tweet_feedback

September 12, 2021

[1]:
```
import spacy
import pandas as pd
from itertools import combinations as combs
from spacy.matcher import Matcher
from spacy import displacy

import nltk

import numpy as np

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, Dropout, SpatialDropout1D
from tensorflow.keras.layers import LSTM
from tensorflow.keras.models import load_model

from collections import Counter
import text_normalizer as tn
import model_evaluation_utils as meu

from keras.preprocessing import sequence
from sklearn.preprocessing import LabelEncoder
```

0.1 Data Pipeline

[2]:
```
nlp = spacy.load('en_core_web_sm')

doc1  = nlp(u'An Englishman, a Scotsman and an Irishman walk into a bar. The␣
 ↪Englishman wanted to go so they all had to leave. #Brexitjokes')
doc2  = nlp(u'Why do we need any colour passport? We should just be able to␣
 ↪shout, “British! Less of your nonsense!” and stroll straight through.')
doc3  = nlp(u'Q: With Britain leaving the EU how much space was created? A:␣
 ↪Exactly 1GB')
doc4  = nlp(u'VOTERS: we want to give a boat a ridiculous name UK: no VOTERS: we␣
 ↪want to break up the EU and trash the world economy UK: fine')
doc5  = nlp(u'#BrexitJokes How did the Brexit chicken cross the road? \"I never␣
 ↪said there was a road. Or a chicken\".')
```

```
doc6  = nlp(u'After #brexit, when rapper 50 cent performs in GBR he\'ll appear␣
 ↪as 10.00 pounds. #brexitjokes')
doc7  = nlp(u'I long for the simpler days when #Brexit was just a term for␣
 ↪leaving brunch early.')
doc8  = nlp(u'Say goodbye to croissants, people. Delicious croissants. We\'re␣
 ↪stuck with crumpets FOREVER.')
doc9  = nlp(u'Hello, I am from Britain, you know, the one that got tricked by a␣
 ↪bus')
doc10 = nlp(u'How many Brexiteers does it take to change a light bulb? None,␣
 ↪they are all walked out because they didn’t like the way the electrician did␣
 ↪it.')

docs = [
    doc1,
    doc2,
    doc3,
    doc4,
    doc5,
    doc6,
    doc7,
    doc8,
    doc9,
    doc10]
```

[26]:
```
#Creating DF for LSTM
tweets = np.array([
    ["An Englishman, a Scotsman and an Irishman walk into a bar. The Englishman␣
 ↪wanted to go so they all had to leave. #Brexitjokes"],
    ["Why do we need any colour passport? We should just be able to shout,␣
 ↪‘‘British! Less of your nonsense!’’ and stroll straight through."],
    ["Q: With Britain leaving the EU how much space was created? A: Exactly␣
 ↪1GB"],
    ["VOTERS: we want to give a boat a ridiculous name UK: no VOTERS: we want to␣
 ↪break up the EU and trash the world economy UK: fine"],
    ["#BrexitJokes How did the Brexit chicken cross the road? \"I never said␣
 ↪there was a road. Or a chicken\"."],
    ["After #brexit, when rapper 50 cent performs in GBR he'll appear as 10.00␣
 ↪pounds. #brexitjokes"],
    ["I long for the simpler days when #Brexit was just a term for leaving␣
 ↪brunch early."],
    ["Say goodbye to croissants, people. Delicious croissants. We're stuck with␣
 ↪crumpets FOREVER."],
    ["Hello, I am from Britain, you know, the one that got tricked by a bus"],
    ["How many Brexiteers does it take to change a light bulb? None, they are␣
 ↪all walked out because they didn’t like the way the electrician did it."]])
```

```
tweet_df = pd.DataFrame(tweets, columns=['tweet_content'])
tweet_df.head()

# Removing Stop words
stop_words = nltk.corpus.stopwords.words('english')
stop_words.remove('no')
stop_words.remove('but')
stop_words.remove('not')
```

