數位語音處理 Final project
新聞稿文本分類

物理四 B03202027 許芝瑜　物理四 B03202042 陳毅

Problem Definition

在如此資訊發達的年代，我們若要知道世界各地發生什麼重大的事件，只需要坐在電腦桌前，到各大新聞網站上看看今日的重點新聞即可，而這些新聞往往都會依照內容被分類在不同的類別下，方便我們進行依照領域去搜尋。

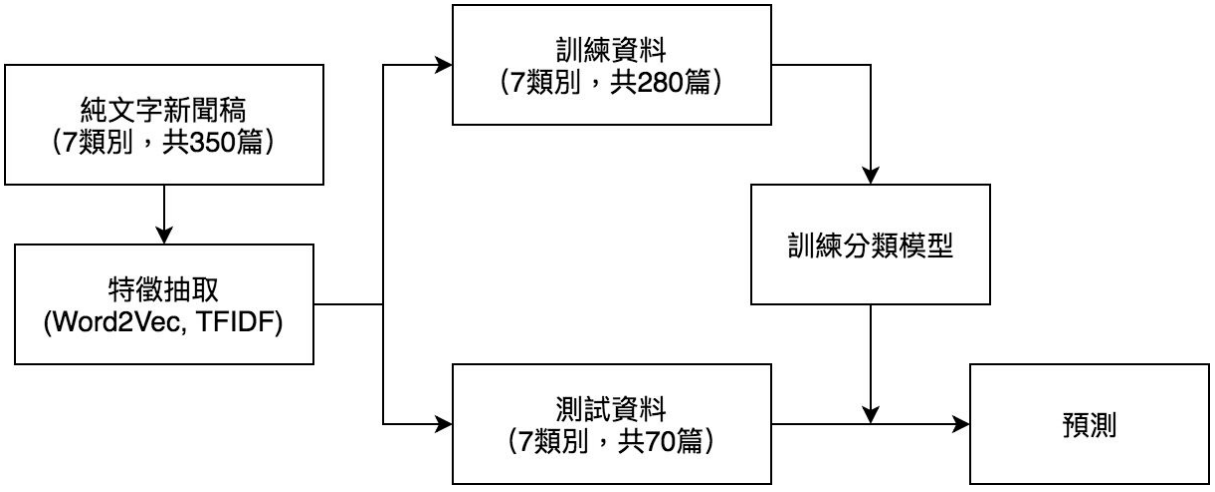
本次的final project，旨在建立許多不同的model，來將大量的新聞稿進行分類，並分析什麼樣的特徵抽取方式及model較為優良。

Environment

我們撰寫程式所使用到的語言有「C/C++」、「Python」以及「Bash shell」。

- C/C++
  - 系統：資工系Workstation
  - 版本：gcc 8.1.1
- Python
  - 系統：Windows 10 64bits
  - 版本：2.7
  - 套件資訊
    - numpy：1.13.1+mkl
    - scipy：0.19.1
    - scikit-learn：0.19.1
    - gensim：3.4.0
- Bash shell
  - 系統：資工系Workstation

Flow Chart



Method

資料收集

新聞稿的收集是這個final project中，最為花費時間的工作。由於我們的目標是想要根據新聞稿的文字內容，來進行新聞稿的分類，所以我們只需收集新聞稿的純文字即可。但當今的新聞網站中，新聞稿時常會穿插不少圖片及圖片說明文字，因此直接手動複製並轉換成純文字文稿，是相當麻煩的工作。

經過分析各個新聞網站的排版方式後，決定撰寫shell script以及C/C++程式來抓取文本，新聞稿的來源網站為British Broadcasting Corperation (BBC)以及New York Times (NYT)。

使用方式：（以BBC為例，NYT的使用方式與之相同，僅為資料夾及檔案名稱不同）

- 輸入指令「./CollectDoc/BBC/BBC.sh」，即可在「CollectDoc/BBC/」目錄下，生成corpusBBC資料夾，其中有7個類別的資料夾，各50篇文章。
- 再手動隨機選取40篇作為訓練資料，10篇作為測試資料。

Source	Topic	Training (articles)	Testing (articles)
British Broadcasting Corperation (BBC)	World-US-Canada (W)	40	10
	Business (B)	40	10
	Entertainment-Arts (EA)	40	10
	Science-Environment (SE)	40	10
	Technology (T)	40	10
	UK-Politics (UK)	40	10
	Health (H)	40	10
New York Times (NYT)	Arts (A)	40	10
	Business (B)	40	10
	Health (H)	40	10
	Science (SC)	40	10
	Technology (T)	40	10
	Politics (P)	40	10
	Sports (SP)	40	10

