Machine Learning --- Final Report

AI CUP 2019 - 新聞立場檢索技術獎金賽

Intent Retrieval from Online News

Team Name: KPCOF

電機三 B05901111 陳建成　　電機三 B05901192 張晁維

電機三 B05901040 蔡松達　　電機三 B05202061 陳威旭

Abstract

在機器學習的範疇之中，目前針對文字處理、分析的問題已經有各式各樣的模型被提出，像是從基本的RNN、LSTM，到Seq2Seq、TF-IDF、BERT不同種類的embedding方式都有其不同的成效，而這些模型提供出embedding的結果往往是透過前後文、語意、出現頻率所得到，對於後續不同的應用帶來不少便利。本實驗主要透過許多不同方法進行embedding，以方便找出與查詢題目(query)相關性最大的幾則新聞，並利用一些模型架構試圖將立場（支持或反對）判斷清楚，以期透過受限的資料訓練過後能夠得到相當的效果。

I Introduction & Motivation

本次project的題目是給定一個議題的名稱，其中大多帶有特定立場（支持或反對），希望能夠訓練出一個類似搜尋引擎的模型，幫助我們輸出與此議題關聯性由高到低、立場由較相同至較不同的新聞排序，此題目提供了100000則新聞連結，4742條某篇新聞與特定議題關聯性強弱的訓練資料，透過這些給定的資料進行模型訓練，期望能夠在輸入一個議題名稱時得到上述新聞排序的輸出。

在現在資訊爆炸、快速流通的時代中，各家傳播媒體在報導同樣的新聞時往往可以由不同的角度切入，由於不同媒體對於特定議題時常會有不同的觀點與看法，報導的面向往往會由不同的角度切入，尤其是針對一些具有較大爭議性的議題，例如經濟、教育、政治等等，同一個議題在不同的新聞媒體呈現出來時卻可能會有截然不同的立場跟說法。本題目希望能夠找出一個可以對於立場相關性排序的搜尋引擎，這樣的想法可以方便我們查詢特定議題中特定立場的理念與想法，不同的觀點、想法能夠促進大家對於特定爭議議題的整體了解，當自己抱持著相當鮮明的立場時，若是也能夠針對完全顛倒的立場進行探究，往往能對於該議題有更深的體會。此外，一般的搜尋引擎基本上不會明確將立場相對的新聞內容區分開來，輸入關鍵字搜尋時得到的結果往往是夾雜著不同立場的，若是能夠明確的找出立場相同的新聞內容，並將立場最相近的新聞排在搜尋結果的最前端，對於使用者將會變得更加方便，且對於研究特定議題也有相當大的助益。而我們這組對於文字處理也較有興趣，期望能夠運用所學來完成此題目。上述大致上為我們這組選擇此題目的主要動機。

Related Work

在進行模型實作之前，我們有經過許多討論與嘗試，對於不同的方法進行了實作。與此相關的論文如下。

1. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [1]

由於上課有提及BERT在NLP上面已經被人應用在許多不同的任務，並且得到了不錯的成效，因此我們嘗試使用pre-trained的BERT模型進行fine-tune，使用next sentence prediction(NSP)進行直接NN分類或是搭配其他方法訓練出一個判斷立場的權重。

2. Distributed Keyword Vector Representation for Document Categorization [2]

這篇提供的實驗條件與本次新聞立場檢索技術獎金賽頗為相近，但是由於其新聞資料分類已經事先有進行類別的區分，而並非我們這次是區分「立場」。雖然我們沒有直接實作其方法，但是此篇論文提供的「關鍵字」作為權重的概念我們將其實做在第二部分Data Representation中Explicit Embedding的keyword embedding，以及第三部分方法中的Keyword Weighted Word2Vec

Embedding，其成效明顯改善模型預測準確性。

3. Words are not Equal: Graded Weighting Model for building Composite Document Vectors [3]

這篇的方法也是我們最後使用的Keyword Weighted Word2Vec Embedding方法的靈感來源之一，不過由於論文本身的方式複雜不易實作，因此我們採用簡單的weighted方式取代。

II Data Preprocessing / Feature Engineering

我們對資料的處理主要分為兩個部分：Data Representation和Ranking Adjustment。前者實作方式有BOW(Bag of Word)、Explicit Embedding和Implicit Embedding，不同的Data Representation方式分別應用於第三部分的不同方法；後者實作方式則是對預測結果以訓練資料進行重新分群再排序。

