere* sowie unvollständige Wörter wie *di* (für *dich* und *dir*) In der Tat wissen wir manchmal nicht, wie viele Zeichen ein Wort umfasst (man sei an das zweite Wort aus dem Beispiel *Alle Auen sind grün.* im Vergleich zu *Auch Autobahnen sind grau.* erinnert). Wir können aber auch genau das durch einen regulären Ausdruck ausdrücken, indem wir nach *d.** suchen. Das Sternchen greift das letzte Zeichen des Ausdrucks (sei es nun konkret oder abstrakt, in diesem Fall der Platzhalter ".") auf und bewirkt, dass es undefiniert oft gesucht wird, d. h. 0 mal, 1 mal, 2 mal, ..., 1000 mal, ... Das Ergebnis für diese Suche würde lauten: *d* (*d*), *dich* (*d...*), *du* (*d.*), *dir* (*d..*), *dere* (*d...*), *dorthin* (*d.....*). Um das d aus den Ergebnis auszuschließen, ließe sich alternativ *d.+* verwenden, wobei das + das letzte Zeichen (konkret oder abstrakt) aufgreift und es 1 mal und häufiger sucht (Ergebnisse: *dich* (*d...*), *du* (*d.*), *dir* (*d..*), *dere* (*d...*), *dorthin* (*d.....*)). Der Vollständigkeit halber sei hier auch auf den Operator *?* verwiesen, welcher anweist, das vorausgehende Zeichen höchstens ein einziges Mal zu finden (Ergebnisse: *d* (*d*) und *du* (*d.*)).

Der Ausdruck *d.+* liefert bereits gute Ergebnisse, allerdings nach wie vor zu viele, denn das definierte Muster passt auch auf (an)*dere* und *dorthin*. Wie lässt sich das vermeiden? *dorthin* lässt sich mithilfe eines weiteren Wiederholungsoperators ausschließen, denn vergleicht man es mit den anderen Ergebnissen von *d.+*, stellt man fest, dass es durch seine abweichende Länge auffällt. Neben den bereits genannten Operatoren lässt sich nämlich mittels geschweifter Klammern festlegen, wie häufig das vorausgehende Zeichen minimal und maximal auftreten darf. So beschreibt In *{1,3}* die erste Zahl (*1*) die minimale, die zweite Zahl die maximale Anzahl (*3*) des vorausgehenden Zeichens. Ersetzt man den Ausdruck *d.+* durch *d.{1,3}*, wird somit ausschließlich nach *d.*, *d..* und *d...* gesucht. Die Ergebnisse lauten folglich: *dich*, *du*, *dir* und *dere*.

Quantitativer Operator	Bedeutung
*	das vorausgehende Zeichen kommt beliebig oft vor (0 bis unendlich)
?	das vorausgehende Zeichen kommt 0 oder 1 mal vor
+	das vorausgehende Zeichen kommt 1 mal oder häufiger vor
{3}	das vorausgehende Zeichen kommt genau 3 mal vor
{1,3}	das vorausgehende Zeichen kommt 1, 2 oder 3 mal vor

Das optimale Ergebnis wird nun nur noch durch *dere* gestört. Es hilft, den Blick nicht nur auf Gemeinsamkeiten, sondern auch auf Unterschiede zu lenken. Was haben die "richtigen" Ergebnisse in Abgrenzung zum "falschen" gemeinsam? Die Position! Sowohl in *dich* als auch in *du* und in *dir* befindet sich das *d* am Zeilenanfang, bei *andere* hingegen in der Wortmitte. Auch für die Positionierung eines Zeichens am Zeilenanfang stellt das Regex-Inventar ein Metazeichen bereit: das ^. *^d.{1,3}* sucht dasselbe wie *d.{1,3}*, allerdings am Anfang einer Zeile. Dieser Ausdruck liefert uns die gewünschten Ergebnisse, nämlich *dich* (*^d...*), *du* (*^d.*) und *dir* (*^d..*).

Positionsanker	Bedeutung
^	am Anfang einer Zeile
$	am Ende einer Zeile
\<	am Anfang eines Wortes
\>	am Ende eines Wortes

Greedy vs. lazy

Gierig und faul, andernorts möglicherweise durchaus vereinbar, im Rahmen der Regex schließen die beiden Eigenschaften einander jedoch aus. "gierig" und "faul" beschreiben das Verhalten der bereits behandelten Wiederholungsoperatoren. Um die beiden Begriffe zu erklären, können wir erneut auf das Listenbeispiel dieses Kapitels zurückgreifen. Diesmal befinden sich die Wörter jedoch in einer einzigen Zeile:

```
d ich ihr dich mir ihn er uns wir sie euch du Ihnen Sie dir sie ihm andere dorthin
```

