**ML final project report**

**Transfer Learning on Stack Exchange Tags**

**指導教授：李宏毅 教授**

**隊名: NTU\_r05922148\_那大家先決定隊名吧**

**組員: 許正忞  陳力維  王冠驊  林家慶**

#1. Team name, members and your work division (5)

* **Introduction**

這個比賽提供了來自六個不同類別的Stack Exchange問題的標題、文本和標籤。我們欲對未知標籤的物理問題進行標籤預測。

* **Observe Real Tags**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Topic** | **Number of Questions** | **Number of Valid Tags** | **Number of 2-gram Tags** | **Number of 3-gram Tags** | **Average Number of Tags in One Question** | **Maximum Number of Tags in One Question** |
| biology | 13196 | 678 | 189 | 15 | 2.51 | 5 |
| cooking | 15404 | 736 | 177 | 7 | 2.31 | 5 |
| crypto | 10432 | 392 | 163 | 40 | 2.44 | 5 |
| diy | 25918 | 734 | 184 | 18 | 2.28 | 5 |
| robotics | 2771 | 231 | 65 | 3 | 2.35 | 5 |
| travel | 19279 | 1645 | 507 | 75 | 3.39 | 5 |
| physics | 81926 | ? | ? | ? | ? | ? |

在這6個類別的主題中，我們發現以下3點:

1. 每個主題中的只有數千個相異的tag
2. 在所有的tags中，2-grams集3-gram占了相大大部份的比例
3. 每的問題大多只有2個或3個tags，且最多不會超過5個，故我們可以針對這3點找出最適合每個問題的tags

* **Tools**
  + nltk.ngrams
  + nltk.pos\_tag
  + nltk.stem.porter
* **Preprocessing/Feature Engineering**

**Preprocessing**

* 移除Stopwords、punctuation

我們以NLTK內建的english stopword濾掉問題中不重要的字詞，並且過濾標點符號，只留下有意義的字詞，避免我們在選擇tag時誤選了stopword或標點符號。

* 移除 html標記和數學式子

與stopword相同，html標記和數學式子有極高機率不是一個tag，故我們在預處理時先把它們刪除。

3. Model Description (At least two different models) (5+5)

* **Methodolog**

**Frequent Words and Phrases**

**Stem**

**POS Tagging**

**Fine Tune based on Output**

**Tags**

Frequent Words and Phrases

* 在一個問題中，我們找出同時出現在title與content的字，並且猜測他可能是此問題的tag
* 在所有的問題中，我們將相鄰的2個字組成一個2-grams，將相鄰的3個字組成一個3-grams，。在所有的問題中統計這些2-grams 以及3-grams出現的頻率，如果出現的頻率夠高，我們即把它視為此問題的tag
* 對於3-grams的tag，剔除中包含2個以上的stopwords的3-grams例如:'up-to-date'，剔除'A-B-A'樣式的3-grams例如: 'step-by-step'、'one-to-one'。

POS Tagging

我們定義一個字的noun\_ratio為一個字有的問題中，被當作NN, NNS, NNP, NNPS的比例。

我們針對每一個字計算它的noun\_ratio，剔除比例低於0.5的字

Stem

同一個字在不同的句子可能會有不同的型態，例如: 'force' vs. 'forces'，'electromagnetics' vs. 'electromagnetism'，但在這些字在之中只有一個會是真正的tag

針對這個問題，若只是單、複數的問題我們統一以複數當做最後的輸出，其餘的我們則取擁有最高noun\_ratio的字當做這些字的最終輸出

Fine Tune based on Output

1. Consider n-grams in estimated tag set  
   若預測出來的tag包含2個字，且此2個字在其他的問題中曾以2-grams形式出現，則我們江子2個字以它的2-grams取代，例如: "relativity special " ⇒ "special-relativity "
2. LSA K-Means clustering

我們假設相似的問題會有相同的tags，故我們將問題分群，統計在同一群問題中最常出的tags，如果在該群的問題我們並沒有預測出該些tag，我們把該些tag加到那些問題最終的tag

* + Example in biology
    - Title: "Is exon order always preserved in splicing?"
    - Content: "Are there any cases in which the splicing machinery constructs an mRNA in which the exons are not in the 5' -> 3' genomic order? I'm interested any such cases, whether they involve constitutive or alternative splicing."
    - Estimated tags: "order"
    - Real tags: "splicing mrna spliceosome introns exons"

Cluster tags: [('exons', 56), ('introns', 56), ('mrna', 14), ('splice', 13), ('sequences', 10)]

1. 剔除出現頻低的tag (n < 25)
2. 若一個問題沒有被預測出任何的tag，我們將一些出現頻率高的tag給它當作最後答案
3. Experiments and Discussion (5+5)

Method3:Classifier

* 我們想從已經有tag的6個類別文章訓練一個NN，我們想立用這樣的一個NN使其在test的data同樣能準確預測出每個文題的tag。針對每一個問題裡的每一個字，給每一個字一些feature，例如tf，idf，是否出現在title，是否出現在context，出現在整個句子(title or context)的位置，POS tag。
* Features
* Term frequency

針對每一個問題，計算某一個字在這整個問題出現的次數。

* Inverse document frequency

若一個字詞只出現在某一個問題中，顯然這個字詞對於此問題是十分獨特、具代表性的。為了突顯一個字詞在一個問題中的獨特程度，我們考慮IDF。

* Position

Position表示這個字詞出現在句子的哪個位置(第幾個字/整個問題的字數)。

* In title/context

這個feature紀錄一個字詞是否出現在title或著是context，若是有出現則紀錄1，沒出現則紀錄0。

* POS tagging

POS tagging是依據字詞在句法結構或語言形態上扮演的角色，經由詞性分類賦予語句中每個字詞適當之詞性符號或標記的過程。此feature需要在尚未濾除stopwords之前執行。

* 在POS這個feature裡，我們本來想以one-hot encoding表示不同詞性，但是當時再存成pickle檔時，因為資料太大存，太久而做罷，最後我們改用正常編碼的方式給不同詞性不同的數字。
* 在實做的時候，這個方法的效果不是很好，主要有2個問題導致訓練成效有限，第一，資料量太大；每個問題中的一個字就會形成一筆資料，導致最終訓練時會有數460幾萬筆資料，訓練時間太長。第二個是抽取的feature量稍嫌不足；在這麼多字詞且feature只有6個情況下，可能無法完整呈現每個字的差異。經過檢討後得到以下改進方法:首先，我們可以將POS tag當做篩選的門檻，這樣可以大幅降低資料量以及提高預測效果。再者，我們可以再加入一些其他的feature，像是字母是否為大寫、是否被” “包覆…等等