Compendio di regular expressions

Linguistica e Filologia Digitale (Università di Cagliari) – a.a. 2019/2020

Simone Ciccolone

Breve compendio sulle regular expressions, le "formule" per la creazione di maschere di ricerca (queries) avanzate.

1. Le regular expressions

Le **regular expressions**, o espressioni regolari, sono stringhe di ricerca variabile, ovvero sequenze di caratteri con funzioni particolari che permettono di cercare insiemi di stringhe sulla base di una serie di criteri.

Una regular expression è una **funzione** che descrive uno **schema astratto** a cui possono corrispondere più stringhe. Anche una regular expression è una stringa, per cui occorre fare attenzione a ricordarsi che alcuni caratteri hanno funzioni specifiche nelle regular expressions, mentre altri sono presi letteralmente.

Ad es., ipotizziamo di voler cercare tutte le parole che iniziano per "mang" (potenzialmente, per cercare tutte le forme del verbo "mangiare", ma con possibili "falsi positivi", come "mango"). Possiamo usare la seguente regular expression:

mang.+

Quando eseguiamo una **query** tramite una regular expression, il software interpreta la stringa della nostra espressione secondo una sintassi stabilita: i caratteri alfabetici vengono interpretati letteralmente, per cui il motore di ricerca del corpus cercherà nei suoi testi tutte le concatenazioni di "m", "a", "n" e "g" (a seconda dello strumento usato, possiamo anche filtrare i risultati in base all'esatta corrispondenza di maiuscole e minuscole). Dopo questi caratteri interpretati letteralmente, ci sono due simboli che, nella sintassi delle regular expressions, hanno una funzione particolare:

. (punto): sostituisce un qualsiasi carattere (alfabetico, numero, punteggiatura etc.)
+ (più): è un quantificatore (cfr. cap. 3 di questo compendio); richiede che il carattere immediatamente precedente sia presente una o più volte

Per cui, la sequenza di simboli ".+", nelle regular expressions, significa: "cerca una sequenza qualsiasi di caratteri, di lunghezza qualsiasi da uno a infinito". La regular expression completa "mang.+" permetterà quindi di cercare tutte le stringhe che iniziano per "mang" e sono seguite da almeno un carattere (qualsiasi).

Le regular expressions sono utilizzate in informatica in vari modi. Il text mining (ovvero l'esplorazione quantitativa di dati in formato testo) si basa sull'uso delle regular expressions per molte operazioni di estrazione di informazioni (information extraction).

2. Classi di caratteri (base)

- . qualsiasi carattere
- [a-z] lettere minuscole
- [A-Z] lettere maiuscole
- [0-9] numeri
- [^a] tutti i caratteri tranne "a"

Il **simbolo . (punto)** ha la funzione di sostituire un qualsiasi carattere. Nella sintassi delle regular expressions, simboli come questo permettono di individuare una **classe di caratteri**.

Per **classe di caratteri** si intende un possibile raggruppamento di simboli di cui può essere composta una stringa; ad es. caratteri alfabetici latini, caratteri numerici, punteggiatura, caratteri dell'alfabeto arabo etc.

Le **parentesi quadre** permettono di creare raggruppamenti ad hoc: all'interno delle parentesi quadre possiamo inserire tutti i possibili caratteri, tra le possibilità di nostro interesse, che possono comparire in una determinata posizione. Ad es.:

[IVX]

permette di rintracciare tutte le occorrenze dei caratteri "I", "V" e "X". Può essere usato per selezionare più forme che variano per uno o più caratteri. Ad es.:

(regular expression) > (possibili risultati)

c[ao]sa	>	casa, cosa
gatt[aoei]	>	gatta, gatto, gatte, gatti
m[ei]s[ei]	>	mese, mise, mesi, misi

All'interno delle parentesi quadre, l'**accento circonflesso (^)** indica negazione: la regular expression [^a] significa "qualsiasi carattere, tranne a".

N.B.: Fuori dalle parentesi quadre, l'accento circonflesso ha un'altra funzione! Cfr. cap. 5 di questo compendio.

3. Quantificatori

- ? 0 o 1
- + 1 o più
- * 0 o più
- {n} esattamente n volte
- {n,} n o più volte
- {n,m} tra n e m volte

I **quantificatori** sono simboli che, nella sintassi delle regular expressions, servono a indicare la quantità di caratteri uguali al precedente richiesti nella query. Ogni quantificatore opera sempre in combinazione col carattere precedente. Alcuni esempi:

- .+ qualsiasi sequenza di 1 o più caratteri
- a+ qualsiasi sequenza di 1 o più "a"
- cas+a > casa, cassa, casssa...

