Propozycja rozszerzenia składni zapytań programu Poliqarp o elementy statystyczne

Aleksander Buczyński

2006.06.26

Poliqarp - stan obecny

Zwracane są kolejne konteksty wystąpień ciągów segmentów pasujących do wzorca zadanego w zapytaniu.
Taka forma wyników pozwala na znajdowanie przykładów użycia konkretnej konstrukcji, ale można sobie wyobrazić szereg problemów korpusowych, dla których przeglądanie setek kolejnych kontekstów może być mało wygodne, np.

- Jakie części mowy mogą występować bezpośrednio po „w”?
- Jakie czasowniki są wykorzystywane najczęściej w tekstach ustaw?
- Jaki jest rozkład częstości form danego leksemu?

Propozycja

Do zapytania dodajemy opcjonalny element *GROUP BY*, informujący o tym, że nie interesują nas konteksty poszczególnych wystąpień, ale częstość określonych zjawisk w rezultatach oryginalnego zapytania (np. form danego rzeczownika, albo czasowników występujących po danym wyrazie).

Przykłady prostych zapytań

Zapytanie o częstość występowania poszczególnych form leksemu woda:

```
[base=woda] group by orth
```

Jw., ale z rozbiciem na liczby i przypadki:

```
[base=woda] group by number, case
```

Jw., ale tak, by w tabelce wyników pojawiła się również forma odpowiadająca danej liczbie i przypadkowi:

```
[base=woda] group by number, case, orth
```

Zapytania obejmujące kilka segmentów

Częstość występowania poszczególnych czasowników w formach finitywnych po wyrazie *woda*:

```
[base=woda][pos=fin] group by 2.base
```

Jw., ale z dopuszczeniem przysłówka pomiędzy wodą a czasownikiem:

```
[base=woda][pos=adv]{0,1}[pos=fin] group by -1.base
```

-1. oznacza tutaj pierwszy segment od końca wyniku. Analogicznie -2. oznaczałby drugi segment od końca, -3. – trzeci itd.

Zapytania obejmujące kilka segmentów (2)

Częstości występowania trójek przysłówków obok siebie:

```
[pos=adv]{3} group by 1.base, 2.base, 3.base
```

Lub:

```
[pos=adv]{3} group by base, 2.base, 3.base
```

Sortowanie wyników

sort by freq	– według częstości wystąpień
sort a fronte	– alfabetycznie
sort a tergo	(w planach) – alfabetycznie od końca
sort ???	(w planach) – w naturalnym porządku tagsetu (np. kolejne przypadki, rodzaje, liczby niekoniecznie alfabetycznie)

Selekcja wyników

min n – tylko wyniki powtarzające się co najmniej n razy

Wieloznaczności

Domyślnie wybierana jest losowa interpretacja spośród ujednoznacznionych (lub wszystkich – w zależności od konfiguracji poliqarpa).
Alternatywnie:

```
[base~mieć] group by base interp combine
```

Inny pomysł – *interp all* – traktowanie wieloznaczności jak wszystkie interpretacje (wyniki mogłyby sumować się wtedy do więcej niż 100%).

Kolokacje

Za pomocą wyżej opisanej składni da się wyrazić proste zapytania o kolokacje, np.

```
[][] group by base, 2.base sort by freq
```

Ważna jest także częstość wystąpień składowych kolokacji:

```
[][] group by base; 2.base sort by freq
```

Symetryczne prawdopodobieństwo warunkowe:

```
[][] group by base; 2.base sort by scp
```

Ze skrzywieniem:

```
[][] group by base; 2.base sort by scp bias 0.5
```

Lub tradycyjnie:

```
[][] group by base; 2.base sort by scp min 2
```

Kolokacje (2)

Funkcje kolokacji mogą korzystać z:

- c – liczby wystąpień
- c1 – liczby wystąpień pierwszego składnika (wyznaczonego przez część grupowania do średnika)
- c2 – liczby wystąpień drugiego składnika

Na próbę zaimplementowano:

- CP – prawdopodobieństwo warunkowe
- SCP – symetryczne prawdopodobieństwo warunkowe
- MAXCP – maksymalne prawdopodobieństwo warunkowe
- DICE – test Dice'a

Prototyp

Na razie zaimplementowana została prosta nakładka na klienta graficznego wyrzucająca wyniki statystyczne w konsoli. Potrafi grupować wg *orth*, *base* i *pos*.