Suchen wir hier nach *d.** oder nach *d.+* wird die gesamte Zeile markiert, da . auch für Leerstellen stehen kann. Warum sucht man dann nicht einfach nach *d .** oder *d.+* gefolgt von einer Leerstelle, also (Achtung: ein grauer Underscore wird in diesem Manual aus Gründen der Sichtbarkeit dazu verwendet, um Leerstellen in einem regulären Ausdruck zu markieren. Ein tatsächlicher Underscore im Ausdruck bleibt schwarz!) *d.*_* oder *d.+_* ? Die Idee dahinter wäre mit "Suche nach *d* gefolgt von einer beliebigen Anzahl von Zeichen (je nach *** oder *+*: mindestens 0 oder 1), aber stelle sicher, dass auf diese eine Leerstelle folgt!" zu umschreiben. Dies funktioniert sogar tatsächlich, allerdings anders als erwartet. Der Grund dafür liegt darin, dass reguläre Ausdrücke in ihrem Wesen gierig sind. Man setze einen regulären Ausdruck an das linke Ende der "d ich ihr dich mir..."-Tafel und sage ihm, er dürfe nun alles verspeisen, dass unter *d.*_* fällt. Im nächsten Augenblick würde er sich gierig bis an das rechte Ende von *dorthin* strecken, doch (kurz bevor er seine Gabel versenken kann) verdutzt feststellen, dass auf *dorthin* keine Leerstelle folgt. Also würde er sich leicht unzufrieden in Richtung seines Sitzplatzes zurückbewegen, bis er die Leerstelle im Anschluss an *andere* findet. Das gilt nicht nur für die die Wiederholungsoperatoren *+* und ***, sondern auch für *?* und *{min,max}*. Steht auf dem Diätplan des regulären Ausdrucks *d.*, so wird er - gierig, wie er ist - *d.* nach Möglichkeit dem mageren *d* bevorzugen.
Ob er für das alles nicht doch zu faul ist, fragt sich der Ausdruck eigentlich nie - es sei denn, man zwingt ihn dazu. Und wie ließe sich der Zweifel an der Größe der eigenen Gier besser ausdrücken als durch ein *??* Fügt man an die bereits besprochenen Ausdrücke d.*, d.+, d.? und d. {1,3} ein Fragezeichen an, so zwingt man den regulären Ausdruck zur laziness. Die Wiederholung des vorausgehenden Zeichens wird wie durch den Wiederholungsoperator vorgeschrieben durchgeführt, allerdings wird sie nur so häufig durchgeführt wie nötig.

Quantitativer Operator	Bedeutung
*?	das vorausgehende Zeichen kommt beliebig oft vor (0 bis unendlich) aber: wähle die geringstmögliche Anzahl
??	das vorausgehende Zeichen kommt 0 oder 1 mal vor aber: lieber 0 als 1 mal
+?	das vorausgehende Zeichen kommt 1 mal oder häufiger vor aber: wähle die geringstmögliche Anzahl

{1,3}?	das vorausgehende Zeichen kommt 1, 2 oder 3 mal vor aber: wähle die geringstmögliche Anzahl

Während also d.*_"d ich ihr dich mir ihn er uns wir sie euch du Ihnen Sie dir sie ihm andere " umfasst, liefert *d.*?_* gleich mehrere (kleinstmögliche) Ergebnisse "d ", "dich ", "du ", "dir " und "dere " (*dorthin* nicht, da es von keiner Leerstelle gefolgt wird). Stellen wir die Verfeinerung der Ergebnisse zunächst zurück und betonen an dieser Stelle stattdessen, dass laziness nicht bedeutet, dass nur Ergebnisse ausgegeben werden, die dem kürzesten Muster entsprechen. Es weist lediglich an, bei mehreren Möglichkeiten den kurzmöglichsten Ausschnitt zu wählen. *d. {1,3}?_* zieht *du dir* nicht vor, jedoch *d* dem längeren *du l(hnen)*.

Was soll ich damit?

Die Frage aller Fragen: Was soll ich damit eigentlich anfangen können? Ich kann zum Beispiel Häufigkeiten für bestimmte Phänomene erfassen. Will ich in einem Korpus des Französischen beispielsweise in Erfahrung bringen, wie häufig das Verb *finir* in all seinen Formen vorkommt, so kann ich dies sehr zeitsparend mit l<fini.{0,6}\>_ (*j'ai fini* bis *nous finissions*) oder einfach mit l<fini.*?\> erledigen (Eine anschließende Bereinigung der Ergebnisse kann dennoch nötig sein!). Ich kann auch nach flektierten Formen des Italienischen suchen, ohne jede Form einzeln eintippen zu müssen, z.B. *<lsan.l>* für *sano*, *sana*, *sani* und *sane*. oder zwei variierende Formen wie *comprare* und *comperare* durch *compe?rare* erfassen. Auch kann ich mir anzeigen lassen, wie viel von dem sprachlichen Kontext des Gesuchten ich in meine Ergebnisse integrieren will. Interessiert mich der Satzkontext von *compe?rare*, kann ich mir durch *.{50}compe?rare_.{50}* jeweils 50 Zeichen vor und nach *compe?rare_* anzeigen lassen.

Um das richtige Muster zu definieren, sollte ich mir Gedanken darüber machen:

- was meine Ergebnisse gemeinsam haben müssen.
- ob die Gemeinsamkeiten konkrete Buchstaben(folgen), Positionen oder eine Mischung aus beidem betreffen.
- welche Bausteine der gewünschten Ergebnisse variabel sind.
- wie viele variabel besetzte stellen darf es (minimal, maximal) zwischen konkret definierten Zeichen oder Positionsankern geben?
- welche Grenzen ich für gierige Operatoren setze (laziness, Begrenzung durch Leerstelle, etc.).
- was ich von der Umgebung des Gesuchten in meine Ergebnisse integrieren will.

Previous Reg-was? Up Reguläre Ausdrücke in der Arbeit mit Textdaten Platzhalter reloaded Next