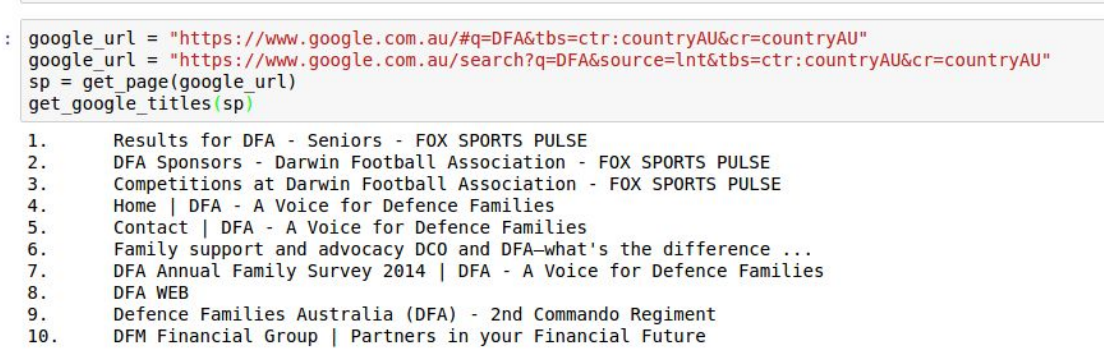
COMP6714 ASSIGNMENT 1

Changjian Zhu   z5082423

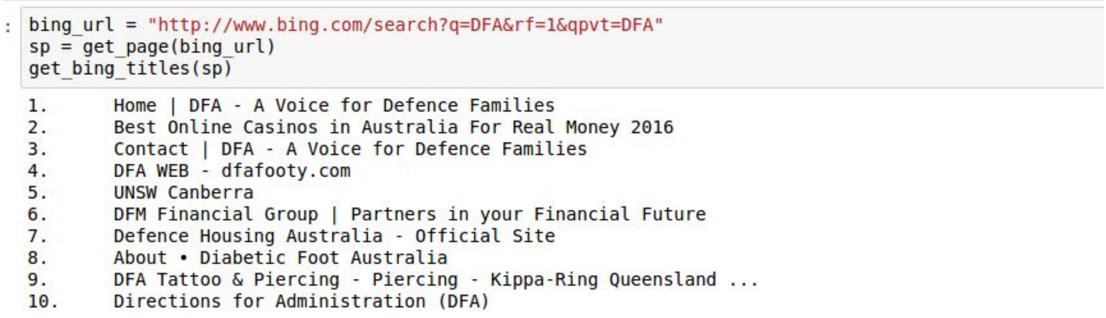
Q1.

1) The top ten search results given by Google and Bing are totally different(I did the web scraping instead of screenshots).



```
google_url = "https://www.google.com.au/#q=DFA&tbs=ctr:countryAU&cr=countryAU"
google_url = "https://www.google.com.au/search?q=DFA&source=lnt&tbs=ctr:countryAU&cr=countryAU"
sp = get_page(google_url)
get_google_titles(sp)
```

```
1.      Results for DFA - Seniors - FOX SPORTS PULSE
2.      DFA Sponsors - Darwin Football Association - FOX SPORTS PULSE
3.      Competitions at Darwin Football Association - FOX SPORTS PULSE
4.      Home | DFA - A Voice for Defence Families
5.      Contact | DFA - A Voice for Defence Families
6.      Family support and advocacy DCO and DFA—what's the difference ...
7.      DFA Annual Family Survey 2014 | DFA - A Voice for Defence Families
8.      DFA WEB
9.      Defence Families Australia (DFA) - 2nd Commando Regiment
10.     DFM Financial Group | Partners in your Financial Future
```



```
bing_url = "http://www.bing.com/search?q=DFA&rf=1&qpvt=DFA"
sp = get_page(bing_url)
get_bing_titles(sp)
```

```
1.      Home | DFA - A Voice for Defence Families
2.      Best Online Casinos in Australia For Real Money 2016
3.      Contact | DFA - A Voice for Defence Families
4.      DFA WEB - dfafooty.com
5.      UNSW Canberra
6.      DFM Financial Group | Partners in your Financial Future
7.      Defence Housing Australia - Official Site
8.      About • Diabetic Foot Australia
9.      DFA Tattoo & Piercing - Piercing - Kippa-Ring Queensland ...
10.     Directions for Administration (DFA)
```

2) a) Token normalization: unnecessary because “ioauen” is already a not bad token.
b) Query expansion: Google only got 3 results and the first one is “6714 ass1 specification”, because it keeps the term as original term . Bing got 87 results because it automatically explain the search term to other candidates.
c) Query suggestion: Google shows ‘Did you mean: "aiou"’. While Bing does not have the option to search original term.

