On Large-Scale Multi-Label Classification for POI Tagging

楊鎧謙
中央大學資工所碩士班
as5374942@gmail.com

張嘉惠
中央大學資工系教授
chiahui@g.ncu.edu.tw

劉胥影
東南大學計算機學系
liuxy@seu.edu.cn

摘要

近年來智慧型手持裝置迅速普及，現在已經達到幾乎人手一機的情況。而交通方式的進步更是使得人們到陌生地點的機會增加。在陌生的環境之中要尋找感興趣的點是不容易的，所以需要提供電子地圖系統以便查詢。因為使用者可能不知道這些點的確切名稱，他們可能只是想找特定類型的點，所以需要提供類別搜尋服務。

為了要提供類別搜尋服務，我們需要將系統中所有的點進行分類。因為系統中有許多筆資料，每筆資料都有一個或多個類別，所以這是一個大數量的多類別分類問題。我們使用中華黃頁的分類方式。類別有共有 1,287 種。因為類別與資料較多使得一般訓練分類器的方式需要訓練較多個分類器。我們利用降低類別維度的方式來加快訓練與測試的速度。

實驗顯示採用 KDE+SVM 的混合模型方式的訓練時間與測試時間皆比一般的 SVM 分類快幾乎一倍，然 Micro-F1 為 0.718 低於 SVM 的 0.783。由於資料為非平衡資料我們試圖利用 Reweighting 和縮減取樣(Downsampling)的方式想增進效能，但其結果顯示在大數量的資料中這兩種方法也不太有效。

關鍵字：機器學習、多類別分類、非平衡資料、興趣點。

I. 緒論

由於智慧型手持裝置以及行動網路的迅速發展，電子地圖服務也跟著日益增加，這些服務提供使用者查詢的功能。使用者在規劃旅遊行程或是到達一個陌生的環境，常常會需要利用這些電子地圖來尋找他們感興趣的景點或店家，這些點我們稱之為 POI（Point of Interest）。

通常使用者要在電子地圖上搜尋 POI 時，往往是直接輸入 POI 名稱來搜尋這些 POI，但是使用者對於不熟悉的地方會不知道附近有哪些 POI。這時就無法使用 POI 名稱來進行搜尋，而是要使用類別名稱來進行 POI 的搜尋。

在提供電子地圖服務時，收集這些 POI 的資料是基本的第一步，這些 POI 的來源十分多樣。如：黃頁、政府登記資料或是使用者標記等等。這些資料通常都會包含 POI 名稱以及位置，但不一定會有 POI 的類別。這些沒有包含類別資訊的 POI 就無法利用類別搜尋被使用者查詢到，所以要將這些不包含類別資訊的 POI 進行分類以便使用者可以利用類別搜尋找到這些 POI。

本論文使用的類別是比照中華黃頁的類別，雖然類別有 1,287 類，由於有些類別 POI 過少，我們將這些類別過濾後有 765 類。然而不是所有的 POI 都只包含一個類別，一個 POI 可能會屬於兩個或兩個以上的類別，所以我們面對的是一個多標籤分類的問題。

II. 相關研究

在解決分類問題的第一步是特徵擷取，在文字特徵擷取的部分，Yang 等人[11]比較關於各個文字特徵擷取方式的效果，最後得到的結論是 Information gain 以及 Chi square statistic 的結果相對較好。

多類別分類最常見的方式就是分別對於每個類別各訓練一個二元分類器，測試時將全部分類器的結果結合成為輸出結果。這種方式稱之為 One-vs.-rest Classification，當類別數量眾多，為了減少訓練分類器數量，Weston 等人[10]在 2002 年提出了一種稱為 KDE(Kernel Dependency Estimation)的方法，這個方法在訓練時有兩個步驟，第一步先將所有訓練資料的類別向量組合成一個矩陣並對矩陣進行資料降維，如：PCA(Principal Components Analysis)[3]使類別維度從 q 降為 k，k 的值由 PCA 過程中所取的特徵向量數量所決定。之後再分別對這 k 個維度訓練回歸模型，在測試時只需要測試這 k 個模型取得 k 個值，並將這 k 個值還原回到 q 個維度就能得知測試結果，如此一來便可以將訓練所需要的分類器個數由 q 減少為 k，可以有效減少訓練和測試的時間，但會因為在降維過程中有丟棄一些資訊而使得分類結果的準確性受到影響。