0.2 Part of Speach Tagging

[37]:
```
tweet_no = 1
for doc in docs:
    print(f'Tweet: {tweet_no}')
    for token in doc:
        print(f'{token.text:{10}} - {token.pos_:{10}} - {token.tag_:{10}} -␣
 ↪{spacy.explain(token.tag_)}')
    tweet_no += 1
```

```
Tweet: 1
An         - DET        - DT         - determiner
Englishman - PROPN      - NNP        - noun, proper singular
,          - PUNCT      - ,          - punctuation mark, comma
a          - DET        - DT         - determiner
Scotsman   - PROPN      - NNP        - noun, proper singular
and        - CCONJ      - CC         - conjunction, coordinating
an         - DET        - DT         - determiner
Irishman   - PROPN      - NNP        - noun, proper singular
walk       - NOUN       - NN         - noun, singular or mass
into       - ADP        - IN         - conjunction, subordinating or preposition
a          - DET        - DT         - determiner
bar        - NOUN       - NN         - noun, singular or mass
.          - PUNCT      - .          - punctuation mark, sentence closer
The        - DET        - DT         - determiner
Englishman - PROPN      - NNP        - noun, proper singular
wanted     - VERB       - VBD        - verb, past tense
to         - PART       - TO         - infinitival "to"
go         - VERB       - VB         - verb, base form
so         - ADV        - RB         - adverb
they       - PRON       - PRP        - pronoun, personal
all        - DET        - DT         - determiner
had        - VERB       - VBD        - verb, past tense
to         - PART       - TO         - infinitival "to"
leave      - VERB       - VB         - verb, base form
```

```
.           - PUNCT      - .          - punctuation mark, sentence closer
#           - SYM        - $          - symbol, currency
Brexitjokes - NOUN        - NNS        - noun, plural
Tweet: 2
Why         - ADV        - WRB        - wh-adverb
do          - AUX        - VBP        - verb, non-3rd person singular present
we          - PRON       - PRP        - pronoun, personal
need        - VERB       - VB         - verb, base form
any         - DET        - DT         - determiner
colour      - NOUN       - NN         - noun, singular or mass
passport    - NOUN       - NN         - noun, singular or mass
?           - PUNCT      - .          - punctuation mark, sentence closer
We          - PRON       - PRP        - pronoun, personal
should      - AUX        - MD         - verb, modal auxiliary
just        - ADV        - RB         - adverb
be          - VERB       - VB         - verb, base form
able        - ADJ        - JJ         - adjective (English), other noun-modifier
(Chinese)
to          - PART       - TO         - infinitival "to"
shout       - VERB       - VB         - verb, base form
,           - PUNCT      - ,          - punctuation mark, comma
“           - PUNCT      - ``         - opening quotation mark
British     - ADJ        - JJ         - adjective (English), other noun-modifier
(Chinese)
!           - PUNCT      - .          - punctuation mark, sentence closer
Less        - ADJ        - JJR        - adjective, comparative
of          - ADP        - IN         - conjunction, subordinating or preposition
your        - PRON       - PRP$       - pronoun, possessive
nonsense    - NOUN       - NN         - noun, singular or mass
!           - PUNCT      - .          - punctuation mark, sentence closer
”           - PUNCT      - ''         - closing quotation mark
and         - CCONJ      - CC         - conjunction, coordinating
stroll      - VERB       - VB         - verb, base form
straight    - ADV        - RB         - adverb
through     - ADV        - RB         - adverb
.           - PUNCT      - .          - punctuation mark, sentence closer
Tweet: 3
Q           - NOUN       - NN         - noun, singular or mass
:           - PUNCT      - :          - punctuation mark, colon or ellipsis
With        - ADP        - IN         - conjunction, subordinating or preposition
Britain     - PROPN      - NNP        - noun, proper singular
leaving     - VERB       - VBG        - verb, gerund or present participle
the         - DET        - DT         - determiner
EU          - PROPN      - NNP        - noun, proper singular
how         - ADV        - WRB        - wh-adverb
much        - ADJ        - JJ         - adjective (English), other noun-modifier
(Chinese)
space       - NOUN       - NN         - noun, singular or mass
```

```
was         - AUX        - VBD        - verb, past tense
created     - VERB       - VBN        - verb, past participle
?           - PUNCT      - .          - punctuation mark, sentence closer
A           - DET        - DT         - determiner
:           - PUNCT      - :          - punctuation mark, colon or ellipsis
Exactly     - ADV        - RB         - adverb
1           - NUM        - CD         - cardinal number
GB          - PROPN      - NNP        - noun, proper singular
Tweet: 4
VOTERS      - NOUN       - NNS        - noun, plural
:           - PUNCT      - :          - punctuation mark, colon or ellipsis
we          - PRON       - PRP        - pronoun, personal
want        - VERB       - VBP        - verb, non-3rd person singular present
to          - PART       - TO         - infinitival "to"
give        - VERB       - VB         - verb, base form
a           - DET        - DT         - determiner
boat        - NOUN       - NN         - noun, singular or mass
a           - DET        - DT         - determiner
ridiculous - ADJ         - JJ         - adjective (English), other noun-modifier
(Chinese)
name        - NOUN       - NN         - noun, singular or mass
UK          - PROPN      - NNP        - noun, proper singular
:           - PUNCT      - :          - punctuation mark, colon or ellipsis
no          - DET        - DT         - determiner
VOTERS      - NOUN       - NNS        - noun, plural
:           - PUNCT      - :          - punctuation mark, colon or ellipsis
we          - PRON       - PRP        - pronoun, personal
want        - VERB       - VBP        - verb, non-3rd person singular present
to          - PART       - TO         - infinitival "to"
break       - VERB       - VB         - verb, base form
up          - ADP        - RP         - adverb, particle
the         - DET        - DT         - determiner
EU          - PROPN      - NNP        - noun, proper singular
and         - CCONJ      - CC         - conjunction, coordinating
trash       - VERB       - VB         - verb, base form
the         - DET        - DT         - determiner
world       - NOUN       - NN         - noun, singular or mass
economy     - NOUN       - NN         - noun, singular or mass
UK          - PROPN      - NNP        - noun, proper singular
:           - PUNCT      - :          - punctuation mark, colon or ellipsis
fine        - ADJ        - JJ         - adjective (English), other noun-modifier
(Chinese)
Tweet: 5
#           - NOUN       - NN         - noun, singular or mass
BrexitJokes - PROPN       - NNP        - noun, proper singular
How         - ADV        - WRB        - wh-adverb
did         - AUX        - VBD        - verb, past tense
the         - DET        - DT         - determiner
```

```
Brexit      - PROPN      - NNP        - noun, proper singular
chicken     - NOUN       - NN         - noun, singular or mass
cross       - VERB       - VB         - verb, base form
the         - DET        - DT         - determiner
road        - NOUN       - NN         - noun, singular or mass
?           - PUNCT      - .          - punctuation mark, sentence closer
"           - PUNCT      - ``         - opening quotation mark
I           - PRON       - PRP        - pronoun, personal
never       - ADV        - RB         - adverb
said        - VERB       - VBD        - verb, past tense
there       - PRON       - EX         - existential there
was         - AUX        - VBD        - verb, past tense
a           - DET        - DT         - determiner
road        - NOUN       - NN         - noun, singular or mass
.           - PUNCT      - .          - punctuation mark, sentence closer
Or          - CCONJ      - CC         - conjunction, coordinating
a           - DET        - DT         - determiner
chicken     - NOUN       - NN         - noun, singular or mass
"           - PUNCT      - ''         - closing quotation mark
.           - PUNCT      - .          - punctuation mark, sentence closer
Tweet: 6
After       - ADP        - IN         - conjunction, subordinating or preposition
#           - NOUN       - NN         - noun, singular or mass
brexit      - NOUN       - NN         - noun, singular or mass
,           - PUNCT      - ,          - punctuation mark, comma
when        - ADV        - WRB        - wh-adverb
rapper      - NOUN       - NN         - noun, singular or mass
50          - NUM        - CD         - cardinal number
cent        - NOUN       - NN         - noun, singular or mass
performs    - NOUN       - NNS        - noun, plural
in          - ADP        - IN         - conjunction, subordinating or preposition
GBR         - PROPN      - NNP        - noun, proper singular
he          - PRON       - PRP        - pronoun, personal
'll         - AUX        - MD         - verb, modal auxiliary
appear      - VERB       - VB         - verb, base form
as          - ADP        - IN         - conjunction, subordinating or preposition
10.00       - NUM        - CD         - cardinal number
pounds      - NOUN       - NNS        - noun, plural
.           - PUNCT      - .          - punctuation mark, sentence closer
#           - NOUN       - NNS        - noun, plural
brexitjokes - NOUN       - NNS        - noun, plural
Tweet: 7
I           - PRON       - PRP        - pronoun, personal
long        - ADV        - RB         - adverb
for         - ADP        - IN         - conjunction, subordinating or preposition
the         - DET        - DT         - determiner
simpler     - ADJ        - JJR        - adjective, comparative
days        - NOUN       - NNS        - noun, plural
```

```
when       - ADV       - WRB       - wh-adverb
#          - NOUN      - NNS       - noun, plural
Brexit     - PROPN     - NNP       - noun, proper singular
was        - VERB      - VBD       - verb, past tense
just       - ADV       - RB        - adverb
a          - DET       - DT        - determiner
term       - NOUN      - NN        - noun, singular or mass
for        - ADP       - IN        - conjunction, subordinating or preposition
leaving    - VERB      - VBG       - verb, gerund or present participle
brunch     - NOUN      - NN        - noun, singular or mass
early      - ADV       - RB        - adverb
.          - PUNCT     - .         - punctuation mark, sentence closer
Tweet: 8
Say        - VERB      - VB        - verb, base form
goodbye    - NOUN      - NN        - noun, singular or mass
to         - ADP       - IN        - conjunction, subordinating or preposition
croissants - NOUN      - NNS       - noun, plural
,          - PUNCT     - ,         - punctuation mark, comma
people     - NOUN      - NNS       - noun, plural
.          - PUNCT     - .         - punctuation mark, sentence closer
Delicious  - ADJ       - JJ        - adjective (English), other noun-modifier
(Chinese)
croissants - NOUN      - NNS       - noun, plural
.          - PUNCT     - .         - punctuation mark, sentence closer
We         - PRON      - PRP       - pronoun, personal
're        - VERB      - VBP       - verb, non-3rd person singular present
stuck      - ADJ       - JJ        - adjective (English), other noun-modifier
(Chinese)
with       - ADP       - IN        - conjunction, subordinating or preposition
crumpets   - NOUN      - NNS       - noun, plural
FOREVER    - ADV       - RB        - adverb
.          - PUNCT     - .         - punctuation mark, sentence closer
Tweet: 9
Hello      - INTJ      - UH        - interjection
,          - PUNCT     - ,         - punctuation mark, comma
I          - PRON      - PRP       - pronoun, personal
am         - AUX       - VBP       - verb, non-3rd person singular present
from       - ADP       - IN        - conjunction, subordinating or preposition
Britain    - PROPN     - NNP       - noun, proper singular
,          - PUNCT     - ,         - punctuation mark, comma
you        - PRON      - PRP       - pronoun, personal
know       - VERB      - VBP       - verb, non-3rd person singular present
,          - PUNCT     - ,         - punctuation mark, comma
the        - DET       - DT        - determiner
one        - NOUN      - NN        - noun, singular or mass
that       - DET       - WDT       - wh-determiner
got        - AUX       - VBD       - verb, past tense
tricked    - VERB      - VBN       - verb, past participle
```

```
by          - ADP        - IN         - conjunction, subordinating or preposition
a           - DET        - DT         - determiner
bus         - NOUN       - NN         - noun, singular or mass
Tweet: 10
How         - ADV        - WRB        - wh-adverb
many        - ADJ        - JJ         - adjective (English), other noun-modifier
(Chinese)
Brexiteers - NOUN        - NNS        - noun, plural
does        - AUX        - VBZ        - verb, 3rd person singular present
it          - PRON       - PRP        - pronoun, personal
take        - VERB       - VB         - verb, base form
to          - PART       - TO         - infinitival "to"
change      - VERB       - VB         - verb, base form
a           - DET        - DT         - determiner
light       - ADJ        - JJ         - adjective (English), other noun-modifier
(Chinese)
bulb        - NOUN       - NN         - noun, singular or mass
?           - PUNCT      - .          - punctuation mark, sentence closer
None        - NOUN       - NN         - noun, singular or mass
,           - PUNCT      - ,          - punctuation mark, comma
they        - PRON       - PRP        - pronoun, personal
are         - AUX        - VBP        - verb, non-3rd person singular present
all         - DET        - DT         - determiner
walked      - VERB       - VBN        - verb, past participle
out         - ADP        - RP         - adverb, particle
because     - SCONJ      - IN         - conjunction, subordinating or preposition
they        - PRON       - PRP        - pronoun, personal
did         - AUX        - VBD        - verb, past tense
n't         - PART       - RB         - adverb
like        - ADP        - IN         - conjunction, subordinating or preposition
the         - DET        - DT         - determiner
way         - NOUN       - NN         - noun, singular or mass
the         - DET        - DT         - determiner
electrician - NOUN        - NN         - noun, singular or mass
did         - VERB       - VBD        - verb, past tense
it          - PRON       - PRP        - pronoun, personal
.           - PUNCT      - .          - punctuation mark, sentence closer
```

