Information Retrieval Project

R93725006 游子賢 (與 R92725028 蔡景祥)

June 2005

1 實驗工具與環境

- 程式語言 – Perl 5.8.6
- 資料儲存 – QDBM: Quick Database Manager (http://qdbm.sourceforge.net/)
- Stemming – Snowball Stemmer (http://snowball.tartarus.org/)
- WordNet – 用來查詢單字原型
- CPU: Pentium 4 3.0 GHz
- Physical RAM: 1GB
- Operating System: FreeBSD 5.4-RELEASE

2 程式與流程

2.1 程式 index.pl (index phase)

1. Index words 的選擇: 從檔案讀入並切割文件, 全部轉爲小寫, 除去太短的字以及 stop words
2. 使用上面提及的 snowball stemmer 對單字進行 stemming
3. 計算每篇文件的 TF
4. 計算每個單字的 IDF
5. 全部文件掃瞄過後, 計算每篇文件的單字的 TFIDF 及 normalize 之後儲存 (logical view) (捨去 DF<6 的單字)
6. Interved tables 按照開頭字母存在 26 個 db files 內, logical view 單獨存爲一個檔案
7. 在 normalization 之前的 logical view 包含的是文件內的 TF, 這也將保留, 留給 incremental 時重新計算 TF-IDF 用

8. TF-IDF 式子:

$$\text{TF-IDF} = \text{TF} * \log \frac{\text{total number of docs}}{\text{document frequency}}$$

2.2 程式 index-incremental.pl

1. 前面步驟與一般 index 一樣.
2. 更新 inverted tables 及 logical view file.
3. 重新計算所有文件 (包括先前的 FBIS3) 的 TF-IDF 值

2.3 程式 query.pl (query phase)

1. 讀入 query topic, 使用 title + description + narrative 作爲比較用的內容
2. 對文章作與上面相同的處理 (stemming 等), 除了只使用 TF 而不是計算 TF-IDF
3. 將 Logical View 讀出, 與每篇 query 比較相似度 (cosine similarity)
4. 對於每篇 query topic, 將相關文章以相似度排序

3 花費時間

3.1 FBIS-3

Index 部分:

```
Elapsed time: 4:38
Writing inverted tables...
Elapsed time: 4:44
Calculating TF-IDF...
384.722u 10.105s 6:55.31 95.0%  10+23317k 2+358io 0pf+0w
```

Query 部分:

```
$ time ./query.pl > db3.out
81.583u 3.617s 1:25.37 99.7%    10+18571k 0+98io 0pf+0w
```

3.2 FBIS-3 + FBIS-4 (incremental)

Index 部分:

```
Elapsed time: 6:01
Writing inverted tables...
Elapsed time: 6:40
Calculating TF-IDF...
576.605u 31.007s 16:31.19 61.8% 10+-17916k 34887+166io 7pf+0w
```

Query 部分:

```
time ./query.pl > db3+4i.out
171.376u 9.833s 3:03.18 98.9%   10+38487k 429+222io 7pf+0w
```

3.3 FBIS-3 + FBIS-4

Index 部分:

```
Elapsed time: 10:31
Writing inverted tables...
Elapsed time: 10:41
Calculating TF-IDF...
825.011u 27.437s 16:47.22 84.6% 10+-11208k 10853+511io 3pf+0w
```

Query 部分:

```
$ time ./query.pl > db34.out
167.834u 4.395s 3:00.25 95.5%   10+3691k 8758+0io 0pf+0w
```

4 效能

4.1 相似度門檻值

我們先做了實驗, 瞭解相似度門檻值設在何處會有最好的結果. 我們分別實驗 0.1, 0.15, 0.2, 0.25 ... 0.5, 結果發現是 0.1 最好.

4.2 使用 topic 中的欄位

在相似度門檻值 0.1 的情形下, 應該使用 topic 中的哪些欄位?

使用欄位	map
Title	0.2072
Desc	0.1216
Title+Desc	0.2542
T+D+Narrative	0.2078

可見 Title + Desc 可得到最好效果. Narrative 欄位內似乎有太多雜訊.

4.3 綜合結果

使用相似度門檻值 0.1, query topic 採用 title+desc 的結果:

	Full Index Time	Incremental Time	Search Time	Average Precision	Precision at R(30%)	Precision at 10 docs
FBIS3	6:55		1:25	0.25	0.35	0.38
FBIS3+4	16:47	16:31	3:00	0.24	0.35	0.44

5 心得

5.1 Stemming V.S. WordNet

一開始我們以爲使用 WordNet 查出字的原型會比 stemming 的效果來得好, 但事實不然, 兩者並無顯著的差異。而且 WordNet 的速度相較之下較慢, 因此沒有需要。

5.2 Caching stemming

一開始的速度不快, 一部份是因爲 stemming (若使用 WordNet 速度更是慢上幾十倍), 後來直接使用 hash 儲存 stemming 的結果, 速度便加快了不少。

5.3 計算 TF-IDF

一開始是在 query 時才計算 TF-IDF, 但是這樣導致速度太慢, 後來改爲在 index 時就計算, 利用記憶體內已有的資料, 可以把總時間縮短。

6 未來工作

1. Feature selection 的方法改進。
2. TF-IDF 權重的調整。
3. 使用 NLP 工具, 只索引名詞。