

Analysis

December 21, 2021

1 Importing Packages

```
[1]: # Load packages for data wrangling:
import os
import glob
import numpy as np
import pandas as pd

# Load packages for fine-tuning BERT model:
from simpletransformers.classification import ClassificationModel

# Load scikit-learn train_test_split:
from sklearn.model_selection import train_test_split

# Load classification metrics:
from sklearn.metrics import (accuracy_score, recall_score, precision_score,
                             f1_score,
                             classification_report, confusion_matrix)

# Load softmax for converting raw model output to probabilities:
from scipy.special import softmax

# Load packages for data cleaning:
import string
import re
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk import pos_tag
from nltk import sent_tokenize, word_tokenize

# Set stopword corpus
stopword = nltk.corpus.stopwords.words('english')
```

```
# Set NLTK lemmatizer
lemmatizer = WordNetLemmatizer()
```

```
[nltk_data] Downloading package stopwords to /home/ucloud/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /home/ucloud/nltk_data...
[nltk_data] Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to /home/ucloud/nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data] /home/ucloud/nltk_data...
[nltk_data] Package averaged_perceptron_tagger is already up-to-
[nltk_data] date!
```

2 Preprocessing of dataset 1

2.1 Data loading and wrangling

```
[2]: # Loading data with fake news as pandas dataframe:
fake_df = pd.read_csv(os.path.join("data", "dataset_1", "Fake.csv"))

# Loading data with true news as pandas dataframe:
true_df = pd.read_csv(os.path.join("data", "dataset_1", "True.csv"))
```

```
[3]: # Inspecting fake data:
fake_df.head(10)
```

```
[3]:
```

	title \
0	Donald Trump Sends Out Embarrassing New Year'...
1	Drunk Bragging Trump Staffer Started Russian ...
2	Sheriff David Clarke Becomes An Internet Joke...
3	Trump Is So Obsessed He Even Has Obama's Name...
4	Pope Francis Just Called Out Donald Trump Dur...
5	Racist Alabama Cops Brutalize Black Boy While...
6	Fresh Off The Golf Course, Trump Lashes Out A...
7	Trump Said Some INSANELY Racist Stuff Inside ...
8	Former CIA Director Slams Trump Over UN Bully...
9	WATCH: Brand-New Pro-Trump Ad Features So Muc...

	text	subject \
0	Donald Trump just couldn t wish all Americans ...	News
1	House Intelligence Committee Chairman Devin Nu...	News
2	On Friday, it was revealed that former Milwauk...	News
3	On Christmas day, Donald Trump announced that ...	News
4	Pope Francis used his annual Christmas Day mes...	News
5	The number of cases of cops brutalizing and ki...	News

```

6 Donald Trump spent a good portion of his day a... News
7 In the wake of yet another court decision that... News
8 Many people have raised the alarm regarding th... News
9 Just when you might have thought we d get a br... News

```

```

                                date
0 December 31, 2017
1 December 31, 2017
2 December 30, 2017
3 December 29, 2017
4 December 25, 2017
5 December 25, 2017
6 December 23, 2017
7 December 23, 2017
8 December 22, 2017
9 December 21, 2017

```

```

[4]: # Inspecting true data:
true_df.head(10)

```

```

[4]:                                     title \
0 As U.S. budget fight looms, Republicans flip t...
1 U.S. military to accept transgender recruits o...
2 Senior U.S. Republican senator: 'Let Mr. Muell...
3 FBI Russia probe helped by Australian diplomat...
4 Trump wants Postal Service to charge 'much mor...
5 White House, Congress prepare for talks on spe...
6 Trump says Russia probe will be fair, but time...
7 Factbox: Trump on Twitter (Dec 29) - Approval ...
8 Trump on Twitter (Dec 28) - Global Warming
9 Alabama official to certify Senator-elect Jone...

```

```

                                text      subject \
0 WASHINGTON (Reuters) - The head of a conservat... politicsNews
1 WASHINGTON (Reuters) - Transgender people will... politicsNews
2 WASHINGTON (Reuters) - The special counsel inv... politicsNews
3 WASHINGTON (Reuters) - Trump campaign adviser ... politicsNews
4 SEATTLE/WASHINGTON (Reuters) - President Donal... politicsNews
5 WEST PALM BEACH, Fla./WASHINGTON (Reuters) - T... politicsNews
6 WEST PALM BEACH, Fla (Reuters) - President Don... politicsNews
7 The following statements were posted to the ve... politicsNews
8 The following statements were posted to the ve... politicsNews
9 WASHINGTON (Reuters) - Alabama Secretary of St... politicsNews

```

```

                                date
0 December 31, 2017
1 December 29, 2017