特徵抽取

由於各個文章長短不一，而且所使用的詞彙也相當分歧，因此當我們成功收集到新聞網上各個分類的文章後，還沒辦法直接將這些文章丟給分類模型去訓練或預測。因此，我們可以先將每篇文章進行特徵向量的抽取，讓這些看似毫無關聯的文章能夠映射到同一個向量空間，再藉此特徵向量，去給分類模型去進行訓練與預測。

我們所使用到的特徵向量抽取方式有兩種，第一種是「*Word2Vec*」，第二種是「*TF-IDF + LSA*」。

- Word2Vec
  - 使用gensim套件。
  - 將所有文本中出現的所有文字丟入gensim的gensim.models.Word2Vec中，並設定model所會計算的word的最少出現次數。
  - 使用model.wv[word]來得到每個文字的word vector，並將文章中所有word的word vector加總平均，可得該文章的特徵向量。

```
sentences = MySentences('../corpus/BBC/train', '../corpus/BBC/test')
model = gensim.models.Word2Vec(sentences, size=dim, min_count=3)
```

- TF-IDF + LSA
  - 使用sklearn套件去計算TF-IDF，使用scipy.sparse去進行LSA的SVD並降維。
  - 將所有文本的文字丟入sklearn.feature_extraction.text的CountVectorizer()中，計算每個文本中出現的文字的字頻，可得一字頻矩陣（row：文本、column：文字）。再藉由此字頻矩陣去計算每篇文本的TF-IDF，作為該文本的特徵向量。
  - 由於TF-IDF的維度為「所有文章中所有出現過的文字數目」，因此直接以此去進行分類模型的訓練會造成計算困難。因此結合LSA，可從20000維左右的維度降至50維，以此進行SVD分解，可得每篇文本的特徵向量。

```
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(corpus)
word = vectorizer.get_feature_names()
#print len(word)

transformer = TfidfTransformer()
tfidf = transformer.fit_transform(X)

tfidfArr = tfidf.toarray()
#print len(tfidfArr)

tmp = np.array_split(tfidfArr, [len(tfidfArr)*4/5])

training_data = np.transpose(tmp[0])
testing_data = np.transpose(tmp[1])
#print type(training_data), len(training_data) # N * 0.8
#print type(testing_data), len(testing_data) # N * 0.2

W = csc_matrix(training_data, dtype=float)
U,sigma,VT = svds(W, k=50)
```

訓練分類模型及預測

總共使用了sklearn套件中的9種分類模型，分別如下：

- Logistic Regression

```
### Logistic Regression ###
lr = LogisticRegression()
lr.fit(np.transpose(training_data_VT), label_train)
print "### Logistic Regression ###"
print lr.predict(np.transpose(testing_data_VT))
print np.array(label_test)
print performance(lr.predict(np.transpose(testing_data_VT)), label_test)
```

- Gaussian Naïve Bayes

```
### Gaussian Naive Bayes ###
gnb = GaussianNB()
gnb.fit(np.transpose(training_data_VT), label_train)
print "### Gaussian Naive Bayes ###"
print gnb.predict(np.transpose(testing_data_VT))
print np.array(label_test)
print performance(gnb.predict(np.transpose(testing_data_VT)), label_test)
```

- Bernoulli Naïve Bayes

```
### Bernoulli Naive Bayes ###
bnb = BernoulliNB()
bnb.fit(np.transpose(training_data_VT), label_train)
print "### Bernoulli Naive Bayes ###"
print bnb.predict(np.transpose(testing_data_VT))
print np.array(label_test)
print performance(bnb.predict(np.transpose(testing_data_VT)), label_test)
```

- Stochastic Gradient Descent

```
### Stochastic Gradient Descent ###
sgd = SGDClassifier(loss='modified_huber', shuffle=True, random_state=101)
sgd.fit(np.transpose(training_data_VT), label_train)
print "### Stochastic Gradient Descent ###"
print sgd.predict(np.transpose(testing_data_VT))
print np.array(label_test)
print performance(sgd.predict(np.transpose(testing_data_VT)), label_test)
```

- K-Nearest Neighbours

```
### K-Nearest Neighbours ###
knn = KNeighborsClassifier(n_neighbors=8)
knn.fit(np.transpose(training_data_VT), label_train)
print "### K-Nearest Neighbours ###"
print knn.predict(np.transpose(testing_data_VT))
print np.array(label_test)
print performance(knn.predict(np.transpose(testing_data_VT)), label_test)
```