[42]:
```python
# POS Counts
tweet_no = 1
for doc in docs:
    print(f'Tweet: {tweet_no}')
    POS_counts = doc.count_by(spacy.attrs.POS)
    for k,v in sorted(POS_counts.items()):
        print(f'{k}: {doc.vocab[k].text:{5}} {v}')

    print('\n')
```

```
    tweet_no += 1
```

```
Tweet: 1
85: ADP   1
86: ADV   1
89: CCONJ 1
90: DET   6
92: NOUN  3
94: PART  2
95: PRON  1
96: PROPN 4
97: PUNCT 3
99: SYM   1
100: VERB  4


Tweet: 2
84: ADJ   3
85: ADP   1
86: ADV   4
87: AUX   2
89: CCONJ 1
90: DET   1
92: NOUN  3
94: PART  1
95: PRON  3
97: PUNCT 7
100: VERB  4


Tweet: 3
84: ADJ   1
85: ADP   1
86: ADV   2
87: AUX   1
90: DET   2
92: NOUN  2
93: NUM   1
96: PROPN 3
97: PUNCT 3
100: VERB  2


Tweet: 4
84: ADJ   2
85: ADP   1
89: CCONJ 1
```

```
90: DET   5
92: NOUN  6
94: PART  2
95: PRON  2
96: PROPN 3
97: PUNCT 4
100: VERB  5


Tweet: 5
86: ADV   2
87: AUX   2
89: CCONJ 1
90: DET   4
92: NOUN  5
95: PRON  2
96: PROPN 2
97: PUNCT 5
100: VERB  2


Tweet: 6
85: ADP   3
86: ADV   1
87: AUX   1
92: NOUN  8
93: NUM   2
95: PRON  1
96: PROPN 1
97: PUNCT 2
100: VERB  1


Tweet: 7
84: ADJ   1
85: ADP   2
86: ADV   4
90: DET   2
92: NOUN  4
95: PRON  1
96: PROPN 1
97: PUNCT 1
100: VERB  2


Tweet: 8
84: ADJ   2
85: ADP   2
```

```
86: ADV   1
92: NOUN  5
95: PRON  1
97: PUNCT 4
100: VERB  2


Tweet: 9
85: ADP   2
87: AUX   2
90: DET   3
91: INTJ  1
92: NOUN  2
95: PRON  2
96: PROPN 1
97: PUNCT 3
100: VERB  2


Tweet: 10
84: ADJ   2
85: ADP   2
86: ADV   1
87: AUX   3
90: DET   4
92: NOUN  5
94: PART  2
95: PRON  4
97: PUNCT 3
98: SCONJ 1
100: VERB  4
```

[55]:
```
# Visualising POS
options = {
    'distance':95,
    'compact':'True'
}

for doc in docs:
    spans = list(doc.sents)
    displacy.render(spans,style='dep',jupyter=True, options = options)
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>
```

0.3 Named Entity Recognition

[56]:
```python
def show_ents(doc):
    no_ents = 0
    if doc.ents:
        for ent in doc.ents:
            print(f'{ent.text} - {ent.label_} - {spacy.explain(ent.label_)}')
            no_ents += 1
        print(f'Total number of entities: {no_ents}')
    else:
        print('No entites found')
```