```

```
2 December 31, 2017
3 December 30, 2017
4 December 29, 2017
5 December 29, 2017
6 December 29, 2017
7 December 29, 2017
8 December 29, 2017
9 December 28, 2017
```

```
[5]: # Adding category-labels to each dataset:
```

```
fake_df["label"]="fake"
true_df["label"]="true"
```

```
[6]: # Merge fake- and true news into a single dataframe:
```

```
merged_df = pd.concat([true_df, fake_df])
```

```
[7]: # Assessing whether merge was succesful:
```

```
len(true_df) + len(fake_df) == len(merged_df)
```

```
[7]: True
```

2.2 Data cleaning

2.2.1 Removing bad rows

```
[8]: # Remove rows with only whitespace and replace it with NaN:
```

```
merged_df.replace(" ", float("NaN"), inplace=True)
```

```
# Remove NA's:
```

```
merged_df.dropna(subset = ["text"], inplace=True)
```

```
[9]: # Remove duplicate texts:
```

```
merged_df = merged_df.drop_duplicates(subset=['text'])
```

```
[10]: # Reset indices:
```

```
merged_df = merged_df.reset_index()
```

```
[11]: # Selecting only relevant columns:
```

```
merged_df = merged_df[["text", "label"]]
```

2.2.2 Regex

Remove “[city name] Reuters -” from true articles

```
[12]: # Define regex pattern:
```

```
pattern = r".*\(Reuters\) - "
```

```
for i in range(len(merged_df['text'])):
```

```
merged_df['text'][i] = re.sub(pattern, '', merged_df['text'][i])
```

Remove hashtags

```
[13]: # Define regex pattern:
pattern = r"#(\S+)"

for i in range(len(merged_df['text'])):
    merged_df['text'][i] = re.sub(pattern, '', merged_df['text'][i])
```

Remove twitter tags (“@[username]”)

```
[14]: # Define regex pattern:
pattern = r"@(\S+)"

for i in range(len(merged_df['text'])):
    merged_df['text'][i] = re.sub(pattern, '', merged_df['text'][i])
```

Remove ‘(CAPSLOCK)’ e.g. from (VIDEO); something which was quite frequent in the fake news dataset

```
[15]: # Define regex pattern:
pattern = r"\([A-Z]*\)"

for i in range(len(merged_df['text'])):
    merged_df['text'][i] = re.sub(pattern, '', merged_df['text'][i])
```

Remove systematic patterns:

```
[16]: # Define regex pattern:
pattern = r"The following statement.*accuracy[.]"

for i in range(len(merged_df['text'])):
    merged_df['text'][i] = re.sub(pattern, '', merged_df['text'][i])
```

```
[17]: # Define regex pattern:
pattern = r"pic\.twitter\.com\/.* "

for i in range(len(merged_df['text'])):
    merged_df['text'][i] = re.sub(pattern, '', merged_df['text'][i])
```

2.2.3 Remove punctuation

```
[17]: # Define function:
def remove_punctuation(text):
    no_punct=[words for words in text if words not in string.punctuation]
    words_wo_punct=' '.join(no_punct)
    return words_wo_punct
```

```
[19]: merged_df['text']=merged_df['text'].apply(lambda x: remove_punctuation(x))
```

2.2.4 Tokenization + Lower

```
[18]: # Define function:
def tokenize(text):
    split=re.split("\W+",text)
    return split
```

```
[21]: merged_df['tokenized']=merged_df['text'].apply(lambda x: tokenize(x.lower()))
```

2.2.5 Remove stopwords

```
[19]: # Define function:
def remove_stopwords(text):
    text=[words for words in text if words not in stopwords]
    #text=' '.join(text)
    return text
```

```
[23]: merged_df['tokenized'] = merged_df['tokenized'].apply(lambda x:
    ↪remove_stopwords(x))
```

2.2.6 Lemmatize

```
[20]: # Define function:
def penn2morphy(penntag):
    morphy_tag = {'NN':'n', 'JJ':'a',
                  'VB':'v', 'RB':'r'}

    try:
        return morphy_tag[penntag[:2]]
    except:
        return 'n'
```

```
[25]: for i in range(len(merged_df['tokenized'])):
    tagged = pos_tag(merged_df['tokenized'][i])
    merged_df['tokenized'][i] = [lemmatizer.lemmatize(word,
    ↪pos=penn2morphy(tag)) for word, tag in tagged]
```

2.2.7 Concatenate tokens into sentences

```
[21]: # Define function:
def concat(text):
    text=[words for words in text]
    text=' '.join(text)
    return text
```

```
[27]: merged_df['text'] = merged_df['tokenized'].apply(lambda x: concat(x))
```

2.2.8 Remove newly induced empty columns

```
[28]: merged_df.replace(" ", float("NaN"), inplace=True)

merged_df.dropna(subset = ["text"], inplace=True)
```

```
[29]: merged_df = merged_df.reset_index()
```

