國立中興大學

人工智慧概論 期末專題

文本情緒分析：word2vec跟bert的比較

報告人: 王致雅
系級：資管三
學號：4110029009

内容

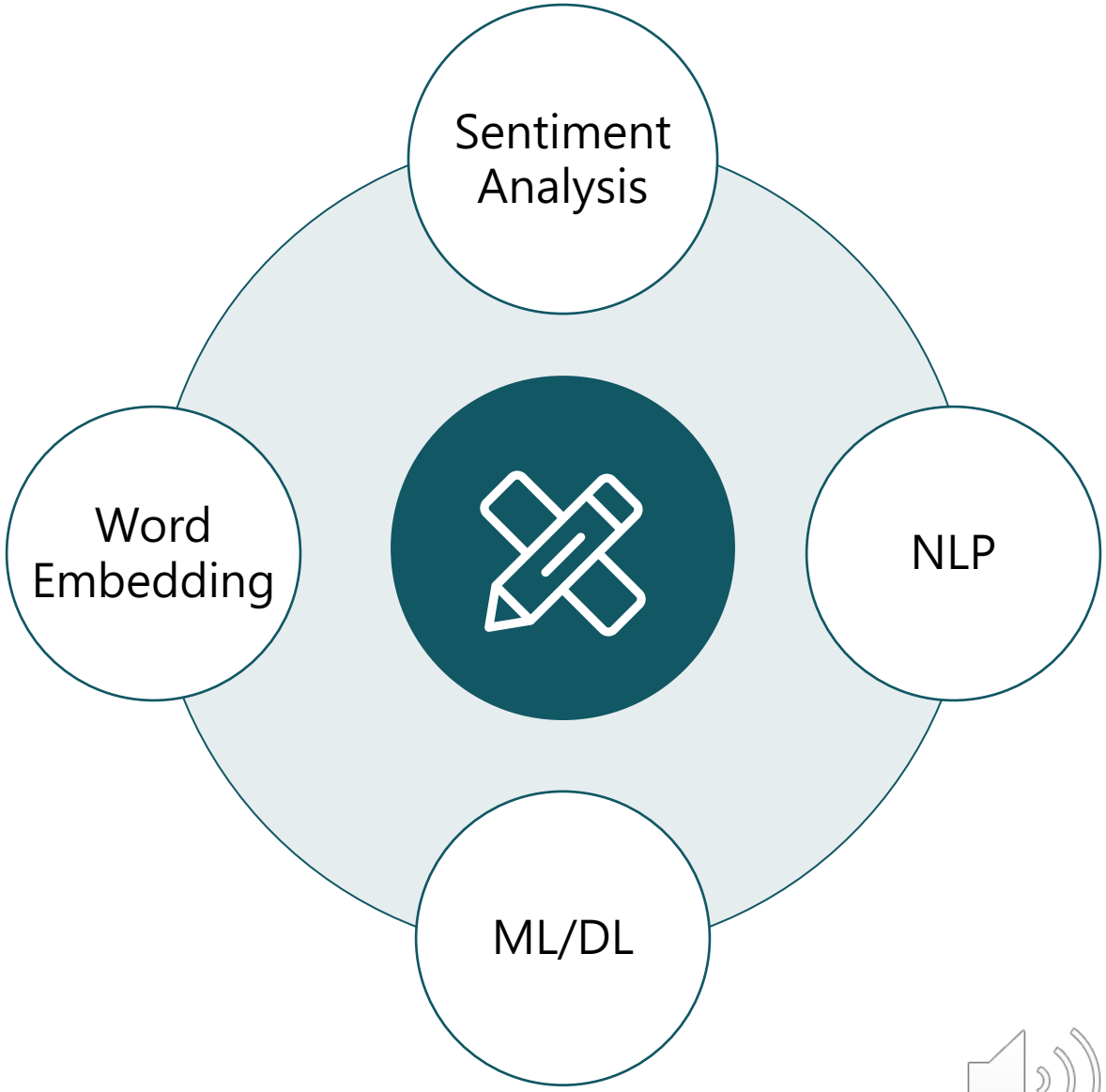
PART.01

介　　紹

研究內容

- 文本情緒分析是自然語言處理（NLP）中的一個重要的研究領域，情緒分析技術可以幫助我們從大量文本中提取出情感傾向。

- 在本次研究中，比較了兩種不同的文本特徵提取技術：Word2Vec和BERT，並使用多種機器學習和深度學習模型來進行情緒分析，探討它們的性能差異。



特徵提取方式

Word2Vec

- 基於詞嵌入的模型，用於將詞轉換為實數向量。它通過上下文詞來訓練，捕捉詞之間的語義關係。

BERT

- 基於 Transformer 的語言表示模型，它使用雙向的 Transformer 結構，可以更好地理解上下文中的詞義。該架構的核心特性之一就是自注意力機制。用來計算序列中每個元素（如詞語）與其他元素的關係，從而理解上下文語義。

採用模型

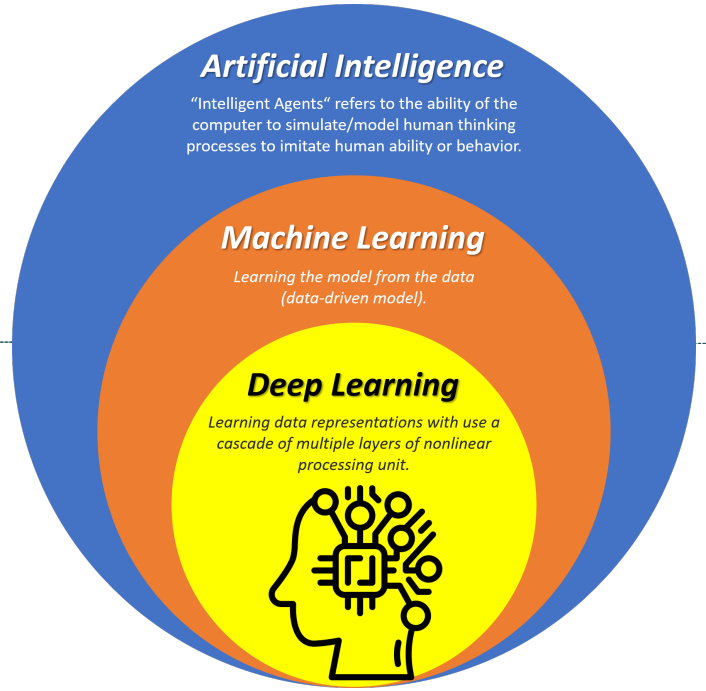
機器學習(ML)

Logistic Regression
簡單、高效、易於解釋，適合線性分離的數據

SVM
擁有良好的分類效果，特別是對於邊界清晰的數據。

Random Forest
基於多個決策樹的集成模型，能夠處理複雜特徵關係

XGBoost
高效的梯度提升樹模型，強大的處理能力。



深度學習(DL)