- Decision Tree

```
### Decision Tree ###
dtree = DecisionTreeClassifier(max_depth=None, random_state=101,
                               max_features=None, min_samples_leaf=1)
dtree.fit(np.transpose(training_data_VT), label_train)
print "### Decision Tree ###"
print dtree.predict(np.transpose(testing_data_VT))
print np.array(label_test)
print performance(dtree.predict(np.transpose(testing_data_VT)), label_test)
```

- Random Forest

```
### Random Forest Tree ###
rfm = RandomForestClassifier(max_depth=4, random_state=0)
rfm.fit(np.transpose(training_data_VT), label_train)
print "### Random Forest Tree ###"
print rfm.predict(np.transpose(testing_data_VT))
print np.array(label_test)
print performance(rfm.predict(np.transpose(testing_data_VT)), label_test)
```

- SVM

```
### Support Vector Machine ###
svm = NuSVC()
svm.fit(np.transpose(training_data_VT), label_train)
print "### Support Vector Machine ###"
print svm.predict(np.transpose(testing_data_VT))
print np.array(label_test)
print performance(svm.predict(np.transpose(testing_data_VT)), label_test)
```

- Neural network

```
### Neural Network ###
nn = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5,20),
                   random_state=1)
nn.fit(np.transpose(training_data_VT), label_train)
print "### Neural Network ###"
print nn.predict(np.transpose(testing_data_VT))
print np.array(label_test)
print performance(nn.predict(np.transpose(testing_data_VT)), label_test)
```

使用訓練資料的特徵向量訓練模型，再以此模型去分類測試資料，可計算出該模型的準確率。而後，調整各個模型的參數，或是抽取特徵向量的維度，以最大化預測準確率。

使用方法

python model/TFIDF_BBC.py
python model/TFIDF_NYT.py
python model/word2vec_BBC.py
python model/word2vec_NYT.py

Result

下表為對BBC及NYT的新聞稿，運用不同特徵抽取方法及分類模型所得之預測正確率表格。

Source	Classification	TF-IDF+ LSA (1/70)	Word2Vec (1/70)
British Broadcasting Corperation (BBC)	Logistic Regression	62	30
	Gaussian Naïve Bayes	44	32
	Bernoulli Naïve Bayes	59	18
	Stochastic Gradient Descent	58	10
	K-Nearest Neighbours	52	34
	Decision Tree	52	29
	Random Forest	58	30
	SVM	61	27
	Neural Network	60	37
New York Times (NYT)	Logistic Regression	57	23
	Gaussian Naïve Bayes	37	24
	Bernoulli Naïve Bayes	48	18
	Stochastic Gradient Descent	52	10
	K-Nearest Neighbours	49	25
	Decision Tree	49	28
	Random Forest	48	30
	SVM	54	27
	Neural Network	54	30

下表為對BBC的新聞稿，以TF-IDF+LSA抽取特徵後，各主題的預測正確率。

(1/10)	W	B	EA	SE	T	UK	H
Logistic Regression	8	6	9	10	9	10	10
Gaussian Naïve Bayes	8	2	6	9	3	10	6
Bernoulli Naïve Bayes	7	6	9	10	10	10	9
Stochastic Gradient Descent	9	1	9	10	10	10	9
K-Nearest Neighbours	9	3	8	10	5	10	7
Decision Tree	8	3	8	9	8	10	7
Random Forest Tree	9	6	6	9	10	10	9
SVM	7	5	9	10	10	10	10
Neural Network	7	6	9	9	9	10	10

下表為對BBC的新聞稿，以Word2Vec抽取特徵後，各主題的預測正確率。

(1/10)	W	B	EA	SE	T	UK	H
Logistic Regression	6	1	1	3	6	8	5
Gaussian Naïve Bayes	7	2	4	4	4	8	3
Bernoulli Naïve Bayes	5	0	2	0	1	1	9
Stochastic Gradient Descent	0	0	10	0	0	0	0
K-Nearest Neighbours	6	2	6	6	4	7	3
Decision Tree	6	1	4	5	4	7	2
Random Forest Tree	6	2	3	5	2	8	4
SVM	7	1	2	0	4	9	4
Neural Network	9	2	4	0	7	9	6

下表為對NYT的新聞稿，以TF-IDF+LSA抽取特徵後，各主題的預測正確率。

(1/10)	A	B	H	SC	T	P	SP
Logistic Regression	10	5	8	8	8	9	9
Gaussian Naïve Bayes	4	2	3	8	3	9	8
Bernoulli Naïve Bayes	8	3	8	7	6	7	9
Stochastic Gradient Descent	6	2	10	8	9	8	9
K-Nearest Neighbours	7	5	8	8	8	9	10
Decision Tree	7	6	7	7	7	7	8
Random Forest Tree	10	3	7	5	8	8	7
SVM	10	5	8	8	7	8	8
Neural Network	8	4	8	10	8	9	9