2.2.9 Assess whether we have missed anything

```
[30]: true_idx = merged_df[merged_df['label']=="true"].index.tolist()
fake_idx = merged_df[merged_df['label']=="fake"].index.tolist()
```

```
[31]: from collections import Counter
Counter(" ".join(merged_df['text'][true_idx]).split()).most_common(10)
```

```
[31]: [('say', 113426),
      ('trump', 53621),
      ('u', 40552),
      ('state', 36143),
      ('would', 31145),
      ('president', 26582),
      ('republican', 20154),
      ('government', 19171),
      ('year', 18520),
      ('house', 16787)]
```

```
[32]: from collections import Counter
Counter(" ".join(merged_df['text'][fake_idx]).split()).most_common(10)
```

```
[32]: [('trump', 58413),
      ('say', 36515),
      ('people', 19204),
      ('president', 18091),
      ('go', 17802),
      ('would', 17078),
      ('make', 16956),
      ('one', 16919),
      ('state', 16195),
      ('get', 14812)]
```

2.2.10 Remove newly found systematic patterns

```
[33]: # Define regex pattern:
      pattern = r"21st century wire say"

      for i in range(len(merged_df['text'])):
          merged_df['text'][i] = re.sub(pattern, '', merged_df['text'][i])

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
"""

[34]: # Define regex pattern:
      pattern = r"21st century wire"

      for i in range(len(merged_df['text'])):
          merged_df['text'][i] = re.sub(pattern, '', merged_df['text'][i])

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
"""

[35]: # Define regex pattern:
      pattern = r"filessupport.*"

      for i in range(len(merged_df['text'])):
          merged_df['text'][i] = re.sub(pattern, '', merged_df['text'][i])

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
"""

[36]: # Define regex pattern:
      pattern = r"21wire"

      for i in range(len(merged_df['text'])):
```



```
merged_df['text'][i] = re.sub(pattern, '', merged_df['text'][i])
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:5:
```

```
SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
"""
```

```
[37]: from collections import Counter
Counter(" ".join(merged_df['text'][fake_idx]).split()).most_common(10)
```

```
[37]: [('trump', 58413),
      ('say', 36151),
      ('people', 19204),
      ('president', 18091),
      ('go', 17802),
      ('would', 17078),
      ('make', 16956),
      ('one', 16919),
      ('state', 16195),
      ('get', 14812)]
```

2.3 Saving and loading cleaned dataset 1

2.3.1 Write dataframe to csv-file

```
[38]: # Selecting only relevant columns
merged_df = merged_df[["text", "label"]]
```

```
[39]: # Write to csv
merged_df.to_csv(os.path.join("data", "generated_data", "cleaned_dataset_1.
↪ csv"), index=False)
```

3 BERT trained- and evaluated on dataset 1

3.1 Load cleaned data and prepare for classification

```
[60]: cleaned_dataset_1 = pd.read_csv(os.path.join("data", "generated_data",
↪ "cleaned_dataset_1.csv"))
```

One row is corrupted when loading CSV and is turned into blank space. This is removed

```
[61]: cleaned_dataset_1.replace(" ", float("NaN"), inplace=True)
```

```
cleaned_dataset_1.dropna(subset = ["text"], inplace=True)
```

Create training-, validation and testing dataset:

```
[103]: # Create train/test split with 20% of all articles in testing data:
train_1, test_1 = train_test_split(cleaned_dataset_1, test_size=0.2)
```

```
[104]: # Create train/val split with 10% of remaining articles in validation data:
train_1, val_1 = train_test_split(train_1, test_size=0.1)
```

```
[105]: # Assess that split was successful:
len(train_1) + len(val_1) + len(test_1) == len(cleaned_dataset_1)
```

```
[105]: True
```

```
[106]: # Convert label column to binary integer (0 = true, 1 = fake):
train_1["label"] = np.where(train_1["label"] == "true", 0,1)
val_1["label"] = np.where(val_1["label"] == "true", 0,1)
test_1["label"] = np.where(test_1["label"] == "true", 0,1)
```

```
[107]: # Inspecting transformed training data:
train_1.head(10)
```

```
[107]:
```

	text	label
8490	billionaire investor wilbur ross stand behind ...	0
2952	white water raft bali visit temple java former...	0
24959	remember donald trump claim ford move job prod...	1
7294	u president barack obama foreign policy legacy...	0
23977	allege president donald trump create worldwide...	1
15028	saudi arabia announce confiscate money asset h...	0
26379	go folk yet another video surface show yet ano...	1
21859	first impaneled grand jury grand jury begin is...	1
5890	iraqi say life danger work u government iraq f...	0
24288	republican long four year think progressive li...	1

```
[108]: # Inspecting transformed validation data:
val_1.head(10)
```

```
[108]:
```

	text	label
7660	trail opinion poll republican presidential nom...	0
30191	young man record maliciously abuse young woman...	1
8830	u house majority leader kevin mccarthy say tue...	0
19380	least 10 people die another 92 miss eastern de...	0
14784	medium outlet cnn radio free europe deutsche w...	0
38499	episode sunday wire show resume sunday may 29 ...	1
33163	judge jeanine pirro leave disrespectful attitu...	1
28982	bad enough oscar year black people nominate in...	1