除了速度的考量，提升準確率更是分類系統最主要的目標。Godbole 等人[6]提出一種類似堆疊的方法，他們的做法是先使用 SVM 訓練一組 One-vs.-rest 分類器，之後將訓練資料當作測試資料測試分類器，將分類器產生的結果和原始的特徵合併成為新的特徵，利用這些新的特徵訓練新的 One-vs.-rest 分類器來提升分類效能。

另外 Tang 等人[8]的做法則是判斷 POI 可能會有幾種類別將 POI 的類別數量當作輸出結果，訓練 MetaLabeler 分類器，例如：POI 只有一個類別時 MetaLabeler 的類別就是為一，兩個類別就為二以此類推。並且藉由這個分類器來輔助 One-vs.-rest 分類器的判斷。

另外由於在 POI 的所有類別之中，數量的分布是十分不平衡的，常見類別數量可能是稀有類別的 100 倍以上，在機器學習中這樣的資料被稱為非平衡資料。Byron C 等人[9]針對非平衡資料嘗試一些改善的方法，其中較常見的方式為改變訓練 SVM 時正例的權重(Reweighting)和縮減取樣的方式，我們也會嘗試利用這兩種方式來改善分類效能。

III. 系統架構

本章節主要描述本論文系統的運作流程及架構(圖 1)，首先會先介紹資料準備及處理，包含 POI 資料的來源、特徵的擷取。接著介紹分類器的訓練以及選擇。最後說明測試資料的過程。

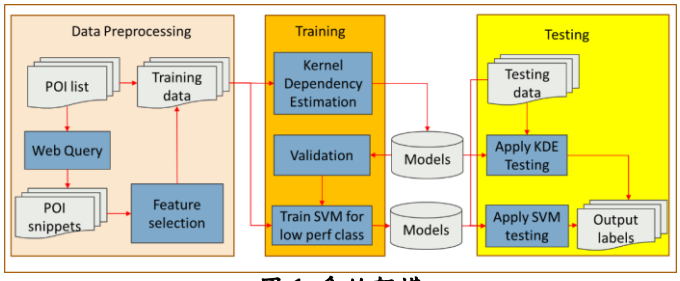


圖 1. 系統架構

A. 資料前處理

從網路上搜集回來的 POI 資料大多只有包含 POI 名稱以及經緯度或地址。這些內容通常很簡短，難以進行分類，所以我們需要額外的 POI 相關描述讓我們可以找到更多的特徵來訓練分類器。

Google 搜尋引擎是目前最熱門的搜尋引擎，所以我們利用 POI 的名稱作為關鍵字在 Google 搜尋引擎上尋找相關文本，並利用網路爬蟲取得 snippets，我們對於每個 POI 分別取得十個 snippets。我們對這些 snippets 進行一些前處理，包括利用中央研究院所研發的中文處理系統 CKIP(Chinese Knowledge Information Processing)[7]中的斷詞功能對 snippets 進行斷詞，並且將中文以外的字符移除。

接著我們對這些處理完的 snippets 萃取特徵，根據 Yang[11]等人實驗完的結果，Information gain(IG)和 Chi square statistic（CHI）的效果是明顯較好。

因為 POI 的類別數量較多所以在計算 IG 分數時會因為公式的關係讓分數變得過低使得我們無法找到相對較好的詞作為特徵，所以我們採用 CHI 來選擇我們在訓練分類器時所使用的特徵。我們分別將每個類別前十高分的詞作為特徵列表，在過濾重複的詞之後在我們的資料集裡會得到 6,516 個特徵。我們採用稀疏矩陣來儲存特徵向量。

B. KDE-based Classification

由於 POI 的類別量較多，在訓練分類器的時候時間成本較高，在測試時花費的時間也較多，所以我們採用 KDE 訓練我們的基本分類器，並且使用 SVM 來輔助 KDE 表現較差的部分，雖然可能會犧牲一些效能，但能夠使得整體的訓練及測試時間下降。

在 KDE 的訓練流程中首先我們會先將訓練資料的類別向量組和成為一個 n*q 的矩陣，其中 n 代表訓練資料的個數，而 q 指的是類別的數量。我們對矩陣進行 PCA[3]處理使得矩陣的維度減少。在本論文中我們採用整體特徵值的 75%和 85%來進行降維並比較各別結果。

