CSML1010 Project Working Copy

Sentiment Analysis with the Sentiment140 dataset

Pete Gray

Import libraries

In [1]:
```
import pandas as pd
import numpy as np
np.set_printoptions(precision=2, linewidth=80)
import warnings
warnings.filterwarnings("ignore")
import model_evaluation_utils as meu
```

Adjust pandas display

In [2]:
```
pd.options.display.max_columns = 30
pd.options.display.max_rows = 100
pd.options.display.float_format = '{:.2f}'.format
pd.options.display.precision = 2
pd.options.display.max_colwidth = -1
```

Import matplotlib and seaborn and adjust defaults

In [3]:
```
%matplotlib inline
%config InlineBackend.figure_format = 'svg'

from matplotlib import pyplot as plt
plt.rcParams['figure.dpi'] = 100

import seaborn as sns
sns.set_style("whitegrid")
```

Read data from local filesystem and csv source

In [4]:
```
df = pd.read_csv("training.1600000.processed.noemoticon.csv", encoding="ISO-8859
```

Check data with quick visual inspection

In [5]:
```
df
```

Out[5]:

	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D
0	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by texting it... and might cry as a result School today also. Blah!
1	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Managed to save 50% The rest go out of bounds
2	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
3	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all. i'm mad. why am i here? because I can't see you all over there.
4	0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei not the whole crew
...	...	...	...	...	...	...
1599994	4	2193601966	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	AmandaMarie1028	Just woke up. Having no school is the best feeling ever
1599995	4	2193601969	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	TheWDBoards	TheWDB.com - Very cool to hear old Walt interviews! â™« http://blip.fm/~8bmta
1599996	4	2193601991	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	bpbabe	Are you ready for your MoJo Makeover? Ask me for details
1599997	4	2193602064	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	tinydiamondz	Happy 38th Birthday to my boo of alll time!!! Tupac Amaru Shakur
1599998	4	2193602129	Tue Jun 16 08:40:50 PDT 2009	NO_QUERY	RyanTrevMorris	happy #charitytuesday @theNSPCC @SparksCharity @SpeakingUpH4H

1599999 rows × 6 columns

Give dataframe columns

In [6]:
```
df.columns = ['sentiment', 'ID', 'Time', 'none', 'username', 'Text']
```

In [7]:
```
df.count()
```

Out[7]:
```
sentiment    1599999
ID           1599999
Time         1599999
none         1599999
username     1599999
Text         1599999
dtype: int64
```

Now it has columns, this seems better.

#

We have to cut this down to size, for iterative development.

##

Don't forget to get rid of this!!! When crunching whole huge dataset.

Set temporary dataset size, for quicker processing

In [8]:
```
dev_data_size = 4000
```

In [9]:
```
start_row = int(800000-(dev_data_size/2))-1
finish_row = int(800000+(dev_data_size/2))-1
df_sm = df[start_row:finish_row]
df_sm.count()
```

Out[9]:
```
sentiment    4000
ID           4000
Time         4000
none         4000
username     4000
Text         4000
dtype: int64
```

In [10]:
```
columns = [col for col in df.columns if not col.startswith('self')]
columns
```

Out[10]:
```
['sentiment', 'ID', 'Time', 'none', 'username', 'Text']
```

In [11]:
```
raw_text = np.array(df_sm['Text'])
sentiments = np.array(df_sm['sentiment'])
raw_text[5:15]
```

Out[11]:
```
array(['@StewartWade Yeah, I know--pigs for sure...which is a great visual on m
y end among all the akimbo-ness. ',
       "ouh @Babe_Franzi was hast du hun'? hoffentlich nichts schlimmes. yes, i
miss you rlly much, mary too. ",
       'Woke up with the worst headache ',
       "@MacekMakeupArt I can't remember the last movie I saw in a theatre!  Ho
pe you guys have fun! What are you going to see?",
       'last day of classes   im going to miss chichi !',
       'Damn, time for another pedicure, just chipped my toenail on an open cab
inet  Shit happens!',
       "@mikegentile i've never been in a walmart  no joke",
       "@amedelrivero Start putting up $100 every paycheck! We have to prepare
ourselves for the future -_-. ONLY $300 is what i'm getting ",
       '@patrickeatworld takboleh. i am so in loveeeeeeeeeeee  life sucks. FMMF
L',
       'I have church thur and am always forgetting I can watch fbc on line unt
il Thurs. '],
      dtype=object)
```