32723	national security council aide craig deare dis...	1
28267	convenient stereotype girl small mishmash suga...	1

```
[109]: # Assess that data is roughly balanced across categories:
train_1.groupby('label').count()
```

```
[109]:      text
label
0      15312
1      12510
```

```
[110]: # Assess that data is roughly balanced across categories:
val_1.groupby('label').count()
```

```
[110]:      text
label
0      1672
1      1420
```

```
[111]: # Assess that data is roughly balanced across categories:
test_1.groupby('label').count()
```

```
[111]:      text
label
0      4207
1      3522
```

```
[112]: # Define number of unique labels:
n_labels = len(train_1['label'].unique())
```

```
[113]: # Create list of texts to predict:
X_dataset_1 = test_1['text'].tolist()
```

```
[114]: # Inspect length
len(X_dataset_1)
```

```
[114]: 7729
```

3.2 Training

```
[19]: # Initialize the model with the specified hyperparameters:
FN_model_1 = ClassificationModel('bert', "bert-base-uncased",
                                num_labels=n_labels, use_cuda=False,
                                args={'reprocess_input_data': True,
→ 'overwrite_output_dir': True,
                                "num_train_epochs": 3, "max_seq_length": 512,
→ "train_batch_size": 128,
```

```
"learning_rate": 1e-5})
```

```
# Fine-tune the model:  
FN_model_1.train_model(train_1)
```

Some weights of the model checkpoint at bert-base-uncased were not used when initializing BertForSequenceClassification: ['cls.predictions.bias', 'cls.predictions.transform.dense.bias', 'cls.predictions.transform.LayerNorm.bias', 'cls.predictions.decoder.weight', 'cls.seq_relationship.bias', 'cls.seq_relationship.weight', 'cls.predictions.transform.LayerNorm.weight', 'cls.predictions.transform.dense.weight']

- This IS expected if you are initializing BertForSequenceClassification from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).

- This IS NOT expected if you are initializing BertForSequenceClassification from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).

Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are newly initialized:

```
['classifier.bias', 'classifier.weight']
```

You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.

```
/opt/conda/lib/python3.7/site-
```

```
packages/simpletransformers/classification/classification_model.py:586:
```

```
UserWarning: Dataframe headers not specified. Falling back to using column 0 as text and column 1 as labels.
```

```
"Dataframe headers not specified. Falling back to using column 0 as text and column 1 as labels."
```

```
0%|          | 0/27822 [00:00<?, ?it/s]
```

```
Epoch: 0%|          | 0/3 [00:00<?, ?it/s]
```

```
Running Epoch 0 of 3: 0%|          | 0/218 [00:00<?, ?it/s]
```

```
Running Epoch 1 of 3: 0%|          | 0/218 [00:00<?, ?it/s]
```

```
Running Epoch 2 of 3: 0%|          | 0/218 [00:00<?, ?it/s]
```

```
[19]: (654, 0.06579963080759448)
```

3.3 Predictions

```
[32]: # Loading trained model, so we don't have to rerun the training each time we
      ↪ restart the kernel:
      FN_model_1 = ClassificationModel("bert", "outputs_dataset_1/",
      ↪ num_labels=n_labels, use_cuda=False)

[56]: # Use the fine-tuned model to predict the testing labels and save the raw model
      ↪ outputs:
      _, raw_pred = FN_model_1.predict(X_dataset_1)

      0%|          | 0/7729 [00:00<?, ?it/s]

      0%|          | 0/967 [00:00<?, ?it/s]

[57]: # Convert raw model outputs to class probabilities:
      probabilities = softmax(raw_pred, axis=1)

[58]: # Assess probabilities:
      probabilities

[58]: array([[4.79282272e-04, 9.99520718e-01],
             [9.99563380e-01, 4.36620433e-04],
             [4.79260344e-04, 9.99520740e-01],
             ...,
             [3.38563774e-03, 9.96614362e-01],
             [5.35239229e-04, 9.99464761e-01],
             [9.99326263e-01, 6.73736558e-04]])

[59]: # Binarize probabilities to the most probable class:
      binary_preds = [np.argmax(pred) for pred in probabilities]

[60]: # Inspect length of predictions:
      len(binary_preds)

[60]: 7729
```

3.4 Results

```
[61]: # Print classification report:
      print(classification_report(test_1.label, binary_preds))

      # Print confusion matrix:
      confusion_matrix(test_1.label, binary_preds)
```