在 PCA 處理過後原本 n*q 的矩陣會乘上由特徵向量組成的 q*k 矩陣變為 n*k 的矩陣，k 的值由選取特徵向量的數量所決定，這使得需要訓練的分類器數量由 q 變成 k。因為由 0 和 1 組成的矩陣會變成由小數組成的矩陣，所以我們採用 SVR（Support Vector Regression）的方式，並且利用 LIBLINEAR[5]來訓練這 k 個 SVR 模組。

因為 KDE 的方式會去除特徵值較低的特徵向量，雖然這些特徵向量包含的資訊量較少，但還是有包含一定程度的資訊量，所以去除這些特徵向量會使得分類器的效能下降。在訓練資料中，由於各個類別包含的 POI 量差距很大，所以 POI 數較小的類別就可能會是被移除的資訊而導致這些類別的效能很差。為了解決小類別預測效能的問題，我們對效能較差的類別分別訓練 SVM 分類器，在時間的考量上這樣是可行的，因為小類別訓練時間較少影響的程度較小，所以可以採用 SVM 做訓練。

在 KDE 的測試中。首先也要取得測試 POI 的相關描述，之後進行特徵擷取產生測試資料，利用 SVR 模組對測試資料進行測試，測試結束後每筆 POI 會得到 k 個結果，將所有測試 POI 的結果組成一個 n*k 的矩陣，這裡 n 為測試資料量，這時我們要將 n*k 的矩陣還原為 n*q 的矩陣，方法就是乘上訓練時降維用矩陣的轉置矩陣。

在將類別還原回到原本的維度後，類別向量中的值並不是 0 跟 1 而是 0 到 1 區間的小數值，我們必須將這些數值映射到 0 跟 1 使得我們可以產生分類結果。我們訂定一個閾值 β，當類別向量中的數值高於 β 時，將之映射到 1，反之映射到 0。

我們利用驗證資料計算各個類別的效能，我們可以找出效能相對較低的類別並記住這些類別，並針對這些類別進行額外的處理以提升效能。我們發現 POI 數量小於一定程度時，該類別的 F1-Measure 會趨近於 0，我們認為原因是這些類別的 POI 量過少以至於在降維的過程中被捨去。我們對於這些類別分別使用工具 LIBSVM[1][4]訓練 SVM 分類器。由於 POI 的資料量較大且類別量也較多所以特徵數量也較多，訓練 SVM 分類器時如果是使用 RBF kernel 訓練那會使得訓練時間加長，所以我們所有的 SVM 分類器在訓練時一律是採用 linear kernel。在我們的資料中有 430 個類別會有這個問題。由於這些類別包含的 POI 數量較少所以在訓練 SVM 分類器時花費的時間較少，所以不會增加太多的訓練時間。

C. 資料測試

訓練完分類器之後，就可以進行資料測試。首先我們會對 POI 作和訓練時一樣的前處理，就是將 POI 作為關鍵字查詢 snippets 並擷取前十筆 snippets 做為測試文本，之後對文本進行斷詞、特徵擷取等前處理產生測試資料，利用 KDE 分類器進行測試後每一筆資料得到 k 個值，將這 k 個維度的資料還原回到原本 q 個維度。另外使用 SVM 分類器測試數量較少的類別，將測試結果取代 KDE 的部分結果，最後將合併的結果作為最終輸出結果。

IV. 實驗結果

本章節將會比較混合模型和其他方法的差異，如：效能、訓練時間、測試時間等等。另外也會比較不同權重與不同取樣方式對於分類結果的影響。

A. 資料集描述

本論文的訓練及測試資料皆採用中華黃頁的資料、類別及其所定義的類別。在中華黃頁中總共有 1,287 個類別，但並不是所有的類別都有許多筆 POI。我們認為這些類別比較冷門且訓練效果不佳，所以濾掉低於 100 筆的類別，在進行過濾之後，數量剩 765 類、908,130 筆的 POI。在實驗中我們使用 1 至 30%的資料量作為訓練資料，以另外 20%的資料作為測試資料，在所有的訓練及測試資料中所有類別的比例都與原資料比例相近。我們進一步對 POI 資料作分析，發現 POI 很少會有高於 5 個類別的 POI。