下表為對NYT的新聞稿，以Word2Vec抽取特徵後，各主題的預測正確率。

(1/10)	A	B	H	SC	T	P	SP
Logistic Regression	4	0	7	0	3	9	0
Gaussian Naïve Bayes	4	2	2	5	2	8	1
Bernoulli Naïve Bayes	1	0	5	0	2	10	0
Stochastic Gradient Descent	1	9	0	0	0	0	0
K-Nearest Neighbours	4	5	3	3	2	6	2
Decision Tree	6	2	2	3	2	7	6
Random Forest Tree	4	2	5	4	1	9	5
SVM	4	2	4	3	1	8	5
Neural Network	5	3	5	3	1	8	5

Discussion

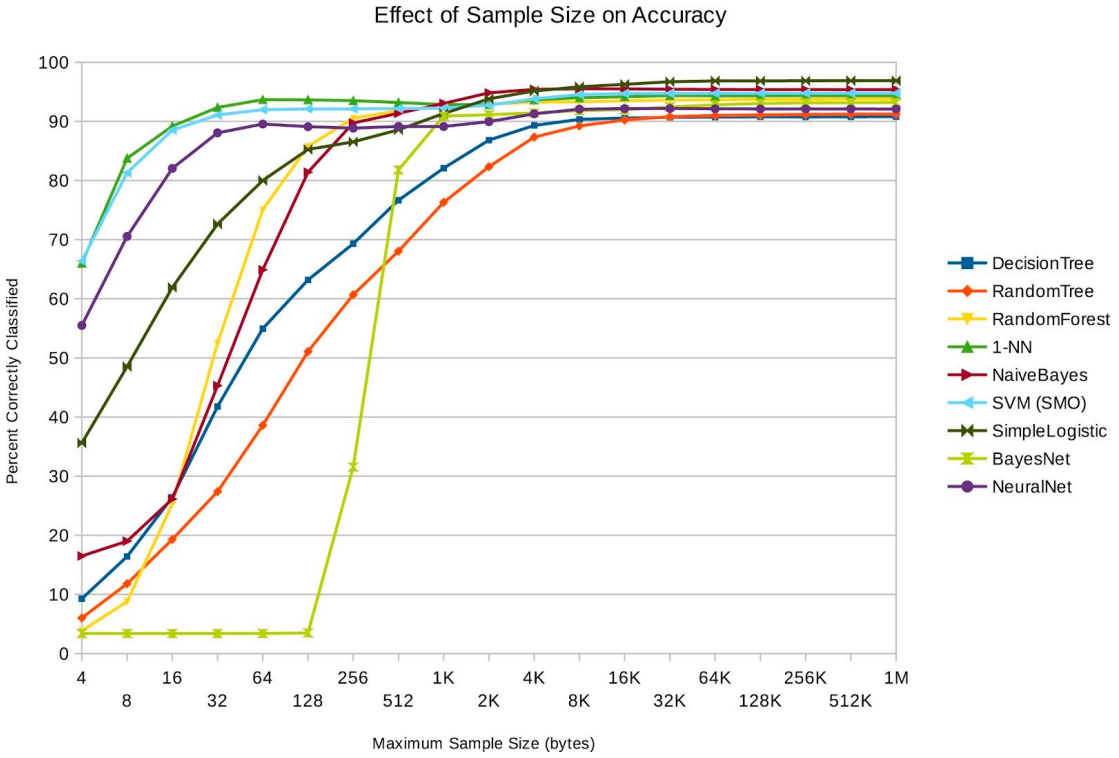
總括來說，用TF-IDF抽取特徵的結果比Word2Vec好很多，使用BBC做為資料來源的結果比NYT稍好一些。

使用TF-IDF抽取特徵時，搭配Logistic Regression來分類最適合，Gaussian Naïve Bayes最不適合。使用Word2Vec抽取特徵時，搭配Neural Network來分類最適合，Stochastic Gradient Descent最不適合。

而這樣的結果，與下面的Reference相符：TF-IDF比Doc2Vec效果好，這可能是因為資料不夠多，所以Doc2Vec捕捉不到詞間的關聯性。因為我們的資料量偏小，所以使用Logistic Regression，SVM，Neuron Network效果好，而不適合用Naïve Bayes。至於SVM適合用在少資料，是因為只需要support vector就能建構hyperplane，而不是用全部的sample，Naïve Bayes效果不佳可能是因為用了許多假設，因此在很多狀況下都不適合。

VSM Technique	Classification Algorithm	Accuracy
TF-IDF	Logistic Regression	95.45%
TF-IDF	Naïve Bayes	73.62%
Doc2Vec	Logistic Regression	76.20%
Doc2Vec	Naïve Bayes	58.40%