3) (a) “Neuro-linguistic” (About 72,700 results)
(b) “otta” (About 19,400 results)
(c) “Neuro-linguistic” “otta” (About 3 results)
(d) “Neuro-linguistic” OR “otta” (About 92,000 results)
(e) “Neuro-linguistic * otta” (About 1 result)
(f) “bugle” (About 384,000 results)
(g) “bugle bugle” (About 2,100 results)
(h) “bugle bugle bugle” (About 2 results)

1. The estimated numbers do make sense in my result, because
   72700(“Neuro-linguistic”) + 19400(“otta”) - 3(AND) ≈ 92000(OR)
2. The range should be **[0, min(A, B)] = [0, 19400]**

Q2.

1) a) “fools rush in” occur in **doc 2(pos 1), doc 4(pos 8), doc 7(pos 3)**.
   b) “fools rush in" AND “angels fear to tread": **doc 2(pos 12)**.
2) One obvious mistake is the **duplicated index 15** in doc 7.
   Another one is that “fools” and “angels” should be “fool” and “angel”.

Q3.

1) **Time complexity:** 1. The outer while loop is doing naive “AND” intersection of document ids, so the worst time complexity is **O(sum(p1)+sum(p2)) = O(n)**
   2. Although there are two while loops, it scans **each element only once** in both p1 and p2, so the time complexity is **O(n)**. In this inner while loop, **deleting pp2 from *l* is O(k)** because the program is doing linear scanning on “l”.
   As a result, **the total time complexity in worst case is O(n^2k) = O(n^3).** In addition**,** It means that the performance will be awful if k is a huge number.

   Modification: Using galloping search in deleting the non matching part in *l,* because the elements in *l* are already sorted, so it’s not necessary to scan each element to delete it one by one from the beginning.
   Eventually the improved time complexity would be **O(n^2logn)**.

2) **/k:** within k words respectively, **/s:** in the same sentence. **/p:** same paragraph.
   /k: same as the algorithm above in (1).
   Here are two methods to support the other two proximity operators.
   1. Recording sentence and paragraph id with each position id.
      For example:
      [1, 15, 75]→[(1,1,1), (15,2,1), (75, 12, 3)], but it’s **space consuming**..
   2. In the pre-processing, transferring “!?. etc” to token(end of sentence) and ”\n” to token(end of paragraph), storing them as inverted indexes.
      For example:
      Positional inverted indexes of “end of sentence”(doc 1): [7, 18, 30, 56, ...]
      Positional inverted indexes of “end of paragraph”(doc 1): [30, 90, ...]
      Then we can use this two inverted indexes to check and filter results.

Q4.

1) **Immediate merge(eager):** there will always only have **one** sub-index. For example: for each new set of docs: *Merge_runs*(I_0, I_1) -> I_0'.
2) **No merge(lazy):** the number of sub-index is C/M (the total occurrence of merging). For example, the sub-index will become: I_0 , $K(D_1\text{-}D_{10})$ -> I_1 , I_2 , I_3....
3) **Logarithmic merge(same idea as binary):** if we assume cannot have **two indexes** in one generation and the size of each subindex want to merge is 1, so the change of sub-indexes will be like this “..16, 8, 4, 2, 1” + “1” → “.. 32” ~~In this case, the answer(number of sub-indexes) is 5, (16+8+4+2+1=31=2⁵-1) is C, size of sub-index(1) is M, as a result:~~ $31(C) = (2^5 - 1) * 1(M)$
   ~~So if the number of sub-indexes is~~ x ~~, we can get this equation:~~
   $C = (2^x - 1) * M$ ~~, then~~ $x = log_2(C/M + 1)$ ~~. (same idea as the depth of binary tree :P)~~

   Sorry my previous thoughts was wrong… the case(“..16, 8, 4, 2, 1”) above is the “worst case”, the correct answer should be the number of 1 in the binary representation of C/M:

   For example: if C/M is 13, 13=1101, there are 3 ‘1’ in ‘1101’, so the number of sub-indexes in that case is 3.