[57]:
```python
tweet_no = 1
for doc in docs:
    print(f'Tweet: {tweet_no}')
    show_ents(doc)
    print('\n')
    tweet_no += 1
```

```
Tweet: 1
Scotsman - PERSON - People, including fictional
Irishman - NORP - Nationalities or religious or political groups
Englishman - PERSON - People, including fictional
Total number of entities: 3


Tweet: 2
British - NORP - Nationalities or religious or political groups
Total number of entities: 1


Tweet: 3
Britain - GPE - Countries, cities, states
EU - ORG - Companies, agencies, institutions, etc.
Total number of entities: 2
```

```
Tweet: 4
UK - GPE - Countries, cities, states
EU - ORG - Companies, agencies, institutions, etc.
Total number of entities: 2


Tweet: 5
Brexit - PERSON - People, including fictional
Total number of entities: 1


Tweet: 6
50 cent - MONEY - Monetary values, including unit
10.00 pounds - MONEY - Monetary values, including unit
Total number of entities: 2


Tweet: 7
the simpler days - DATE - Absolute or relative dates or periods
Brexit - PERSON - People, including fictional
Total number of entities: 2


Tweet: 8
FOREVER - WORK_OF_ART - Titles of books, songs, etc.
Total number of entities: 1


Tweet: 9
Britain - GPE - Countries, cities, states
Total number of entities: 1


Tweet: 10
Brexiteers - WORK_OF_ART - Titles of books, songs, etc.
Total number of entities: 1
```

[33]:
```
tweet_no = 1
for doc in docs:
    print(f'Tweet: {tweet_no}')
    displacy.render(doc, style="ent")
    tweet_no += 1
```

```
Tweet: 1
```

```
<IPython.core.display.HTML object>
Tweet: 2
<IPython.core.display.HTML object>
Tweet: 3
<IPython.core.display.HTML object>
Tweet: 4
<IPython.core.display.HTML object>
Tweet: 5
<IPython.core.display.HTML object>
Tweet: 6
<IPython.core.display.HTML object>
Tweet: 7
<IPython.core.display.HTML object>
Tweet: 8
<IPython.core.display.HTML object>
Tweet: 9
<IPython.core.display.HTML object>
Tweet: 10
<IPython.core.display.HTML object>
```

0.4 Feature Extraction

[91]:
```
tweet_df.isnull().sum() #delete at a later date
```

[91]:
```
tweet_content    0
dtype: int64
```

[10]:
```
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer,␣
 ↪TfidfVectorizer
```

[12]:
```
tfidf = TfidfVectorizer(min_df=2, max_df=0.5, ngram_range=(1,2))
```

[17]:
```
doc1  = ('An Englishman, a Scotsman and an Irishman walk into a bar. The␣
 ↪Englishman wanted to go so they all had to leave. #Brexitjokes')
doc2  = ('Why do we need any colour passport? We should just be able to shout,␣
 ↪“British! Less of your nonsense!” and stroll straight through.')
```

```
doc3  = ('Q: With Britain leaving the EU how much space was created? A: Exactly␣
 ↪1GB')
doc4  = ('VOTERS: we want to give a boat a ridiculous name UK: no VOTERS: we␣
 ↪want to break up the EU and trash the world economy UK: fine')
doc5  = ('#BrexitJokes How did the Brexit chicken cross the road? \"I never said␣
 ↪there was a road. Or a chicken\".')
doc6  = ('After #brexit, when rapper 50 cent performs in GBR he\'ll appear as 10.
 ↪00 pounds. #brexitjokes')
doc7  = ('I long for the simpler days when #Brexit was just a term for leaving␣
 ↪brunch early.')
doc8  = ('Say goodbye to croissants, people. Delicious croissants. We\'re stuck␣
 ↪with crumpets FOREVER.')
doc9  = ('Hello, I am from Britain, you know, the one that got tricked by a bus')
doc10 = ('How many Brexiteers does it take to change a light bulb? None, they␣
 ↪are all walked out because they didn’t like the way the electrician did it.')

fe_docs = [
    doc1,
    doc2,
    doc3,
    doc4,
    doc5,
    doc6,
    doc7,
    doc8,
    doc9,
    doc10]
```

[21]:
```
features = tfidf.fit_transform(fe_docs)
```

[22]:
```
fe_df = pd.DataFrame(features.todense(),columns=tfidf.get_feature_names())
```

[23]:
```
fe_df
```

[23]:

	all	and	brexit	brexitjokes	britain	did	eu	\
0	0.427075	0.373640	0.000000	0.373640	0.000000	0.000000	0.000000	
1	0.000000	0.379486	0.000000	0.000000	0.000000	0.000000	0.000000	
2	0.000000	0.000000	0.000000	0.000000	0.391305	0.000000	0.391305	
3	0.000000	0.313682	0.000000	0.000000	0.000000	0.000000	0.358542	
4	0.000000	0.000000	0.434107	0.434107	0.000000	0.496189	0.000000	
5	0.000000	0.000000	0.549943	0.549943	0.000000	0.000000	0.000000	
6	0.000000	0.000000	0.411017	0.000000	0.000000	0.000000	0.000000	
7	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
8	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	
9	0.371304	0.000000	0.000000	0.000000	0.000000	0.371304	0.000000	

```
        how      just   leaving    the eu      they        to       was  \
0  0.000000  0.000000  0.000000  0.000000  0.427075  0.596656  0.000000
1  0.000000  0.433757  0.000000  0.000000  0.000000  0.302996  0.000000
2  0.342346  0.000000  0.391305  0.391305  0.000000  0.000000  0.342346
3  0.000000  0.000000  0.000000  0.358542  0.000000  0.500911  0.000000
4  0.434107  0.000000  0.000000  0.000000  0.000000  0.000000  0.434107
5  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
6  0.000000  0.469798  0.469798  0.000000  0.000000  0.000000  0.411017
7  0.000000  0.000000  0.000000  0.000000  0.000000  0.465343  0.000000
8  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
9  0.324847  0.000000  0.000000  0.000000  0.742609  0.259370  0.000000

         we      when      with
0  0.000000  0.000000  0.000000
1  0.758972  0.000000  0.000000
2  0.000000  0.000000  0.391305
3  0.627365  0.000000  0.000000
4  0.000000  0.000000  0.000000
5  0.000000  0.628591  0.000000
6  0.000000  0.469798  0.000000
7  0.582818  0.000000  0.666168
8  0.000000  0.000000  0.000000
9  0.000000  0.000000  0.000000
```