DNN
深層神經網絡能夠自動提取數據的非線性特徵。

RNN
適合分析時間序列數據或具有上下文關係的文本數據。

CNN
善於提取局部特徵，具有平移不變性。

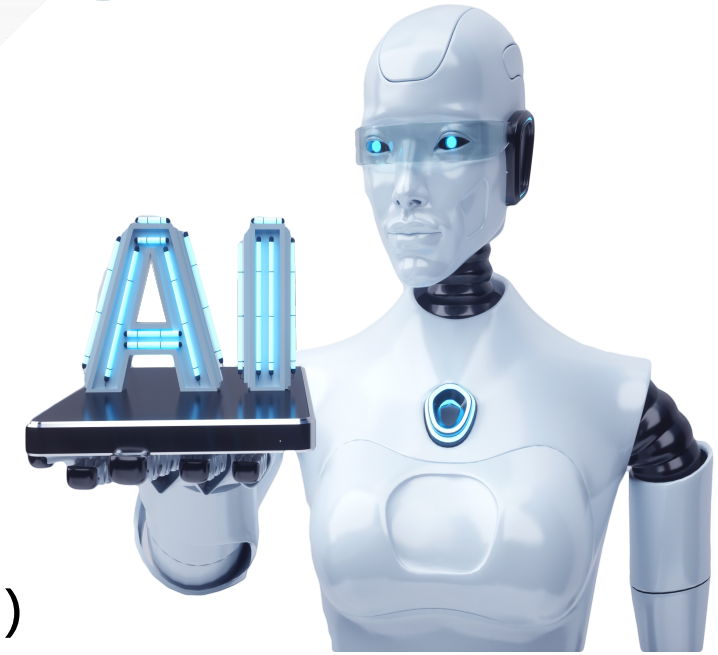
PART.02

研 究 方 法

02 研究方法（1/15）

特徵提取
（Word2Vec和BERT）

模型訓練
（機器學習和深度學習）



資料前處理
（清洗、分詞、轉換）

模型評估
（混淆矩陣、精確度、召回率和F1分數）

02

研究方法（2/15）

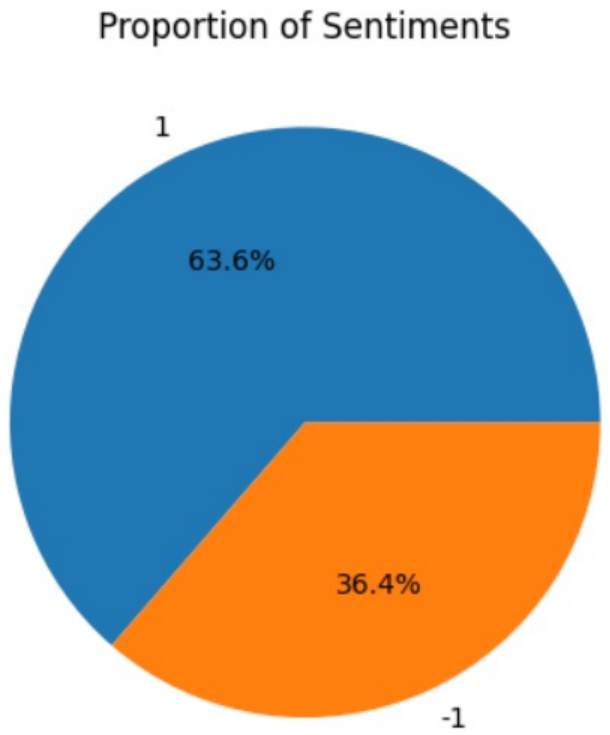
	Text	Sentiment
0	Kickers on my watchlist XIDE TIT SOQ PNK CPW B...	1
1	user: AAP MOVIE. 55% return for the FEA/GEED i...	1
2	user I'd be afraid to short AMZN – they are lo...	1
3	MNTA Over 12.00	1
4	OI Over 21.37	1
...	...	...
5786	Industry body CII said #discoms are likely to ...	–1
5787	#Gold prices slip below Rs 46,000 as #investor...	–1
5788	Workers at Bajaj Auto have agreed to a 10% wag...	1
5789	#Sharemarket LIVE: Sensex off day's high, up 6...	1
5790	#Sensex, #Nifty climb off day's highs, still u...	1

```
[ ] data.isnull().sum()
    #資料集無殘缺
```

```
Text          0
Sentiment     0
dtype: int64
```

```
data['Sentiment'].value_counts()
```

```
Sentiment
 1    3685
-1    2106
Name: count, dtype: int64
```



02 研究方法（3/15）

```
# 資料清理函數
def clean_text(text):
    text = text.lower()  # 轉小寫
    text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)  # 移除URL
    text = re.sub(r'\@w+|\#','', text)  # 移除@和#
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)  # 移除標點符號
    text = re.sub(r'\d+', '', text)  # 移除數字
    text = text.strip()  # 去除首尾空格
    return text
```

```
# 清理文本
data['Text'] = data['Text'].apply(clean_text)
```

將文本轉為小寫、使用正則表達式移除URL、移除 @符號 和 #符號、移除標點符號、移除數字、去除字串首尾的空白字符。

02 研究方法（4/15）

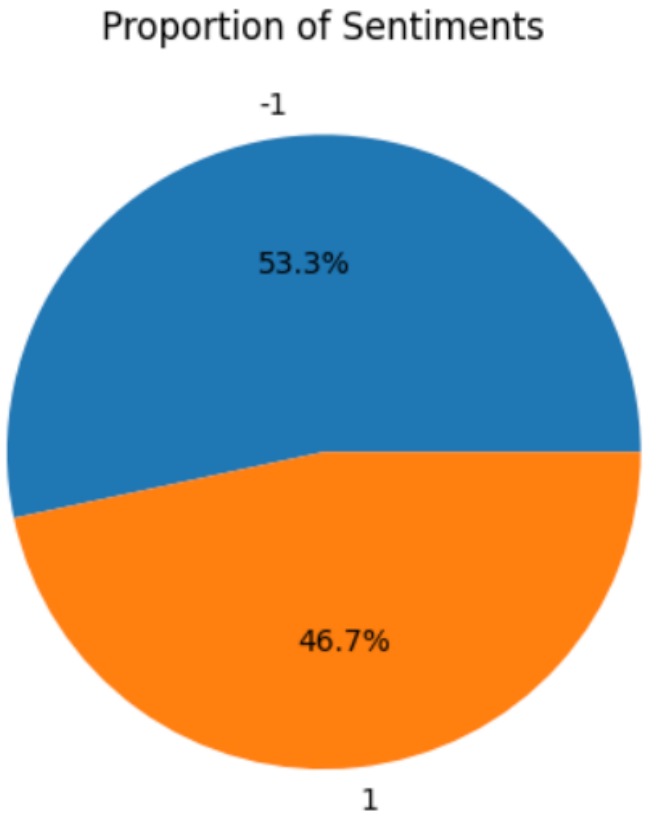
```python
# 同義詞替換增強函數
def synonym_replacement(sentence, n=1):
    words = sentence.split()
    new_words = words.copy()
    for _ in range(n):
        word_to_replace = random.choice(words)
        synonyms = wordnet.synsets(word_to_replace)
        if synonyms:
            synonym = random.choice(synonyms).lemmas()[0].name()
            new_words = [synonym if word == word_to_replace else word for word in new_words]
    return ' '.join(new_words)

# 打亂句子結構增強函數
def shuffle_sentence(sentence):
    words = sentence.split()
    random.shuffle(words)
    return ' '.join(words)

# 綜合增強函數
def augment_text(sentence):
    sentence = synonym_replacement(sentence, n=1)
    sentence = shuffle_sentence(sentence)
    return sentence
```

```python
# 綜合增強數據
augmented_texts = []
for text, emotion in zip(texts, emotions):
    if emotion == -1:  # 只對負情緒進行增強
        augmented_text = augment_text(text)
        augmented_texts.append({'Text': augmented_text, 'Sentiment': emotion})
```

因為負面情緒樣本較少，因此只對負面情緒做資料增強



02 研究方法（5/15）

```
# 分割訓練集和測試集
X_train, X_test, y_train, y_test = train_test_split(augmented_data['Text'], emotions_encoded, test_size=0.2, random_state=42)
```