一、Data Representation：

(一)BOW(Bag of Word)：

BOW方法為最基本的Embedding表示法之一，將新聞資料與查詢問題通過jieba.cut斷詞後進行one-hot encoding，並將其one-hot encoding結果加總，使一個查詢問題與一則新聞資料皆以一個約57萬維度的向量來表示，第N維的元素表示被one-hot encoded至該維度的詞在該則新聞資料或是該句查詢問題的出現次數。

(二)Explicit Embedding：

Explicit Embedding方法中Embedding model和之後的分類器是分開來訓練的，如使用gensim套件的Word2Vec embedding和Doc2Vec embedding，再分別經過不同的方法改良embedding的結果。兩者共通點為皆利用jieba套件進行中文斷詞後再進行embedding，斷詞辭典使用作業六提供的中文繁體辭典。

1. gensim Word2Vec embedding：

引用HW6以及許多RNN相關任務會使用的Gensim套件提供之Word2Vec，在實作時我們分別存下了400維與1500維兩種model，針對所有的文件（100000篇新聞資料 + 4742條訓練資料 + 20條查詢題目）作為corpus進行embedding的訓練（訓練12個iterations）。

(1) Average embedding：

這是一個naive method，將一篇文章斷詞後的所有字經過embedding後直接平均起來，依照其向量分佈的中心點進行比較，詞彙的分佈會使得最後中心偏向其主題與立場相關的方向。但其問題是「平均向量」的想法明顯會被無關字眼與stop words干擾。

(2) TFIDF weighted embedding：

為了改善詞彙的權重問題，加上實驗過程中使用TF-IDF BOW的優異表現，因此我們使用每個詞彙的TF-IDF值當作仲要性去加權word vectors[1]。此法表現確實有稍微改進前者的缺點，但問題是詞彙太多，在實作時記憶體經常會被用盡導致系統崩潰。

(3) keyword embedding：

依照前面論文與網站提及的方法，我們最後直接採用jieba.analyse裡面提供的pretrained weight當作每個詞彙重要性的代表，取出前N名重要的keyword進行加權平均。由於詞彙數量變少許多，執行的時間顯著減少，結果甚至超過所有嘗試過的方法。

2. gensim Doc2Vec embedding：

取得所有新聞資料後先用jieba.cut函式進行斷詞，再將每篇斷詞過後的新聞資料重組成gensim Doc2Vec模型需要的TaggedDocument物件，調整Doc2Vec需要的模型參數（min_count, vector_size等等）後對Doc2Vec模型進行訓練，再利用此模型取得每篇新聞資料與查詢題目的embedding vector。查詢題目的embedding vector取得方式為將斷詞後的查詢題目傳入訓練完畢的Doc2Vec模型的infer_vector() method，便會回傳該文章（句子）

[1] 參考的Colaboratory Notebook (https://colab.research.google.com/drive/1wetOy4UIRHBv_G3MDaMHUWiihctSQpTU)

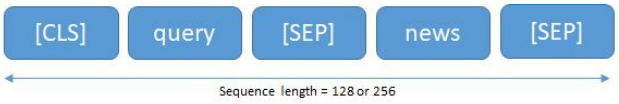
依照此Doc2Vec模型得到的embedding vector；而新聞資料的embedding vector則是呼叫Doc2Vec模型的物件docvecs，用index的方式得到該篇新聞資料的embedding vector。

(三) Implicit Embedding：

Implicit Embedding方法為將Embedding Vector直接接著輸入NN(Neural Network)或是其他分類器進行相關性，和其他Embedding方式有所區別故分開討論。此方法主要為使用BERT Pre-trained model[2]來實作Sentence Embedding，使用BERT Tokenizer 將查詢題目與新聞資料依照BERT模型輸入方式進行預處理[3]，採用BERT模型Pre-training任務中NSP方式得到每個查詢題目與新聞資料的Embedding Vector。我們認為BERT進行NSP的任務可以類比於訓練資料中查詢題目與新聞資料的相關性。

欲使用BERT模型進行NSP任務我們需要將查詢問題與新聞資料tokenize後串接成sequence length小於512的陣列，而實作上BERT Pretrained Model輸入的sequence length只要大於256我們的硬體都無法負荷，故我們輸入BERT的sequence length僅有嘗試總長度128與256，如下圖所示。

圖一：BERT模型輸入sequence示意圖



二、Ranking Adjustment：

經過一些方法（如第三部分所述）後我們得到各篇新聞資料對各個查詢題目的立場與相關性的排序，觀察查詢題目與訓練題目的相關性與相似程度之後發現可以利用訓練題目已知的相關性來改善我們對相同題目的排序結果。