0.5 Sentiment Analysis

[29]:
```
# Load pre-trained model
model = load_model('LSTM_model.h5')
```

[27]:
```
norm_tweets = tn.normalize_corpus(tweet_df['tweet_content'],␣
 ↪stopwords=stop_words)
tokenized_tweets  = [tn.tokenizer.tokenize(text) for text in norm_tweets]

# build word to index vocabulary
token_counter = Counter([token for review in tokenized_tweets for token in␣
 ↪review])
vocab_map     = {item[0]: index+1 for index, item in␣
 ↪enumerate(dict(token_counter).items())}
max_index     = np.max(list(vocab_map.values()))

vocab_map['PAD_INDEX']        = 0
vocab_map['NOT_FOUND_INDEX'] = max_index+1

vocab_size    = len(vocab_map)
```

```
# view vocabulary size and part of the vocabulary map
print('Vocabulary Size:', vocab_size)
print('Sample slice of vocabulary map:', dict(list(vocab_map.items())))

#get max length of train corpus and initialize label encoder
le          = LabelEncoder()
num_classes = 2 # positive -> 1, negative -> 0
max_len     = np.max([len(review) for review in tokenized_tweets])


## Test reviews data corpus
# Convert tokenized text reviews to numeric vectors
tweet_ready = [[vocab_map[token] for token in tokenized_review] for␣
 ↪tokenized_review in tokenized_tweets]
tweet_ready = sequence.pad_sequences(tweet_ready, maxlen=max_len) # pad


# view vector shapes
print('Max length of tweet review vectors:', max_len)
print('Tweet vectors shape:', tweet_ready.shape)
```

```
Vocabulary Size: 84
Sample slice of vocabulary map: {'englishman': 1, 'scotsman': 2, 'irishman': 3,
'walk': 4, 'bar': 5, 'want': 6, 'go': 7, 'leave': 8, 'brexitjoke': 9, 'need':
10, 'colour': 11, 'passport': 12, 'able': 13, 'shout': 14, 'british': 15,
'less': 16, 'nonsense': 17, 'stroll': 18, 'straight': 19, 'q': 20, 'britain':
21, 'eu': 22, 'much': 23, 'space': 24, 'create': 25, 'exactly': 26, 'gb': 27,
'voter': 28, 'give': 29, 'boat': 30, 'ridiculous': 31, 'name': 32, 'uk': 33,
'no': 34, 'break': 35, 'trash': 36, 'world': 37, 'economy': 38, 'fine': 39,
'brexitjokes': 40, 'brexit': 41, 'chicken': 42, 'cross': 43, 'road': 44,
'never': 45, 'say': 46, 'rapper': 47, 'cent': 48, 'perform': 49, 'gbr': 50,
'appear': 51, 'pound': 52, 'long': 53, 'simple': 54, 'day': 55, 'term': 56,
'brunch': 57, 'early': 58, 'goodbye': 59, 'croissant': 60, 'people': 61,
'delicious': 62, 'stick': 63, 'crumpet': 64, 'forever': 65, 'hello': 66, 'know':
67, 'one': 68, 'got': 69, 'trick': 70, 'bus': 71, 'many': 72, 'brexiteer': 73,
'take': 74, 'change': 75, 'light': 76, 'bulb': 77, 'none': 78, 'nt': 79, 'like':
80, 'way': 81, 'electrician': 82, 'PAD_INDEX': 0, 'NOT_FOUND_INDEX': 83}
Max length of tweet review vectors: 17
Tweet vectors shape: (10, 17)
```

[30]:
```
my_pred_test = model.predict(tweet_ready)
```

```
WARNING:tensorflow:Model was constructed with shape (None, 1473) for input
KerasTensor(type_spec=TensorSpec(shape=(None, 1473), dtype=tf.float32,
name='embedding_input'), name='embedding_input', description="created by layer
'embedding_input'"), but it was called on an input with incompatible shape
(None, 17).
```

[31]:
```
pred_score = [1 if p > 0.5 else 0 for p in my_pred_test]
pred_sent = ['Positive' if p > 0.5 else 'Negative' for p in my_pred_test]
```

[32]:
```
for i in range(len(pred_score)):
    print(f'Tweet {i+1}:\nActual Score: {my_pred_test[i]} - Score:␣
 ↪{pred_score[i]} - Sentiment: {pred_sent[i]}')
```

```
Tweet 1:
Actual Score: [0.5145975] - Score: 1 - Sentiment: Positive
Tweet 2:
Actual Score: [0.9946981] - Score: 1 - Sentiment: Positive
Tweet 3:
Actual Score: [0.78269374] - Score: 1 - Sentiment: Positive
Tweet 4:
Actual Score: [0.8127065] - Score: 1 - Sentiment: Positive
Tweet 5:
Actual Score: [0.06928542] - Score: 0 - Sentiment: Negative
Tweet 6:
Actual Score: [0.4466458] - Score: 0 - Sentiment: Negative
Tweet 7:
Actual Score: [0.37085027] - Score: 0 - Sentiment: Negative
Tweet 8:
Actual Score: [0.92935675] - Score: 1 - Sentiment: Positive
Tweet 9:
Actual Score: [0.91288126] - Score: 1 - Sentiment: Positive
Tweet 10:
Actual Score: [0.2754283] - Score: 0 - Sentiment: Negative
```

0.6 Tweet Similarity Scoring

0.6.1 Document Similarity

[28]:
```
doc_df = pd.DataFrame()

for each_pair in id_combs:
    doc_similarity = docs[each_pair[0]-1].similarity(docs[each_pair[1]-1])
    doc_results = {
        'tweet1': int(each_pair[0]),
        'tweet2': int(each_pair[1]),
        'similarity': doc_similarity,
        'text 1': docs[each_pair[0]-1],
        'text 2': docs[each_pair[1]-1]
    }

    doc_df = doc_df.append(doc_results, ignore_index=True)
```

```
<ipython-input-28-7d924ec05226>:4: UserWarning: [W007] The model you're using
has no word vectors loaded, so the result of the Doc.similarity method will be
based on the tagger, parser and NER, which may not give useful similarity
judgements. This may happen if you're using one of the small models, e.g.
`en_core_web_sm`, which don't ship with word vectors and only use context-
sensitive tensors. You can always add your own word vectors, or use one of the
larger models instead if available.
  doc_similarity = docs[each_pair[0]-1].similarity(docs[each_pair[1]-1])
```

[29]:
```
doc_df['tweet1'] = doc_df['tweet1'].astype(int)
doc_df['tweet2'] = doc_df['tweet2'].astype(int)
doc_df.head()
```

[29]:

	similarity	text 1	text 2	tweet1	tweet2
0	0.246874	(An, Englishman, ,, a, Scotsman, and, an, Iris...	(Why, do, we, need, any, colour, passport, ?, ...	1	2
1	0.338287	(An, Englishman, ,, a, Scotsman, and, an, Iris...	(Q, :, With, Britain, leaving, the, EU, how, m...	1	3
2	0.490278	(An, Englishman, ,, a, Scotsman, and, an, Iris...	(VOTERS, :, we, want, to, give, a, boat, a, ri...	1	4
3	0.575611	(An, Englishman, ,, a, Scotsman, and, an, Iris...	(#, BrexitJokes, How, did, the, Brexit, chicke...	1	5
4	0.198380	(An, Englishman, ,, a, Scotsman, and, an, Iris...	(After, #, brexit, ,, when, rapper, 50, cent, ...	1	6

[30]:
```
doc_df_ordered = doc_df.sort_values(by=['similarity'], ascending=False)
doc_df_ordered.head(10)
```