Word2vec

```
from gensim.models import Word2Vec
import numpy as np

# 將文本標記化
tokenized_texts = [word_tokenize(text) for text in X_train]

# 訓練Word2Vec模型
w2v_model = Word2Vec(sentences=tokenized_texts, vector_size=300, window=10, min_count=1, workers=4)

# 生成詞向量的函數
def get_w2v_vector(text):
    tokens = word_tokenize(text)
    vector = np.mean([w2v_model.wv[token] for token in tokens if token in w2v_model.wv], axis=0)
    if isinstance(vector, np.ndarray):
        return vector
    else:
        return np.zeros(300)

# 將訓練集和測試集轉換為向量
X_train_w2v = np.array([get_w2v_vector(text) for text in X_train])
X_test_w2v = np.array([get_w2v_vector(text) for text in X_test])

# 如果有缺失值（全是未出現詞），則補零向量
X_train_w2v = np.nan_to_num(X_train_w2v)
X_test_w2v = np.nan_to_num(X_test_w2v)
```

```
X_train_w2v
```

```
array([[ 0.05409548,  0.35170597,  0.02849689, ..., -0.15267354,
         0.21513155, -0.06981268],
       [ 0.03072447,  0.22819868,  0.04118833, ..., -0.14038286,
         0.17107585, -0.05239639],
       [ 0.05599502,  0.38509396,  0.06511904, ..., -0.23589393,
         0.28587756, -0.09014736],
       ...,
       [ 0.03667165,  0.25665265,  0.04617003, ..., -0.15994772,
         0.19195423, -0.05939126],
       [ 0.05166622,  0.34984416,  0.05608506, ..., -0.20489849,
         0.254414  , -0.07955785],
       [ 0.00204742,  0.00801015,  0.00157459, ..., -0.00377869,
         0.00739131, -0.00156006]], dtype=float32)
```

02 研究方法（6/15）

Bert

```
# 加載BERT模型和標記器
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')
model.to(device)  # 將模型移動到 GPU

# 設置為評估模式
model.eval()
```

```
# 生成BERT詞向量的函數
def get_bert_vector(texts, batch_size=16):
    all_cls_embeddings = []

    for i in tqdm(range(0, len(texts), batch_size)):
        batch_texts = texts[i:i+batch_size]
        inputs = tokenizer(batch_texts.tolist(), return_tensors='pt', truncation=True, padding=True, max_length=128)
        inputs = {key: val.to(device) for key, val in inputs.items()}  # 移動到GPU

        with torch.no_grad():
            outputs = model(**inputs)

        cls_embeddings = outputs.last_hidden_state[:, 0, :].cpu().numpy()  # 移到CPU並轉換為numpy
        all_cls_embeddings.extend(cls_embeddings)

    return np.array(all_cls_embeddings)

# 將訓練集和測試集轉換為向量
X_train_bert = get_bert_vector(X_train)
X_test_bert = get_bert_vector(X_test)
```

```
X_train_bert
```

```
array([[-0.412744  , -0.06327974,  0.31435785, ..., -0.532389  ,
        -0.0357389 ,  0.09567103],
       [-0.2774287 ,  0.04613997,  0.0443465 , ..., -0.42565525,
         0.1341548 ,  0.5799537 ],
       [-0.23087057,  0.04627104,  0.28377265, ..., -0.16905557,
         0.28360623,  0.24268745],
       ...,
       [ 0.08582894, -0.01247747,  0.34396395, ..., -0.4801729 ,
         0.2709672 ,  0.27456674],
       [-0.17246386,  0.04487026, -0.03098492, ..., -0.57692903,
        -0.10229421,  0.41448888],
       [-0.10032072,  0.22234513, -0.0270679 , ..., -0.3075844 ,
         0.15486053,  0.25975654]], dtype=float32)
```

02 研究方法（7/15）

```python
# 訓練Logistic Regression模型 (Word2Vec)
clf_w2v = LogisticRegression(max_iter=1000)
clf_w2v.fit(X_train_w2v, y_train)
y_pred_w2v_lr = clf_w2v.predict(X_test_w2v)
```

```python
# 訓練Logistic Regression模型 (BERT)
clf_bert = LogisticRegression(max_iter=1000)
clf_bert.fit(X_train_bert, y_train)
y_pred_bert_lr = clf_bert.predict(X_test_bert)
```

```python
# 訓練RandomForest模型 (Word2Vec)
rf_w2v = RandomForestClassifier(n_estimators=100, random_state=42)
rf_w2v.fit(X_train_w2v, y_train)
```

```python
# 訓練RandomForest模型 (BERT)
rf_bert = RandomForestClassifier(n_estimators=100, random_state=42)
rf_bert.fit(X_train_bert, y_train)
```

```python
# 訓練SVM模型 (Word2Vec)
clf_svm_w2v = SVC(kernel='linear', C=1.0, random_state=42)
clf_svm_w2v.fit(X_train_w2v, y_train)
```

```python
# 訓練SVM模型 (BERT)
clf_svm_bert = SVC(kernel='linear', C=1.0, random_state=42)
clf_svm_bert.fit(X_train_bert, y_train)
```

```python
# 訓練XGBoost模型(Word2Vec)
clf_xgb_w2v = XGBClassifier(eval_metric='mlogloss', use_label_encoder=False)
clf_xgb_w2v.fit(X_train_w2v, y_train)
```

```python
# 訓練XGBoost模型 (BERT)
clf_xgb_bert = XGBClassifier()
clf_xgb_bert.fit(X_train_bert, y_train)
```

02

研究方法（8/15）

```python
# 创建MLP模型 (Word2Vec)
model_mlp_w2v = Sequential()
model_mlp_w2v.add(Dense(512, activation='relu', input_dim=300))
model_mlp_w2v.add(Dropout(0.5))
model_mlp_w2v.add(Dense(1, activation='sigmoid'))  # 输出层改为1，适用于二分类

# 编译模型
model_mlp_w2v.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# 定义早停策略
early_stopping = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)

# 训练模型
model_mlp_w2v.fit(X_train_w2v, y_train, epochs=30, batch_size=32, validation_data=(X_test_w2v, y_test), callbacks=[early_stopping])
```

```python
# 定義CNN模型
model_cnn_w2v = Sequential()
model_cnn_w2v.add(Conv1D(128, kernel_size=5, activation='relu', input_shape=(300, 1)))
model_cnn_w2v.add(MaxPooling1D(pool_size=2))
model_cnn_w2v.add(Flatten())
model_cnn_w2v.add(Dense(128, activation='relu'))
model_cnn_w2v.add(Dense(1, activation='sigmoid'))
# 編譯模型
model_cnn_w2v.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
# 轉為三維輸入
X_train_w2v_cnn = np.expand_dims(X_train_w2v, axis=2)
X_test_w2v_cnn = np.expand_dims(X_test_w2v, axis=2)
# 訓練模型
model_cnn_w2v.fit(X_train_w2v_cnn, y_train, epochs=10, batch_size=32, validation_data=(X_test_w2v_cnn, y_test))
```

```python
# Reshape，使其適應 RNN 輸入要求
X_train_w2v_reshaped = np.expand_dims(X_train_w2v, axis=1)
X_test_w2v_reshaped = np.expand_dims(X_test_w2v, axis=1)
```

```python
# 定義RNN模型
model_w2v_rnn = Sequential()
model_w2v_rnn.add(SimpleRNN(128, input_shape=(1, 300), dropout=0.2, recurrent_dropout=0.2))
model_w2v_rnn.add(Dropout(0.2))
model_w2v_rnn.add(Dense(2, activation='softmax'))
# 編譯模型
model_w2v_rnn.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
# 訓練模型
model_w2v_rnn.fit(X_train_w2v_reshaped, y_train_categorical, epochs=30, batch_size=32, validation_split=0.2)
```