(截自Reference 7)



(截自Reference 6)

另外，使用不同抽取特徵的方法，和不同來源的新聞稿，在各主題的表現上也有些不同。

以TF-IDF抽取特徵時，BBC在UK-Politics和Science-Environment預測的結果最準確，在Business的預測結果最差，通常是被錯分到Technology，UK-Politics，或Health。在UK-politics預測準確，可能是因為testing data的10篇裡有9篇都有出現UK這個關鍵詞，相較之下，Business的文章較無特定使用的詞彙。NYT則在Sports和Politics預測的結果最準確，在Business的預測結果最差，通常是被錯分到Technology或Politics。 在Politics預測準確，應該是幾乎都有Washinton這個關鍵詞，Sports則有各類運動的聯盟或比賽名稱，有助於分類，而NYT的Bussiness的用詞跟BBC的一樣也不特定，所以分類結果不佳。

以Word2Vec抽取特徵時，BBC在UK-Politics預測的結果最好，在Business的預測結果最差；NYT在Politics預測的結果最好，在Technology的預測結果最差。

所以若要將數百篇新聞分類，使用TF-IDF抽取特徵，搭配Logistic Rgression、SVM、Neuron Network分類，效果會好很多。

Conclusion

根據本次final project的數據結果可知，在進行新聞稿的特徵抽取時，TF-IDF能夠比Word2Vec更好的表達一個文本的特徵，此外結合LSA的技術，可以有效地幫助我們將特徵向量空間的維度降低，進而提升訓練模型的速度。

此外，在我們所使用的9種分類模型中，Logistic Regression、SVM以及NN，是較適合進行此種分類工作的模型。

未來，若能收集更多新聞稿來訓練模型，將預期可以繼續提升預測準確率。

Reference

1. 7 Types of Classification Algorithms
   https://analyticsindiamag.com/7-types-classification-algorithms/
2. 奇异值分解 SVD 的数学解释
   https://blog.csdn.net/u010099080/article/details/68060274
3. [python] 使用scikit-learn工具计算文本TF-IDF值
   https://blog.csdn.net/Eastmount/article/details/50323063
4. 【实践】用深度学习解决大规模文本分类问题
   http://www.sohu.com/a/130593183_633698
5. tf-idf - Wikipedia
   https://zh.wikipedia.org/wiki/Tf-idf
6. Automatic classification of object code using machine learning
   https://www.sciencedirect.com/science/article/pii/S1742287615000523
7. Document feature extraction and classification
   https://towardsdatascience.com/document-feature-extraction-and-classification-53f0e813d2d3
8. BBC News
   https://www.bbc.co.uk/news
9. The New York Times
   https://www.nytimes.com/

Appendix 1: 分工表

許芝瑜

- 收集NTY的新聞稿350篇的網址（CollectDoc/NTY/link/*）
- 寫程式
    - 文章特徵抽取：Word2Vec
    - 調整各個模型參數，並統計結果
- 撰寫書面報告

陳毅

- 收集BBC的新聞稿350篇的網址（CollectDoc/BBC/link/*）
- 寫程式
    - 收集新聞稿的shell script及C/C++
    - 文章特徵抽取：TF-IDF + LSA
    - 模型建立：共9個model
- 撰寫書面報告

Appendix 2: Raw Data

BBC的主題類別對應表

1	2	3	4	5	6	7
W	B	EA	SE	T	UK	H

NYT的主題類別對應表

1	2	3	4	5	6	7
A	B	H	SC	T	P	SP

BBC TF-IDF+LSA

Logistic Regression
[2 5 7 5 2 5 2 2 2 2 3 3 3 3 3 3 3 3 1 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 4
 4 4 4 3 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 2 1 1 3]
[2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 4
 4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 1 1 1 1]
62 / 70
Gaussian Naive Bayes
[6 6 7 5 2 6 6 6 2 6 3 6 3 3 6 3 3 6 3 1 7 7 7 6 6 3 7 7 7 1 4 4 4 4 4 4 4 6
 4 4 4 6 5 5 6 6 6 1 6 5 4 6 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 2 1 1 6]
[2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 4
 4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 1 1 1 1]
44 / 70
Bernoulli Naive Bayes
[2 5 7 5 2 5 2 2 2 2 3 3 3 1 3 3 3 3 3 3 7 7 7 7 7 1 7 7 7 4 4 4 4 4 4 4 4
 4 4 4 4 5 5 5 4 5 5 5 5 5 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 2 1 3 3]
[2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 4