實作方法有兩種，其一是直接讀取任一方法排序後的結果比較查詢題目與訓練題目的相似度，紀錄這些和訓練題目相似的查詢題目並且找出該訓練題目與訓練資料中新聞資料標記之相關性(Relevance)，取相關性大於0的新聞資料，將這些新聞資料依據相關性做降序排序後做為輸出結果，如有不足三百名的部分則是再加入原本查詢題目使用方法得到排序結果的前幾名補足三百名的排序結果，如果出現補足三百名的某則新聞資料已經由訓練資料依據相關性排入的情形，那麼該則新聞資料會被略過，轉而觀察下一順位的另一則新聞資料，直到補足三百名為止。

此種實作方是的優點是快速且方便實作，由於直接使用各查詢題目對各新聞資料的排序結果，可以很方便的接在各種搜尋方法後面進行結果修正。然而此方法的缺點也是顯而易見的，即不考慮先前方法所得到的排序結果依據為何便直接由訓練資料內相關性高的新聞資料取代，可能導致具有高相關性的資料被排序至低順位。

第二種實作方法則是取得全部新聞資料對各查詢題目排序的依據，即演算結果的相似度（cosine similarity等等），以及訓練資料中和查詢題目相似之訓練題目對應新聞資料與其相關性，對一相似訓練題目的查詢題目的所有排序新聞資料而言，如有出現在訓練題目內的新聞資料則將其演算結果的相似度加上訓練題目與該新聞資料的相似度，而如果該新聞在訓練資料中相似度為0則將其演算結果相似度乘以0.5，全部相似題目內的新聞資料相似度都重新處理過後再進行排序。

此方法利用訓練資料的相似度對預測結果做分群的，可以保留原本演算結果得到的相似度關係，如相似度判斷為較高的新聞資料加上其原本在訓練資料中對相同查詢題目的相似度仍可以維持其相似程度，而若是沒有出現在訓練資料中而演算結果為高相似程度的新聞資料在最後的排序中仍有1左右的排序相關性。

III Methods

此部分將會介紹我們曾經使用過並且較具有代表性的分析方法，配合不同的Data Representation有不同的Similarity Analysis方法；某些Similarity Analysis方法則不使用第二部分中提及的Data Representation方式，直接使用rule-based的方法對資料進行相關性的排序。此外也有使用不同方法進行ensemble的方式來得到更好的排序結果。

[2] BERT Pre-trained Model pytorch實現版本請參考VI-[5]，keras實現版本請參考VI-[6]。
[3] BERT模型架構與輸入資料預處理方式請參考 VI-[1]。

一、Doc2vec + cosine similiarity：

先用gensim同時找出每篇文章以及查詢題目的document vector，再用每篇文章的vector和查詢題目的vector做cosine similiarity，拿和各個查詢題目最相近的前300名出來輸出。

二、關鍵字搜尋法（Rule-based）：

針對文章中，所有出現的jieba分詞進行搜尋，假設有出現與查詢題目相同的特定的詞，就給那篇文章加分，此外也針對部分查詢題目進行限制，若是特定詞彙沒有出現就算得分很高也會將分數歸零，例如當查詢題目為「通姦應該除罪化」，但其實很多新聞時常報導有關於外遇的新聞，這些新聞往往與除罪化議題毫無關聯，因此必須設定若是沒有出現「除罪」詞彙時即便「通姦」出現多次，分數也將會歸零，以排除不具關聯性的新聞。最後統計出每則新聞對於特定的查詢題目所得的分數後，將前300高分的新聞排序出來，即為所求答案。

三、TF-IDF Bag of words by sklearn library[4]：

先經過jieba斷詞處理，接著使用scikit-learn套件的Vectorizer和TfidfTransformer，將原本的字串轉換成one-hot vector之後，再根據文件中每個單詞出現的詞頻（term frequency，TF）與反向文件頻率（inverse document frequency，IDF），得到權重過後的BOW向量（約10萬×57萬維的sparse matrix）。接著將Query Set(QS_1.csv)的每個query vector分別與news vector進行cosine similarity直接比對。

四、Average Word Embedding：

將所有news、TD與QS合併起來當做一個大corpus，經過jieba斷詞處理後使用Gensim的Word2Vec套件，embed到400維的向量上，然後將每個新聞資料出現文字的embedding vectors直接進行平均，得到10萬×400維的matrix。接著與前面相同，將Query Set的每個query vector分別與news vector進行cosine similarity直接比對。