[30]:

	similarity	text 1	text 2	tweet1	tweet2
34	0.576191	(#, BrexitJokes, How, did, the, Brexit, chicke...	(How, many, Brexiteers, does, it, take, to, ch...	5	10
3	0.575611	(An, Englishman, ,, a, Scotsman, and, an, Iris...	(#, BrexitJokes, How, did, the, Brexit, chicke...	1	5
16	0.490846	(Why, do, we, need, any, colour, passport, ?, ...	(How, many, Brexiteers, does, it, take, to, ch...	2	10
2	0.490278	(An, Englishman, ,, a, Scotsman, and, an, Iris...	(VOTERS, :, we, want, to, give, a, boat, a, ri...	1	4
14	0.489872	(Why, do, we, need, any, colour, passport, ?, ...			
8	0.462386	(An, Englishman, ,, a, Scotsman, and, an, Iris...			
11	0.458674	(Why, do, we, need, any, colour, passport, ?, ...			
18	0.456565	(Q, :, With, Britain, leaving, the, EU, how, m...			
20	0.439537	(Q, :, With, Britain, leaving, the, EU, how, m...			
7	0.406573	(An, Englishman, ,, a, Scotsman, and, an, Iris...			

```
14   (Say, goodbye, to, croissants, ,, people, ., D...       2       8
8    (How, many, Brexiteers, does, it, take, to, ch...       1      10
11   (#, BrexitJokes, How, did, the, Brexit, chicke...       2       5
18   (#, BrexitJokes, How, did, the, Brexit, chicke...       3       5
20   (I, long, for, the, simpler, days, when, #, Br...       3       7
7    (Hello, ,, I, am, from, Britain, ,, you, know,...       1       9
```

0.6.2 Term Similarity

[3]:
```
spans = {}
```

[4]:
```
for j,doc in enumerate(docs):
    named_entity_span = [doc[i].text for i in range(len(doc)) if doc[i].ent_type␣
 ↪!= 0]
    print(named_entity_span)
    named_entity_span = ' '.join(named_entity_span)
    named_entity_span = nlp(named_entity_span)
    spans.update({j:named_entity_span})
```

```
['Scotsman', 'Irishman', 'Englishman']
['British']
['Britain', 'EU']
['UK', 'EU']
['Brexit']
['50', 'cent', '10.00', 'pounds']
['the', 'simpler', 'days', 'Brexit']
['FOREVER']
['Britain']
['Brexiteers']
```

[7]:
```
df = pd.DataFrame()

tweet_id = [i for i in range(1,11)]
id_combs = list(combs(tweet_id, 2))

for each_pair in id_combs:
    similarity = spans[each_pair[0]-1].similarity(spans[each_pair[1]-1])
    #print(f'doc{each_pair[0]} is similar to doc{each_pair[1]} by:␣
 ↪{similarity}') #Un-comment if you want to see individual scores printed.
    results = {
        'tweet1': int(each_pair[0]),
        'tweet2': int(each_pair[1]),
        'similarity': similarity,
        'tweet1 NE Span': spans[each_pair[0]-1],
        'tweet2 NE Span': spans[each_pair[1]-1]
    }
```

```
    df = df.append(results, ignore_index=True)
```

```
<ipython-input-7-cfc05c34a82a>:7: UserWarning: [W007] The model you're using has
no word vectors loaded, so the result of the Doc.similarity method will be based
on the tagger, parser and NER, which may not give useful similarity judgements.
This may happen if you're using one of the small models, e.g. `en_core_web_sm`,
which don't ship with word vectors and only use context-sensitive tensors. You
can always add your own word vectors, or use one of the larger models instead if
available.
  similarity = spans[each_pair[0]-1].similarity(spans[each_pair[1]-1])
```

[8]:
```
# Chaning Data Types
df['tweet1'] = df['tweet1'].astype(int)
df['tweet2'] = df['tweet2'].astype(int)
```

[12]:
```
# Saving to/loading from CSV
#df = pd.read_csv('similarity_scores_v2.csv') #Uncomment to load.
#df.to_csv('similarity_scores_v2.csv') #Uncomment to resave.
```

[9]:
```
df_ordered = df.sort_values(by=['similarity'], ascending=False)
```

[10]:
```
# Display the Top 10 Simialr Combinations
df_ordered.head(10)
```

[10]:

	similarity	tweet1	tweet1 NE Span	tweet2	tweet2 NE Span
17	0.857896	3	(Britain, EU)	4	(UK, EU)
1	0.788178	1	(Scotsman, Irishman, Englishman)	3	(Britain, EU)
33	0.771924	5	(Brexit)	9	(Britain)
2	0.720223	1	(Scotsman, Irishman, Englishman)	4	(UK, EU)
18	0.688950	3	(Britain, EU)	5	(Brexit)
22	0.646520	3	(Britain, EU)	9	(Britain)
24	0.598866	4	(UK, EU)	5	(Brexit)
16	0.549264	2	(British)	10	(Brexiteers)
7	0.510660	1	(Scotsman, Irishman, Englishman)	9	(Britain)
3	0.510251	1	(Scotsman, Irishman, Englishman)	5	

```
3        (Brexit)
```

[11]:
```
# Display the Bottom 10 Similar Combinations
df_ordered.tail(10)
```

[11]:

	similarity	tweet1	tweet1 NE Span	tweet2	tweet2 NE Span
6	0.198919	1	(Scotsman, Irishman, Englishman)	8	(FOREVER)
30	0.186382	5	(Brexit)	6	(50, cent, 10.00, pounds)
39	0.185533	7	(the, simpler, days, Brexit)	8	(FOREVER)
41	0.124216	7	(the, simpler, days, Brexit)	10	(Brexiteers)
19	0.123947	3	(Britain, EU)	6	(50, cent, 10.00, pounds)
36	0.097287	6	(50, cent, 10.00, pounds)	8	(FOREVER)
4	0.075894	1	(Scotsman, Irishman, Englishman)	6	(50, cent, 10.00, pounds)
38	0.065650	6	(50, cent, 10.00, pounds)	10	(Brexiteers)
25	0.036557	4	(UK, EU)	6	(50, cent, 10.00, pounds)
12	-0.025753	2	(British)	6	(50, cent, 10.00, pounds)

0.7 Utterence Pattern Matching

[13]:
```
def dep_pattern(doc):
    for i in range(len(doc)-1):
        if doc[i].dep_ == 'nsubj' and doc[i+1].dep_ == 'aux' and doc[i+2].dep_
 ↪== 'ROOT':
            for tok in doc[i+2].children:
                if tok.dep_ == 'dobj':
                    return True
    else:
        return False
```

[16]:
```
for i in docs:
    if dep_pattern(i):
        print(f'Found in: {i}')
    else:
```

```
        print('Not Found')
```

```
Not Found
Not Found
Not Found
Not Found
Not Found
Found in: After #brexit, when rapper 50 cent performs in GBR he'll appear as
10.00 pounds. #brexitjokes
Not Found
Not Found
Not Found
Not Found
```

0.8 Finding Word Sequence Patterns

[30]:
```
matcher = Matcher(nlp.vocab)
pattern = [{
    'DEP':"nsubj"},
    {"DEP":"aux"},
    {"DEP":"ROOT"}
    ]

matcher.add("NsubjAuxRoot", [pattern])

tweet_no = 1

for doc in docs:
    matches = matcher(doc)
    print(f'Tweet: {tweet_no}')
    for match_id, start, end in matches:
        span = doc[start:end]
        print(f"Span: {span.text}")
        print(f"The position in the doc are: {start} - {end}\n")
    else:
        print("None found.\n")
    tweet_no += 1
```

```
Tweet: 1
None found.

Tweet: 2
None found.

Tweet: 3
None found.
```

```
Tweet: 4
None found.

Tweet: 5
None found.

Tweet: 6
Span: he'll appear
The position in the doc are: 11 - 14

None found.

Tweet: 7
None found.

Tweet: 8
None found.

Tweet: 9
None found.

Tweet: 10
None found.
```