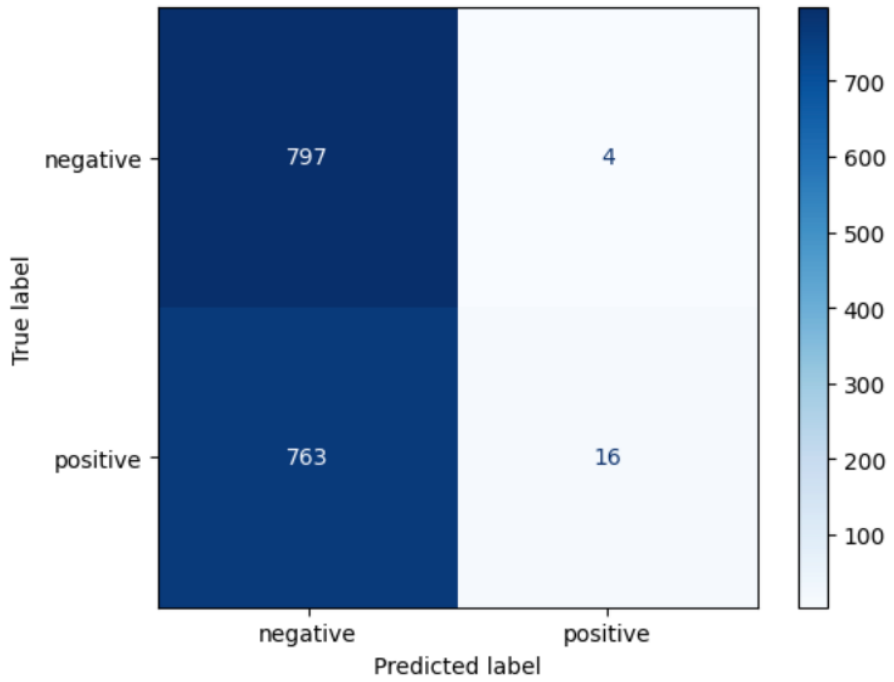
研究方法（9/15）

使用LogisticRegression

Classification Report (Word2Vec + LogisticRegression):

	precision	recall	f1-score	support
negative	0.51	1.00	0.68	801
positive	0.80	0.02	0.04	779
accuracy			0.51	1580
macro avg	0.66	0.51	0.36	1580
weighted avg	0.65	0.51	0.36	1580

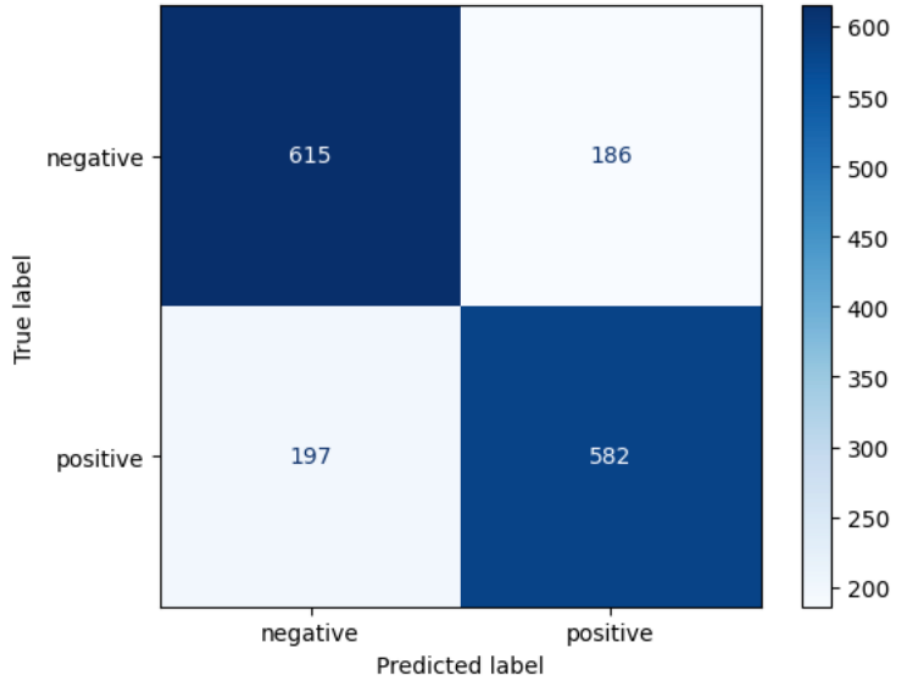
Confusion Matrix (Word2Vec + LogisticRegression):



Classification Report (BERT + LogisticRegression):

	precision	recall	f1-score	support
negative	0.76	0.77	0.76	801
positive	0.76	0.75	0.75	779
accuracy			0.76	1580
macro avg	0.76	0.76	0.76	1580
weighted avg	0.76	0.76	0.76	1580

Confusion Matrix (BERT + LogisticRegression):



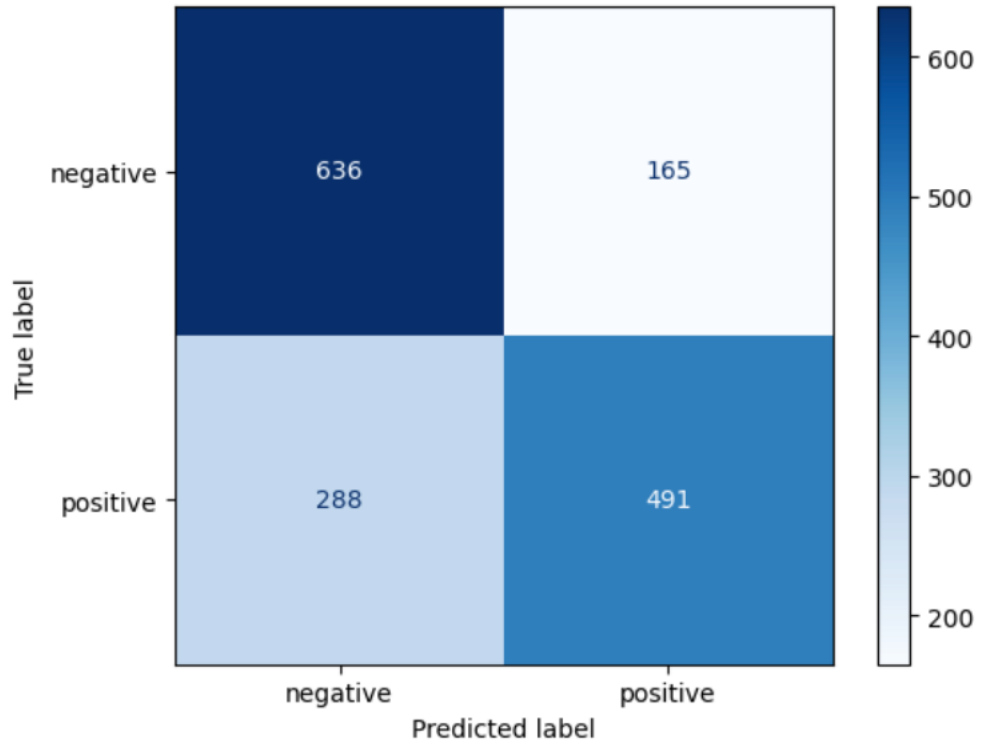
研究方法（10/15）

使用RandomForest

Classification Report (Word2Vec + Random Forest):

	precision	recall	f1-score	support
negative	0.69	0.79	0.74	801
positive	0.75	0.63	0.68	779
accuracy			0.71	1580
macro avg	0.72	0.71	0.71	1580
weighted avg	0.72	0.71	0.71	1580