4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 1 1 1 1]
59 / 70
Stochastic Gradient Descent
[5 5 7 5 3 5 5 5 2 5 3 3 3 3 3 3 3 3 1 7 7 7 7 7 7 7 7 5 4 4 4 4 4 4 4
4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 1 1 1 5]
[2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 4
4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 1 1 1 1]
58 / 70
K-Nearest Neighbours
[3 2 7 5 2 4 4 4 2 4 4 3 3 3 4 3 3 3 3 3 7 7 7 4 7 3 7 7 7 4 4 4 4 4 4 4 4 4
4 4 4 3 5 5 4 4 5 5 2 5 4 6 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 1 1 1 3]
[2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 4
4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 1 1 1 1]
52 / 70
Decision Tree
[5 5 7 2 4 5 5 2 2 7 3 3 3 3 1 3 3 3 3 1 7 7 7 7 7 6 7 3 7 5 4 4 4 4 4 4 2
4 4 4 7 5 5 5 5 5 5 5 5 2 6 6 6 3 6 6 6 6 6 6 1 1 1 1 1 1 1 1 3 2]
[2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 4
4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 1 1 1 1]
52 / 70
Random Forest Tree
[6 2 7 2 2 5 2 2 4 2 2 7 3 7 3 3 3 3 3 1 7 7 7 7 7 7 7 7 1 7 4 4 4 4 4 4 4 4
4 4 5 5 5 5 5 5 5 2 5 5 5 6 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 2 1 1 1]
[2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 4
4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 1 1 1 1]
58 / 70
Support Vector Machine
[2 5 7 5 2 5 5 2 2 2 3 3 3 3 3 3 3 3 3 1 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 4
4 4 4 5 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 2 1 3 3]
[2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 4
4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 1 1 1 1]
61 / 70
Neural Network
[2 1 7 2 2 5 2 1 2 2 3 3 3 3 3 3 3 3 3 1 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 1
4 4 4 1 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 2 1 3 3]
[2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 4
4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 1 1 1 1]
60 / 70

NYT TF-IDF+LSA

Logistic Regression
[1 1 1 1 1 1 1 1 1 1 7 2 2 6 2 2 2 5 1 6 3 3 3 3 3 1 3 3 4 3 6 6 6 6 6 6 6
6 6 2 4 4 2 4 4 1 4 4 4 4 7 4 7 7 7 7 7 7 7 7 2 5 5 5 5 5 5 5 2 5]
[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6 6
6 6 6 4 4 4 4 4 4 4 4 4 4 4 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5]
57 / 70

Gaussian Naive Bayes
[1 6 1 1 6 1 6 6 6 6 6 6 6 6 2 2 5 6 6 3 6 6 3 6 6 6 3 4 6 6 6 6 6 6 6
 2 6 6 4 4 6 4 4 6 4 4 4 4 7 6 7 7 7 6 7 7 7 7 2 6 6 6 5 2 6 5 2 5]
[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6
 6 6 6 4 4 4 4 4 4 4 4 4 4 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5]
37 / 70
Bernoulli Naive Bayes
[1 1 1 1 1 1 2 2 1 1 1 1 2 6 2 5 2 5 1 1 3 3 3 3 3 7 3 3 4 3 6 6 6 6 6 6 6
 2 2 5 5 4 2 4 4 2 4 4 4 4 7 3 7 7 7 7 7 7 7 7 2 5 5 5 5 3 2 5 5 3]
[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6
 6 6 6 4 4 4 4 4 4 4 4 4 4 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5]
48 / 70
Stochastic Gradient Descent
[1 1 1 1 1 1 3 2 3 2 7 5 5 2 2 5 5 5 4 5 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6
 2 6 5 4 4 2 4 4 7 4 4 4 4 7 3 7 7 7 7 7 7 7 7 2 5 5 5 5 5 5 5 5 5]
[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6
 6 6 6 4 4 4 4 4 4 4 4 4 4 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5]
52 / 70
K-Nearest Neighbours
[1 7 1 1 1 1 3 4 1 1 7 1 2 2 2 2 2 5 4 7 3 3 7 3 3 3 3 3 4 3 6 6 6 6 6 6 6
 6 6 2 4 4 2 4 4 7 4 4 4 4 7 7 7 7 7 7 7 7 7 2 2 2 5 2 2 2 5 2 2]
[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6
 6 6 6 4 4 4 4 4 4 4 4 4 4 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5]
49 / 70
Decision Tree
[1 1 1 1 1 1 3 2 2 1 1 2 2 6 2 2 2 5 2 4 3 3 2 3 3 1 3 3 4 3 6 2 6 6 6 6 6
 5 6 2 4 4 4 4 4 1 4 4 3 3 7 2 7 7 7 2 7 7 7 7 5 2 3 5 2 5 5 5 5 5]
[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6
 6 6 6 4 4 4 4 4 4 4 4 4 4 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5]
49 / 70
Random Forest Tree
[1 1 1 1 1 1 1 1 1 1 7 2 5 6 1 2 2 5 1 1 3 3 2 3 3 7 3 3 4 3 6 2 6 6 6 6 6
 6 6 2 4 4 2 4 7 1 4 1 4 1 7 1 7 7 7 7 1 7 7 3 5 5 2 5 5 5 2 5 5 5]
[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6
 6 6 6 4 4 4 4 4 4 4 4 4 4 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5]
48 / 70
Support Vector Machine
[1 1 1 1 1 1 1 1 1 1 2 2 6 2 2 2 5 1 1 3 3 3 3 3 1 3 3 4 3 6 6 6 6 6 6 6
 2 6 2 4 4 2 4 4 1 4 4 4 4 7 4 7 7 7 1 7 7 7 7 2 5 2 5 5 5 5 5 2 5]
[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6
 6 6 6 4 4 4 4 4 4 4 4 4 4 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5]
54 / 70
Neural Network
[1 1 1 1 1 1 4 2 1 1 5 5 2 6 2 2 2 5 4 4 3 7 3 3 3 3 3 3 4 3 6 6 6 3 6 6 6
 6 6 5 4 4 1 4 4 4 4 4 4 4 7 3 7 7 7 7 7 7 7 7 2 5 5 5 2 5 5 5 5 5]
[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6

