Machine Learning Final Project - Proposal

隊名：NTU_b05901063_39得第一4
隊員：黃世丞 王鈺能 楊采綸
題目：AI CUP 2019 - 新聞立場檢索技術獎金賽

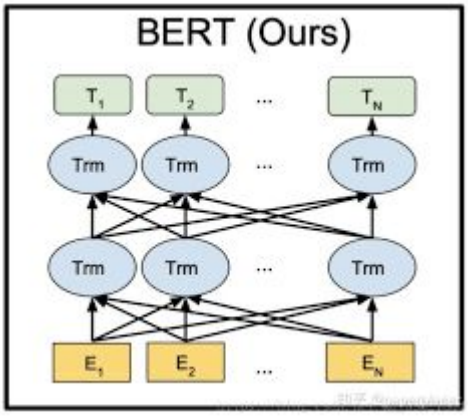
1. Paper Study

TF-IDF

概念十分直覺，給定一個詞，他和一個文本的相關程度會正比於該詞在文本內出現的次數，例如阿扁保外就醫的文章中，"阿扁"可能出現很多次、"鱷魚"可能就不會出現；反比於在整個corpus出現的次數，例如"的"雖然在文章中都出現很多次，但其實只是因為它是每篇文章都會用到的字，因此和文本的相關性反而不高。

BERT

Google所提出的BERT模型一推出就在NLP領域全面刷新紀錄，使用transformer來提取語彙特徵也比LSTM效果要好。BERT的模型架構龐大，參數量也非常多，因此我們比較難以原論文所提出的方法，改造BERT的輸入以執行下游任務，幸好Google不但開源了中文版的BERT，甚至有人寫了bert-as-service，可以簡單快速的呼叫pre-trained的BERT模型進行embedding。BERT在抽取文本特徵方面的能力十分強大，但是在本題中除了分析文本相似度外，還要考慮立場是否相同，BERT模型本身是沒有辦法判斷語意的，除非加上classifier並進行fine-tune，但我們缺乏訓練資料，GPU也不夠強大，因此這次只使用對文本進行embedding的部分。



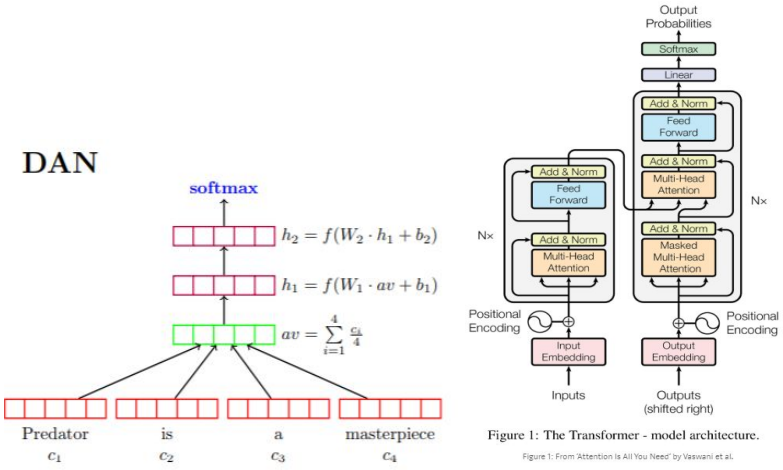
Google Universal Setence Encoder

原本是想要拿來嘗試不同embedding的效果，可惜google還沒提供中文的版本所以最後沒有使用，以下簡單介紹概念。

Universal Sentence Encoder可以用來判斷語意的相似度，這是另一個很強大的NLP模型Bert比較不容易做到的部分

考慮到準確度、訓練複雜度及資源消耗，Universal Sentence Encode的model有兩個架構，一個是Deep Average Network(DAN)，模型較為簡單，由

input的word vector求和平均再通過幾層非線性層，準確率較低但訓練較快；另一個是Transfomer，一種升級版的Seq2Seq模型，但encoder和decoder都只有用到Attention而無RNN(改由positional encoding來提供input的位置訊息)，準確率高但訓練較慢。



圖左：DAN 圖右：Transformer

因為Universal Sentence Encoder是multi-task learning，可以讓模型夠general而不會過於fit在訓練資料上，隨意input一些句子他都能有效抓出相似的pair。

2. Proposed Method

a. Preprocess

我們發現dataset中很多新聞內容遊戲、賭盤結果、廣告等等，因此我們將助教提供的文本中長度小於30或是符合特定patter(例如：開頭是"掌握即時戰報")從dataset中去除。