I **quantificatori** indicati sono di tipo **"greedy"**: selezionano la sequenza più lunga possibile. Ad es, se ho il testo:

ABCABCABC

la regular expression "A.+C" troverà una sola occorrenza: "ABCABCABC".

I **quantificatori "lazy"**, invece, selezionano la sequenza più corta possibile corrispondente alla regular expression.

La regular expression "A.+?C" (con quantificatore "lazy": "+?") troverà tre occorrenze (le tre sequenze "ABC").

4. Classi di caratteri (avanzato)

- \X grafema completo (con diacritici, diversamente da .)
- \d qualsiasi numero (anche non 0-9)
- \s spazio
- \n riga nuova (a capo)
- \p{} qualsiasi carattere con specifica proprietà Unicode (es. "Uppercase")
- \w word character (lettera, numero, marcatori diacritici)
- \D, \W complementari di \d, \w (ad es. \W: qualsiasi carattere che non sia lettera o numero)

Qui sono indicati **classificatori** di caratteri predefiniti: "\d" individua una classe di caratteri corrispondenti a cifre di qualsiasi sistema numerico (quindi non solo le cifre da 0 a 9, ma anche simboli numerici usati in altre lingue); il classificatore "\X" permette di selezionare il grafema completo, compreso di eventuali segni diacritici "aggiunti" (si veda la codifica Unicode dei diacritici per maggiori informazioni).

Particolarmente utili possono essere i classificatori **"\s"**, per selezionare lo **spazio**, e **"\n"**, per selezionare l'**a capo** (e comporre quindi queries che includono più paragrafi, che altrimenti rispondono sempre individualmente a queries con regular expressions).

A ognuno di questi classificatori corrisponde un **classificatore complementare**, rappresentato dalla versione maiuscola: "\S" individua qualsiasi carattere che non sia uno spazio; "\W" qualsiasi carattere non alfabetico, numero o marcatore diacritico.

5. Delimitatori

- \b confine di parola
- \n a capo (nuova riga)
- ^ inizio stringa
- $ fine stringa

Queste sequenze di simboli permettono di individuare non dei caratteri visibili (realmente digitati), ma dei confini tra unità (tokens) rintracciati automaticamente da sistemi di analisi testuale (e dalla sintassi delle regular expressions).

Il **delimitatore "\b"** individua tutti i **"confini" di "parola"** (con "parola", si intende qui una sequenza continua di caratteri della classe \w, ovvero caratteri alfabetici, numerici e diacritici), ovvero spazi, a capo e caratteri non "di parola", come la punteggiatura.

Se ad es. facciamo una "Trova e sostituisci" del delimitatore "\b" con un a capo ("\n"), trasformiamo il testo in una serie di righe composte da singole parole o (sequenze di) caratteri di punteggiatura. In pratica, facciamo un'operazione di **tokenizzazione**, mantenendo come tokens separati anche la punteggiatura.

6. Operatori AND/OR

Ci sono inoltre due simboli speciali che funzionano come **operatori logici** (cfr. algebra booleana).

- a|b a oppure b
- (ab) raggruppamento (a E b insieme)
- gr(e|a)y "grey", "gray"

La barra verticale funziona come **operatore OR**: viene richiesto di cercare stringhe corrispondenti alla sequenza di simboli che precede O alla sequenza di simboli che segue l'operatore.

Le parentesi tonde sono l'**operatore AND**: creano un raggruppamento di caratteri e simboli in sequenza, compresenti.

Nell'esempio riportato sopra, le parentesi tonde delimitano il raggio d'azione dell'operatore OR ai caratteri compresi nel raggruppamento, da considerare quindi alternativi.

7. Carattere di Escape

Abbiamo visto quindi come le regular expressions siano stringhe con una serie di simboli "speciali" che permettono di selezionare insiemi di stringhe sulla base di **criteri di corrispondenza**.

Alcuni caratteri sono quindi interpretati letteralmente (caratteri alfabetici e numerici), **altri invece hanno funzioni e valori specifici** (di classificatori, delimitatori, operatori etc.).

Ma se dobbiamo cercare proprio uno dei caratteri con funzioni speciali, come il punto, le parentesi o l'asterisco?

In questi casi, abbiamo bisogno di utilizzare un altro carattere con una funzione speciale: il **carattere di escape**. Questo simbolo speciale indica che il carattere immediatamente successivo è da interpretare letteralmente, e non come simbolo di una regular expression. Ad es., per cercare i punti interrogativi in un testo, visto che "?" è un simbolo speciale (un quantificatore), dobbiamo farlo precedere dal carattere di escape "\". Per cercare i punti interrogativi in un testo, devo quindi usare la seguente regular expression:

```
\?
```