五、Keyword Weighted Word2Vec Embedding：

將經過jieba斷詞後的新聞資料、訓練題目與查詢題目中的關鍵詞經過jieba.analyse.extract_tag函式取出前幾十個（或幾百個）關鍵詞，將每則新聞資料或是每個查詢題目僅用關鍵詞embedding vector表示，非關鍵詞的詞則會以十萬筆新聞資料訓練的Word2Vec模型的embedding vector mean作為embedding vector，將每則新聞資料與每個查詢題目以關鍵詞的weighted embedding vector和模型embedding vector mean組合而成，再根據各自的L2-norm正規化作為比較cosine similarity的embedding vector。

六、Binary Stance Classification：

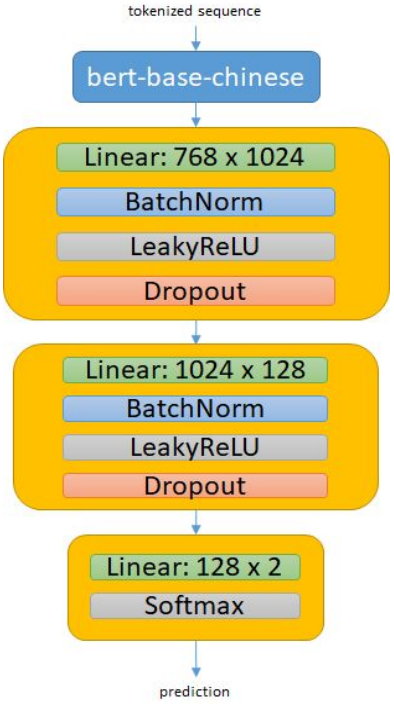
基於前敘方法多為考慮查詢題目與新聞資料的相似度為主的Similarity Analysis，然而題目闡述中提及內容相近和與查詢題目相反立場的新聞資料其relevance會標示為0，故以查詢題目和新聞資料相似度為主的分析方法考慮立場判別的有效性較相似度（cosine similarity等）較差，故另外建構一個立場的二元分類器，欲以此方法來彌補其他分析方法的缺點。

實作上Data Representation為Implicit Embedding方式，使用BERT Pre-trained Model的pytorch版本來實作，模型為BertModel的bert-base-chinese，輸入tokenized後的查詢題目與新聞資料（如上圖一），以模型輸出結果作為NN分類器的輸入，NN分類器輸出二維向量，第一維代表立場不同或無關的機率，第二維代表立場相同的機率，如下圖三所示。訓練用損失函數為交叉熵，訓練資料中查詢題目與新聞資料的立場以其相關性為區分標準，相關性非0者我們將此資料視為立場相同，即該查詢題目與新聞資料的立場標籤會被標示為1，相關性為0者我們將此資料視為立場不同，該查詢題目與新聞資料的立場標籤會被標示為0。

訓練完畢後將每筆新聞資料對每個查詢問題tokenize過後載入模型進行立場判斷，將預測結果向量第一維機率乘以0、第二維機率乘以1加總後得到每個查詢問題與每筆新聞資料的立場分數，再與其他分析方法結果相乘得到排序結果。

[4] 請參考VI-[4]。

圖三：Stance Classifier架構圖



七、Ensemble + Ranking Adjustment

為了得到最高的準確率，我們對表現最好的兩個模型：TF-IDF BOW與Weighted Keyword vectors預測出來的cosine similarity，先分別標準化到0~1之間，接著以平臺上得到出來的準確率當作權重進行ensemble之後，再將TD上面的Relevance值直接加在原本對100000篇新聞0~1的ratings上。如此一來便可以保證對於與TD相似或相同的查詢題目而言，TD中標示的相關性最為優先考量，其次再考量我們預測出來的ratings。至於Relevance 0的部分，我們將我們預測出來的ratings對這些新聞乘上0.1以使這些新聞很容易被排到最後面。

IV Experiment and Discussion

一、Doc2vec + cosine similiarity：

準確率：0.0124075

可能問題：Gensim使用的Doc2Vec演算法，對於每個document的embedding的時候容易造成嚴重震盪，因此無法得到穩定的embedding，進而無法在後續的比對上得到很好的效果。

