Hàm tính trọng số từ khóa:

Weight(Class, Keyword, Type)

+ Class: Lớp đối tượng

+ Keyword: Từ khóa

+ Type: Độ đo (TFIDF, Chi-Square, BNS, OR,…)

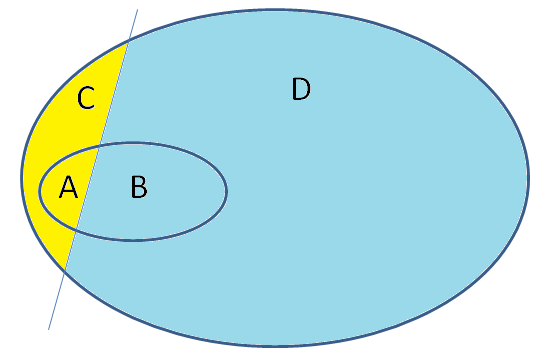
Lưu ý: Trong các độ đo trên thì chỉ có độ đo TFIDF tính tần suất xuất hiện của từ khóa trong nội dung mô tả class (class description). Các phương pháp còn lại được tính toán dựa trên nội dung mô tả function (function description).

Để tính hàm đánh trọng số này ta cần thêm 1 số thông tin sau:

+ Gọi N = A+B+C+D là toàn bộ số functions có trong cơ sở dữ liệu.

+ Giả sử c là 1 lớp đối tượng bất kỳ trong cơ sở dữ liệu, c có chứa N1 = A+B functions

+ Cho k là 1 keywords bất kỳ trong cơ sở dữ liệu, giả định có N2 = A+C functions trong cơ sở dữ liệu có chứa từ khóa k.



Bây giờ ta sẽ tính toán A,B,C,D dựa trên N,N1,N2. Dễ thấy, mổi độ đo được tính toán dựa trên 4 hệ số A,B,C,D. Mổi cặp (c, k) sẽ có 1 bộ hệ số (A(c,k), B(c,k), C(c,k), D(c,k)) khác nhau. Do đó công việc của ta là phải tính toán cho được 4 hệ số này ứng với mổi cặp (class c, keyword k) tương ứng.

Anh đề xuất 2 hướng tiếp cận:

1. **Tính trọng số cho toàn bộ từ khóa ngay khi index** (Đánh trọng số cho toàn bộ dữ liệu ngay khi bắt đầu chương trình):

Trong phương pháp này: Ta phải quét qua toàn bộ các từ khóa đã được đánh chỉ mục (bẳng lucene) cũng như toàn bộ các lớp đối tượng có trong cơ sở dữ liệu, ứng với mổi cặp (class, keyword) ta sẽ tính trọng số tương ứng với từng phương pháp (OR, TFIDF, BSN, …).

*Foreach (k : lucene.ListAllKeywordsInIndex())*

*Foreach (c : database.ListAllClassesInDatabase()){*

*//do something* for each pair (c,k)

*}*

Đầu tiên, ta tính N = A+B+C+D là toàn bộ số functions có trong CSDL. Lưu ý các bộ (A(c,k), B(c,k), C(c,k), D(c,k)) có thể khác nhau theo từng cặp (c,k) nhưng tổng của chúng luôn bằng N.

Tiếp theo, ta dễ dàng tìm được số functions có trong class c, tức là tính được N1 = A+B. Lưu ý, N1 chỉ phụ thuộc vào c (không phụ thuộc vào k)

Trong lucene, ta đã đánh index cho class và function. Do đó ta dễ dàng tìm được tất cả function chứa từ khóa k (dựa trên function index), tức là tính được N2 = A+C. Lưu ý, N2 chỉ phụ thuộc k (không phụ thuộc c)

Gọi ListKeywordFunctions(k) là tập tất cả các function có chứa từ khóa k. Giờ ta duyệt qua từng function trong list đó và kiểm tra xem nó có thuộc class c hay không. Sau vòng lặp này ta sẽ tính được A là số functions trong c có chứa từ khóa k.

Sau đó, ta tính B,C,D dựa trên A,N,N1,N2.

Cuối cùng là tính trọng số cho mổi cắp (c,k) ứng với từng độ đo dựa trên 4 hệ số A,B,C,D vừa tìm được (đọc lại cách tính các độ đo bài báo).