In [12]:
```
sentiments[4995:5005]
```

Out[12]:
```
array([], dtype=int64)
```

Data Cleaning

Type *Markdown* and LaTeX: α^2

Cleaning function

In [13]:
```
import re
def clean(s):
    s = s.replace(r'<lb>', "\n")
    s = s.replace(r'<tab>', "\i")

    # As a sanity check - s = s.replace(r'W', "Q")

    s = re.sub(r'<br */*>', "\n", s)
    s = s.replace("&lt;", "<").replace("&gt;", ">").replace("&amp;", "&")
    s = s.replace("&amp;", "&")
    # markdown urls
    s = re.sub(r'\(https*://[^\)]*\)', "", s)
    # normal urls
    s = re.sub(r'https*://[^\s]*', "", s)
    s = re.sub(r'_+', ' ', s)
    s = re.sub(r'"+', '"', s)
    return str(s)
```

Create new column in dataframe

In [14]:
```
df_sm["text_clean"] = ''
```

Iterate and clean

In [15]:
```
for i, row in df_sm.iterrows():
    if i % 1000 == 0:
            print('processed:'.format(i), i)
    df_sm.at[i, "text_clean"] = clean(row.Text)
```

```
processed: 798000
processed: 799000
processed: 800000
processed: 801000
```

Check results

In [16]:
```
df_sm.head()
```

Out[16]:

	sentiment	ID	Time	none	username	Text	text
797999	0	2328378861	Thu Jun 25 09:30:33 PDT 2009	NO_QUERY	skelekitty	Work is so slow, I'm seriously considering quitting my job this week	Work is s I'm s cons quitting my
798000	0	2328379014	Thu Jun 25 09:30:33 PDT 2009	NO_QUERY	DjinniGenie	@davidvancamp That's awful. I wish mine would stop making fat jokes.	@davidva That's awfu mine wou making fa
798001	0	2328379041	Thu Jun 25 09:30:33 PDT 2009	NO_QUERY	Unrated7String	Well, i guess i need to start a new chapter in professional my life	Well, i need to star ch professiona
798002	0	2328379271	Thu Jun 25 09:30:34 PDT 2009	NO_QUERY	jamesebradford	@SandraBernhard Miss Lady, since you brought up your web store - it is notoriously known that it takes AGES to rec'v your merch.	@SandraB Miss Lad you bro your web s is not known that AGES to re
798003	0	2328379299	Thu Jun 25 09:30:34 PDT 2009	NO_QUERY	njandecrox	@CarterTwinsZach Im sorry I hope u feel better cuz I love u and it makes feel horrible when ur sick or sad or mad or hurt	@CarterTwi Im sorry I feel bett love makes feel when ur sad or mac

Additional pre-processing: tokenization, removing extra whitespaces, lower casing and more advanced operations like spelling corrections, grammatical error corrections, removing repeated characters.

In [17]:
```
import nltk
wpt = nltk.WordPunctTokenizer()
nltk.download("stopwords")
stop_words = nltk.corpus.stopwords.words('english')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Dell\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Define normalization function

In [18]:
```
def normalize_document(doc):
    # lower case and remove special characters\whitespaces
    doc = re.sub(r'[^a-zA-Z0-9\s]', '', doc, re.I)
    doc = doc.lower()
    doc = doc.strip()
    # tokenize document
    tokens = wpt.tokenize(doc)
    # filter stopwords out of document
    filtered_tokens = [token for token in tokens if token not in stop_words]
    # re-create document from filtered tokens
    doc = ' '.join(filtered_tokens)
    return doc
```

In [19]:
```
normalize_corpus = np.vectorize(normalize_document)
```

In [20]:
```
df_sm["text_normalized"] = ''
```

In [21]:
```
for i, row in df_sm.iterrows():
    if i % 1000 == 0:
            print('processed:'.format(i), i)
    df_sm.at[i, "text_normalized"] = normalize_corpus(row.text_clean)
```

