**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**

**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**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **主題** | **問題數** | **相異標籤數** | **相異二連字標籤數** | **相異三連字標籤數** | **每題平均標籤數** | **每題最大標籤數** |
| biology | 13196 | 678 | 189 (28%) | 15 | 2.51 | 5 |
| cooking | 15404 | 736 | 177 (24%) | 7 | 2.31 | 5 |
| crypto | 10432 | 392 | 163 (42%) | 40 | 2.44 | 5 |
| diy | 25918 | 734 | 184 (25%) | 18 | 2.28 | 5 |
| robotics | 2771 | 231 | 65 (28%) | 3 | 2.35 | 5 |
| travel | 19279 | 1645 | 507 (31%) | 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. 每個主題只有數百個或一千多個相異的tags
2. 在所有的tags中，2-grams及3-grams占了不小的比例 (25%以上)
3. 每個問題大多只有2個或3個tags，且最多不會超過5個

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

**Methodology**

**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，並且猜測它可能是此問題的tag
* 在所有的問題中，我們將相鄰的2個字組成一個2-gram，將相鄰的3個字組成一個3-gram。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2-grams，剔除
  + 總長度< 5個字母(包含hyphen: '-')者
  + 'A-A'樣式者，例如: ''
* 對於3-grams，剔除
  + 總長度< 7個字母(包含hyphen: '-')者
  + 包含2個以上stopwords者，例如'up-to-date'
  + 'A-B-A'樣式者，例如 'step-by-step'、'one-to-one'

POS Tagging

在不刪去標點符號及stopwords的情況下，統計整份文本所有的字的詞性。定義一個字的noun\_ratio為一個字被當作名詞('NN', 'NNS', 'NNP', or '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

同一種字在不同的句子可能會有不同的型態，例如: 'force' vs. 'forces'，'electromagnetics' vs. 'electromagnetism'。這些字只有一個會是真正的tag。針對這個問題，我們將每個字還原成它們的詞根(stem)，將同一字根的字以下列兩個規則選出一個字來代表

* 若只是單複數的差異，統一以複數當做最後的輸出
* 其餘則取擁有最高noun\_ratio的字

用上述規則導向的方法所建立的tags的Kaggle score為0.09897，還沒辦法過strong baseline。

Estimate the probability of being a valid tag for each word (uni-gram)

我們針對每一個主題，各自建立每個字在該主題內的各項統計特性做為特徵，建立二元分類模型來預測每個字是否為valid tag。我們建立的特徵如下表所示：

|  |  |
| --- | --- |
| **欄位** | **描述** |
| word | 所有出現在某一主題的字(有被算過noun\_ratio者) |
| n\_occurence | 出現的總次數 |
| n\_title | 出現在幾篇標題 |
| n\_question | 出現在幾篇問題(標題或內容) |
| ratio\_capital | 字首大寫的比例 |
| ratio\_upper | 全部字母大寫的比例 |
| ratio\_before\_ques | 出現在問號之前的比例 |
| ratio\_noun | 被當作名詞的比例 |
| ratio\_around\_kw\_tag | 前後兩個字包含與'tag'同字根的字的比例 |
| ratio\_around\_kw\_understand | 前後兩個字包含與'understand'同字根的字的比例 |
| ratio\_around\_kw\_study | 前後兩個字包含與'study'同字根的字的比例 |
| ratio\_around\_kw\_introduction | 前後兩個字包含與'introduction'同字根的字的比例 |
| ratio\_around\_kw\_explain | 前後兩個字包含與'explain'同字根的字的比例 |
| ratio\_around\_kw\_principle | 前後兩個字包含與'principle'同字根的字的比例 |
| ratio\_around\_kw\_interpret | 前後兩個字包含與'interpret'同字根的字的比例 |
| ratio\_around\_kw\_difference | 前後兩個字包含與'difference'同字根的字的比例 |
| ratio\_around\_kw\_book | 前後兩個字包含與'book'同字根的字的比例 |
| ratio\_around\_kw\_knowledge | 前後兩個字包含與'knowledge'同字根的字的比例 |
| mean\_occur\_in\_title | 在標題出現的平均次數 |
| mean\_position\_in\_title | 在標題出現的平均位置 |
| ratio\_in\_title\_first | 出現在標題句首的比例 |
| ratio\_in\_title\_last | 出現在標題句尾的比例 |
| mean\_occur\_in\_content | 在內容出現的平均次數 |
| mean\_position\_in\_content | 在內容出現的平均位置 |
| ratio\_in\_content\_first | 出現在內容句首的比例 |
| ratio\_in\_content\_last | 出現在內容句尾的比例 |
| mean\_occur\_in\_question | 在一個問題當中(標題和內容)出現的平均次數 |
| is\_end\_with\_s | 是否以's'結尾 |
| is\_end\_with\_ing | 是否以'ing'結尾 |
| str\_length | 字串長度 |
| is\_tag | 是否為tag (只有physics以外的其他六項主題才有此欄，為模型的反應變數) |