B. 評估方式

在本小節中將介紹我們所使用的評估方式。首先我們將y_i訂為標準答案（如果 POI 包含此類別就表示為 1 反之則為 0），而$\hat{y}_i$則是系統輸出結果。我們假定 N 為實例數量，Q 則表示為類別數量。P^q代表第 q 類的精確率(Precision)，R^q代表第 q 類的召回率(Recall)，F_1^q是第 q 類的 F1-Measure，其計算方式如下：

$$P^q = \frac{\sum_{i=1}^{N} y_i^q \hat{y}_i^q}{\sum_{i=1}^{N} \hat{y}_i^q} \quad R^q = \frac{\sum_{i=1}^{N} y_i^q \hat{y}_i^q}{\sum_{i=1}^{N} y_i^q} \quad F_1^q = \frac{2P^q R^q}{P^q + R^q}$$

在計算完每個類別的效能之後，我們需要進一步的計算整體的效能，這時我們會計算兩種數據，分別為 Macro-F1 以及 Micro-F1。Macro-F1 是計算每個類別效能的平均其計算公式如下：

$$\text{Macro-F1} = \frac{1}{Q}\sum_{q=1}^{Q} F_1^k$$

Micro-F1 則是全體結果的 F1-score，計算方式如下：

$$\text{Micro-F1} = \frac{2\sum_{q=1}^{Q}\sum_{i=1}^{N} y_i^q \hat{y}_i^q}{\sum_{q=1}^{Q}\sum_{i=1}^{N} y_i^q + \sum_{q=1}^{Q}\sum_{i=1}^{N} \hat{y}_i^q}$$

Macro-F1 以及 Micro-F1 的不同處在於：Macro-F1 是將所有的類別權重視為相等，所以當有些類別效能不佳時，整體分數會受到較大的影響。Micro-F1 則是不分類別直接以整體實例作計算，當有些類別校能不好時，如果這些類別包含的實例數不多，那對於整體也不會有太大的影響。

C. 實驗分析與討論

我們將會在這一小節中介紹各種實驗的流程方法及結果，並探討這些結果的原因，實驗包括各參數的比較、不同方法的訓練與測試時間的比較、效能的測試與比較。

1) Pre-image 方法對 KDE 結果的影響

前面提到 KDE 在測試後需要將結果映射到 0 和 1，由於測試結果為 0 到 1 之間的小數，所以我們設定一個閾值 β，高於 β 就映射到 1 反之則映射到 0。本段將比較不同的閾值 β 對於 KDE 模組最終分類的效能會有哪些影響。

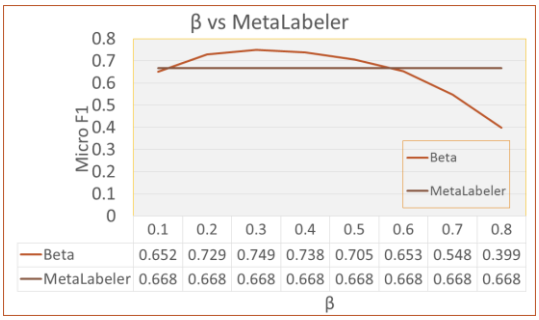


圖 2. β 值與 Micro-F1 的關係

在本次實驗中我們取 30%的資料作為訓練資料訓練 KDE 分類器，以另外 20%的資料作為測試資料，降維的過程中我們取前 85%特徵值所對應的特徵向量做為映射空間。在測試完畢後我們將 β 從 0.1 遞增到 0.8 並計算各個值的 Micro-F1 值。另外我們也嘗試使用 MetaLabeler 作為映射到 0 或 1 的依據，我們會先將資料對 MetaLabeler 進行測試，得到每筆資料可能會屬於 n 個類別，之後再將矩陣還原後數值前 n 大的值映射到 1 其餘則為 0。兩種方式的比較如圖 2 我們發現在 β 值為 0.3 的時候分類的結果是最好的，並且高於 MetaLabeler 的效能。所以在之後 KDE 的相關實驗我們都會使用 β 值做為映射條件，並將其設定為 0.3 以達到最好的效能。

2) 訓練與測試時間結果

我們嘗試比較各種不同的分類模組的訓練時間以及測試時間，我們分別選擇 SVM、KDE 前 75%特徵值、KDE 前 85%特徵值與我們的方法進行比較。SVM 會有 765 個分類器，KDE 分別需要產生 220 和 330 個 SVR 分類器，而我們改良的 KDE 會再加上 430 個 SVM 分類器來改善小類別分類。

