國立臺灣師範大學資訊工程學系

專題報告

笑話自動蒐集及預測系統

專題指導教授　　柯佳伶教授

專題製作參與　　蒲立年　40447025S

李柏徹　40447038S

劉秝瑋　40447044S

中華民國 108 年 1 月 13 日

(一)摘要

在人工智慧蓬勃發展的世代裡，機器學習已成為現在科技的先驅。本研究以研究讓電腦自動判斷短文幽默程度為目標，將機器學習方法應用在笑話跟一般文章，希望讓電腦在學習一定數量關於笑話及非笑話的資料後，建立出自己判斷短文幽默性的預測模型，藉此讓電腦能從網際網路中自動蒐集幽默的文章，紓解使用者生活壓力。

(二)研究動機與研究問題

研究動機

「幽默」可以是肢體、語言(文字)、行為上的表現，它也是人們在日常交流中常用的互動技巧，其中語言(文字)上的幽默頗為常見，像是講出令人覺得好笑的笑話。笑話可幫助交際上良好的互動，在人工智慧日益發展的世代中，人們想要機器能夠像人一樣有自然語言溝通的能力，幽默感是其中一項值得研究的項目。在網路上有許多笑話內容，可提供人們紓解生活壓力。不過一段短文是不是笑話，且其好笑的程度不盡相同。若能使電腦判斷一個笑話好笑的要素及好笑的程度，就能應用在蒐集網際網路中的笑話進行自動分析，建立出自己判斷短文幽默性的預測模型。因此我們希望透過此研究，運用機器學習及深度學習技術，嘗試讓電腦建立幽默預測模型，判斷笑話的好笑程度，未來這個技術可用於電腦更具人性化互動的發展和邁進。

研究問題

本研究將專注研究中文笑話，建立判斷笑話與非笑話的分類模型，並計畫嘗試表達短文的幽默程度，如何抽取出笑話的重要特徵或語意概念的表示，建立一個可判斷文章內容好笑與否的分類預測模型，是本研究要深入探討的部分。

(三)文獻回顧與探討

幽默是人們生活中的調劑，為了理解幽默，Hay 曾在 1995 年提出一些關於幽默的分類：軼事、幻想、侮辱、反諷、自嘲、俗氣、文字遊戲等[2]。而到 2006 年 Rada Mihalcea 和 Carlo Strapparava 在論文中指出在哲學、語言學、甚至心理學的領域一直有嘗試對「幽默」做解釋，但仍是個眾人困惑的概念。然而各種在電腦辨識幽默的實驗上，已有顯著的資料來證明機器有能力自動辨識出幽默以及非幽默，而他們的論文也提到大部分的統計資料主要以 one-liners 的案例來做統計[3]。

關於如何讓電腦學習人的幽默，在 2012 年 Yishay Riaz 的 Automatic Humor Classification on Twitter 一文中，整理出一些從英文文章中擷取出特徵，用以判斷笑話的方式，如句法、詞彙、型態、音韻、文體等，希望能夠藉由對這些特徵進行分類學習，讓電腦懂得人類的幽默[4]。

除 Yu-Hsiang Huang 等人，Peter Potash, Alexey Romanov, 及 Anna Rumsh-isky

亦投入相關的研究，對於已給定主題標籤的回應，研究如何進行推文的幽默程度預測。在研究中他們描述這個新資料集的半自動蒐集過程，以及使用非監督式方法和監督式方法對該資料集進行初始實驗。並認為以字母層級建立模型比字層級的模型更適合解決這個問題，這可能是由於英文大量的雙關語要採用字母層級的特徵模型才能表達[5]。

孫瑛澤和陳建良等人所提出的中文短句情緒分類方法，提到不少可判斷詞彙跟詞彙、詞彙及類別間相依關係的演算法，例如 point-wise mutual informat-ion(PMI)、naive bayes classifier、CKIP_SVM 法等，有助於取出關鍵詞彙進行笑話分類[8]。而蔡惠萍和王麗丹等人採用 word embedding 和卷積神經網路(CNN)來將情感分類的研究，更是搜集到大量資料後可以嘗試效仿的深度學習方法[9]。