```
processed: 798000
processed: 799000
processed: 800000
processed: 801000
```

check results

In [22]:
```
df_sm
```

Out[22]:

	sentiment	ID	Time	none	username	Text	
797999	0	2328378861	Thu Jun 25 09:30:33 PDT 2009	NO_QUERY	skelekitty	Work is so slow, I'm seriously considering quitting my job this week	Wo quittir
798000	0	2328379014	Thu Jun 25 09:30:33 PDT 2009	NO_QUERY	DjinniGenie	@davidvancamp That's awful. I wish mine would stop making fat jokes.	@d That's mir mal
798001	0	2328379041	Thu Jun 25 09:30:33 PDT 2009	NO_QUERY	Unrated7String	Well, i guess i need to start a new chapter in professional my life	need profes
798002	0	2328379271	Thu Jun 25 09:30:34 PDT 2009	NO_QUERY	jamesebradford	@SandraBernhard Miss Lady, since you brought up your web store - it is notoriously known that it takes AGES to rec'v your merch.	@San Mis yo your know AGES
798003	0	2328379299	Thu Jun 25 09:30:34 PDT 2009	NO_QUERY	njandecrox	@CarterTwinsZach Im sorry I hope u feel better cuz I love u and it makes feel horrible when ur sick or sad or mad or hurt	@Car Im s fe make w sad o
...	...	...	...	...	...	...	
801994	4	1468163268	Tue Apr 07 00:03:40 PDT 2009	NO_QUERY	jerichoK	@FizzyDuck Five? Seems a little bit too late in the morning but what the hell !	@Fiz See mor
801995	4	1468163291	Tue Apr 07 00:03:40 PDT 2009	NO_QUERY	ex1up	ryanodonnell: @AttractMode Thanks for putting on such a great event. Can't wait for the inevitable sequels! [.. http://tinyurl.com/c3e3ub	@ Thar on eve for
801996	4	1468163300	Tue Apr 07 00:03:39 PDT 2009	NO_QUERY	Mmmbaileys	@damygeebo Carli's my friend	@ Ca
801997	4	1468163315	Tue Apr 07 00:03:39 PDT 2009	NO_QUERY	jasminejoejonas	I feel so great for starting twitter at suzanne but still hardly anyone has it.	I fe sta suz hardl

	sentiment	ID	Time	none	username	Text	
801998	4	1468163384	Tue Apr 07 00:03:40 PDT 2009	NO_QUERY	rsuenaga	@Luke_Stephens: I just said I was wondering about it, not that I wanted it.	@Luk ju wond not th

4000 rows × 8 columns

In [24]:
```
import spacy
nlp = spacy.load('en_core_web_sm')
```

In [25]:
```
for i, row in df_sm.iterrows():
    if i % 1000 == 0:
        print(i)
    if(row["text_normalized"] and len(str(row["text_normalized"])) < 1000000):
        doc = nlp(str(row["text_normalized"]))
        adjectives = []
        nouns = []
        verbs = []
        lemmas = []

        for token in doc:
            lemmas.append(token.lemma_)
            if token.pos_ == "ADJ":
                adjectives.append(token.lemma_)
            if token.pos_ == "NOUN" or token.pos_ == "PROPN":
                nouns.append(token.lemma_)
            if token.pos_ == "VERB":
                verbs.append(token.lemma_)

        df_sm.at[i, "text_lemma"] = " ".join(lemmas)
        df_sm.at[i, "text_nouns"] = " ".join(nouns)
        df_sm.at[i, "text_adjectives"] = " ".join(adjectives)
        df_sm.at[i, "text_verbs"] = " ".join(verbs)
        df_sm.at[i, "text_nav"] = " ".join(nouns+adjectives+verbs)
        df_sm.at[i, "no_tokens"] = len(lemmas)
```