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	1.00	1.00	1.00	4187
1	1.00	1.00	1.00	3542
accuracy			1.00	7729
macro avg	1.00	1.00	1.00	7729
weighted avg	1.00	1.00	1.00	7729

```
[61]: array([[4182,    5],
           [ 13, 3529]])
```

4 Preprocess dataset 2

4.1 Data loading and wrangling

```
[2]: file_list = glob.glob(os.path.join(os.getcwd(), "data", "dataset_2", "fake", "*.
      ↪txt"))

fake = []

for file_path in file_list:
    with open(file_path, encoding='windows-1252') as f_input:
        encoded_f = f_input.read().replace("\n", " ")
        fake.append(encoded_f)
```

```
[3]: file_list = glob.glob(os.path.join(os.getcwd(), "data", "dataset_2", "real", "*.
      ↪txt"))

real = []

for file_path in file_list:
    with open(file_path, encoding='windows-1252') as f_input:
        encoded_f = f_input.read().replace("\n", " ")
        real.append(encoded_f)
```

4.2 Data cleaning

```
[4]: # Remove \ from the data:
for i in range(len(fake)):
    fake[i] = fake[i].replace("\'", "")
```

```
[5]: # Remove \ from the data:
for i in range(len(real)):
    real[i] = real[i].replace("\'", "")
```

```

[6]: # Convert data to pandas dataframe:
fake_new = pd.DataFrame(fake)

[7]: # Rename column with texts to text:
fake_new = fake_new.rename({0: "text"},axis = 'columns')

[8]: # Add label-column with fake labels:
fake_new['label'] = 'fake'

[9]: # Convert data to pandas dataframe:
real_new = pd.DataFrame(real)

[10]: # Rename column with texts to text:
real_new = real_new.rename({0: "text"},axis = 'columns')

[11]: # Add label-column with fake labels:
real_new['label'] = 'true'

[12]: # Merge fake- and true news into a single dataframe:
merged_new = pd.concat([fake_new, real_new])

[13]: # Reset indeces:
merged_new = merged_new.reset_index()

[14]: # Selecting only relevant columns:
merged_new = merged_new[["text", "label"]]

[15]: # Inspecting:
merged_new

[15]:
                                     text label
0    The warranty on ‘Make America Great Again’ bas...  fake
1    Calling it a total disaster, president-elect D...  fake
2    WASHINGTON, D.C. - Former presidential inter...  fake
3    President Barack Obama’s legacy might soon be ...  fake
4    atican City - In a final speech to the synod, ...  fake
..
243  WASHINGTON - Republicans are united on repeali...  true
244  President-elect Donald Trump escalated his rhe...  true
245  Congress is preparing to do major battle next ...  true
246  PALM BEACH, Fla. -- President-elect Donald Tru...  true
247  This is my last column until after the electio...  true

[248 rows x 2 columns]

```

4.2.1 Remove punctuation

```
[22]: merged_new['text']=merged_new['text'].apply(lambda x: remove_punctuation(x))
```

4.2.2 Tokenize and lower

```
[23]: merged_new['tokenized']=merged_new['text'].apply(lambda x: tokenize(x.lower()))
```

4.2.3 Remove stopwords

```
[24]: merged_new['tokenized'] = merged_new['tokenized'].apply(lambda x: ↵  
    ↵remove_stopwords(x))
```

4.2.4 Lemma

```
[25]: for i in range(len(merged_new['tokenized'])):  
    tagged = pos_tag(merged_new['tokenized'][i])  
    merged_new['tokenized'][i] = [lemmatizer.lemmatize(word, ↵  
    ↵pos=penn2morphy(tag)) for word, tag in tagged]
```

4.2.5 Concatenate tokens into sentences

```
[26]: merged_new['text'] = merged_new['tokenized'].apply(lambda x: concat(x))
```

4.2.6 Write dataframe to csv-file

```
[27]: # Selecting only relevant columns:  
merged_new = merged_new[["text", "label"]]
```

```
[28]: # Write to csv:  
merged_new.to_csv(os.path.join("data", "generated_data", "cleaned_dataset_2.  
    ↵csv"), index=False)
```

5 BERT trained on dataset 1, evaluated on dataset 2

5.1 Load cleaned data

```
[29]: cleaned_dataset_2 = pd.read_csv(os.path.join("data", "generated_data", ↵  
    ↵"cleaned_dataset_2.csv"))
```

```
[30]: # Change labels to binary integers:  
cleaned_dataset_2["label"] = np.where(cleaned_dataset_2["label"] == "true", 0,1)
```

```
[31]: # Define number of unique labels:  
n_labels = len(cleaned_dataset_2['label'].unique())
```