除前列所述關於判斷幽默的方法，Donghai Zhang 等人採用一個新的概念：情境知識，他們以人為中心的角度設計兩種特徵來捕捉樣本中的情感和主觀性，可使幽默認知判斷的準確性大幅提高[10]。

Quoc Le, Tomas Mikolov 曾提出一種叫 paragraph vector(PV)的文本表示方法，在 word2vec 中從原本以單詞之間的關聯程度，加入了文本與單詞的關係去預測單詞，即新增了文本與單詞間的關聯程度來訓練向量[13]

(四)研究方法與步驟

根據本研究探討的問題，將進行以下工作：

(1)網路資料蒐集

(2)資料前處理

(3)Pre-trained word embeddings 詞向量預訓練

(4)Keras LSTM & RNN 與實作成果

(5)Validation method

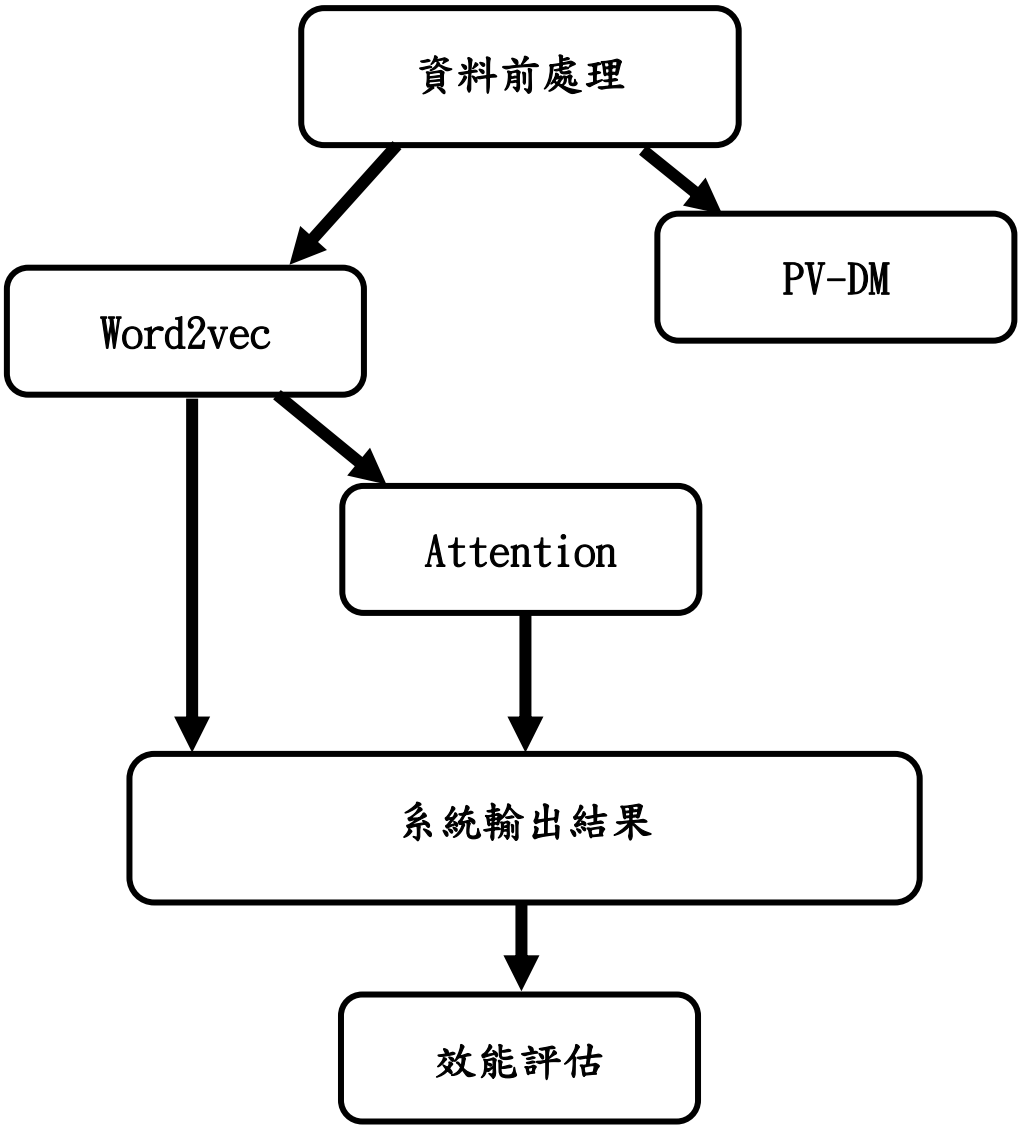
(5)Distributed Memory of Paragraph Vector(PV-DM)

