Machine Learning Final Project - Intent Retrieval from Online News

R07922104林傳祐、B06902127鄭人愷
B06502149 張琦琛、R07922108陳鎰龍

- Introduction & Motivation

  具爭議性議題的新聞一直是閱聽人關注與討論的焦點。不論是政治、經濟、教育、兩性、能源、環保等公共議題，新聞媒體常需報導不同的立場。若能從大量的新聞文件裡，快速搜尋各種爭議性議題中具特定立場的新聞，不但有助於人們理解不同的立場對這些議題的認知與價值觀，對制定決策的過程而言，也相當有參考價值。

  近年來，隨著機器學習與深度學習等技術在各領域上蓬勃發展，在大數據的前提下處理各種需求也已經展現出比傳統方法更優秀的效能，而自然語言處理(NLP)便是其中一大領域，對於分析新聞議題便能藉由NLP建立語言模型，讓電腦學會從訓練的資料中，歸納出語言的特性，透過詞向量、語意分析等技術使搜尋新聞變得更加準確。

  我們參加此競賽的動機，是因為我們在作業中有使用過RNN的技術，來作文本的分析。我們認為在這個議題上，也是可以透過類似的方式，來找出各個新聞的立場與議題，並且探討傳統的技術TF-IDF，與現在流行的Machine Learning技術(如RNN、BERT)，何者能在此競賽獲得較好的成績。

- Data Preprocessing & Feature Engineering
  1. TF-IDF
     a. JIEBA斷詞：Term Frequency(TF)與Inverse Document Frequency(IDF)的計算，都是以"詞"為單位，因此我們必須先用JIEBA幫助我們將每篇新聞進行斷詞。我們有load我們自己的辭典因為在Query中有些詞jieba沒有辦法準確的進行斷詞，此外我們也用jieba.analyse中的set_stop_words，載入一些中文的停止詞，避免讓這些停止詞影響我們最後的計算結果。
     b. Word2Vector：將每個詞都賦予一個向量，就可以知道與它相似的詞有哪些，當我們計算一個詞的TF-IDF時，也可以同時計算它的相似詞的TF-IDF。
  2. BERT
     a. 斷句：Pretrain好的BERT，只接受512個字的輸入，一篇新聞通常都大於這個數字，因此要將新聞斷成很多句子。
     b. 賦予label：官方給的label非常地少，並且只有q_16~q_20的。這裡選擇label的方式是，利用TF-IDF輸出的結果，每個query的300篇文章的label就是此query。

- Method
    1. TF-IDF+Word2Vec
        a. Term Frequency(TF)：表示一個詞(Term)出現在一篇文章的頻率。
        b. Inverse Document Frequency(IDF) ：表示某個詞對應的文章數目，若一個詞在每篇文章中都有出現，如一些語助詞或是無意的連結詞，這些對於判斷並無任何幫助，但若只出現在特定文章，例如一些專業術語，則會提供較大的信息量。
        c. TF-IDF：將上述兩項乘積做比較，便可得到每個詞的TF-IDF。給定一系列關鍵詞(標題)，便可以透過計算文章中含有每個關鍵詞的比例，以及每個關鍵詞提供的信息量比例，找出與這些關鍵詞最相近的文章。
        d. 我們利用pretrained好的word2vec model，用most_similar找出每個斷詞後的query中，與那些詞相似的前N個詞，加進query中，也就是增加query中的關鍵詞數量，再做TF-IDF的計算。因為原始query中的關鍵詞數量太少，會導致對此query有相關的文章，被我們的算法認為是不相關的。
    2. BERT
        a. 將每篇news根據標點符號分句，利用BERT將句子與Query皆轉成向量，再與Query做cosine similarity，由於一篇文章有很多句子，我們將每個句子與Query的cosine similarity加起來後再除以這篇文章的句子數量當作此文章對於某個Query的分數，最後排序取前300大的當作結果。
        b. 與上面的方始類似，但是改成直接將整篇文章丟進BERT，計算文章與Query向量的cosine similarity，一樣排序前300大的當作結果。
        c. 利用TF-IDF的結果，配合前面提到的data preprocessing，來Train一個Bert Classifier。
        比較不同的是Input為一篇文章，而非一個句子。Output則為q_01~q_20其中一個。
        d. 與上面的方式類似，但是Input改成句子，但這樣一篇文章內其實不會所有句子都是與query相關的，因此此處還要先利用a.的方法，找出比較好的句子，才丟進Model，Train一個句子的Classifier。最終輸出則是由這篇文章的句子投票，看哪個query多，Output就是那個query。
    3. Word2vec+word mover’s distance
        a. 由於Query的字數太少，我們將每個Query的句子重複增加直到長度大致與第i個斷詞後的news相等。
        b. 利用gensim.Word2Vec中的wmdistance，計算出Query對十萬篇news的詞移距離，排序後取前300小的文章當作結果。