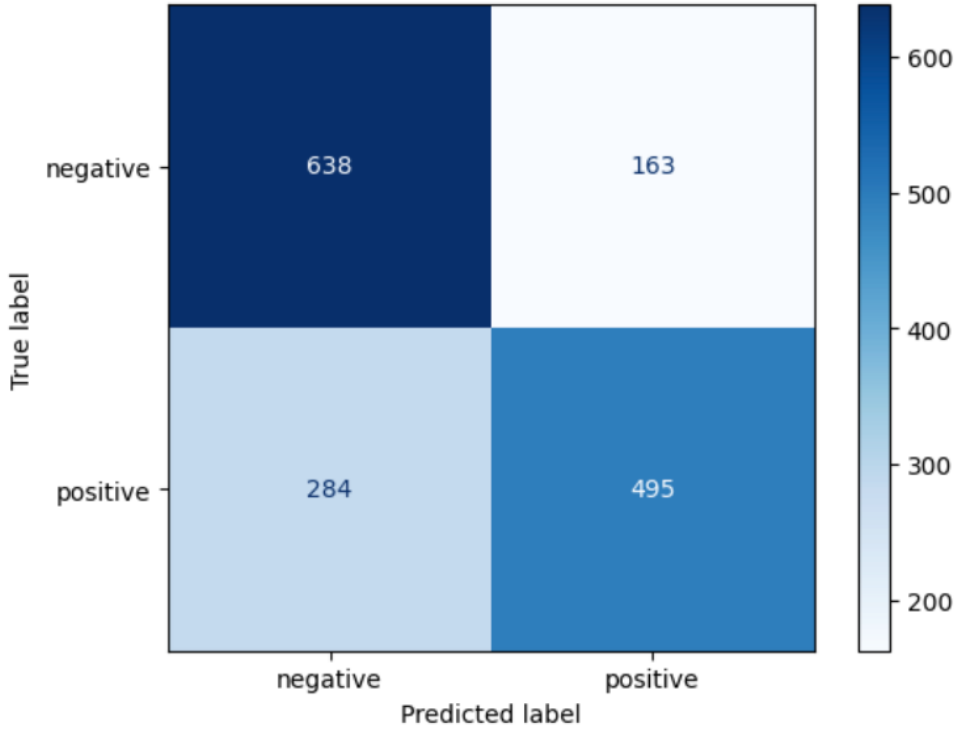
Confusion Matrix (Word2Vec + Random Forest):



Classification Report (BERT + Random Forest):

	precision	recall	f1-score	support
negative	0.69	0.80	0.74	801
positive	0.75	0.64	0.69	779
accuracy			0.72	1580
macro avg	0.72	0.72	0.71	1580
weighted avg	0.72	0.72	0.72	1580

Confusion Matrix (BERT + Random Forest):



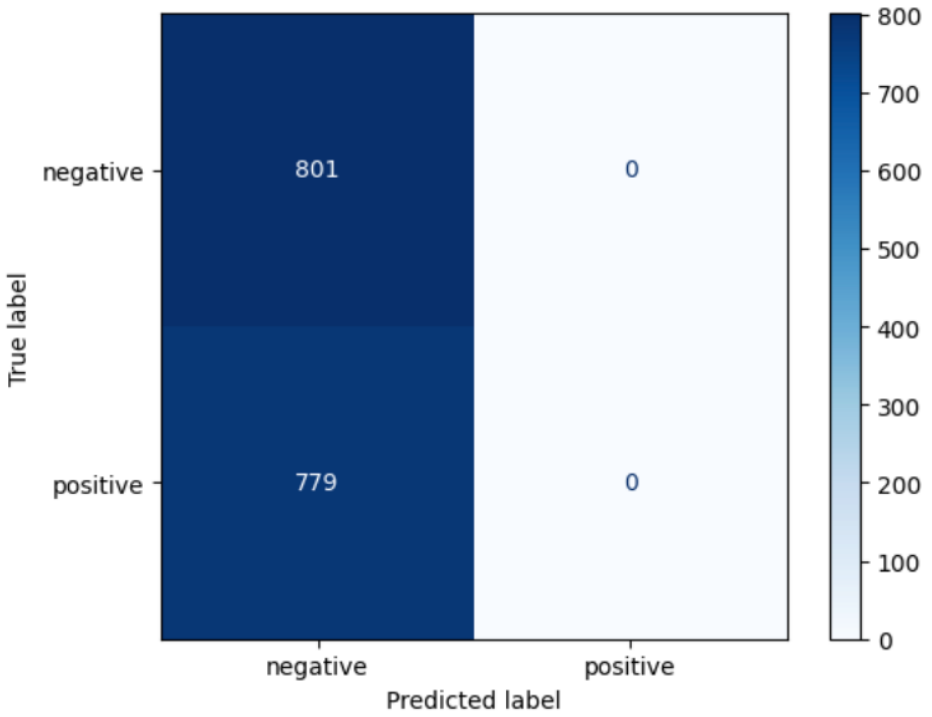
研究方法（11/15）

使用SVM

Classification Report (Word2Vec + SVM):

	precision	recall	f1-score	support
negative	0.51	1.00	0.67	801
positive	0.00	0.00	0.00	779
accuracy			0.51	1580
macro avg	0.25	0.50	0.34	1580
weighted avg	0.26	0.51	0.34	1580

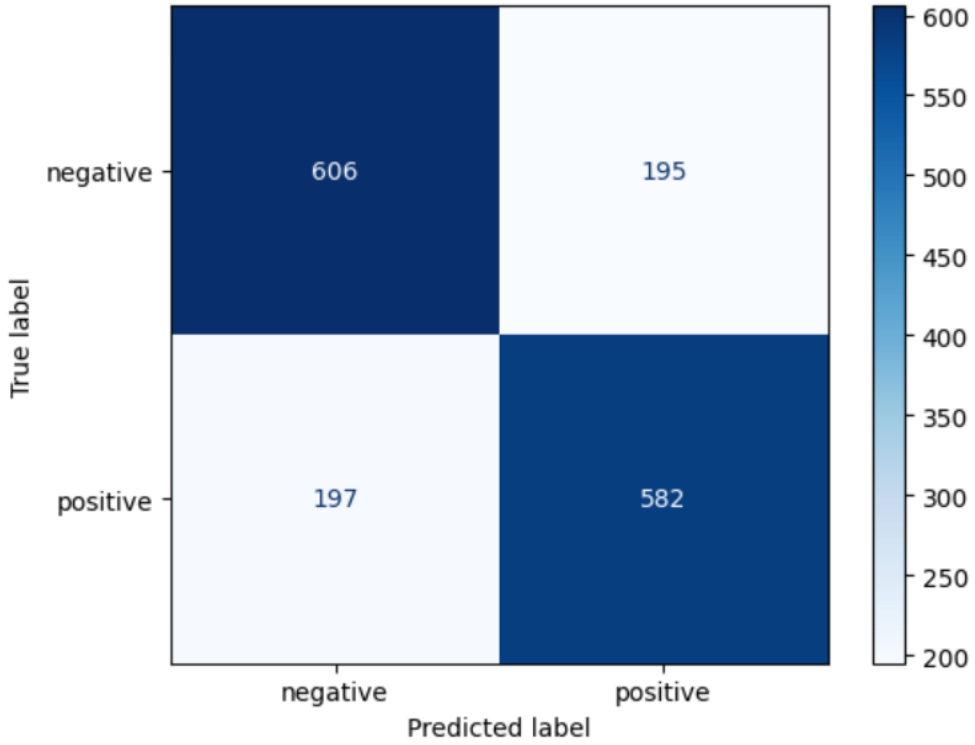
Confusion Matrix (Word2Vec + SVM):



Classification Report (BERT + SVM):

	precision	recall	f1-score	support
negative	0.75	0.76	0.76	801
positive	0.75	0.75	0.75	779
accuracy			0.75	1580
macro avg	0.75	0.75	0.75	1580
weighted avg	0.75	0.75	0.75	1580

Confusion Matrix (BERT + SVM):