Giải thuật đề xuất như sau:

*Foreach (k : lucene.ListAllKeywordsInIndex())*

*Foreach (c : database.ListAllClassesInDatabase()){*

*N = database.totalFunctionInDatabase();*

*N1 = c.totalFunctionInThisClass();*

*N2 = lucene.totalFunctionIndex(k);*

*A = 0;*

*Foreach (f : lucene.ListKeywordFunctions(k)){*

*If (f ϵ c) A++;*

*}*

*B = N1-A;*

*C = N2-A;*

*D = N-A-B-C;*

c.setBNS(k) = BNS(A,B,C,D); //Lưu độ đo BNS của từ khóa k vào class c

c.setChi-Square(k) = Chi-Square(A,B,C,D); //Lưu độ đo Chi-Square của k vào c

c.setOR(k) = OR(A,B,C,D); //Lưu độ đo OR của k vào c

…

*}*

Sau khi tính xong hết các độ đo cho tất cả các cặp (c,k) ta lưu chúng vào bảng sau:

Bảng độ đo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **Class1** | **Class2** | **…** |
| **Keyword1** | TFIDF |  |  |  |
| BNS |  |  |  |
| OR |  |  |  |
| … |  |  |  |
| **Keyword2** | TFIDF |  |  |  |
| BNS |  |  |  |
| OR |  |  |  |
| … |  |  |  |
| **…** | … |  |  |  |
| … |  |  |  |
| … |  |  |  |
| … |  |  |  |

**Ưu điểm**: Truy vấn nhanh, vì khi truy vấn ta chỉ sử dụng lại các dữ liệu đã được tính toán trong bảng độ đo.

**Hạn chế**: Tốn nhiều chi phí để xây dựng bảng độ đo. Ngoài ra khi database thay đổi ta phải tốn nhiều chi phí để cập nhật lại bảng.

1. **Tính trọng số từ khóa khi thực thi câu truy vấn** (Truy vấn tới đâu tính toán trọng số tới đó)

Đầu tiên, ta cần tìm tập từ khóa và tập class. Tập từ khóa sẽ được rút trích từ câu truy vấn. Dựa trên tập từ khóa này, ta rút trích tất cả các class bẳng lucene (dùng class index) để làm tập class.

*Foreach (k : Query.ListKeywordInQuery())*

*Foreach (c : lucene.ListKeywordClasses ( k ) ){*

*N = database.totalFunctionInDatabase();*

*N1 = c.totalFunctionInThisClass();*

*N2 = lucene.totalFunctionIndex(k);*

*A = 0;*

*Foreach (f : lucene.ListKeywordFunctions(k)){*

*If (f ϵ c) A++;*

*}*

*B = N1-A;*

*C = N2-A;*

*D = N-A-B-C;*

c.setBNS(k) = BNS(A,B,C,D); //Lưu độ đo BNS của từ khóa k vào class c

c.setChi-Square(k) = Chi-Square(A,B,C,D); //Lưu độ đo Chi-Square của k vào c

c.setOR(k) = OR(A,B,C,D); //Lưu độ đo OR của k vào c

…

*}*

Lưu ý: Trong phương pháp này, sau khi tính toán xong các độ đo cho từng cắp (c,k), ta sẽ không lưu chúng vào bảng độ đo. Do đó, mổi lần truy vấn ta phải tính toán lại tất cả.

Ưu điểm: Làm tới đâu tính toán tới đó, không phải mất công xây dựng bảng độ đo.

Hạn chế: Truy vấn tốn nhiều thời gian để tính toán lại các độ đo.

1. **Đo lường hiệu quả các độ đo:**

Sau khi truy vấn, ta lưu tất cả class c tìm được dựa trên tập khóa vào 1 listCalculatedClass. Sau đó làm thêm một số tác vụ sắp xếp (sort) các class dựa trên “mức độ tương tự” với câu truy vấn (đọc lại bài báo mục IV–a.Class Retrieval để biết thêm chi tiết). Lưu ý, ứng với mổi độ đo ta sẽ làm 1 hàm sắp xếp tương ứng. Ví dụ: listCalculatedClass.sortByOR() – sắp xếp các class c theo độ đo OR, hay sortByBNS(), sortByChi-Square(),…

Tới đây thì xong giai đoạn 1, tiếp theo là tìm cách đánh giá xem phương pháp sắp xếp dựa trên độ đo nào là tối ưu nhất (cái này thì em suy nghĩ thêm).