(6)Attention

(1)網路資料蒐集

本計畫以 Python 網路爬蟲來進行資料的收集，利用 Python 裡頭的 BeautifulSoup 及 Request 套件來抓取笑話及非笑話資料。目前抓取的網站包括「ZOL 笑話大全」、「巴哈姆特」、「痞克邦」、「學習電子報」等網站，其中「ZOL 笑話大全」為檢體網站，因此我們在爬蟲時使用 openCC 來進行簡轉繁的處理。

以下是我們這學期的工作流程。下方圖一為流程規劃圖。



圖一 系統工作流程圖

(2)資料前處理

我們的資料分為笑話及非笑話，並且總共有分兩組訓練集。資料集會先採取斷詞、排除標點符號、每滿 16 詞或遇到結束的標點符號進行截斷，最後在句子前加上標記，1 代表笑話，0 代表非笑話，目的為了在訓練時以句子為單位監督學習。

第一個資料集來源是「zol 笑話大全」網站的笑話文章以及維基語料庫的非笑話句子，標記為 1 的共 36044 句，標記為 0 的共 37539 句。

第二個資料集來源是「zol 笑話大全」網站的笑話文章以及「學習電子報」網站，總共的句子有 64041 句，標記 0 和 1 的句子數大約各占一半。

5	0	我們 要 與 你們 同 去
6	0	因為 我們 聽見 上帝 與 你們 同 在 了
7	0	甚至 馬 的 鈴鐺 上 也 刻有
8	0	林肯 的 演講 於 當天 第二 順位 發表
9	0	以 不足 三百 字 的 字數
0	1	她 因爲 別人 都 不 同情 你 我 才 做 你 的 妻子 他 你 總算
1	1	成功 了 現在 每個 人 都 因此 同情 我
2	1	女 爲什麼 從前 你 對 我 百依百順 可 結婚 才 三天 你 就 跟 我 吵
3	1	了 兩天 的 架 男 因爲 我 的 忍耐 是 有 限度 的
4	1	燕爾新婚 新娘 對 新郎 說 今後 咱們 不興 說 、 我 的 ， 了 要說 、
5	1	我們 的 ， 新郎 去 洗澡 良久 不 出 新娘 問 你 在 幹什麼 哪 親愛
6	1	的 我 在 刮 我們 的 鬍子 呢
7	1	有 一個 男人 當他 妻子 臨死時 做 得 非常地 開 他 說 妻 呀 你 死

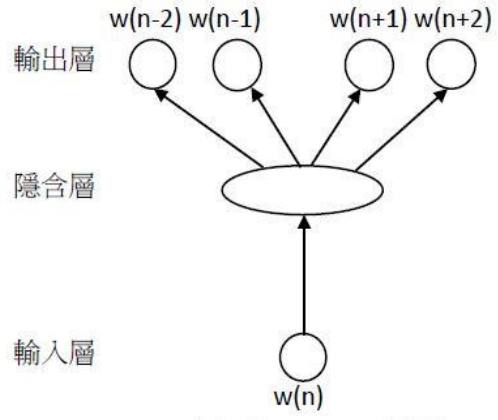
圖二 資料示意圖

測試資料則分成巴哈、痞克邦、學習電子報三組，分別進行斷詞，並且每個笑話獨立存進 txt，且第一行為一標籤，1 表示笑話，0 表示非笑話，藉此讓測試時能以一個笑話為單位得到最終一個笑話好笑與否的回饋。另外，學習電子報中的測試資料和前述的訓練資料並不相同。