6 6 6 4 4 4 4 4 4 4 4 4 4 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5]
54 / 70

BBC Word2Vec

Logistic Regression
[5 5 5 6 1 5 1 2 4 5 1 5 5 5 6 3 6 5 5 1 7 4 6 7 5 5 7 7 7 1 7 4 5 4 4 5 7
1 1 5 5 5 7 1 7 5 5 5 7 5 6 6 6 6 6 5 6 6 5 6 1 1 5 5 1 1 6 1 1 6]
[2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 4
4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 1 1 1 1]
30/70
Gaussian Naive Bayes
[5 5 5 5 1 5 2 2 7 5 1 5 3 3 1 3 3 5 5 5 7 4 7 4 5 5 4 4 7 1 7 4 5 4 2 5 4
2 4 3 3 5 7 4 4 5 5 2 4 5 6 6 6 6 6 3 6 6 7 6 1 1 3 7 1 1 1 1 1 6]
[2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 4
4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 1 1 1 1]
32/70
Bernoulli Naive Bayes
[7 7 5 7 1 7 1 6 7 5 1 7 3 3 7 7 7 5 7 7 7 7 7 7 7 7 7 7 1 7 6 7 7 7 7 7
1 6 5 5 7 7 7 7 7 7 7 7 7 7 7 7 7 5 6 7 7 7 1 6 7 7 1 1 7 1 1 7]
[2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 4
4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 1 1 1 1]
18/70
Stochastic Gradient Descent
[3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3]
[2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 4
4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 1 1 1 1]
10/70
K-Nearest Neighbours
[5 5 3 5 2 5 1 3 7 2 3 4 3 3 6 4 1 3 3 3 7 4 3 7 4 7 4 5 5 1 4 4 5 4 4 4 4
3 1 3 3 5 5 4 4 4 5 5 4 2 6 6 6 6 6 7 6 5 4 6 1 1 3 7 1 1 2 1 1 6]
[2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 4
4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 1 1 1 1]
34/70
Decision Tree
[5 5 1 4 1 5 2 4 1 5 3 5 3 3 6 4 3 5 5 5 7 2 5 4 2 2 4 4 7 1 6 4 2 4 4 5 4
1 4 5 3 4 5 5 1 5 5 2 3 2 6 6 6 6 6 3 6 3 7 6 1 1 6 2 1 1 7 1 1 6]
[2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 4
4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 1 1 1 1]
29/70
Random Forest Tree
[5 5 2 1 1 5 1 2 4 3 4 4 3 3 1 4 1 5 3 5 7 4 1 7 4 5 4 7 7 6 1 4 5 4 2 4 4
2 4 3 3 3 5 3 4 3 5 2 3 2 6 6 6 6 6 3 6 6 7 6 1 1 3 7 1 6 1 1 1 6]
[2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 4
4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 1 1 1 1]
30/70