0.9 Topic Modelling

[89]:
```
## Is this needed?
```

[ ]:
```

```

0.10 Key Phrases

[1]:
```
def keyphrase(doc):
    for t in doc:
        if t.dep_ == 'probj' and (t.pos_ == 'NOUN' or t.pos_ == "PROPN"):
            return (' '.join([child.text for child in t.lefts]) + ' ' + t.text).
 ↪lstrip()
    for t in reversed(doc):
        if t.dep_ == 'nsubj' and (t.pos_ == 'NOUN' or t.pos_ == 'PROPN'):
            return t.text + ' ' + t.head.text
    for t in reversed(doc):
```

```
        if t.dep_ == 'dobj' and (t.pos_ == 'NOUN' or t.pos_ == 'PROPN'):
            return t.head.text + ' ' + 'ing' + ' ' + t.text
    return False
```

[5]:
```
tweet_no = 1
for doc in docs:
    print(keyphrase(doc))
    tweet_no += 1
```

```
Englishman wanted
need ing passport
Britain leaving
trash ing UK
chicken cross
appear ing performs
Brexit was
Say ing goodbye
False
electrician did
```

---

1 Result Analysis

[2]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from matplotlib.colors import LogNorm
```

1.1 Combination Distribution

[3]:
```
df = pd.read_csv('Comparison Times Tracker.csv', index_col=0)
```

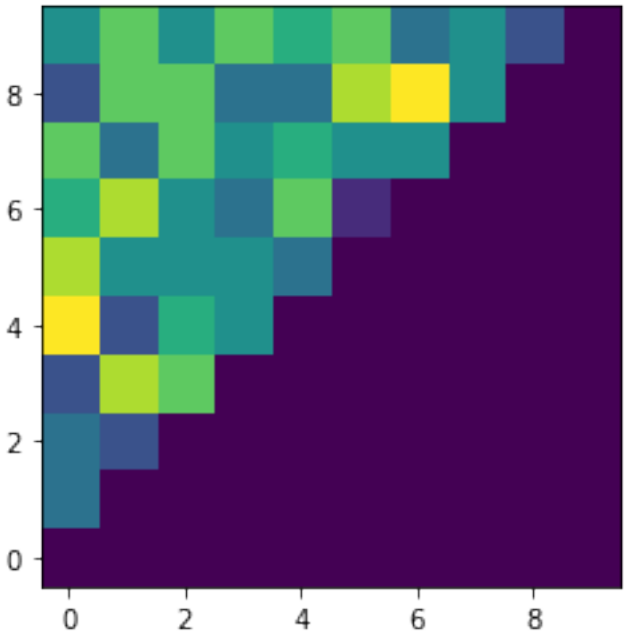
[11]:
```
df.head(10)
```

[11]:

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	3	0	0	0	0	0	0	0	0	0
3	3	2	0	0	0	0	0	0	0	0
4	2	7	6	0	0	0	0	0	0	0
5	8	2	5	4	0	0	0	0	0	0
6	7	4	4	4	3	0	0	0	0	0
7	5	7	4	3	6	1	0	0	0	0
8	6	3	6	4	5	4	4	0	0	0
9	2	6	6	3	3	7	8	4	0	0
10	4	6	4	6	5	6	3	4	2	0

[13]:
```
plt.imshow(np.array(df.values.tolist()).astype('float'),interpolation␣
↪='nearest', origin ='lower')
```

[13]:
```
<matplotlib.image.AxesImage at 0x7ffd182bc2b0>
```

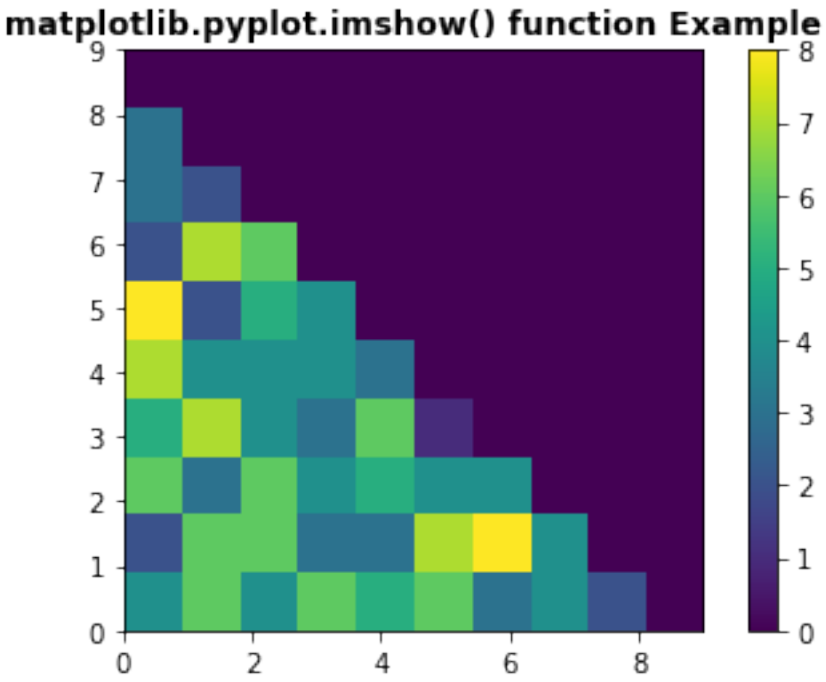


[14]:
```
df.shape
```

[14]:
```
(10, 10)
```

[34]:
```
y, x = np.mgrid[slice(0, 10),
                slice(0, 10)]

c = plt.imshow(df,
                extent =[x.min(), x.max(), y.min(), y.max()],
                   interpolation ='nearest')
plt.colorbar(c)