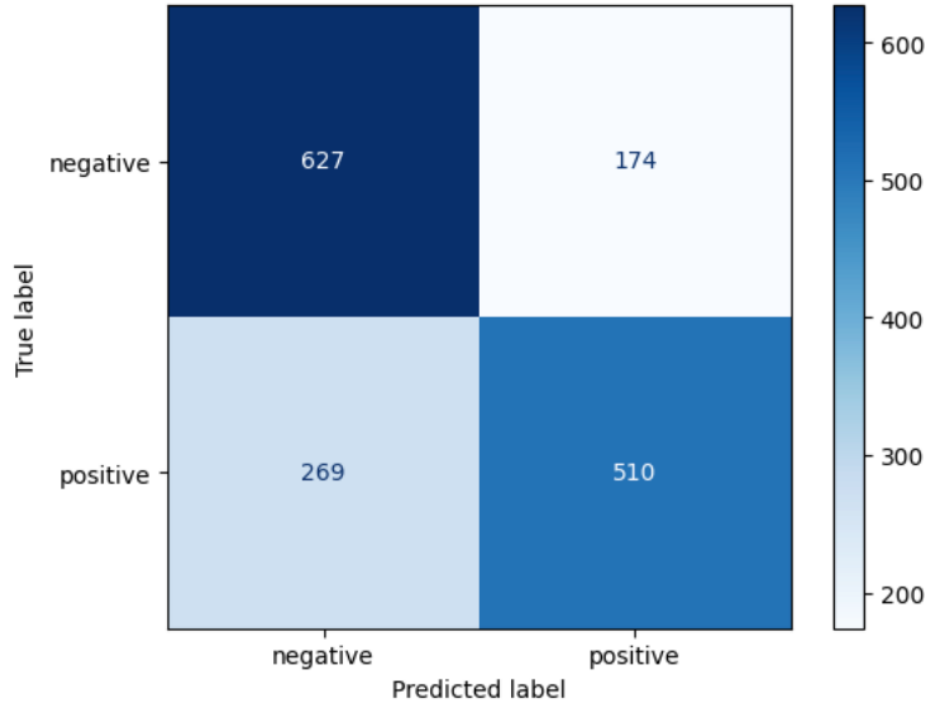
研究方法（12/15）

使用XGBoost

Classification Report (Word2Vec + XGBoost):

	precision	recall	f1-score	support
negative	0.70	0.78	0.74	801
positive	0.75	0.65	0.70	779
accuracy			0.72	1580
macro avg	0.72	0.72	0.72	1580
weighted avg	0.72	0.72	0.72	1580

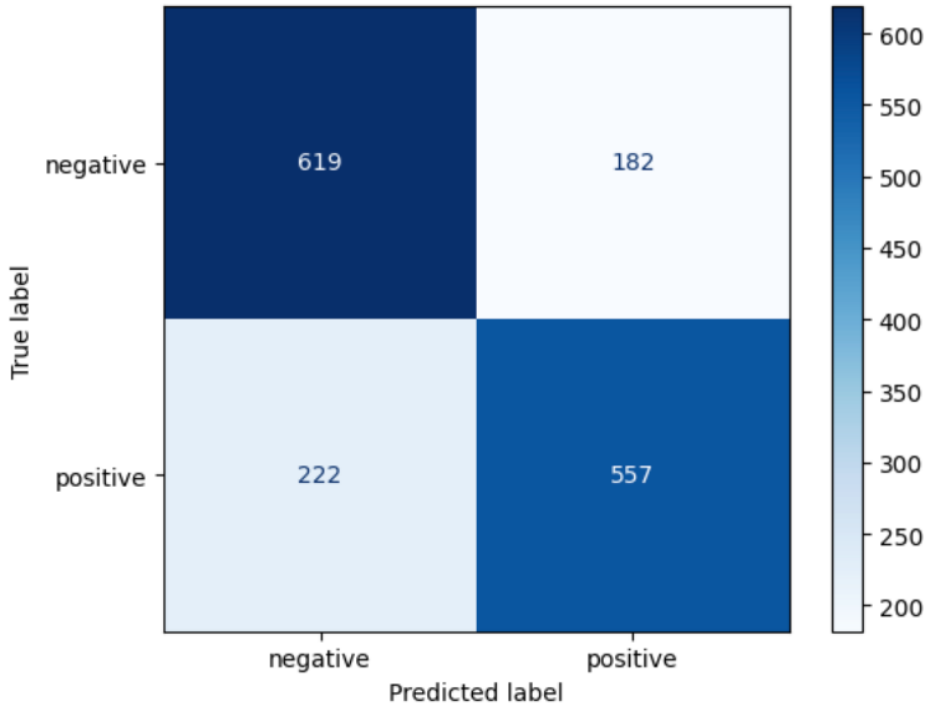
Confusion Matrix (Word2Vec + XGBoost):



Classification Report (BERT + XGBoost):

	precision	recall	f1-score	support
negative	0.74	0.77	0.75	801
positive	0.75	0.72	0.73	779
accuracy			0.74	1580
macro avg	0.74	0.74	0.74	1580
weighted avg	0.74	0.74	0.74	1580

Confusion Matrix (BERT + XGBoost):



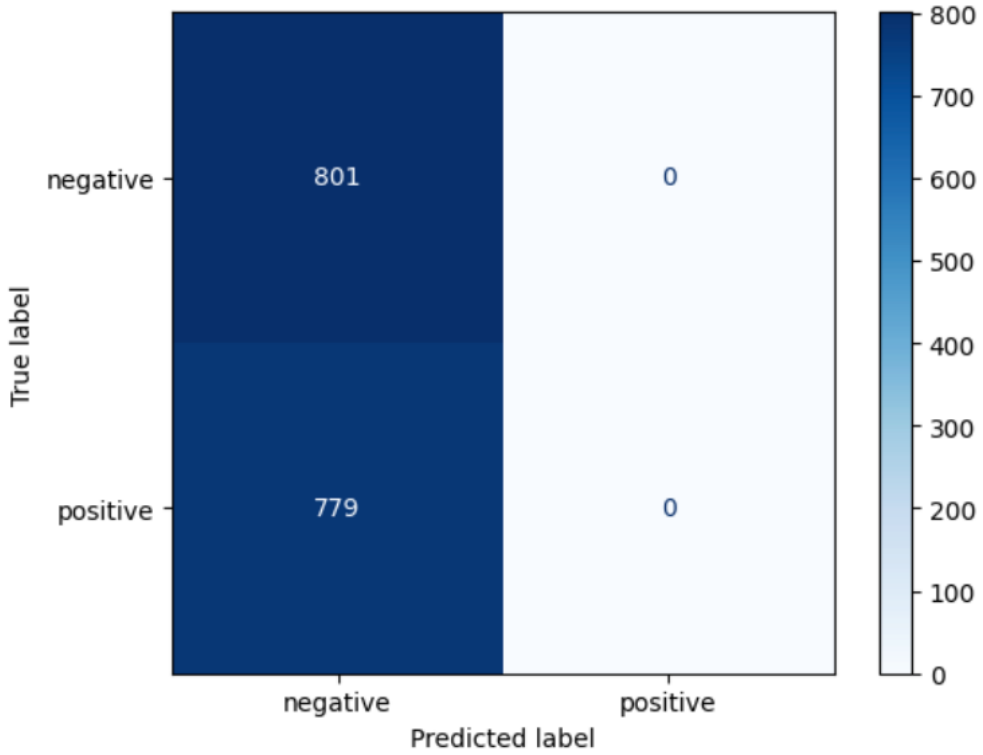
研究方法（13/15）

使用DNN

Classification Report (Word2Vec + DNN):

	precision	recall	f1-score	support
negative	0.51	1.00	0.67	801
positive	0.00	0.00	0.00	779
accuracy			0.51	1580
macro avg	0.25	0.50	0.34	1580
weighted avg	0.26	0.51	0.34	1580