Support Vector Machine
[3 5 4 5 1 5 1 2 4 5 1 4 5 5 1 3 5 5 5 3 4 7 7 7 4 5 4 7 6 1 6 7 5 7 2 7 7
 2 2 5 5 5 4 2 7 5 5 2 7 2 6 6 6 6 6 6 6 6 4 6 1 1 6 5 1 1 1 1 1 6]
[2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 4
 4 4 4 5 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 1 1 1 1 1]
27/70
Neural Network
[5 5 7 3 1 5 2 2 7 5 1 5 3 5 6 3 1 5 3 3 7 7 7 2 7 5 2 7 7 6 7 2 5 7 7 5 7
 2 2 5 5 5 5 5 7 5 7 5 5 7 6 6 6 6 6 6 6 6 7 6 1 1 1 5 1 1 1 1 1 1 1]
[2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 4 4 4 4 4 4 4 4
 4 4 4 5 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 1 1 1 1 1 1 1 1 1 1 1]
37/70

NYT Word2Vec

Logistic Regression
[5 6 1 1 5 3 3 5 1 1 6 1 5 6 1 6 3 7 1 6 3 3 7 5 3 3 3 3 1 3 6 1 6 6 6 6 6
 6 6 6 3 5 1 5 6 6 6 5 5 3 6 6 1 1 1 5 1 6 3 3 3 5 3 3 1 3 3 5 5 3]
[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6
 6 6 6 4 4 4 4 4 4 4 4 4 4 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5]
23/70
Gaussian Naive Bayes
[4 4 1 1 4 4 3 4 1 1 6 2 4 6 2 6 4 7 1 4 3 2 7 4 4 3 4 2 1 5 6 1 6 6 6 6 4
 6 6 6 2 4 1 4 6 6 1 4 4 4 6 1 1 1 1 3 7 6 2 2 2 4 3 4 1 4 4 5 4 5]
[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6
 6 6 6 4 4 4 4 4 4 4 4 4 4 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5]
24/70
Bernoulli Naive Bayes
[6 6 7 6 6 6 5 6 1 6 1 6 6 6 6 6 3 3 6 6 3 3 6 6 6 3 3 6 6 3 6 6 6 6 6 6 6
 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 1 6 6 3 6 6 6 3 6 6 3 6 6 4 6 5 6 5]
[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6
 6 6 6 4 4 4 4 4 4 4 4 4 4 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5]
18/70
Stochastic Gradient Descent
[2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6
 6 6 6 4 4 4 4 4 4 4 4 4 4 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5]
10/70
K-Nearest Neighbours
[2 7 1 1 2 5 3 4 1 1 1 2 4 2 2 5 2 7 1 2 7 2 7 4 3 3 3 2 2 4 6 1 6 5 6 1 1
 6 6 6 2 1 1 4 6 6 7 2 4 4 7 1 1 1 1 3 7 2 2 3 3 6 3 4 5 4 4 5 3 4]
[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6
 6 6 6 4 4 4 4 4 4 4 4 4 4 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5]
25/70
Decision Tree
[4 1 1 1 4 4 1 4 1 1 1 3 5 7 2 5 2 7 6 6 7 2 2 4 4 7 3 2 1 3 6 5 6 5 6 6 1

6 6 6 2 1 1 2 1 4 1 2 4 4 1 7 7 7 6 7 7 7 2 3 4 4 7 2 6 2 4 5 4 5]
[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6
6 6 6 4 4 4 4 4 4 4 4 4 4 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5]
28/70
Random Forest Tree
[4 7 1 1 2 4 5 4 1 1 7 2 2 7 1 6 3 7 1 7 3 2 7 4 3 3 3 3 1 5 6 1 6 6 6 6 6
6 6 6 2 1 1 4 1 6 4 4 4 2 7 1 1 1 1 3 7 7 7 7 3 2 3 2 2 2 2 5 2 4]
[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6
6 6 6 4 4 4 4 4 4 4 4 4 4 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5]
30/70
Support Vector Machine
[5 7 1 1 5 4 2 4 1 1 7 4 5 2 2 6 3 7 1 4 7 3 2 5 3 7 3 3 5 5 6 5 6 6 6 6 1
6 6 6 2 4 2 5 6 7 4 5 5 4 7 5 1 1 1 4 7 7 7 7 2 4 3 4 2 2 4 4 4 5]
[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6
6 6 6 4 4 4 4 4 4 4 4 4 4 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5]
27/70
Neural Network
[1 7 7 1 4 1 4 2 1 1 1 2 2 6 2 6 3 7 1 7 6 3 3 4 3 6 3 3 4 4 6 7 6 6 6 6 1
6 6 6 1 4 7 2 7 7 2 4 4 3 7 7 7 1 1 2 7 7 1 6 3 3 3 2 7 1 2 5 2 4]
[1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 6 6 6 6 6 6 6
6 6 6 4 4 4 4 4 4 4 4 4 4 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5]
30/70