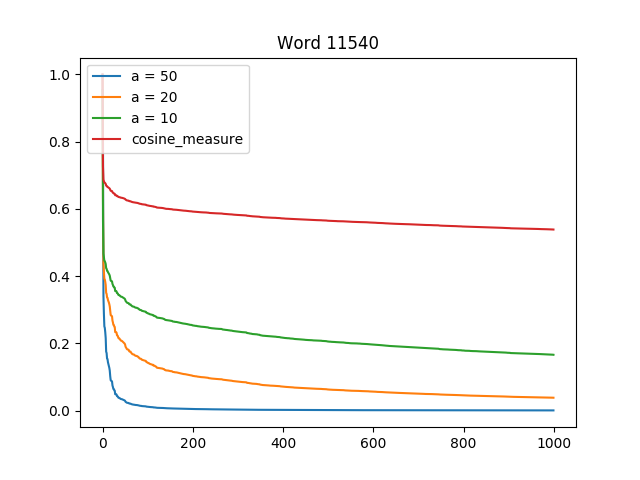
Homework 2

60547047S 羅天宏

一、方法概述：

　　這次實作的方法為Embedding-based Query Language Models，主要的做法是將維基百科作為外部的訓練資料，產生word embeddings，再藉由word與word之間的vector的相似度做度量衡，決定要不要更新query model裡的字詞。核心概念是藉由訓練外部的資料，來得到更多語義間的關聯，在更新query model後，改善one-shot和feedback的查詢結果。

　　這論文的方法與其他相似的query expansion不同之處，在於這篇論文強調的是如何更新字，如能不能讓很像與非常像之間的距離分得更開，指更新在相似度上非常像的word，因此作者用了一些巧思，將cosine similarity的結果，用常用在classification的sigmoid來做線性轉換，讓相似度的值下降得更快，有些discriminant的意思。



Sigmoid和cosine similarity的比較

　上圖是Word 11540在不同的相似度計算下的結果，縱軸是相似度，橫軸是前一千個較相似的word排序。a為sigmoid函數裡的gamma，a值越大，sigmoid函數的值就降得越快。

Sigmoid公式如下

Sigmoid(x) =

二、Embedding-based Query Language Models

　　目的是找出不存在query裡，但語意十分相似的term。作者設計了兩個相似度的計算想法都是源自於relevance和irrelevance，盡量與query term相似，然後與其他的term越遠越好。

1. **Conditional Independence of Query Terms**

　　假設query term彼此之間的產生機率是獨立、不互相影響。因此term與query的相似度可以用term與每一個query term獨立計算。因為law of total probability，所以可以將公式推導為連乘積。

計算公式為：

(w|) ∝ = (w)

1. **Query-Independent Term Similarities**

　　假設term和term之間的相似度與query model產生query term的機率有關。一樣假設每個query term之間的關係為independent，但在計算相似度的過程中，考慮了第三方的term.

計算公式為

(w|) ∝ \*

三、實驗：

實驗設計：

Word2Vec (gensim)

300 dimension

5 window size

cbow

Min-count = 5

Parameter:

a =10, 20, 30, 40, 50

c = 0.1, 0.2 …. 0.7

m = 1 ~ 80

Interpolated alpha = 0.1, 0.2 … 1

a 和c為sigmoid的參數。

m：

為在與query計算完相似度並排序之後，要更新的前幾個term。

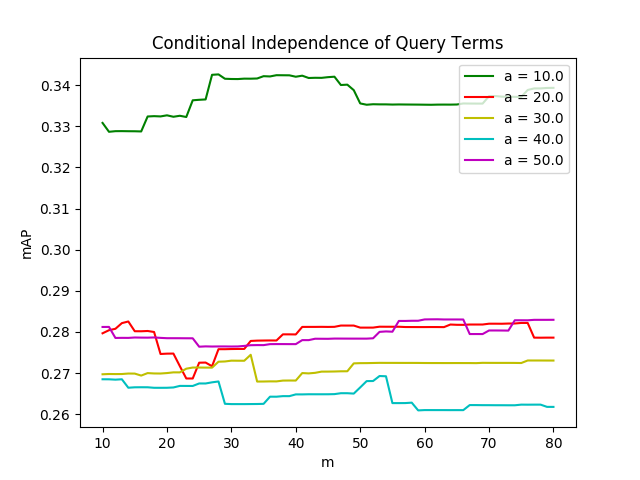
Interpolated alpha：

為兩種expansion計算後的query model q’，要以多少比例內插回原本的query model。

實驗結果：

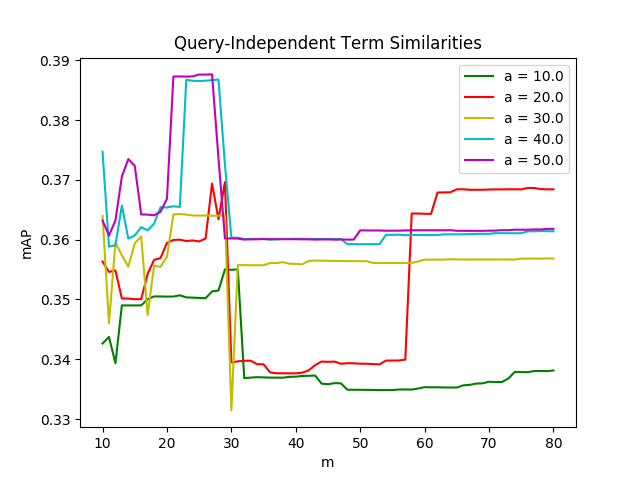
實驗結果發現c不影響mAP，主要影響的地方是a 和 m，因此將這兩個參數之間的交互關係畫成下圖：