- Experiment & Discussion
    1. TF-IDF

Query	News_01	Score
性交易,TF=0.3 應該,TF=0.3 合法化,TF=0.3	性交易:TF=0.1,IDF=10 應該:TF=0.2,IDF=1 合法化TF=0.2,IDF=3	0.3*0.1*10+ 0.3*0.2*1+ 0.3*0.2*3
同意,TF=0.3 動物,TF=0.3 實驗,TF=0.3	同意:TF=0.1,IDF=3 動物:TF=0,IDF=3 實驗:TF=0,IDF=3	0.3*0.1*3+ 0.3*0*3+ 0.3*0*3

Table. 1

此為最簡單的方法，利用前面介紹的方法，計算三個值，分別為Query每個詞的TF、News每個詞的TF、News每個詞的IDF。計算完後，在Query看到一個詞時，就將這個詞的上述三個值相乘，就會得到這個詞在這個Query與這篇News的分數，再將這個Query有的所有詞的分數加總，即為這個Query與這篇News的分數。就這樣計算完各個Query對於各篇News的分數後，輸出前300名即為解答。

以Table 1.為例，我們就能看出News_01跟"性交易合法化"是比較有關係的，而與"同意動物實驗"是幾乎沒關係的。

這裡我有犯了一個主要的錯誤是，Jieba斷詞並沒有載入辭典，因此切出來的詞並不好，對結果會有影響。

但這個方法仍有0.1753940的分數，顯示TF-IDF方法其實是很強的，因此改善切出來的詞，應該能使結果更好。

2. TF-IDF + Word2Vector

Figure 1.

而適當的利用TD.csv所提供的資訊也可以有效地提高準確度。在query相同的情況下，我嘗試把TD.csv中query的結果作為QS_1.csv中query的答案丟進去，剩餘的新聞隨機選取，準確度可達0.1662742。在之前TF-IDF的結果中導入用TD.csv進行的後處理也可使準確度上升高達約0.08。可見利用TD.csv對輸出的結果進行後處理可有效提高搜索的準確度。

透過Word2vec，可以找出相似詞來實作Query Expansion。一開始，我對query做完斷詞所得出的每一個詞用Word2vec找出前三相似的詞，再用這些詞實作TF-IDF的運算，準確度可從0.3068106上升至0.3246627，可見用Word2vec可以彌補TF-IDF方法。會導致此結果的原因可能為：

即便是語意一樣的詞，但只要是不同的詞就無法被計入TF-IDF的結果中。

藉由IDF，還能夠把query中對主題影響較小的字剔除。比如說「是」、「的」這些過於常出現的詞較不易成為判斷主旨的要素。通過只留下IDF前三大的詞可以將準確度由0.3246627上升至0.3367950。

此方法的流程如 Figure 1. 所示。

3. Word2vec+word mover's distance

   這個方法最後得出的分數為0.1494345，我認為比我們預期的分數高，因為整篇文章中，真正的關鍵詞並不多，儘管已經去除掉停止詞及特殊標點符號，仍有許多冗言及不相關的文字，例如網址、某某報社等等，因此若拿整篇文章對query計算word mover's distance，此距離的大小將會因為一些不重要的詞嚴重影響最後的結果。而且我認為query的句子長度及文章的長度嚴重不一致也是導致這個方法失敗的重要因素，雖然有將query padding成與文章長度大致相等，但是若在幾何空間中來看，這種重複增加query中的句子，例如'abc' → 'abcabc'，來增加長度的方法，不過只是在空間中增加了重複的點而已，與沒有padding差別不大，甚至更差。因此我認為不論用哪種方法padding，結果差異不大，綜合以上兩個難以解決的問題，我認為這種方法必須再搭配一些類似TDIDF等提取關鍵詞的技術，分數才能有效提升。

4. BERT

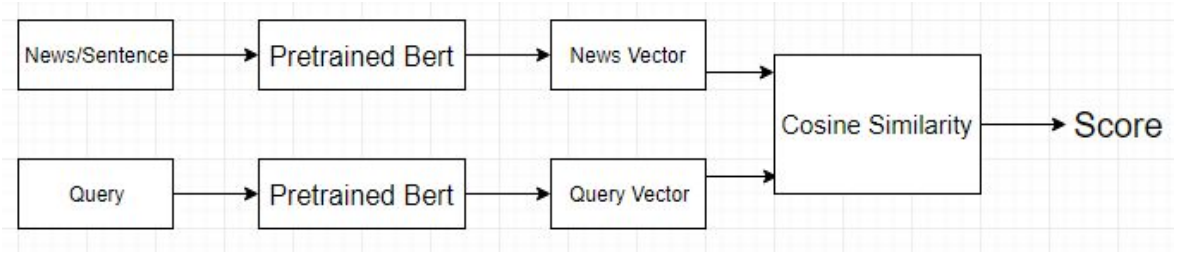

Figure 2.

   a. 若單純將每篇news根據標點符號分句，利用BERT將句子轉成向量，再與Query做cosine similarity，將所有cosine similarity加起來後取平均當作此文章的分數，最後在平台上得到的分數非常悽慘。