(3) Pre-trained word embeddings 預訓練詞向量

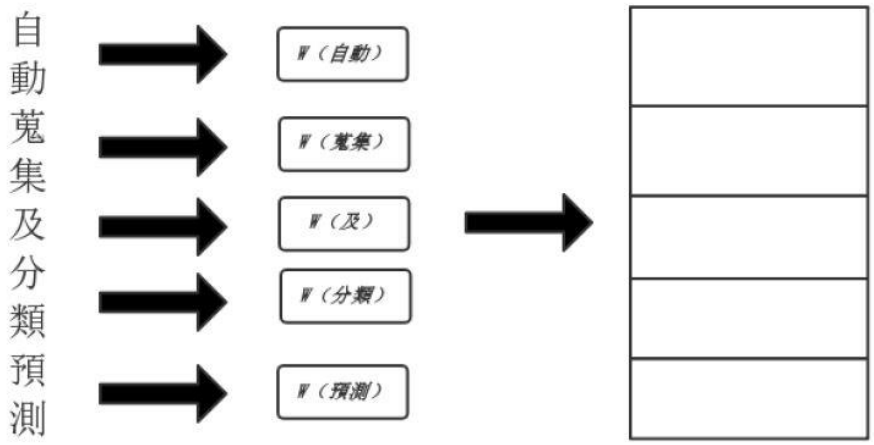
詞向量(Word embeddig)的概念是用來幫助電腦將文章中的詞彙，一個詞彙對應到一個向量，且向量可以有好幾個維度。Word embedding 的概念主要是將符號形式的字，映射(或嵌入)到一個數學空間裡，以詞向量來進行表示，同時詞向量之間的位置關係也能夠幫助在語意上取得聯繫，幫助機器更有效的進行訓練。

我們使用的模型為 Word2vec，為現今最廣泛的詞向量模型。利用 Word2vec 通過學習字與字之間關聯的方法預先進行訓練，使程式在訓練神經網路之時，達到更好更快速的訓練成效。在 Word2vec 兩種的模型中，我們選擇以中間單字來預測上下文的 Skip-gram 模型來訓練詞向量，其結構如下圖三。



圖三 Skip-gram 模型(左)

當訓練完成後，每個單詞皆能從其 embedding 層取出該詞的 embedding，也就是該詞的詞向量。將每一個詞向量結合成為一個二維矩陣後，令其作為之後 RNN&LSTM 所使用的起始 embedding，這些詞向量稱為預訓練詞向量。



圖四 將所有單詞以 word embedding 轉成一個二維矩陣

為了實現詞向量的預訓練，我們使用功能強大的 Gensim 套件，其中 window_size 設置為 5，代表讓每個詞針對前五及後五個詞學習彼此之間的關聯，而 embedding_size 設置為 10，將每個詞的維度壓縮為 10。

(4) Keras LSTM & RNN

網路上有許多通過 Keras 套件提供的 LSTM 層和 RNN 模型來解決情感的分類或情緒的分析。一位 CSDN 博客網站帳號名為 William_2015 曾發表一篇運用此方法來判斷語句中的情緒，把一句話當作一個 many-to-one 的 RNN 中，並得到一個值的輸出，0 代表消極、1 代表積極。

我們使用了相似的方法，將屬於笑話裡的句子標記為 1，非笑話的句子標記為 0 做訓練，在最後 testing 的部份我們會將笑話一個一個輸入進來，利用訓練好的模型，為"一個"笑話中的"每個"句子進行預設，此時每個笑話的句子會有屬於自己的"浮點數"預測值，我們將其加總平均，此平均數為該笑話的好笑程度，越接近 1 其好笑程度越高，其值如果大於 0.4 則歸類為笑話，反之則歸類為非笑話。