我們對於這些方法分別統計使用全部資料中的 1%至 30%資料量作為訓練資料時所花費的訓練時間，由圖 3 可以知道 SVM 的訓練時間最久，成長幅度也是最明顯，我們的方法次之但與全部使用 SVM 訓練仍可節省不少時間，KDE 彼此之間的差距不明顯都是花最少時間。

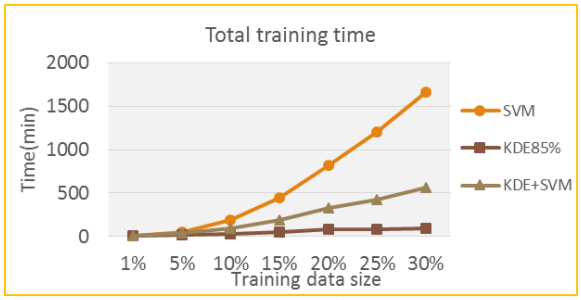


圖 3. 訓練時間

由於分類器的總類以及數量不同，測試資料時所花費的時間也是不相同的。我們使用總資料量的 20%作為測試資料集，計算各個模組的測試時間，結果如圖 4 所示，我們可以發現結果與訓練時間相差無幾，一樣是 SVM 花費最多時間，我們的方法次之，但由於測試與訓練不同，不會被正例個數影響所以差距沒有訓練時大，不過還是有一倍以上的差距，KDE 一樣遠低於其他方法。

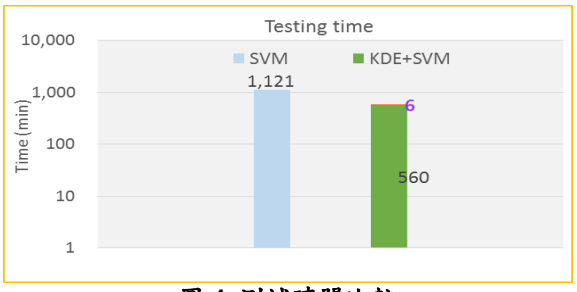
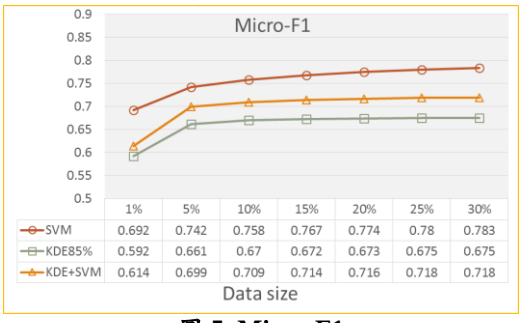


圖 4. 測試時間比較

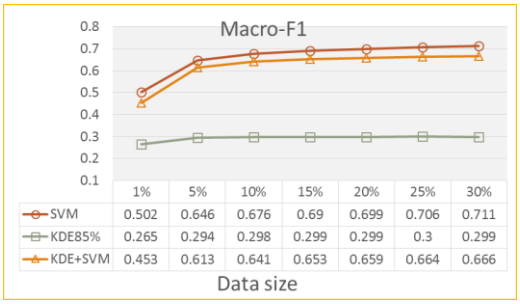
3) 各個方法效能結果

最後我們計算 SVM、KDE 的 85%特徵值以及我們結合兩種方法的混合模型的效能，分別計算 Micro-F1 和 Macro-F1，最終結果表示於圖 5 與圖 6，我們可以從圖中發現 SVM 的效能都是最好的，我們的方法次之，KDE 的結果最差，各方法 Micro-F1 的差距都不大，但 KDE 的 Macro-F1 值明顯與其他方法有著一段不小的落差，這是因為在 Macro-F1 的算法中所有類別的重要性是一致的而 KDE 的少量 POI 類別效能是極低的，這樣會造成 Macro-F1 的結果不佳。而我們的方法是利用 SVM 來分辨這些小類別使得 Macro-F1 不會被拉低。



	1%	5%	10%	15%	20%	25%	30%
SVM	0.692	0.742	0.758	0.767	0.774	0.78	0.783
KDE85%	0.592	0.661	0.67	0.672	0.673	0.675	0.675
KDE+SVM	0.614	0.699	0.709	0.714	0.716	0.718	0.718