      再嘗試分析失敗原因後，我發現用BERT將句子轉為向量的結果並不是很好，明明是兩句相反意思的句子或是詞，或者是毫無相關的句子或詞，得到的cosine similarity竟然都可以到0.8以上，只有再兩個句子的長度落差過大時，才有明顯差異。

      另外一個可能的問題是，Bert內有非常多的hidden layer，要挑哪一層來當作我們需要的向量，也是很大的問題，或許是沒有挑到真正需要的hidden layer，所以導致結果很差。

      此方法的流程如 Figure 2. 所示。

   b. 將整篇news丟進Bert產生了很多問題。也得到不好的結果。

第一個是，Pretrained好的Bert，Input最大只有512維，而在Bert中，一個字就是一維，但一篇News通常都不只512個字，因此不能整篇文章丟進Bert。
第二個是，若將文章裁減至512個字後再丟進Bert，會產生與上面相同的問題。
此方法的流程如Figure 2. 所示。

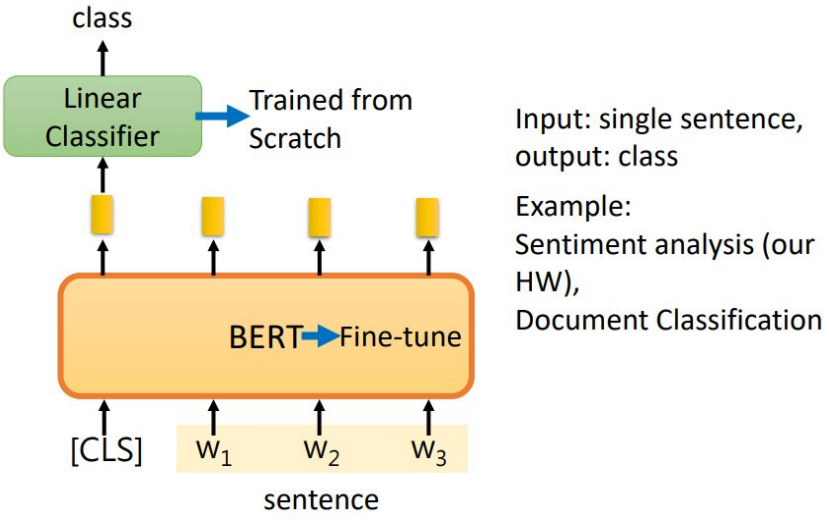

Figure 3.

c. Bert Classifier是我們有個pretrain好的bert，加上一個沒有train過的linear classifier，此處利用文章-Query來進行traning，輸入文章，輸出它屬於哪個query。
但很不幸的，這個Model完全Train不起來，accuracy始終是0。
仔細想想，依據Bert的概念，它應該是以句子為單位的，因此整篇News對它來說，可能太龐大了，也有可能前後文不是那麼相館，會對它造成影響。
此方法如 Figure 3. 所示。

d. 因為有了上面的經驗，所以決定再嘗試以句子為輸入。並且在句子的挑選上，先去除掉一些跟query較沒有關係的。
這個Model的accuracy不再是0，但經過很多Epoch後，accuracy竟不停地下降，Validation也始終是0。
經過多次的調整後，還是無法以這個方法成功train出一個classifier，Bert有關的方法就在此結束。
此方法如 Figure 3. 所示。

- Conclusion

延續在Introduction所提到的，我們希望能比較傳統的TF-IDF方法，與近年熱門的機器學習方法Bert，在這個競賽上的表現。
我們本來預期，強大的Bert能在這個競賽上"也"有精采的表現，但依結果來看，我們並沒有如願。這次競賽的幾個特點可能是造成Bert無法發揮實力的原因，像是有label的資料太少、新聞內容可能會夾雜很多髒的資訊導致Model失靈、

結果需要考慮立場等。我們花了很多心力與時間在使用不同的輸出輸入方式來Train Bert，卻無法有好的結果，我們覺得相當可惜。
至於傳統的TF-IDF，以結果來看，讓我們知道這個方法仍然很強的，最簡單的使用此方法就能通過simple baseline。但也不是說機器學習的方法就沒有用，因為我們利用了Word2Vec來找出相似詞，並利用它們之間的關係，再來算與原本TF-IDF稍微不一樣的分數，幫助我們通過Strong baseline，而Word2Vec就是機器學習的一個技巧。
綜上所述，在此競賽中，我們學習到了TF-IDF的技巧，也學習到了如何使用Bert這個厲害的Model，不管是傳統的方法還是較新的技術，都有他們厲害的地方。而我們也同時將兩者一起使用，做出更好的結果。

- Reference
https://radimrehurek.com/gensim/models/tfidfmodel.html
https://github.com/google-research/bert
http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML19.html
https://github.com/fxsjy/jieba
https://github.com/huggingface/pytorch-pretrained-BERT?fbclid=IwAR0Kg3vlAD5245OxrdC64M330ZiQTfCMMCGoYw2eW1BxF6bLYroBM8jegNc
https://radimrehurek.com/gensim/index.html