此外，除了文本內容之外，我們認為標題可能也對判斷立場有幫助，因此我們把標題和文本用jieba分詞的結果，作為後面training時的feature使用。

b. Models

- **RNN**

  我們嘗試了三種架構：(1) 先將news內文丟進去gensim word2vec train好embedding，再將query和title接起來當作input丟進由兩層Bi-LSTM加一層DNN的model中(如下圖)，不過結果會有某種規律，每個query的output會大致相同，且都只包含training data中的特定新聞。

```
    model = Sequential()
    model.add(Embedding(embedding_matrix.shape[0], embedding_dim, input_length=max_time_steps, weights=
[embedding_matrix], trainable=False))
    model.add(Bidirectional(LSTM(embedding_dim, return_sequences=True, unit_forget_bias=False, dropout=dropout_rate,
recurrent_dropout=dropout_rate), merge_mode='sum'))
    model.add(Bidirectional(LSTM(embedding_dim, return_sequences=True, unit_forget_bias=False, dropout=dropout_rate,
recurrent_dropout=dropout_rate), merge_mode='sum'))
    model.add(Attention())
    model.add(Dense(256))
    model.add(LeakyReLU(0.2))
    model.add(Dense(1))
    model.add(Activation('sigmoid'))
```

  (2) 我們將model改為由兩個小model並聯的形式(如下圖)，並把層數減少、LSTM改為GRU，結果會呈現另一種規律

```
    query_input = Input(shape=(max_time_steps,))
    query_model = Embedding(embedding_matrix.shape[0], embedding_dim, input_length=max_time_steps, weights=
[embedding_matrix], trainable=False)(query_input)
    query_model = Bidirectional(GRU(embedding_dim, return_sequences=True, dropout=dropout_rate,
recurrent_dropout=dropout_rate), merge_mode='sum')(query_model)
    query_model = Attention()(query_model)

    news_input = Input(shape=(max_time_steps,))
    news_model = Embedding(embedding_matrix.shape[0], embedding_dim, input_length=max_time_steps, weights=
[embedding_matrix], trainable=False)(news_input)
    news_model = Bidirectional(GRU(embedding_dim, return_sequences=True, dropout=dropout_rate,
recurrent_dropout=dropout_rate), merge_mode='sum')(news_model)
    news_model = Attention()(news_model)
```

(3) 最後覺得標題可能不太夠，嘗試加入內文一起train，model改為三個小model並聯(增加以下model)，結果也不如預期

```
    context_input = Input(shape=(context_time_steps,))
    context_model = Embedding(embedding_matrix.shape[0], embedding_dim, input_length=context_time_steps, weights=
[embedding_matrix], trainable=False)(context_input)
    context_model = Bidirectional(GRU(embedding_dim, return_sequences=True, dropout=dropout_rate,
recurrent_dropout=dropout_rate), merge_mode='sum')(context_model)
    context_model = Attention()(context_model)
```

經過以上嘗試，發現這個任務使用end to end model的話會train不起來，可能是因為官方給的training data太少而testing又必須在十萬個新聞中找到好的，硬用這些data訓練會讓model自己學到很奇怪的東西，表現也完全比不上直接用word embedding去算相似度。

- **Doc2vec**

  使用gensim的Doc2vec計算新聞內文和query的相似度，作法為將preprocess後的文本作為training corpus丟入doc2vec model，訓練完後將query做分詞到model中計算前300相似的文本輸出。

  此作法相較前一作法有改善(但仍然很差)，觀察其輸出發現的確部分query有抓到相關的文章，但大部分都抓到的內文都是很短的文本(很短的文本剛好幾乎都是廣告、樂透等等，因此後來才決定preprocess時把長度太短的刪掉)，結果仍不夠好。

- **BERT**

  BERT抽取文本特徵的能力果然十分強大，光用BERT直接對新聞內文和query做embedding再以兩者的cosine similarity當作排序標準，就讓我們的分數飆升一萬倍。但是把前三百名的新聞內文印出來檢視後會發現，雖然內容類似，但有許多立場相反或是詞彙類似但在講不同件事的新聞也會被排在前面。

- **TF-IDF**

  使用gensim的TfidfModel計算新聞內文和query的相似度

c. How to improve

- **N-gram TF-IDF**

  將"詞"的單位改成"N個詞"，這樣可以考慮到前後文的語意，或許對於搜尋更有幫助，需要注意的是OOV的狀況可能會更明顯，如何back-up要仔細考慮。

- **Combining TF-IDF with pre-trained Word embedding**

用TF-IDF的分數乘上每個詞彙的word embedding，flatten之後再算cosine similarity

- **Information Retrieval**
  https://www.csie.ntu.edu.tw/~pjcheng/course/wm2019/resources.html

3. Reference

[1] Bert: https://arxiv.org/pdf/1810.04805.pdf
[2] Universal Sentence Encoder: https://arxiv.org/pdf/1803.11175.pdf
[3] TF-IDF: https://arxiv.org/pdf/1607.01759.pdf
[4] Gensim Word2vec: https://radimrehurek.com/gensim/models/word2vec.html
[5] Gensim Doc2vec: https://radimrehurek.com/gensim/models/doc2vec.html
[6] Gensim tfidfmodel: https://radimrehurek.com/gensim/models/tfidfmodel.html