我們以robotics所建立的data frame為testing data，以biology、cooking、diy、crypto四類所建立的data frame合併為training data (捨棄travel，因為此主題的tag統計特性與其他五類差異較大)。在建模之前，我們對於不是tag的字進行隨機取樣，使training data平衡 (balanced，tag與非tag的sample數一樣多)。我們採用的模型為隨機森林(random forest)，使用R語言的'randomForest' package，以預設的參數 (ntree = 500, mtry = floor(sqrt(ncol(x))) = 5，表示建立500顆決策樹，每個節點隨機考慮5個變數)以及training data建立二元分類模型，並驗證模型在testing data上的預測正確率 (accuracy)。重複上述步驟(平衡、建模、預測)10次，得到的平均正確率為86.08%，F1-score為。結果顯示上述特徵對於判斷一個字是否為tag確實是有用的。我們再利用randomForest package內建的importance函式，計算每個特徵對於正確率的重要性 (採用MeanDecreaseAccuracy，其意義為：打亂順序後會使得正確率下降多少，越多表示特徵越重要)，得到下表：

|  |  |  |
| --- | --- | --- |
| **排名** | **特徵** | **重要性** |
| 1 | ratio\_noun | 49.52 |
| 2 | n\_title | 47.45 |
| 3 | mean\_occur\_in\_question | 32.90 |
| 4 | ratio\_capital | 32.61 |
| 5 | ratio\_in\_title\_last | 28.82 |
| 6 | mean\_occur\_in\_content | 28.10 |
| 7 | ratio\_before\_ques | 27.75 |
| 8 | n\_occurence | 25.91 |
| 9 | ratio\_in\_title\_first | 24.02 |
| 10 | n\_question | 23.68 |
| 11 | mean\_position\_in\_title | 20.21 |
| 12 | str\_length | 18.97 |
| 13 | ratio\_in\_content\_first | 18.02 |
| 14 | ratio\_in\_content\_last | 17.41 |
| 15 | mean\_position\_in\_content | 13.20 |
| 16 | ratio\_upper | 8.67 |
| 17 | mean\_occur\_in\_title | 8.13 |
| 18 | is\_end\_with\_ing | 7.61 |
| 19 | is\_end\_with\_s | 5.93 |
| 20 | ratio\_around\_kw\_difference | 4.75 |
| 21 | ratio\_around\_kw\_study | 4.18 |
| 22 | ratio\_around\_kw\_explain | 3.47 |
| 23 | ratio\_around\_kw\_book | 2.77 |
| 24 | ratio\_around\_kw\_knowledge | 2.34 |
| 25 | ratio\_around\_kw\_understand | 1.84 |
| 26 | ratio\_around\_kw\_principle | 0.01 |
| 27 | ratio\_around\_kw\_introduction | -0.94 |
| 28 | ratio\_around\_kw\_tag | -1.36 |
| 29 | ratio\_around\_kw\_interpret | -3.13 |

從上表可以看出，除了較顯而易見的「名詞比例」、「出現在標題的次數(類似document frequency)」，以及「在一個問題當中(標題和內容)出現的平均次數 (即平均term frequency)」之外，還有一些有趣的特徵，例如「字首為大寫的比例」、「出現在標題句尾的比例」，以及「出現在問號前的比例」等等。另外，「出現在關鍵字附近」這樣的特徵比較沒有用，可能是因為我們僅考慮前後各兩個字做為鄰居來計算此特徵，因此大多數的字的這類特徵都很接近0，使得此類特徵比較沒有鑑別度。之後可以考慮擴大定義鄰居的window大小，或著甚至出現在同一句話裡面就視為鄰居。

由於用robotics為testing data得到不錯的預測正確率，我們就直接將此模型套用到physics類問題所建立的data frame上，並將預測的結果形式改為機率 (predict(…, type = "prob"))，預測出全部16834個字是valid tag的機率 (存成physics\_prob.csv)，用在前述方法的最後一步：將tag機率小於某個threshold的tag拿掉。我們試了幾種可能的threshold (0.5、0.7、0.75)，最後採用Kaggle分數較高的0.75做為threshold，分數達到了0.13802。

Fine Tune based on Output

最後，我們將上述方法所得到這份答案進行下列微調

1. Consider n-grams in estimated tag set  
   若某一個問題預測出來的tags的某兩個字，在其他的問題中曾以2-grams形式出現，則直接將這兩個字以它的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，故我們將所有問題分群。分群依據的特徵是LSA (TF-IDF + SVD)取200維，演算法則使用K-Means。我們將所有問題分成500群，並且重覆4次(每次都設定n\_init=10，取10次不同初始化之下最好的分群結果)。因此最終每個問題都會分別屬於4個不同的群。藉由統計每一個問題所屬分群的所有tags，找出出現頻率高的tags，將它們加入此問題的tags當中

* + 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."
    - Real tags: "splicing mrna spliceosome introns exons"
    - Estimated tags: "order"
    - Top-5 frequent cluster tags (數字代表出現總次數): [('exons', 56), ('introns', 56), ('mrna', 14), ('splice', 13), ('sequences', 10)]

從以上例子可以看出，原本對此問題只預測出'order'一個tag，經由分群之後，我們知道此問題所屬分群當中最常出現的tags有'exons'、'introns'、'mrna'等等，都是正確的答案。其中'introns'是完全沒有出現在此問題的標題或文本當中的字，也透過這個方法找到了。

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