(5)Validation method

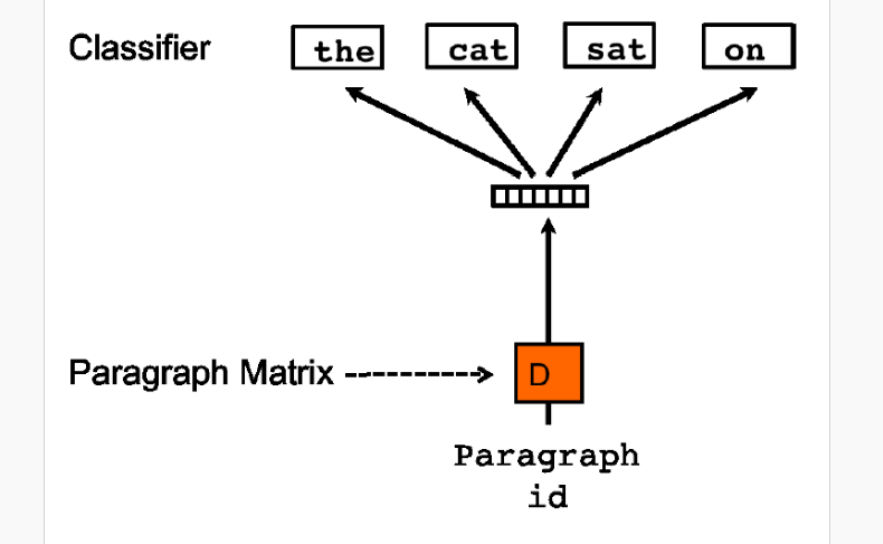
在驗證模型方面，我們使用計算 precision 和 recall 的方法。

Presition=標示為 1 的句子數/所有預測為 1 的句子數，也就是我們所預測的笑話中，真正符合笑話的比例。

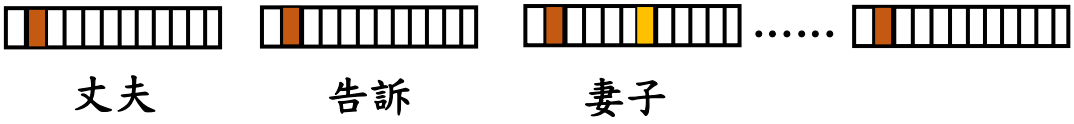
Recall=預測為 1 的句子數/所有標示為 1 的句子數，也就是在所有笑話中，被我們判定為笑話的比例

(6) Distributed Memory of Paragraph Vector(PV-DM)

Paragraph Vector(PV)模型的概念很簡單，在原先 word2vec 只有單詞的向量中，加入了文本(句子)向量，來 pre-train 單詞與單詞間、單詞與句子間的關係。原先的 PV 模型是用來預測單詞，而這裡的用法傾向於透過在單詞加入與哪些句子的關聯(有出現在該句子則標記)並轉換成 embedding 後，來訓練之後判斷文章是否為笑話。圖五、圖六為文本(句子)向量與單詞向量關聯的示意圖。



圖五 文本(句子)向量與單詞的關聯



圖六 同句出現的單詞標記該句子向量(橘色)
重複出現在別的句子仍要標記(黃色)

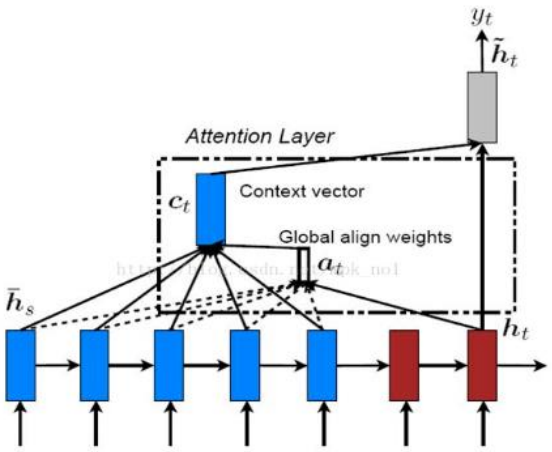
而 PV-DM 連結到 LSTM 的 model 訓練，必須有第二個 layer 來 input document vectors，下圖六為建立 model 的 code

```
EMBEDDING_SIZE = 32
HIDDEN_LAYER_SIZE = 128
BATCH_SIZE = 32
NUM_EPOCHS = 10

model_1 = Sequential()
model_1.add(Embedding(vocab_size, EMBEDDING_SIZE,input_length=MAX_SENTENCE_LENGTH))
model_1.add(LSTM(HIDDEN_LAYER_SIZE, dropout=0.2, recurrent_dropout=0.2))

model_2 = Sequential()

model_2.add(Dense(128, input_shape=(16,)))

model = Sequential()
model.add(Merge([model_1,model_2],mode='sum'))

model.add(Dense(1))
model.add(Activation("sigmoid"))
# binary_crossentropy:二分法
model.compile(loss="binary_crossentropy", optimizer="adam",metrics=["accuracy"])
```

圖七

(7)Attention

在傳統的 LSTM 中，我們訓練模型只能得到模型，難以直接看到中間的參數，而 Attention 是通過保留 LSTM 對輸入序列的中間輸出結果，訓練一個模型來對這些輸入進行選擇性的學習，並且在訓練完後時將輸出序列與之進行關聯。



圖八 Attention 示意圖

此方法能夠幫助我們理解在訓練過程中，輸入序列是如何影響最後生成序列，協助我們理解模型內部運作機制。

(五) 訓練及預測結果

以下的模型參數皆為:

HIDDEN_LAYER_SIZE = 128, BATCH_SIZE = 32, NUM_EPOCHS = 10

Traning data 和 testing data 的比例為 8:2

MAX_SENTENCE_LENGTH = 16

當使用第一組資料集(zol+維基)時，訓練的結果如下圖

```
Train on 59186 samples, validate on 14797 samples
Epoch 1/10
59186/59186 [==============================] - 49s 831us/step - loss: 0.2049 - precision:
0.9874 - recall: 0.9671 - acc: 0.9779 - val_loss: 0.1047 - val_precision: 0.9948 -
Epoch 10/10
59186/59186 [==============================] - 43s 733us/step - loss: 0.0085 - precision:
0.9976 - recall: 0.9976 - acc: 0.9977 - val_loss: 0.0318 - val_precision: 0.9976 -
val_recall: 0.9872 - val_acc: 0.9922
14797/14797 [==============================] - 2s 166us/step

Test score: 0.032, precision: 0.998, recall: 0.987, accuracy:0.992
```

圖九

當使用第一組資料集(zol+學習電子報)時，訓練的結果如下圖

```
Train on 51233 samples, validate on 12809 samples
Epoch 1/10
51233/51233 [==============================] - 60s 1ms/step - loss: 0.5923 - precision: 0.6826 - recall:
0.6767 - acc: 0.6764 - val_loss: 0.5206 - val_precision: 0.7475 - val_recall: 0.7000 - val_acc: 0.7321
Epoch 2/10
51233/51233 [==============================] - 62s 1ms/step - loss: 0.5085 - precision: 0.7559 - recall:
0.7394 - acc: 0.7462 - val_loss: 0.4508 - val_precision: 0.7685 - val_recall: 0.8707 - val_acc: 0.8044
Epoch 3/10
51233/51233 [==============================] - 62s 1ms/step - loss: 0.3821 - precision: 0.8416 - recall:
0.8315 - acc: 0.8344 - val_loss: 0.3475 - val_precision: 0.8648 - val_recall: 0.8294 - val_acc: 0.8500
Epoch 10/10
51233/51233 [==============================] - 61s 1ms/step - loss: 0.1718 - precision: 0.9324 - recall:
0.9395 - acc: 0.9347 - val_loss: 0.4420 - val_precision: 0.8478 - val_recall: 0.8508 - val_acc: 0.8490
12809/12809 [==============================] - 4s 278us/step

Test score: 0.442, precision: 0.848, recall: 0.851, accuracy:0.849
```

圖十

由上可觀察到(zol+維基)的資料集基本上第一個 epoch 就幾乎收斂了。

PV-DM 模型:

使用資料集(zol+維基)，訓練的結果如下圖

```
Train on 20512 samples, validate on 5128 samples
Epoch 1/10
20512/20512 [==============================] - 20s 986us/step - loss: 0.1328 - acc:
0.9583 - val_loss: 0.0873 - val_acc: 0.9702
Epoch 2/10
20512/20512 [==============================] - 18s 860us/step - loss: 0.0473 - acc:
0.9847 - val_loss: 0.0620 - val_acc: 0.9838
Epoch 10/10
20512/20512 [==============================] - 18s 871us/step - loss: 0.0110 - acc:
0.9963 - val_loss: 0.1313 - val_acc: 0.9793
5128/5128 [==============================] - 1s 106us/step
```

圖十一

接著我們使用三組測試集包括巴哈(168 則)及痞克邦(122 則)的笑話，和學習電子報中另外沒加入訓練(72 則)的非笑話進行模型測試

	巴哈 (122 則笑話)	痞克邦 (168 則笑話)	學習電子報 (72 則非笑話)
zol+維基	97.6%	99.2%	7.0%
zol+學習電子報	85.7%	90.2%	94.4%
PV-DM(zol+維基)	71.4%	72.1%	50.7

表 1 各測試資料集在不同模組上的準確率

我們發現(zol+維基)的資料集，在不管是笑話還是非笑話的預測裡都偏好預測為笑話，表現和 traing 時的準確率有天壤之別，此為 overfitting 的狀況，我們分析其中的原因有以下兩種可能:

1. 維基語料庫的詞彙通常較為正式，專有名詞較多，與一般口語化差別很大，但在我們測試資料集裡，所有的網路文章比較不會使用專業的詞彙及文筆，因此就算是非笑話文章，模型也會認其為笑話。
2. 維基語料庫不適用我們的斷詞系統，因為其斷詞完的句子普遍較短，在測試資料集裡斷完詞的結果普遍較長，因此也會誤認斷詞後較長的非笑話文章為笑話。

而(zol+學習電子報)的資料集不管在笑話與非笑話都表現不錯，並且沒有出現極端的情形，而在巴哈笑話的表現上比在痞克邦上的表現欠佳，可能的原因為巴哈的笑話較為新穎，其中間會加入很多時事梗，以及一些並非常見的字詞，這些字詞可能從動畫、遊戲、卡通裡出現等等。

Attention:

在訓練出結果後，我們嘗試使用 attention 來幫助我們了解哪些字對於影響整片的判斷結果比重較大，以下使用 zol+學習電子報的模型及資料集進行探討。

```
好笑程度        句子
0.00531560    剛才 我 打電話 給台 哥大 電信 客服 台哥 大 客服 您好 很 高興 爲您服務 我 說

0.98936439    你 高興 的 太 早 了 然後 就 把 電話 掛了
```

圖十二

以上圖為例，這篇笑話分成兩個句子，且笑點集中在第二句。

```
[0.06127213 0.07128049 0.06809079 0.0646585  0.06697059 0.06767157
 0.06343083 0.0664574  0.05600865 0.05362067 0.05315977 0.05137975
 0.06013263 0.06062319 0.06631946 0.06892361]
打電話 1509 0.06809079
說 47 0.06892361
我 67 0.071280494
```

圖十三

```
[0.0697031  0.06860354 0.053947   0.05559666 0.05477224 0.0622255
 0.06508859 0.05840695 0.06556657 0.0657476  0.07219455 0.06791734
 0.06325964 0.06084209 0.05917461 0.05695405]
高興 635 0.06860354
你 38 0.0697031
掛了 8950 0.07219455
```

圖十四

圖十三和圖十四分別是此笑話中第一和第二句中每個字的 attention 權重，及取前三高的權重及其代表的字，可是這些代表字與我們的笑點不盡相同，可能的原因是機器學習到的笑點跟人有差異，或者是機器學習到的笑點是有許多字組合在一起，不適合以單一字的權重來象徵笑點。

在 PV-DM 的模型的測試上，發現前兩者測試資料集準確率都落在 7 成左右，沒有展現出前面高達 8.9 成的準確率，原因是在訓練的時候 document vecctors 會記錄詞彙裡那些字的出現，但是在 load model 之後，當我們要一筆一筆資料測試的時候，沒辦法準確訓練該則笑話的 document vectors，導致 predict 時對應 document layer 的 input 是較不準確的。

而在第三者資料上預測結果不甚理想，原因為其大多資料為非笑話為主，以及裡頭的用詞與訓練資料相比較為不同。

(五)未來展望

本研究將結合自然語言分析、資料探勘、機器學習技術，預期完成以下成果：

(1) 能讓電腦從網際網路自動蒐集並分析文章是否為笑話。

(2) 系統能對多數人評估出的答案預測出相同結果。

開心的情緒是人生存必要條件，讓機器學習預測幽默與否也是未來必不可缺的要件之一。此計劃探討的笑話辨識系統，能讓使用者透過機器分析文章的幽默程度，未來也能進一步於人機互動對話內容中加入幽默特徵，提供更人性化的溝通方式。

(六)參考文獻

[1] Python 網路爬蟲實戰(松崗出版)

[2] Hay, J. 1995. Gender and Humour: Beyond a Joke. A Thesis of the Victoria University of Wellington.

[3] Rada Mihalcea and Carlo Strapparava. Learning to laugh(automatically)： Computational

models for humor recognition, Computational Intelligence, Volume 22, Number 2, 2006.

[4] Yishay Raz. Automatic Humor Classification on Twitter, Proceedings of the NAACL HLT 2012 Student Research Workshop, pages 66–70, 2012.

[5] Peter Potash, Alexey Romanov, and Anna Rumshisky. #HashtagWars: Learning a Sense of Humor, Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017), pages 49–57, 2016.

[6] Renxian Zhang, and Naishi Liu. Recognizing Humor on Twitter. CIKM '14 Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, pages 889-898, 2014.

[7] Dafna Shahaf, Eric Horvitz, Robert Mankoff. Inside Jokes: Identifying Humorous Cartoon Captions. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1065-1074, 2015.

[8] 孫瑛澤, 陳建良, 劉峻杰, 劉昭麟, 蘇豐文.中文短句之情緒分類. 第廿二屆自然語言與語音處理研討會, 2016.

[9]蔡惠萍, 王麗丹, 段書凱. 基於word embedding和CNN的情感模型. 計算机應用研究第33卷. 2016.

[10] Donghai Zhang, Wei Song, Lizhen Liu1, Chao Du1, and Xinlei Zhao. 2017. Investigations in Automatic Humor Recognition. Proceedings of the 2017 10th International Symposium on Computational Intelligence and Design, pages 272-275.

[11] FUKUBALL (2014). 如何使用Jieba中文分詞程式 http://blog.fukuball.com/ru-he-shi-yong-jieba-jie-ba-zhong-wen-fen-ci-cheng-shi/ (2018.05.04)

[12] William_2015. (2017). 利用Keras下的LSTM進行情感分析 檢索至https://blog.csdn.net/William_2015/article/details/72978387(2018.05.04)

[13] Quoc Le, Tomas Mikolov. Distributed Representations of Sentences and Documents. Google Inc, 1600 Amphitheatre Parkway, Mountain View, CA 94043. 2014-5-16.

[14] Keras for attention https://blog.csdn.net/u010041824/article/details/78855435

[15] 理解LSTM/RNN中的attention機制 http://www.jeyzhang.com/understand-attention-in-rnn.html

[16] aneesh joshi. (2017). Learn Word2Vec by implementing it in tensorflow https://towardsdatascience.com/learn-word2vec-by-implementing-it-in-tensorflow-45641adaf2ac