5.2 Prepare and predict

```
[33]: # Use the 1st fine-tuned model to predict dataset 2 save the raw model outputs:
_, raw_pred = FN_model_1.predict(cleaned_dataset_2['text'].tolist())
```

```
0%|          | 0/248 [00:00<?, ?it/s]
```

```
0%|          | 0/31 [00:00<?, ?it/s]
```

```
[34]: # Convert raw model outputs to class probabilities:
probabilities = softmax(raw_pred, axis=1)
```

```
[35]: # Binarize probabilities to the most probable class:
binary_preds = [np.argmax(pred) for pred in probabilities]
```

```
[36]: # Inspect length of predictions:
len(binary_preds)
```

```
[36]: 248
```

```
[37]: # Print classification report:
print(classification_report(cleaned_dataset_2.label, binary_preds))

# Print confusion matrix:
confusion_matrix(cleaned_dataset_2.label, binary_preds)
```

	precision	recall	f1-score	support
0	0.59	0.62	0.61	128
1	0.57	0.55	0.56	120
accuracy			0.58	248
macro avg	0.58	0.58	0.58	248
weighted avg	0.58	0.58	0.58	248

```
[37]: array([[79, 49],
           [54, 66]])
```

6 BERT trained- and evaluated dataset 2:

```
[38]: # Create train/test split with 20% of all articles in testing data:
train_2, test_2 = train_test_split(cleaned_dataset_2, test_size=0.2)
```

```
[39]: # Create list of texts to predict:
X_dataset_2 = test_2['text'].tolist()
```

```
[40]: # Define number of unique labels:
n_labels = len(train_2['label'].unique())

[42]: # Initialize the model with the specified hyperparameters:
FN_model_2 = ClassificationModel('bert',"bert-base-uncased",
                                num_labels=n_labels, use_cuda=False,
                                args={'reprocess_input_data': True,
→ 'overwrite_output_dir': True,
                                "num_train_epochs": 15, "max_seq_length":
→ 512, "train_batch_size": 16,
                                "learning_rate": 1e-5})

# Fine-tune the model:
FN_model_2.train_model(train_2)
```

Some weights of the model checkpoint at bert-base-uncased were not used when initializing BertForSequenceClassification:

```
['cls.predictions.transform.dense.weight', 'cls.seq_relationship.bias',
'cls.predictions.transform.LayerNorm.weight', 'cls.predictions.decoder.weight',
'cls.predictions.transform.dense.bias', 'cls.predictions.bias',
'cls.seq_relationship.weight', 'cls.predictions.transform.LayerNorm.bias']
```

- This IS expected if you are initializing BertForSequenceClassification from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).

- This IS NOT expected if you are initializing BertForSequenceClassification from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).

Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are newly initialized:

```
['classifier.bias', 'classifier.weight']
```

You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.

```
/opt/conda/lib/python3.7/site-
```

```
packages/simpletransformers/classification/classification_model.py:586:
```

```
UserWarning: Dataframe headers not specified. Falling back to using column 0 as
text and column 1 as labels.
```

```
"Dataframe headers not specified. Falling back to using column 0 as text and
column 1 as labels."
```

```
0%|          | 0/198 [00:00<?, ?it/s]
```

```
Epoch: 0%|          | 0/15 [00:00<?, ?it/s]
```

```
Running Epoch 0 of 15: 0%|          | 0/13 [00:00<?, ?it/s]
```

```

Running Epoch 1 of 15:  0%|          | 0/13 [00:00<?, ?it/s]

Running Epoch 2 of 15:  0%|          | 0/13 [00:00<?, ?it/s]

Running Epoch 3 of 15:  0%|          | 0/13 [00:00<?, ?it/s]

Running Epoch 4 of 15:  0%|          | 0/13 [00:00<?, ?it/s]

Running Epoch 5 of 15:  0%|          | 0/13 [00:00<?, ?it/s]

Running Epoch 6 of 15:  0%|          | 0/13 [00:00<?, ?it/s]

Running Epoch 7 of 15:  0%|          | 0/13 [00:00<?, ?it/s]

Running Epoch 8 of 15:  0%|          | 0/13 [00:00<?, ?it/s]

Running Epoch 9 of 15:  0%|          | 0/13 [00:00<?, ?it/s]

Running Epoch 10 of 15: 0%|          | 0/13 [00:00<?, ?it/s]

Running Epoch 11 of 15: 0%|          | 0/13 [00:00<?, ?it/s]

Running Epoch 12 of 15: 0%|          | 0/13 [00:00<?, ?it/s]

Running Epoch 13 of 15: 0%|          | 0/13 [00:00<?, ?it/s]

Running Epoch 14 of 15: 0%|          | 0/13 [00:00<?, ?it/s]

```

[42]: (195, 0.3403184807071319)

```

[95]: # Loading trained model, so we don't have to rerun the training each time we
      ↪ restart the kernel:
      FN_model_2 = ClassificationModel("bert", "outputs_dataset_2/",
      ↪ num_labels=n_labels, use_cuda=False)