Format wyników

Wynikiem zapytania z grupowaniem prostym jest tabela o $k + 1$ kolumnach i p wierszach, gdzie:
k – liczba parametrów po *group by*
$p \leqslant v_1 v_2 ... v_k$ (v_i - liczba możliwych wartości i-tego parametru)
Pierwsze k kolumn zawiera odpowiednie wartości kategorii;
Ostatnia – ilość wystąpień o parametrach określonych we wcześniejszych kolumnach.
Zapewne przydałaby się także częstość względną – ilość dotychczas znalezionych wystąpień podzielona przez liczbę dotychczas przeszukanych segmentów.
Dla kolokacji dochodzą dodatkowe 2-3 kolumny (częstość wystąpień składników + ew. obliczona miara).

Technikalia

- Raz obliczone wyniki zapytania mogą być grupowane wielokrotnie wg różnych kryteriów.
- Domyślnie statystyka tworzona jest na podstawie próbki 1000 (prawie) losowo wybranych trafień, można zmienić dodając np. *count 10000* albo *count all*.
- Domyślnie pokazywanych jest max. 20 wyników, można zmienić dodając np. *display 100* albo *display all*.
- Jest problem z niezamawianą transmisją kontekstów, jak się go rozwiąże, to program powinien nieco przyśpieszyć.
- Po stronie serwera ograniczenie do 500000 trafień.

Czego składnia nie obejmie?

```
[warunek1][]{,m}[warunek2][]{,n}[warunek3]
group by warunek1, warunek2
(lub warunek2, warunek 3)
```

Czy tego typu zapytania byłyby popularne?

Możliwe modyfikacje składni

Marcin Woliński – numerować części zapytania, nie tylko segmenty wyniku, np.

```
[warunek1][]{,m}[warunek2][]{,n}[warunek3]
group by 1.1.base, 3.1.base
```

Łukasz Dębowski – dodać możliwość etykietowania fragmentów zapytania, np.

```
([warunek1])/A []{,m} ([warunek2])/B []{,n} [warunek3]
group by A.1.base, B.1.base
```

Problem: obie modyfikacje wymagają wprowadzenia zmian na wszystkich poziomach Poliqarpa, tak by przy wyszukiwaniu i przekazywaniu zachowywać informację, który fragment zapytania odpowiada któremu fragmentowi wyniku.

Grupowanie po metadanych

Częstość używania słowa „woda” przez poszczególnych autorów:

```
woda group by meta.author
```

Porównanie częstości dopełniacza i biernika w poszczególnych stylach:

```
[case="gen|acc"] group by meta.superCat, case
```

Grupowanie po metadanych – uwagi

Przydałoby się...

1. „Przetasować” korpus przed zindeksowaniem.
2. Podając częstość względną porównywać liczbę dotychczas znalezionych wyników nie z całkowitą ilością przejrzanych segmentów, ale z liczbą przejrzanych segmentów pasującą do określonych metadanych.
3. Dopuścić w protokole możliwość przesyłania metadanych od razu z segmentami, bez dodatkowego odpytywania.
4. Stworzyć nową konstrukcję dla przedziałów czasowych, np:

   ```
   ustawa group by meta.published(,1950,1990,2000,)
   ```

   albo dodać aliasy podziału chronologicznego, np.

   ```
   ustawa group by meta.published.century
   ```

Efektywność zapytań

Z pierwszych eksperymentów z profilowaniem wynika, że większość czasu zajmuje sprawdzenie jaki ciąg segmentów odpowiada poszczególnym trafieniom, operacje wykonywane przez sam moduł statystyczny liczą się szybko.
Obecne rozwiązanie:

- jest dobre dla zapytań „specjalistycznych” (z relatywnie niedużą ilością trafień);
- kiepsko się nadaje do zapytań „ogólnych” (z dużą ilością trafień);
- byłoby dobre dla szybkiego znajdowania przybliżonych wyników zapytań ogólnych pod warunkiem „przetasowania” korpusu.

Zapytania ogólne - co z tym można zrobić?

- Nowe indeksy (ew. zupełnie nowy interfejs) dla zapytań o [], [][] itp.
- Jednorazowo obliczyć wyniki dla najbardziej typowych / sensownych zapytań i udostępniać je niezależnie od wyszukiwarki, np.:
  - [] group by base sort by freq
  - [] group by base, meta.superCat sort a fronte min 20
  - [][] group by base; 2.base sort by scp bias 0.5
- Przenieść moduł statystyczny na stronę serwera (wcześniej: sprawdzić czy to ma sens).
- Wbudować statystyki w konstrukcje automatu rozpoznającego wyrażenie regularne (ryzyko wywołania nowej fali błędów).
- Prosić użytkowników o przysyłanie logów z zapytaniami, by stwierdzić, na jakie optymalizacje jest tak naprawdę zapotrzebowanie.

Luźne pomysły na przyszłość

- dodatkowe (pseudo)atrybuty – np. długość wyniku w segmentach, długość segmentu w literach
- dwuwymiarowa wizualizacja wyników
- Document Frequency / RIDF
- średnia, mediana itp. wyników