二、關鍵字搜尋法：

準確率：0.1358650

可能問題：只有對於文本出現的關鍵字進行搜尋評分，完全沒有考慮到訓練資料，無法對於立場進行判斷，此外這裡所使用的詞彙僅限於查詢題目中有出現的詞彙，因此有些新聞如果使用相似詞則無法得到分數，也是可能的因素之一。

三、TF-IDF Bag of words by sklearn library：

準確率：0.1918345

可能問題：完全沒有考慮訓練資料，純粹就文件本身的資料特性去做unsupervised統計。另外，因為資料量實在太大，容易將工作站(Google Colaboratory)的14G RAM全部使用殆盡而導致工作站多次崩潰。

四、Average Word Embedding：

準確率：0.1646697

可能問題：一樣完全沒有考慮到訓練資料，word embedding的模型訓練也是unsupervised的。另外就是每個詞語的權重不應該是完全相同的。

五、Keyword Weighted Word2Vec Embedding：

此方法實驗經過不同Word2Vec模型的embedding size與取不同數量的關鍵字可以得到不同的結果，如下表一：

表一：Keyword Weighted Word2Vec Embedding不同參數實驗結果

Embedding size	keyword number	predict score
400	40	0.2106434
400	100	0.2106434
1500	100	0.2179311

比較上表實驗結果可以發現，Keyword Weighted Word2Vec Embedding的embedding size越大可以得到較好的predict score，keyword數量越多也對predict score有貢獻，最重要的keyword在只取40個keyword時便被取出來，對較小的新聞資料影響不大，keyword數量取較多的時候較能影響到將較長的新聞資料的embedding結果。

六、Binary Stance Classification with Other Method：

實驗此立場判斷方法套用在其他相似性分析方法的結果皆和原本方法表現較差，因為在訓練模型的時候，模型本身可能有過擬合(overfitting)的現象，比較原來使用cosine similarity的效果因此多少有些打折。我們分別對TF-IDF BOW與Keyword weighted Word2Vec進行了實驗，結果都較不使用此立場判斷方法差。

七、Ensemble + Ranking Adjustment

由於是所有目前找得到最有效方式的ensemble，因此得到了目前最好的準確率。

最後，對於各個方法的實驗結果比較表格如下表二：

表二：各種方法進行實驗後得到之準確率比較

Method	predict score
Doc2vec + cosine similiarity	0.0124075
關鍵字搜尋法 (Rule-Based)	0.1358650
TF-IDF Bag of words by sklearn library	0.1918345
Average Word Embedding	0.1646697
Keyword Weighted Word2Vec Embedding (best)	0.2179311
Binary Stance Classification + TF-IDF BOW by sklearn	0.1867411
Ensemble + Ranking Adjustment	0.3721916

V Conclusion

1. 只依賴統計特性進行相似性的比對，算是unsupervised learning的一種方式，但其找出來的分佈與訓練資料並不完全相同，因此只能得到有限的準確率。考慮訓練資料中查詢題目相近的部分去重新進行排序，可以明顯得到準確率上的改善。當然最好的方法還是使用semi-supervised的方法適當的將訓練資料進行generalization，但是此方法相當不容易，模型十分容易出現過擬合的狀況。

2. 除了BERT pretrained model之外，其他所有的方法都是單純的把新聞資料當成一個集合來看，並沒有考慮新聞資料中句子與詞彙的順序。但是因為其他如RNN autoencoder的嘗試並未得到更好的效果而先暫緩嘗試。之所以單純考慮詞彙分佈即可得到不錯結果的理由，應當是由於本次任務主題只是retrieval，因此可以由觀察詞彙分佈來得到不錯的結果。

VI Reference

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” *arXiv preprint arXiv:1810.04805* (2019)

[2] Yu-Lun Hsieh, et al. (2015) “Distributed Keyword Vector Representation for Document Categorization.” *2015 Conference on Technologies and Applications of Artificial Intelligence (TAAI), IEEE*. Retrieved from doi:10.1109/TAAI.2015.7407126

[3] Pranjal Singh, Amitabha Mukerjee. "Words are not Equal: Graded Weighting Model for building Composite Document Vectors." *arXiv preprint arXiv:1512.03549* (2015)

[4] Joachims, T. 1997. "A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization." *In Proceedings of ICML-97, 14th International Conference on Machine Learning (Nashville, US, 1997)*, pp. 143–151.

[5] pytorch-pretrained-bert library github:

https://github.com/huggingface/pytorch-pretrained-BERT

[6] keras-bert github: https://github.com/CyberZHG/keras-bert