plt.title('matplotlib.pyplot.imshow() function Example',
                                      fontweight ="bold")
#plt.set_xticks(np.arange(len(tweet_id)))
#plt.set_yticks(np.arange(len(tweet_id)))
# ... and label them with the respective list entries
plt.set_xticklabels(tweet_id)
plt.set_yticklabels(tweet_id)
plt.show()
```

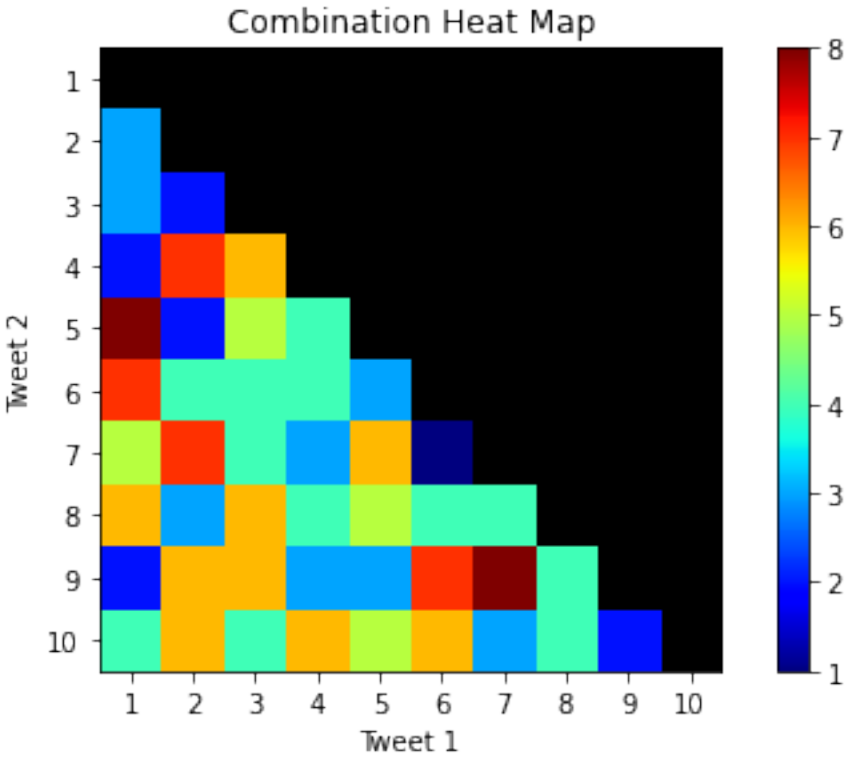
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-34-0cc887716462> in <module>
     12 #plt.set_yticks(np.arange(len(tweet_id)))
     13 # ... and label them with the respective list entries
---> 14 plt.set_xticklabels(tweet_id)
     15 plt.set_yticklabels(tweet_id)
     16 plt.show()

AttributeError: module 'matplotlib.pyplot' has no attribute 'set_xticklabels'
```



[24]:
```
tweet_id = ["1", "2", "3", "4",
            "5", "6", "7","8","9","10"]

results = df

results[results==0] = np.nan

fig, ax = plt.subplots()
cmap = plt.cm.jet
cmap.set_bad(color='k')
im = ax.imshow(results, cmap=cmap)
```

```
# We want to show all ticks...
ax.set_xticks(np.arange(len(tweet_id)))
ax.set_yticks(np.arange(len(tweet_id)))
# ... and label them with the respective list entries
ax.set_xticklabels(tweet_id)
ax.set_yticklabels(tweet_id)
plt.colorbar(im)

ax.set_title("Combination Heat Map")
fig.tight_layout()
plt.xlabel("Tweet 1")
plt.ylabel("Tweet 2")
plt.show()
```

```
<ipython-input-24-23d8e32f7d70>:10: MatplotlibDeprecationWarning: You are
modifying the state of a globally registered colormap. In future versions, you
will not be able to modify a registered colormap in-place. To remove this
warning, you can make a copy of the colormap first. cmap =
copy.copy(mpl.cm.get_cmap("jet"))
  cmap.set_bad(color='k')
```



[11]:
```
win_df = pd.read_csv('Comparison Win Tracker.csv', index_col=0)
```

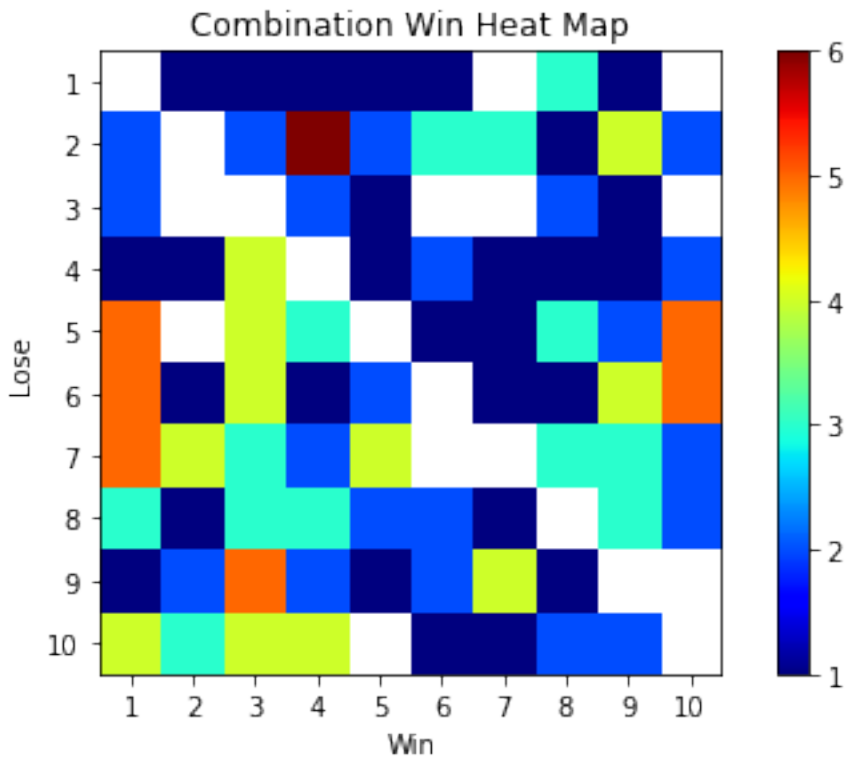
[12]:
```
win_df.head()
```

[12]:

	1	2	3	4	5	6	7	8	9	10
1	0	1	1	1	1	1	0	3	1	0
2	2	0	2	6	2	3	3	1	4	2
3	2	0	0	2	1	0	0	2	1	0
4	1	1	4	0	1	2	1	1	1	2
5	5	0	4	3	0	1	1	3	2	5

[27]:
```
tweet_id = ["1", "2", "3", "4",
              "5", "6", "7","8","9","10"]

results = win_df

results[results==0] = np.nan

fig, ax = plt.subplots()
cmap = plt.cm.jet
cmap.set_bad(color='white')
im = ax.imshow(results, cmap=cmap)

# We want to show all ticks...
ax.set_xticks(np.arange(len(tweet_id)))
ax.set_yticks(np.arange(len(tweet_id)))
# ... and label them with the respective list entries
ax.set_xticklabels(tweet_id)
ax.set_yticklabels(tweet_id)
plt.colorbar(im)

ax.set_title("Combination Win Heat Map")
fig.tight_layout()
plt.xlabel("Win")
plt.ylabel("Lose")
plt.show()
```

```
<ipython-input-27-37a7286085db>:10: MatplotlibDeprecationWarning: You are
modifying the state of a globally registered colormap. In future versions, you
will not be able to modify a registered colormap in-place. To remove this
warning, you can make a copy of the colormap first. cmap =
copy.copy(mpl.cm.get_cmap("jet"))
  cmap.set_bad(color='white')
```


Combination Win Heat Map
Lose
Win
1
2
3
4
5
6
7
8
9
10
1
2
3
4
5
6

[ ]: