統計計算期末報告

tags: `EDA` , `LDA` , `SVM` , `Machine Learning`

📝資料集 🔗 (https://www.kaggle.com/datatattle/covid-19-nlp-text-classification)

📌 Coronavirus tweets NLP - Text Classification

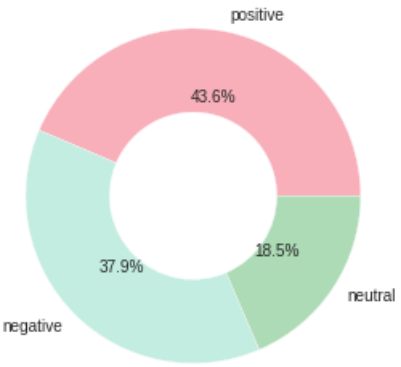
資料描述: 對數據進行文本分類。這些推文是從Twitter中提取的，然後進行了人工標記。姓名和用戶名已經被賦予了代碼，以避免任何隱私問題。

> Columns:
> 1. Location
> 2. Tweet At
> 3. Original Tweet
> 4. Label

📍 一、探索式資料分析 (EDA)

1. 圓餅圖

由圖可知label的分布並不均勻，佔比最大的是positive，接著是negative。

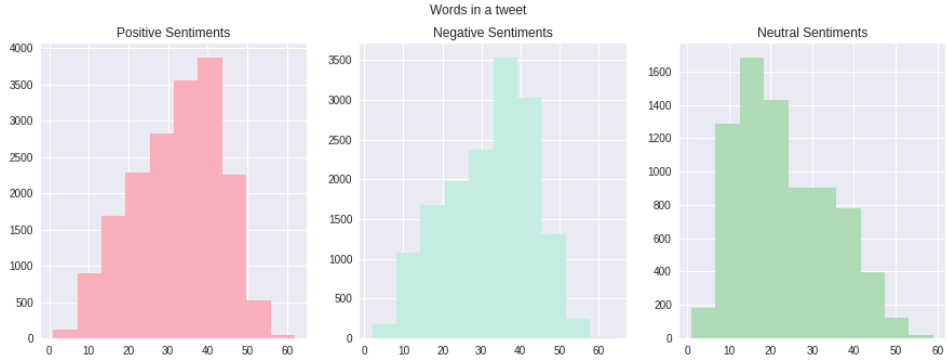


2. 訓練、測試資料集的比較

這兩個資料集基本在label的分布上十分相似。

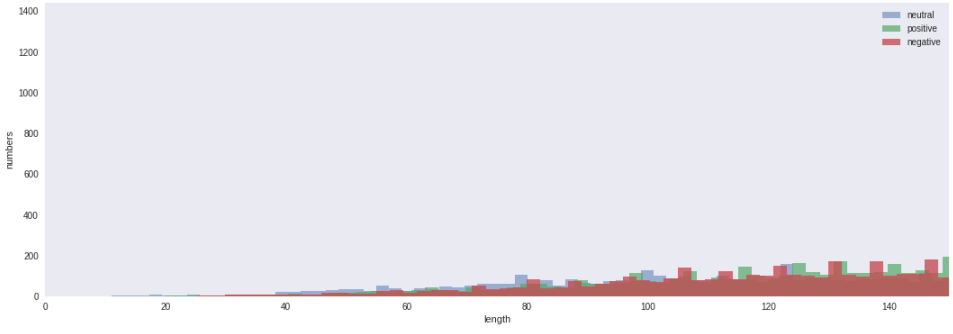
3. 每則推文的文字數量

從圖片中可以看出positive 和 negative 推文的文字數量分布相似，然而和neutral的分布卻很不同。



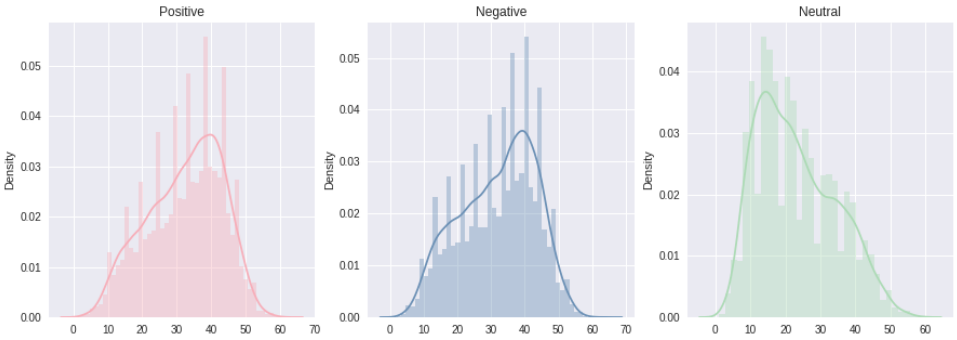
4. 文字長度

下圖畫出了三個標籤下文字的長度分布。



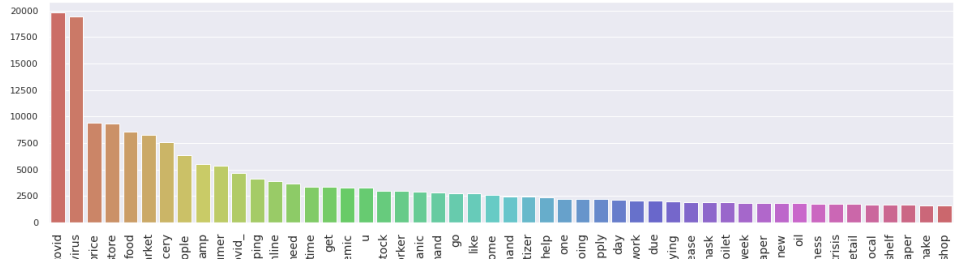
5. 平均文字長度

下圖為三種情緒下平均文字長度的分布。



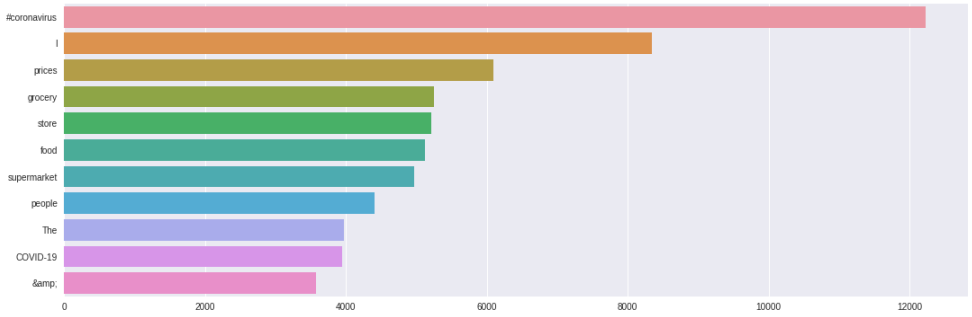
6. 文字出現頻率

下圖為資料集中文字出現的頻率。

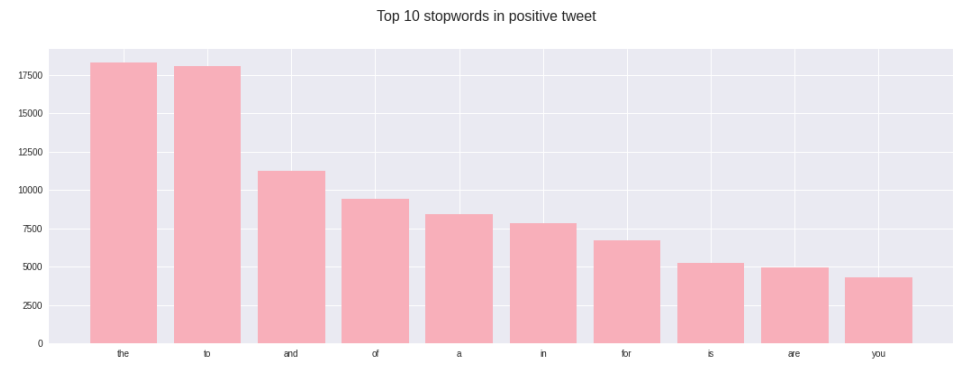


7. Top words

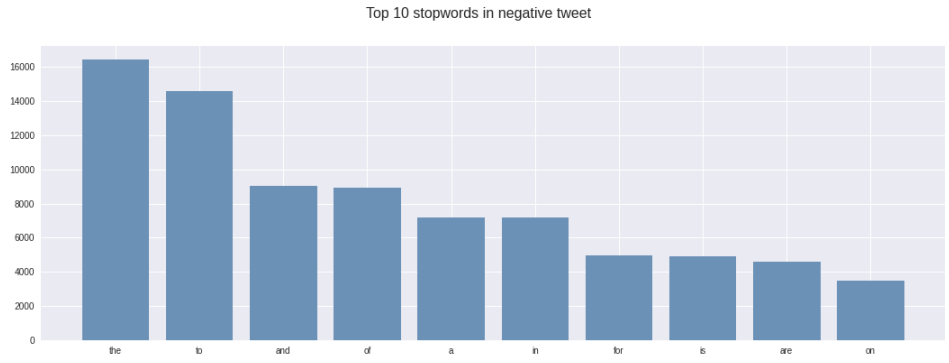
下圖提取出了文章中頻率最高的十個單字。



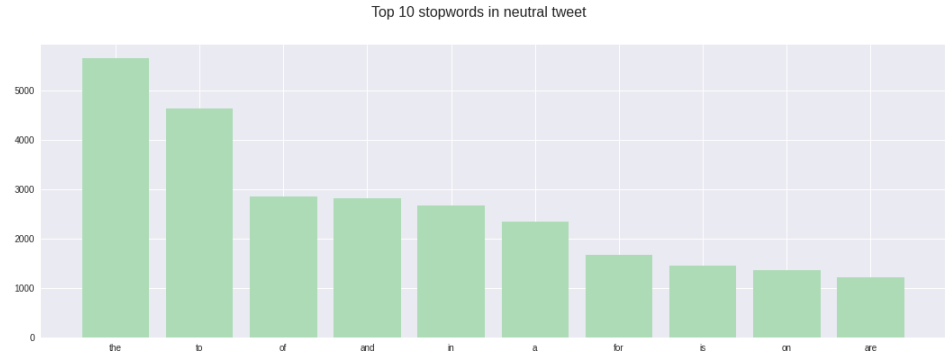
8. 標籤為positive的前十個高頻stopwords



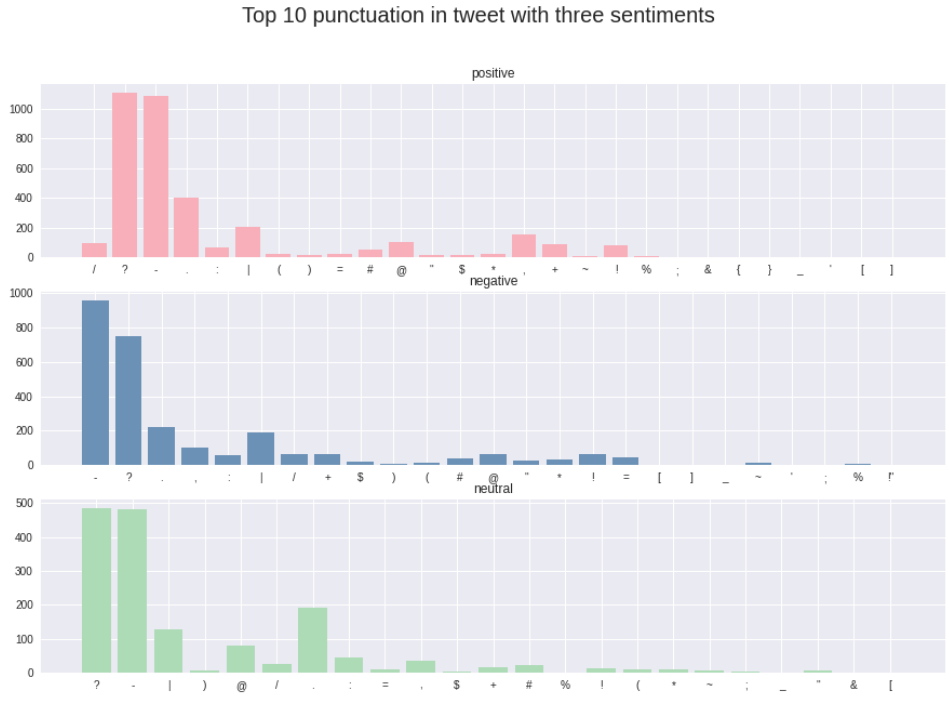
9. 標籤為negative的前十個高頻stopwords



10. 標籤為neutral的前十個高頻stopwords

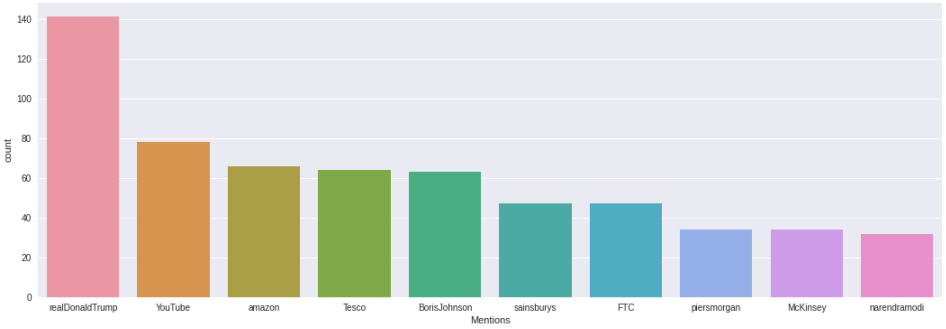


11. 三個標籤中出現率最高的標點符號



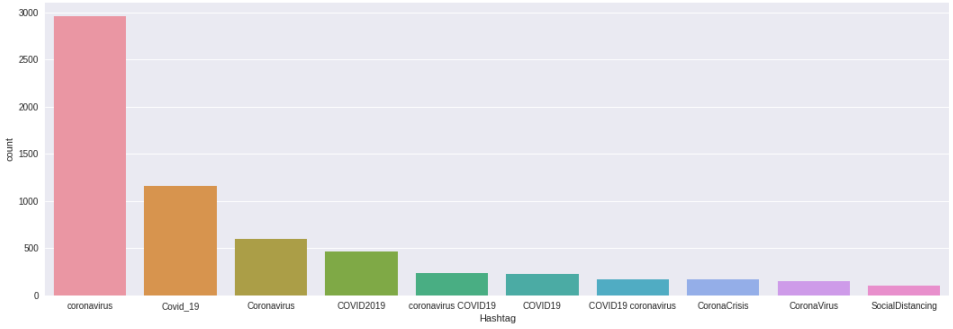
12. 推文中最常出現的提及(mention)

這些提及大多為人名，而提及率最高的是美國前總統川普，為當時疫情爆發時美國的決策者。其中也出現了Youtube、Amazon等網站，推測是和居家防疫的線上活動有關。有趣的是英國首相Boris Johnson也在其中，而他在去年也曾感染新冠病毒。

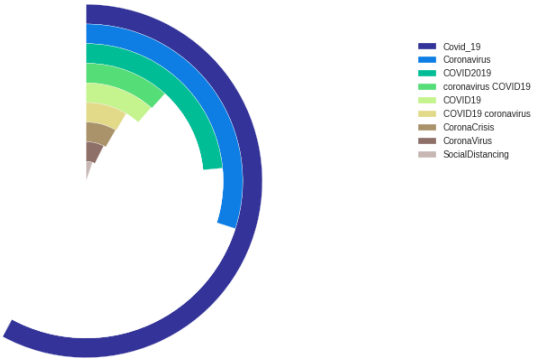


13. 推文中最常出現的標籤(hashtag)

最常出現的標籤皆為COVID-19、coronavirus，等文字，因為資料集為covid-19的推文，人們在自己的推文下都會留下相關的hashtag。



14. 最高頻標籤的圓型分布圖



15. 文章中stopwords的文字雲

p.s手的形狀代表STOP!!

📍 二、資料前處理(Preprocessing)

在做文字探勘以前需要先將文章做前處理，去除掉不需要的字詞，避免得出不良的結果。所利用的套件為nltk。

步驟:

1. 移除urls：避免之後分析的文字有網址等無意義詞彙。
2. 移除htmls：避免之後分析的文字有網址等無意義詞彙。
3. 將文字轉成小寫：維持文字的一致性。
4. 移除數字：數字對文字的分析並無直接關聯，因此移除。
5. 移除標點符號：標點符號對文字的分析並無直接關聯，因此移除。
6. 移除停用詞：停用詞即是文章中出現率極高，然而卻對文本分析沒有貢獻的詞彙(the、and、a等等)，因此利用nltk套件中的stopwords將資料中的停用詞移除。
7. 移除提及：可能會提及一些不相關的人物(例如朋友等等)，因此移除。
8. 移除標籤：標籤內容皆為covid相關詞，而我們已知該資料集為新冠肺炎相關推文，因此covid等詞彙也可以算是停用詞，因此將其移除。
9. 移除空白：對文字的分析並無直接關聯，因此移除。
10. 對文字做lemmitize(詞形還原)，消除詞性問題，所利用的套件是nltk.stem 中的WordNetLemmatizer。

```
#Remove Urls
def remove_urls(text):
  url_remove = re.compile(r'https?://\S+|www\.\S+')
  return url_remove.sub(r'', text)
#Remove HTML links
def remove_html(text):
  html=re.compile(r'<.*?>')
  return html.sub(r'',text)
#Lower text
def lower(text):
  low_text= text.lower()
  return low_text
# Number removal
def remove_num(text):
  remove= re.sub(r'\d+', '', text)
  return remove
from nltk.corpus import stopwords
", ".join(stopwords.words('english'))
STOPWORDS = set(stopwords.words('english'))
#Remove Punctuations
def punct_remove(text):
  punct = re.sub(r"[^\w\s\d]","", text)
  return punct
# Remove stopwords
def remove_stopwords(text):
  return " ".join([word for word in str(text).split() if word not in STOPWORDS])
# Remove mention
def remove_mention(x):
  text=re.sub(r'@\w+','',x)
  return text
# Remove hash
def remove_hash(x):
  text=re.sub(r'#\w+','',x)
  return text
#Remove extra white space left while removing stuff
def remove_space(text):
    space_remove = re.sub(r"\s+"," ",text).strip()
    return space_remove
# Lemmitizing
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
wnl = WordNetLemmatizer()
lem1=[]
for i in range(len(df['text_new'])):
  temp = [wnl.lemmatize(word) for word in df['text_new'][i].split()]
  tem=" ".join(temp)
  lem1+=[tem]
```

📍 三、資料分析

LDA model

LDA分析的目的最主要是想要找出文字的主題分布。這一段內容會包含如何找出好的LDA模型，並選出適合的主題數。

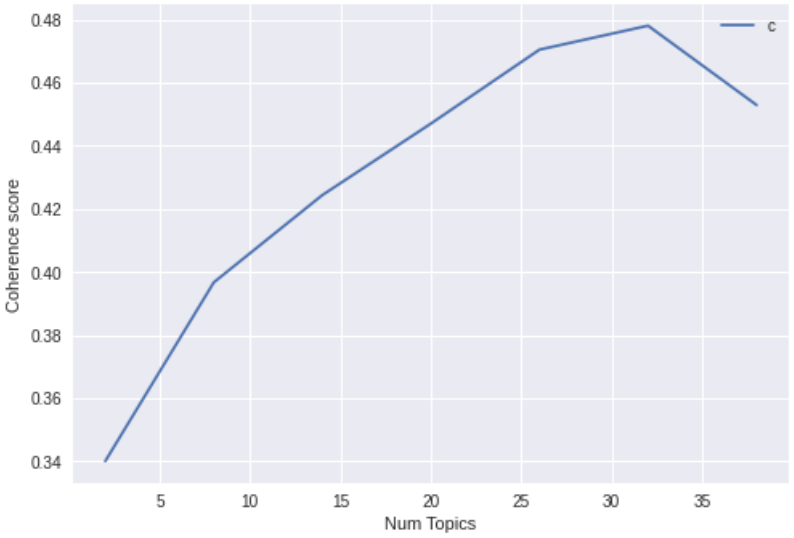
步驟:

1. 在做LDA之前，首先利用spacy的模型對剛剛processed好的text做處理，並且對text做tokenization，建立一個tokens list。

2. 利用python中的gensim模型建立字典，輸入的資料為tokenized過後的詞。

3. 利用doc2bow套件建立適合的語料庫。

4. 首先我使用gensim 套件的LDA model, 他是使用online variational Bayes (VB) algorithm for LDA 演算法。接著，找出Ldamodel的coherence value。coherence value 衡量的是一個主題內詞與詞之間的相對距離，因此我利用coherence value選出表現良好的主題數，並繪製出coherence value的折線圖。



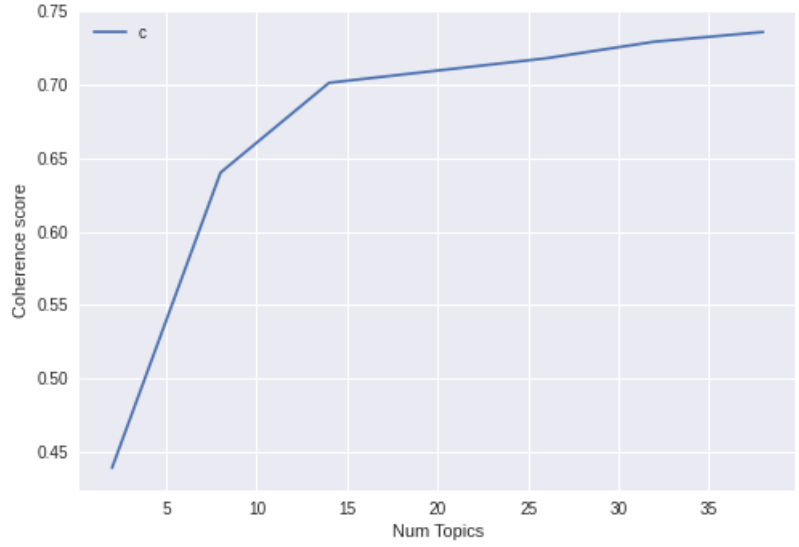

```
Num Topics = 2  has Coherence Value of 0.3227
Num Topics = 8  has Coherence Value of 0.3811
Num Topics = 14  has Coherence Value of 0.4177
Num Topics = 20  has Coherence Value of 0.4478
Num Topics = 26  has Coherence Value of 0.4631
Num Topics = 32  has Coherence Value of 0.4589
Num Topics = 38  has Coherence Value of 0.4597
```

5. 將選出的主題數(在這裡我選32)代入gensim套件中的LdaModel訓練，最終得出整體模型的coherence score約為0.42。

```
Coherence Score:  0.41945860673679325
```

6. 我期望我的coherence score為0.7左右，因為若是小於0.7，LDA模型配適的並不好，但若太大會有overfitting的狀況。上述利用LDAmodel得出的coherence score只有0.42，因此我利用LDAmallet模型重新做一次主題數選擇，LDAmallet模型使用gensim 套件的models.wrappers.LdaMallet , 他是使用optimized Gibbs sampling algorithm演算法。

7. 利用LDAmallet model 的 coherence value選出表現良好的主題數，並繪製出coherence value的折線圖。

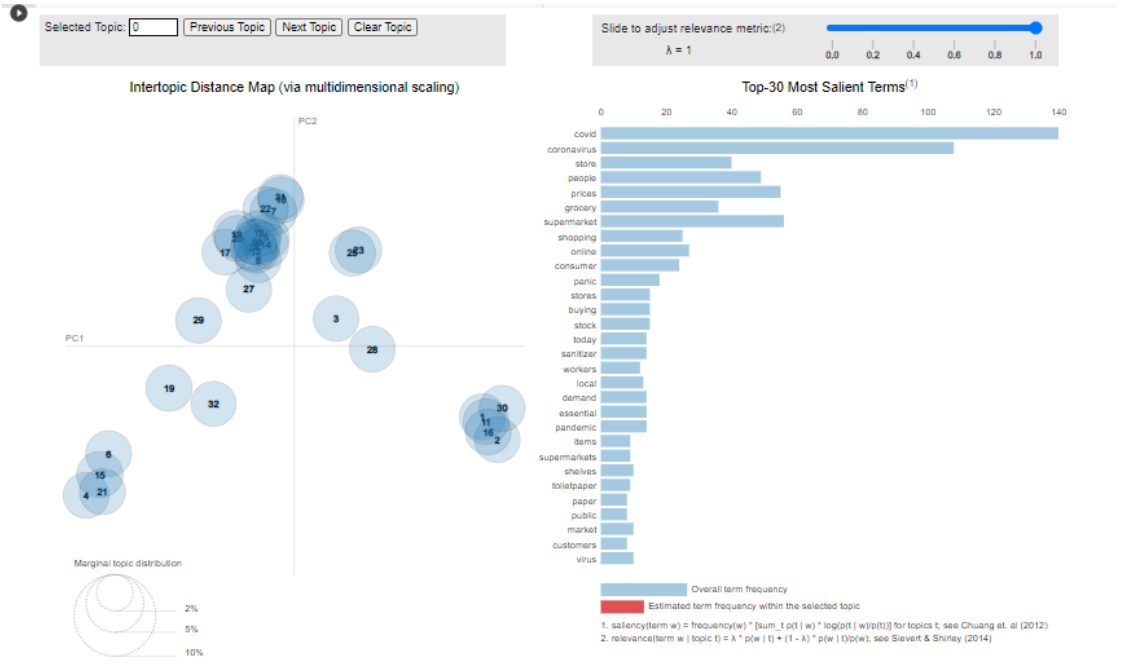



```
Num Topics = 2  has Coherence Value of 0.4396
Num Topics = 8  has Coherence Value of 0.64
Num Topics = 14  has Coherence Value of 0.7009
Num Topics = 20  has Coherence Value of 0.7092
Num Topics = 26  has Coherence Value of 0.7175
Num Topics = 32  has Coherence Value of 0.7288
Num Topics = 38  has Coherence Value of 0.7354
```

8. 將選出的主題數(在這裡我仍選32)代入gensim套件中的LdaModel訓練，最終得出整體模型的coherence score約為0.71，是我理想中的coherence score值，因此我將選擇LDA mallet的這個模型做之後的視覺化以及主題分析。

```
Coherence Score:  0.7146544197944664
```

9. 首先是pyLDAvis視覺化，事實上這個此圖為互動式圖表，透過氣泡的選擇可以看出各個主題最常出現的詞語，氣泡的位置則代表主題的分布。以此圖來說，氣泡大致分為五群，每一群的氣泡十分分散，這可以解釋最主要的主題大約是五群，而那些堆疊起來的氣泡各自代表五個主題下的分類。



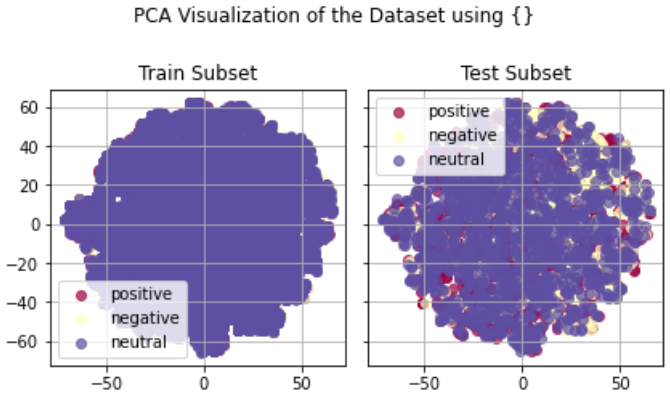
10. 下圖為各個topic下的關鍵字，可以通過關鍵字內容找出主題。

	Terms per Topic
Topic1	covid, prices, putting, price, lives, idaho, ashrafghani, medicines, poetry, combat, idahocovid, 15396,online, received, citizensadvice, outta, ignoring, measures, 8526,basically, 26497,department, repatriation
Topic2	covid, country, sense, closed, soaps, insurance, countries, events, yangon, taste, employee, hunte, ã£â¢reportã£â¢, market, direct, idontusetwittermuch, 23359,covid, fatalities, 23401,access, texas
Topic3	retail, china, covid, virus, spain, netherlands, morons, fresh, older, payment, bought, black, article, station, dodheld, 36067,bryandbender, managing, director, withering, stealing
Topic4	coronavirus, toiletpaper, stocked, donã£â¢t, pharmacy, brings, tesco, spotify, employees, askreuters, panel, surrounded, financial, magic, offices, 24957,iã£â¢ll, 43089,fuck, rahulsh, stranger, supermarket
Topic5	local, workers, delivery, staff, alert, panicking, related, america, advertising, qurantine, thrown, points, present, hungry, uncertainty, callout, overblown, company, wrong, bradpaisley
Topic6	coronavirus, people, supply, waiting, important, slowing, spreading, testing, galway, attention, stocking, centers, hoarders, tells, forget, online, unemployment, interesed, afternoon, dooms
Topic7	prices, virus, corona, staff, higher, delivered, advice, listen, learn, infected, villages, collapse, spread, tripling, running, segment, 12897,million, 22631,meat, special, clean
Topic8	supermarkets, drivers, years, major, point, community, fresh, sainsburys, stupidity, profits, spread, morning, selfish, limits, coronoavirusoutbreak, constantly, bored, 26594,please, grandchildren, stabilize
Topic9	sanitizer, quarantine, isolation, crisis, markets, things, calls, income, weekly, households, michaeljackson, cough, engagement, recover, 36463,dmkts, 39830,incredible, trust, theo_km, jacked, restaurant
Topic10	supermarket, prices, easter, handsanitizer, safety, visit, cleaning, coronavirusoutbreak, resilient, heard, happy, blame, friend, chains, stayathome, visits, pharmacies, companies, adjusting, 33770,covid

11. 將各topic的關鍵字做成文字雲，可以簡單看出各個topic的主要內容。

12. 利用TomotopyLDAVectorizer和剛剛選擇出的topic數量對資料集進行訓練，利用PCA的方法將訓練完稱的資料集投影至平面，並對上原本資料集中的三個標籤，進而生成下圖。從訓練和測試資料集得出的結果可以很明顯的發現各個推文的主題和其所帶有的情緒(label:positive，negative，

neutral)並不會有很明顯的分類。也就是說，每個主題之下都有持相反意見或是中立的人存在，並且沒有特別正面或是特別負面的主題。



LDA+SVM model

在這個模型中我結合了LDA和SVM的演算法對資料進行分類，以下為簡短程式碼和分類結果。

```
from sklearn.decomposition import PCA
folds = RepeatedStratifiedKFold(n_splits=10, n_repeats=10)
vectorizer = TomotopyLDAVectorizer(num_of_topics=32, workers=workers, min_df=min_df,
                                   rm_top=rm_top)
clf = SVC()
pca = PCA(n_components=0.95)
pipe = Pipeline([("vectorizer", vectorizer), ("scalar", StandardScaler()),
                 ("classifier", clf)])

results = cross_val_score(pipe, test['lemma_text'], test.label, cv=folds, n_jobs=2, v
                          scoring="accuracy")


print("Accuracy -> mean: {}\t std: {}".format(results.mean(), results.std()))
```

```
[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done  46 tasks      | elapsed: 17.2min
Accuracy -> mean: 0.4175101374809054     std: 0.02148635843751845
[Parallel(n_jobs=2)]: Done 100 out of 100 | elapsed: 37.6min finished
```

- 由上述結果可以看出利用LDA+SVM的演算法對於情緒(label)的分群非常不理想，只有約42%的準確率，並且分類時間相較於其他的演算法長許多。
- 這個結果也可以說明利用SVM將主題分類的結果對於情緒指標再做分類的結果十分不良好，原因可以由上一張PCA的結果看出，個個主題下皆有持不同意見的人，在這種情況下進行分類結果將會非常不準確。

📍 四、其他分類方法

- 有鑑於SVM+LDA對於情緒指標(label)的分類情況都不理想，因此我決定嘗試其他機器學習的方法找出更適合的分類模型。
- 我試用以下了三種機器學習模型找出分類準確率最高者:
  - Linear SVC(使用SVM進行分類，無LDA)
  - Random forest
  - Multinomial NB

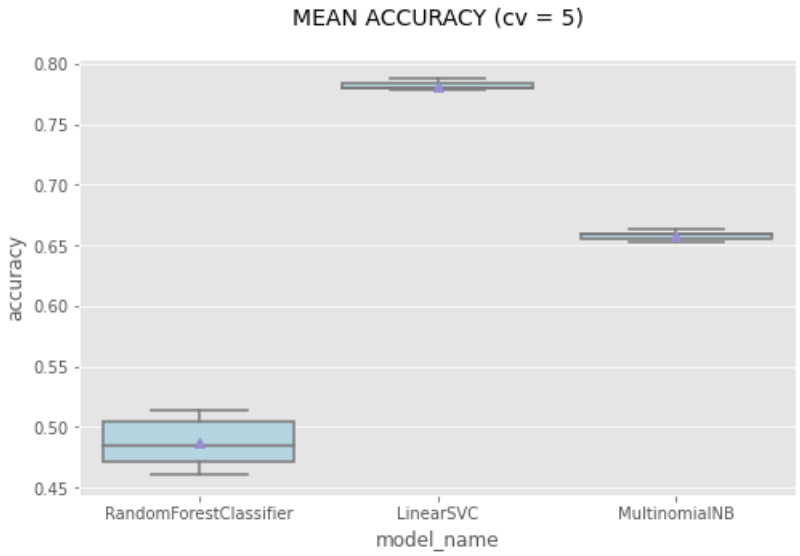
步驟:

1. 對label做encoding

	sentiment	label
0	Neutral	1
1	Positive	2
4	Negative	0

2. 利用Tf-idf算出各個文字的向量並找出features，利用這些features對資料集做訓練。
3. 由以下的表格和圖可以看出Linear SVC有最好的表現，有接近80%的準確率。

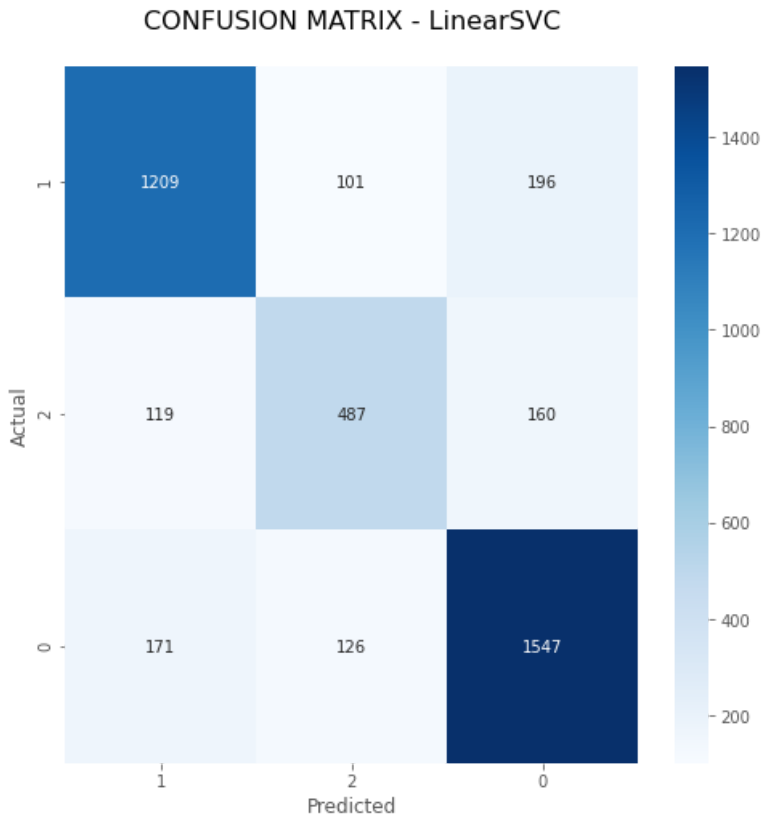
model_name	Mean Accuracy	Standard deviation
LinearSVC	0.782103	0.003729
MultinomialNB	0.658162	0.004504
RandomForestClassifier	0.486891	0.022214



4. 下圖為Linear SVC的Precision matrix 和confusion Matrix.

CLASSIFICATIION METRICS

	precision	recall	f1-score	support
1	0.81	0.80	0.80	1506
2	0.68	0.64	0.66	766
0	0.81	0.84	0.83	1844
accuracy			0.79	4116
macro avg	0.77	0.76	0.76	4116
weighted avg	0.79	0.79	0.79	4116




```
* SVM=Support Vector Machine : 支持向量機。
* SVC=Support Vector Classification: 支持向量機用於分類。
```

- 由上述的結果得知單純使用SVM對於資料進行分類得到的結果準確率可以達到80%，並且表現得比其他兩個分類器還好，我認為這是因為在不同情緒下所使用的詞彙會非常不一樣，而這使得SVM可以進行良好的分類。

📍 五、討論(Disscussion)

對於LDA主題分類的結果，通過關鍵字的觀察可以歸納成以下五類:

- **關於國家議題的推文，關鍵字有:**
  - 物價、人民、州、公眾、經濟等等
- **關於民生用品需求的推文，關鍵字有:**
  - 零售、超市、貨架、民生用品、需求、必需品等等。
- **關於新型態消費模式衍生出的議題的推文，關鍵字有:**
  - 市場、詐騙、線上消費、購買、恐慌、銀行等等
- **關於民生用品供給的問題，關鍵字有:**
  - 庫存、衛生紙、人、供給、消費者、工作、顧客、距離等等
- **關於疫情導致的實體消費問題、隔離措施、政府作為、封城等推文，而關於此主題的推文所佔的數量最大，關鍵字有:**
  - 在地、工人、司機、社區、超市、檢查、到達、安全、人群、影響、商店、川普、政府、封城、限制、恐懼、社會、商業、實施、保護、隔離、消毒、危機等等。

由以上的分類可以簡單地了解當時美國疫情最危急的時候人民最關心、及最擔心的議題。

未來應用:

- 在這份報告中，我利用了SVM對推文的情緒指標進行訓練，並且得到了不錯的準確率，我認為這個結果可以拿來當作分類依據，並對目前關於COVID-19的推文進行情緒預測，並透過轉換製作成情緒指數的資料。藉由這份情緒預測指數的資料，我認為可以將其拿來比較市場上的一些指數、股價等等，因為人民的情緒常常會影響市場的指數。若是能找出情緒指數和市場指數存在關聯性，或許可以拿來當作預測某些股票價格(疫苗股票、藥廠股票、零售股票等等)的參考依據
- 目前美國疫情已趨穩定，然而台灣正陷在疫情的恐慌之中，我認為目前可以做的應用是利用爬蟲得到目前台灣的推文，將其翻譯並透過目前所使用的SVM分類器進行分類並計算情緒值。比較該情緒值和目前台灣的股價(高端疫苗)、市場指數的趨勢，並進行觀察和預測。