```

```

[43]: # Use the fine-tuned model to predict the testing labels and save the raw model
      ↪ outputs:
      _, raw_pred = FN_model_2.predict(X_dataset_2)

```

```

0%|          | 0/50 [00:00<?, ?it/s]

```

```
0%|          | 0/7 [00:00<?, ?it/s]
```

```
[44]: # Convert raw model outputs to class probabilities:
probabilities = softmax(raw_pred, axis=1)
```

```
[45]: # Binarize probabilities to the most probable class:
binary_preds = [np.argmax(pred) for pred in probabilities]
```

```
[46]: # Inspect length of predictions:
len(binary_preds)
```

```
[46]: 50
```

```
[47]: # Print classification report:
print(classification_report(test_2.label, binary_preds))

# Print confusion matrix:
confusion_matrix(test_2.label, binary_preds)
```

	precision	recall	f1-score	support
0	0.77	0.88	0.82	26
1	0.85	0.71	0.77	24
accuracy			0.80	50
macro avg	0.81	0.80	0.80	50
weighted avg	0.81	0.80	0.80	50

```
[47]: array([[23,  3],
           [ 7, 17]])
```

7 BERT trained on dataset 2, evaluated on dataset 1

```
[51]: # Use the fine-tuned model to predict the texts from dataset 1 and save the raw_
      ↪ model outputs:
_, raw_pred = FN_model_2.predict(cleaned_dataset_1['text'].tolist())
```

```
0%|          | 0/38643 [00:00<?, ?it/s]
```

```
0%|          | 0/4831 [00:00<?, ?it/s]
```

```
[52]: # Convert raw model outputs to class probabilities:
probabilities = softmax(raw_pred, axis=1)
```

```
[53]: # Binarize probabilities to the most probable class:
binary_preds = [np.argmax(pred) for pred in probabilities]
```

```
[54]: # Inspect length of predictions:
len(binary_preds)
```

```
[54]: 38643
```

```
[62]: cleaned_dataset_1["label"] = np.where(cleaned_dataset_1["label"] == "true", 0,1)
```

```
[63]: # Print classification report:
print(classification_report(cleaned_dataset_1.label, binary_preds))

# Print confusion matrix:
confusion_matrix(cleaned_dataset_1.label, binary_preds)
```

	precision	recall	f1-score	support
0	0.66	0.51	0.58	21191
1	0.54	0.69	0.60	17452
accuracy			0.59	38643
macro avg	0.60	0.60	0.59	38643
weighted avg	0.61	0.59	0.59	38643

```
[63]: array([[10848, 10343],
          [ 5466, 11986]])
```

8 Periods - for temporal word embedding analysis

8.1 Data wrangling

```
[106]: import numpy as np
import regex as re
from datetime import *
```

```
[107]: # Load data:
fake = pd.read_csv(os.path.join("data", "dataset_1", "Fake.csv"))

# NA for wrong entries:
fake["date"] = [re.sub("^.*:.*|^.* .* .*|^\.d.*", "NA", date) for date in
→fake["date"]] # All webpages, entries that start with a number and sequences
→of words upon words upon words should be NA
```

```

# Drop rows with NAs:
fake = fake[(fake!='NA').all(1)]

# Streamline dates:
months = ["January", "February", "March", "April", "May", "June", "July",
↪ "August", "September", "October", "November", "December"]
for i in months:
    fake["date"] = [re.sub(f"^{i}", f"{i[0:3]}", date) for date in fake["date"]]

# Convert to date format:
fake["date"] = [datetime.strptime(date, "%b %d, %Y").date() for date in
↪ fake["date"]]

```

```

[108]: # Find date range:
date_range = max(fake["date"]) - min(fake["date"])

```

```

[109]: # Create categorical variable pertaining to split:
period = []
for date in fake["date"]:
    if date <= min(fake["date"]) + date_range/5:
        period.append(1)
    if date > min(fake["date"]) + date_range/5 and date <= min(fake["date"]) +
↪ date_range/5*2:
        period.append(2)
    if date > min(fake["date"]) + date_range/5*2 and date <= min(fake["date"]) +
↪ date_range/5*3:
        period.append(3)
    if date > min(fake["date"]) + date_range/5*3 and date <= min(fake["date"]) +
↪ date_range/5*4:
        period.append(4)
    if date > min(fake["date"]) + date_range/5*4:
        period.append(5)

```

```

[110]: # Create column with periods:
fake["period"] = period

```

```

[111]: # Ensure that the unique entries in the period-column is correct:
fake['period'].unique()

```

```

[111]: array([5, 4, 3, 2, 1])

```

```

[112]: # Write data to csv:
fake.to_csv(os.path.join("data", "generated_data", "fake_periods.csv"),
↪ index=False)

```

```
[113]: # Load data from csv:
fake = pd.read_csv(os.path.join("data", "generated_data", "fake_periods.csv"))
```

8.2 Data cleaning

```
[114]: # Remove rows with only whitespace and replace it with NA's
fake.replace(" ", float("NaN"), inplace=True)

# Remove NA's
fake.dropna(subset = ["text"], inplace=True)
```

```
[115]: # Remove duplicate texts:
fake = fake.drop_duplicates(subset=['text'])
```

```
[116]: # Reset indeces:
fake = fake.reset_index()
```

8.2.1 Remove reuters

```
[117]: # Define regex pattern:
pattern = r".*\(Reuters\) - "

for i in range(len(fake['text'])):
    fake['text'][i] = re.sub(pattern, '', fake['text'][i])
```

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:5:

SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

"""

8.2.2 Remove hashtags

```
[118]: # Define regex pattern:
pattern = r"#(\S+)"

for i in range(len(fake['text'])):
    fake['text'][i] = re.sub(pattern, '', fake['text'][i])
```

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:5:

SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: <https://pandas.pydata.org/pandas->

```
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
"""
```

8.2.3 Remove tags

```
[119]: # Define regex pattern:
pattern = r"@(\S+)"

for i in range(len(fake['text'])):
    fake['text'][i] = re.sub(pattern, '', fake['text'][i])
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:5:
```

```
SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
"""
```

8.2.4 Remove (capslock)

```
[120]: # Define regex pattern:
pattern = r"\([A-Z]*\)"

for i in range(len(fake['text'])):
    fake['text'][i] = re.sub(pattern, '', fake['text'][i])
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:5:
```

```
SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
"""
```

8.2.5 Remove systematic patterns

```
[122]: # Define regex pattern:
pattern = r"The following statement.*accuracy[.]"

for i in range(len(fake['text'])):
    fake['text'][i] = re.sub(pattern, '', fake['text'][i])
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:5:
```

```
SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
```



```
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
"""
```

```
[123]: # Define regex pattern:
pattern = r"pic\.twitter\.com\/.* "

for i in range(len(fake['text'])):
    fake['text'][i] = re.sub(pattern, '', fake['text'][i])
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
"""
```

8.2.6 Remove punctuation:

```
[124]: fake['text']=fake['text'].apply(lambda x: remove_punctuation(x))
```

8.2.7 Tokenize and lower

```
[125]: fake['tokenized']=fake['text'].apply(lambda x: tokenize(x.lower()))
```

8.2.8 Remove stopwords

```
[126]: fake['tokenized'] = fake['tokenized'].apply(lambda x: remove_stopwords(x))
```

8.2.9 Lemmatize

```
[127]: for i in range(len(fake['tokenized'])):
        tagged = pos_tag(fake['tokenized'][i])
        fake['tokenized'][i] = [lemmatizer.lemmatize(word, pos=penn2morphy(tag))
                                for word, tag in tagged]
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
This is separate from the ipykernel package so we can avoid doing imports
until
```

8.2.10 Concatenate words

```
[129]: fake['text'] = fake['tokenized'].apply(lambda x: concat(x))
```

8.2.11 Remove newly induced empty columns

```
[130]: fake.replace(" ", float("NaN"), inplace=True)

fake.dropna(subset = ["text"], inplace=True)
```

```
[131]: fake = fake.reset_index()
```

8.2.12 Remove more systematic patterns

```
[132]: # Define regex pattern:
pattern = r"21st century wire say"

for i in range(len(fake['text'])):
    fake['text'][i] = re.sub(pattern, '', fake['text'][i])
```

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:5:

SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

"""

```
[133]: # Define regex pattern:
pattern = r"21st century wire"

for i in range(len(fake['text'])):
    fake['text'][i] = re.sub(pattern, '', fake['text'][i])
```

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:5:

SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

"""

```
[134]: # Define regex pattern:
pattern = r"filessupport.*"

for i in range(len(fake['text'])):
    fake['text'][i] = re.sub(pattern, '', fake['text'][i])
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
"""
```

```
[135]: # Define regex pattern:
pattern = r"21wire"

for i in range(len(fake['text'])):
    fake['text'][i] = re.sub(pattern, '', fake['text'][i])
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
"""
```

8.3 Write data for word-embedding analysis

```
[136]: # Concatenate
period_texts = []
for i in range(1, 6):
    period_text = " ".join(fake.loc[fake['period'] == i]["text"])
    period_texts.append(period_text)

# Write as .txt files
for i, n in zip(period_texts, range(1,6)):
    text_file = open(os.path.join("word_embeddings", "output", "texts",
    ↪f"00{n}0.txt"), "w")
    n = text_file.write(i)
    text_file.close()
```