1. **Conditional Independence of Query Terms**



　　在第一個方法裡面，我們發現若將a調整得太高，讓鑑別度變得太大。不論是加入多少term，都無法挽回mAP下降的趨勢。因此最好是將a設為10，然後加入一點點的term，效果就會變得不錯。

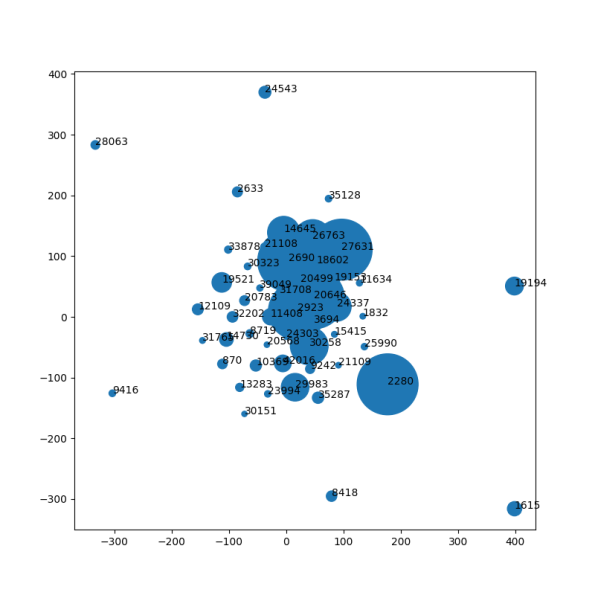
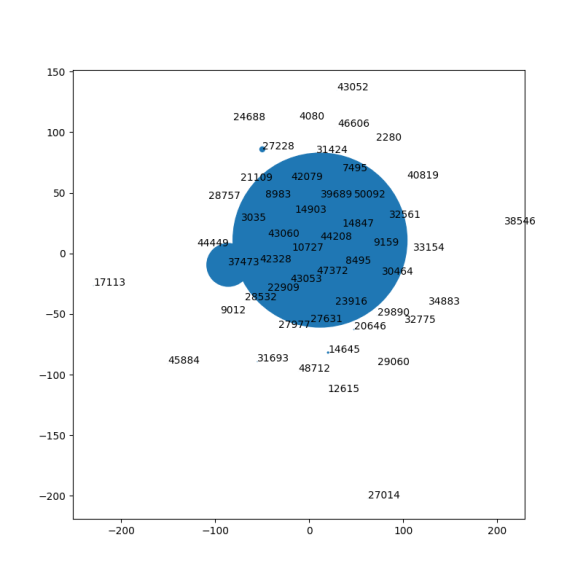
1. **Query-Independent Term Similarities**



　　第二個expansion的方式則是將a調至50會有最好的效果，不過在40的時候，效果也就差距一點，兩者可說是不分軒輊。趨勢都是在m為20到30中間，會讓mAP的值達到最大。

實驗延伸：

**一、觀察word的分佈**



　　上圖分別為兩種方法在20001.query學習後的結果。左圖為第一個expansion方法，右圖為第二個expansion的方法。這是將word vector降至兩維呈現的結果，座標的遠近可以視為字與字之間的距離，越近代表越相似。而圓的面積則是代表這個字利用兩種expansion方法後學習到的相似度，越大代表與20001.query越相關。

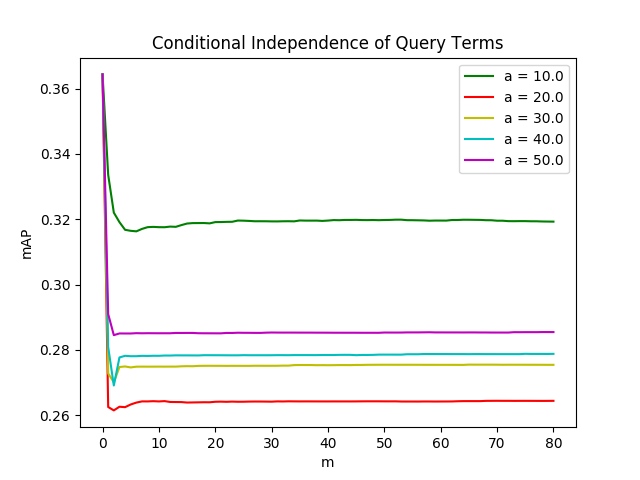
　　可以看出雖然兩種方法學到的分佈差不多，但第一種的方法只找到一個最像的字，其他的機率都非常低，而第二種方法則是可以比較均勻的分給周遭的的字詞。

　　上述結果造成的影響是，第一種方法的鑑別度不能太二元(sigmoid不能驟降)，且加入的term不能太多，不然會因為內插的關係，拉低原先在query 的term的機率。而第二種方法則是相反，因為學習到的字的機率較為平均，因此可以允許較大的鑑別度(sigmoid函數驟降)，且盡量加入多個字詞。

**二、只加入missing word**

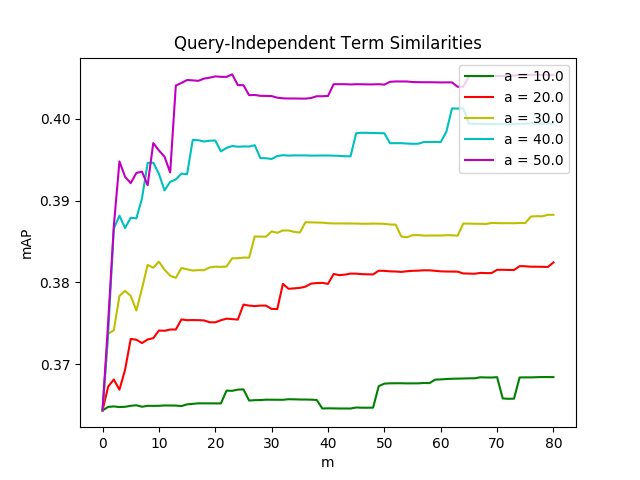
　　因為兩個方法的結果，都呈現相似度較高的term，本身就是query model裡的term，因此在這裡就設計了在計算相似度時，如果該term本身就存在於query裡面，那麼強制令該term與query的相似度為0，以此來避免該term加入query model.

1. **Conditional Independence of Query Terms**



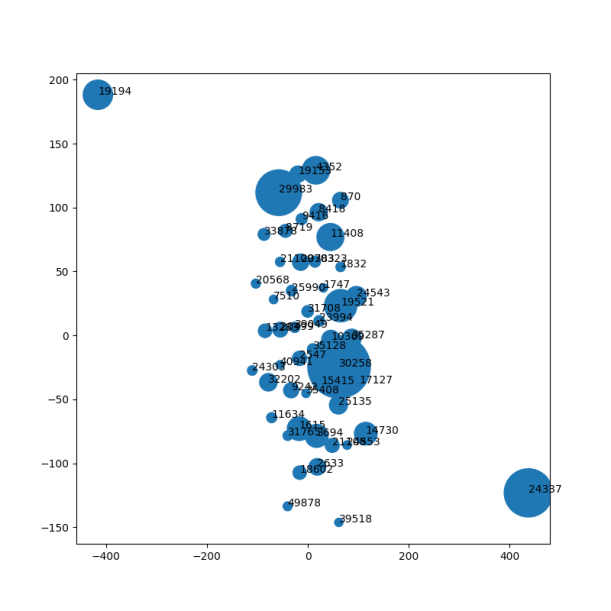
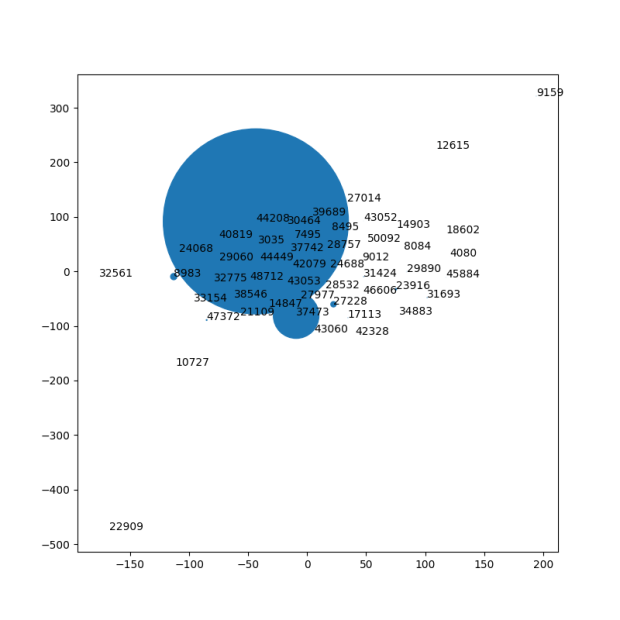
　　上圖可以看出，這樣設計的方式對第一個結果有比較不良的影響，最高的值在不應該加入任何query以外的term。

**2.Query-Independent Term Similarities**



　　與第一個方法不同，只針對missing word對第二個方法影響較為顯著，讓整體的mAP都提升了不少。最好可讓mAP上升到0.40。

**加入missing word後兩個方法呈現的分佈**



可以看出兩個方法學到的分佈就已經不太相同，第一個方法一樣維持一個最像的，其餘的都不太相似，但學到的相似字與原先學到的相似字有些差距；第二個方法比較有趣，學習的結果與原先的只差一些，更像是將原先較為集中的分佈給平攤出來，讓variance變得更亂一些。

統整最佳結果和參數在ppt裡面。