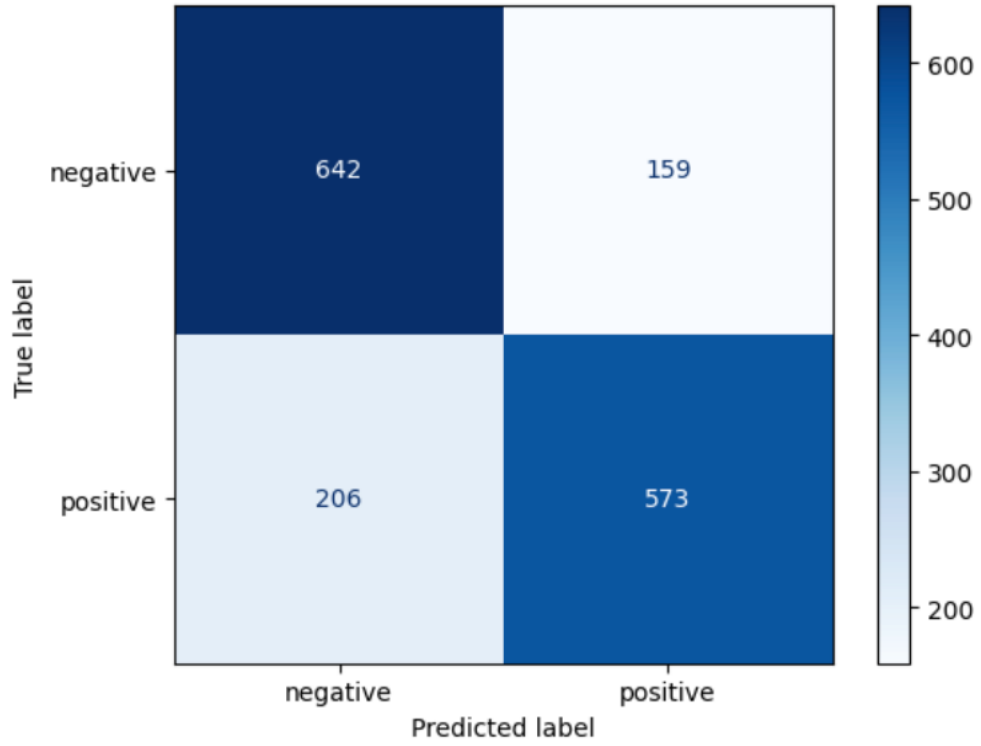
Confusion Matrix (Word2Vec + DNN):



Classification Report (BERT + DNN):

	precision	recall	f1-score	support
negative	0.76	0.80	0.78	801
positive	0.78	0.74	0.76	779
accuracy			0.77	1580
macro avg	0.77	0.77	0.77	1580
weighted avg	0.77	0.77	0.77	1580

Confusion Matrix (BERT + DNN):

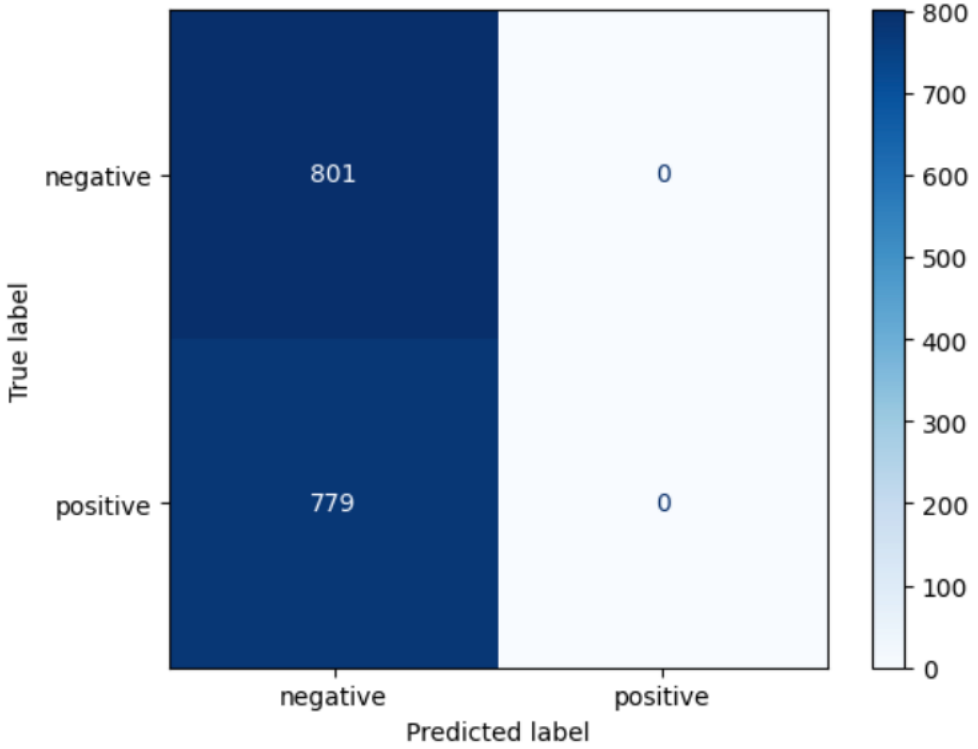


研究方法（14/15）

使用CNN

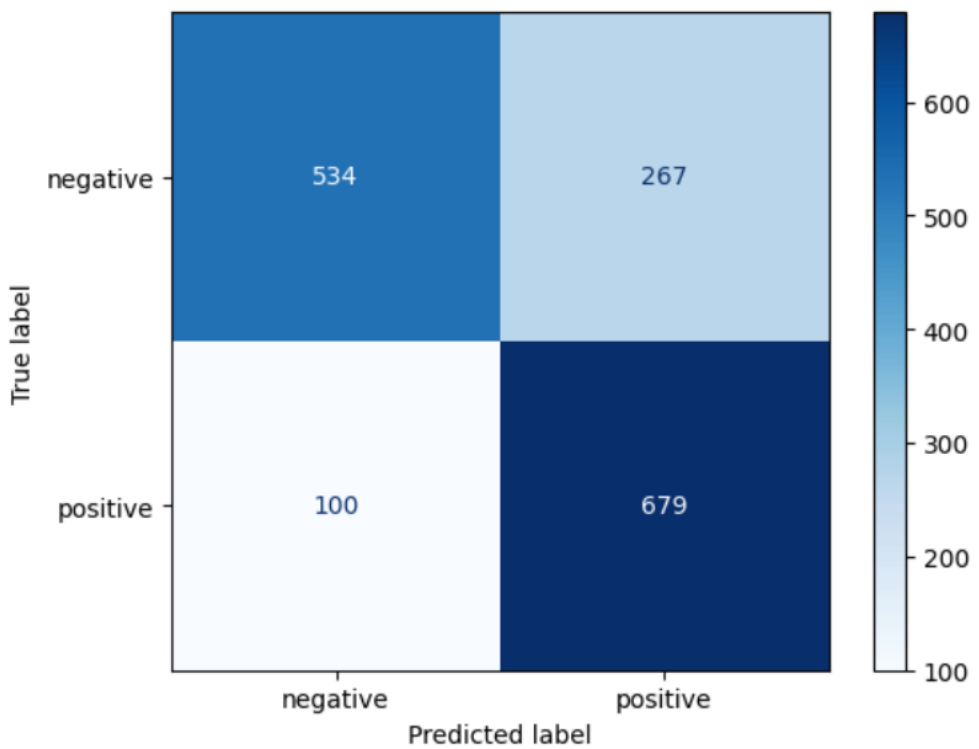
Classification Report (CNN + Word2Vec):

	precision	recall	f1-score	support
negative	0.51	1.00	0.67	801
positive	0.00	0.00	0.00	779
accuracy			0.51	1580
macro avg	0.25	0.50	0.34	1580
weighted avg	0.26	0.51	0.34	1580



Classification Report (BERT + CNN):

	precision	recall	f1-score	support
negative	0.8423	0.6667	0.7443	801
positive	0.7178	0.8716	0.7872	779
accuracy			0.7677	1580
macro avg	0.7800	0.7691	0.7657	1580
weighted avg	0.7809	0.7677	0.7654	1580