圖 5. Micro-F1



	1%	5%	10%	15%	20%	25%	30%
SVM	0.502	0.646	0.676	0.69	0.699	0.706	0.711
KDE85%	0.265	0.294	0.298	0.299	0.299	0.3	0.299
KDE+SVM	0.453	0.613	0.641	0.653	0.659	0.664	0.666

圖 6. Macro-F1

4) SVM 的非平衡資料改善實驗

由於我們的資料是屬於非平衡資料，而 SVM 訓練的部分又是較少 POI 數的類別，不平衡的情況會更明顯，我們嘗試使用兩種方式來改善這個問題。第一種方式為改變訓練 SVM 時正例的權重，在實驗中我們訓練時嘗試使用 5 倍及 10 倍的正例權重。第二種方法則是採用縮減取樣的方式，縮減取樣是將負例的個數縮減成與正例數差不多再進行訓練。我們將原始 SVM 與兩種方式比較，比較結果表示與圖 7，從結果我們可以發現到兩種方式都沒有比原始的方法好，我們認為縮減取樣的方式雖然可以提升召回率，但精確率則會下降，在原始方法召回率本身就不低所以提升程度有限，所以精確率的下降對整體影響較大。

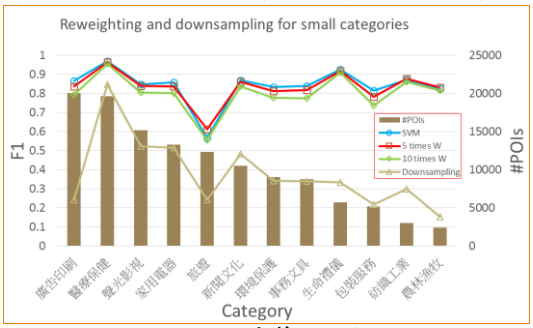


圖 7. 訓練策略比較

V. 結論

本論文比較三種不同訓練分類器的方式，分別是 SVM、KDE 以及我們提出的混合模型。就結果而言，SVM 的效果是最好的，我們的方法次之(約低 5-7%)，KDE 最差。各個方法的 Micro-F1 都有一點差距但差距沒有非常大，KDE 在 Macro-F1 的表現遠低於 SVM 及我們的方法。原因在於 KDE 對於比較稀有的類別表現結果很差才會使 Macro-F1 大幅度的下降，稀有類別在訓練 SVM 分類器並不會占用太多時間，因此結合 KDE 與 SVM 在整體的訓練時間還是比 SVM 快上許多。

由於 POI 的資料屬於非平衡資料，我們嘗試兩種方式分別是 Reweighting 以及縮減取樣來改善非平衡資料的效能，但是效能不如預期中有所提升反而有點下降。

REFERENCES

[1] C.-C. Chang and C.-J. Lin, “LIBSVM: a library for support vector machines,” ACM Trans. Intell. Syst. Technol., vol. 2(27), pp. 1–27, 2011.

[2] Q Chen, et al. “Improvement of Kernel Dependency Estimation and Case Study on Skewed Data.” National Central University, 2013

[3] Dunteman, George H. Principal components analysis. No. 69. Sage, 1989.

[4] Fan, Rong-En, Pai-Hsuen Chen, and Chih-Jen Lin. "Working set selection using second order information for training support vector machines." Journal of machine learning research 6.Dec (2005): 1889-1918.

[5] Fan, Rong-En, et al. "LIBLINEAR: A library for large linear classification." Journal of machine learning research 9.Aug (2008): 1871-1874.

[6] Godbole, Shantanu, and Sunita Sarawagi. "Discriminative methods for multi-labeled classification." Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer Berlin Heidelberg, 2004.

[7] Ma, Wei-Yun, and Keh-Jiann Chen. "Introduction to CKIP Chinese word segmentation system for the first international Chinese Word Segmentation Bakeoff." Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17. Association for Computational Linguistics, 2003.

[8] Tang, Lei, Suju Rajan, and Vijay K. Narayanan. "Large scale multi-label classification via metalabeler." Proceedings of the 18th international conference on World wide web. ACM, 2009.

[9] Wallace, Byron C., et al. "Class imbalance, redux." Data Mining (ICDM), 2011 IEEE 11th International Conference on. IEEE, 2011.

[10] Weston, Jason, et al. "Kernel dependency estimation." Advances in neural information processing systems. 2003.

[11] Yang, Yiming, and Jan O. Pedersen. "A comparative study on feature selection in text categorization." Icml. Vol. 97. 1997