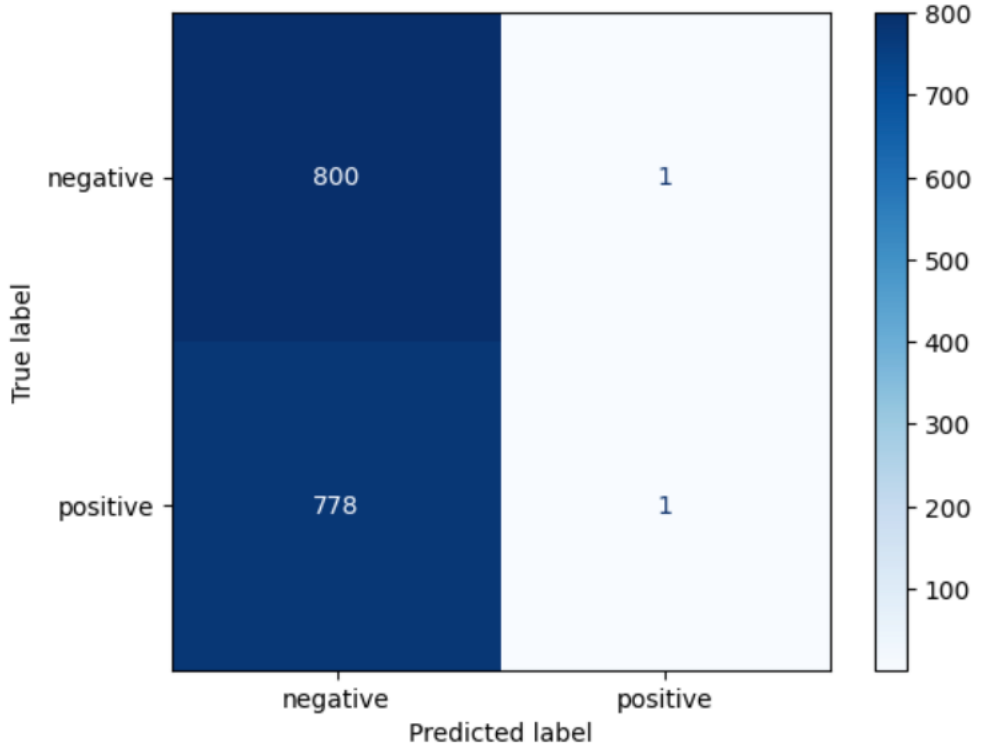
研究方法（15/15）

使用RNN

Classification Report (Word2Vec + RNN):

	precision	recall	f1-score	support
negative	0.51	1.00	0.67	801
positive	0.50	0.00	0.00	779
accuracy			0.51	1580
macro avg	0.50	0.50	0.34	1580
weighted avg	0.50	0.51	0.34	1580

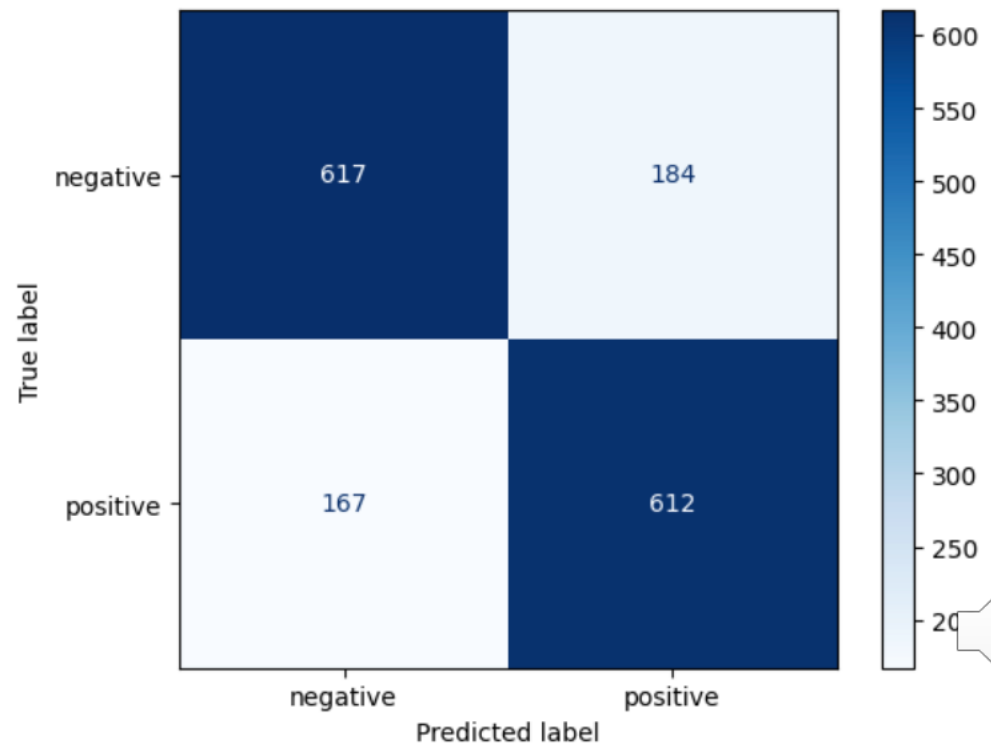
Confusion Matrix (Word2Vec + RNN):



Classification Report (BERT + RNN):

	precision	recall	f1-score	support
negative	0.79	0.77	0.78	801
positive	0.77	0.79	0.78	779
accuracy			0.78	1580
macro avg	0.78	0.78	0.78	1580
weighted avg	0.78	0.78	0.78	1580

Confusion Matrix (BERT + RNN):



PART.03

結 果 分 析

03 結果分析(1/2)

Word2Vec模型	準確率	主要問題
Logistic Regression	0.51	對於正面情緒的識別效果極差，精確率和F1得分都很低。
Random Forest	0.71	相對於Logistic Regression有所提升，對負面情緒的識別效果較好。
SVM	0.51	無法有效識別正面情緒，召回率和F1得分均為0。
XGBoost	0.72	對於負面情緒和正面情緒的識別效果均較為穩定，整體表現良好。
DNN	0.51	無法有效識別正面情緒，整體性能與Logistic Regression和SVM類似。
CNN	0.51	無法有效識別正面情緒，整體性能較差。
RNN	0.51	無法有效識別正面情緒，整體性能不佳。

03 結果分析(2/2)

BERT模型	準確率	主要問題
Logistic Regression	0.76	對於負面情緒和正面情緒的識別效果均較為穩定，整體性能較好。
Random Forest	0.72	對於負面情緒和正面情緒的識別效果均相對穩定，與Logistic Regression接近。
SVM	0.75	對於負面情緒和正面情緒的識別效果較好，比Logistic Regression和Random Forest有所提升。
XGBoost	0.74	對於負面情緒和正面情緒的識別效果較為穩定，整體性能表現良好。
DNN	0.77	對於負面情緒和正面情緒的識別效果均較好，整體性能優於基於Word2Vec的DNN模型。
CNN	0.77	對於負面情緒和正面情緒的識別效果較為均衡，整體性能較穩定。
RNN	0.78	對於負面情緒和正面情緒的識別效果均較好，整體性能優於基於Word2Vec的RNN模型。

PART.04

結　　論

04 結論(1/1)

01.模型性能比較

Word2Vec

- ✓ 較低的性能
- ✓ 準確度和F1分數較低
- ✓ 捕捉正面情感方面的表現最差

BERT

- ✓ 表現穩定且優異
- ✓ 高準確度、較高的F1分數
- ✓ 良好的混淆矩陣結果
- ✓ 更好地捕捉文本的語境和情感

02. Precision & Recall

BERT在情感分析任務中展示出更平衡的精確度和召回率，這表明其對於正負兩類情感的識別能力較為均衡和準確。

03.模型穩定性

在各種模型中，BERT的性能相對穩定，不受不同模型架構的影響過大，這表明BERT在處理不同情感分析任務時有較好的泛化能力和穩定性。

E N D