```
798000
799000
800000
801000
```

In [26]:
```
df_sm
```

Out[26]:

username	Text	text_clean	text_normalized	text_lemma	text_no
skelekitty	Work is so slow, I'm seriously considering quitting my job this week	Work is so slow, I'm seriously considering quitting my job this week	work slow im seriously considering quitting job week	work slow - PRON- be seriously consider quit job week	work job w
DjinniGenie	@davidvancamp That's awful. I wish mine would stop making fat jokes.	@davidvancamp That's awful. I wish mine would stop making fat jokes.	davidvancamp thats awful . wish mine would stop making fat jokes .	davidvancamp that s awful . wish mine would stop make fat joke .	davidvanca wish mine j
Unrated7String	Well, i guess i need to start a new chapter in professional my life	Well, i guess i need to start a new chapter in professional my life	well guess need start new chapter professional life	well guess nee start new chapter professional life	cha professiona
jamesebradford	@SandraBernhard Miss Lady, since you brought up your web store - it is notoriously known that it takes AGES to rec'v your merch.	@SandraBernhard Miss Lady, since you brought up your web store - it is notoriously known that it takes AGES to rec'v your merch.	sandrabernhard miss lady since brought web store - notoriously known takes ages rec ' v merch .	sandrabernhard miss lady since bring web store - notoriously know take age rec ' v merch .	sandrabernh miss lady store age r me
njandecrox	@CarterTwinsZach Im sorry I hope u feel better cuz I love u and it makes feel horrible when ur sick or sad or mad or hurt	@CarterTwinsZach Im sorry I hope u feel better cuz I love u and it makes feel horrible when ur sick or sad or mad or hurt	cartertwinszach im sorry hope u feel better cuz love u makes feel horrible ur sick sad mad hurt	cartertwinszach -PRON- be sorry hope u feel better cuz love u make feel horrible ur sick sad mad hurt	cartertwinsz hope love
...	...	...	...	...	
jerichoK	@FizzyDuck Five? Seems a little bit too late in the morning but what the hell !	@FizzyDuck Five? Seems a little bit too late in the morning but what the hell !	fizzyduck five seems little bit late morning hell !	fizzyduck five seem little bit late morning hell !	fizzyduc morning
ex1up	ryanodonnell: @AttractMode Thanks for putting on such a great event. Can't wait for the inevitable sequels! [.. http://tinyurl.com/c3e3ub	ryanodonnell: @AttractMode Thanks for putting on such a great event. Can't wait for the inevitable sequels! [..	ryanodonnell attractmode thanks putting great event . ' wait inevitable sequels ! [..	ryanodonnell attractmode thank put great event . ' wait inevitable sequel ! [..	ryanodor thank e se
Mmmbaileys	@damygeebo Carli's my friend	@damygeebo Carli's my friend	damygeebo carlis friend	damygeebo carlis friend	damyge carlis fri
jasminejoejonas	I feel so great for starting twitter at suzanne but still hardly anyone has it.	I feel so great for starting twitter at suzanne but still hardly anyone has it.	feel great starting twitter suzanne still hardly anyone	feel great start twitt suzanne still hardly anyone	suza

username	Text	text_clean	text_normalized	text_lemma	text_no
rsuenaga	@Luke_Stephens: I just said I was wondering about it, not that I wanted it.	@Luke Stephens: I just said I was wondering about it, not that I wanted it.	luke stephens said wondering , wanted .	luke stephens say wonder , want .	luke steph

Explore Data

Show data types in each column

In [29]:
```
df_sm.dtypes
```

Out[29]:
```
sentiment          int64
ID                 int64
Time              object
none              object
username          object
Text              object
text_clean        object
text_normalized   object
text_lemma        object
text_nouns        object
text_adjectives   object
text_verbs        object
text_nav          object
no_tokens        float64
dtype: object
```

Summary of numerical features

Not the most useful thing, but helpful as a quick sanity check.

In [37]:
```
df_sm.describe().transpose()
```

Out[37]:

	count	mean	std	min	25%	50%	
sentiment	4000.00	2.00	2.00	0.00	0.00	2.00	
ID	4000.00	1898398236.12	430453578.94	1467822272.00	1468005627.25	1898271122.50	232
no_tokens	4000.00	9.49	5.50	1.00	5.00	8.00	

Exploring word frequencies

In [38]:
```
df_sm[['text_clean','text_normalized','text_lemma','text_nav']].sample(10)
```

Out[38]:

	text_clean	text_normalized	text_lemma	text_nav
801407	@timjennion morning mr hows things?	timjennion morning mr hows things	timjennion morning mr how s thing	timjennion morning mr s thing
799078	So sad to see you go Farrah. Thanks for being my hottie girl I wanted to be when I grew up!	sad see go farrah thanks hottie girl wanted grew	sad see go farrah thank hottie girl want grow	farrah thank hottie girl sad see go want grow
798397	@janportfolio I have my colleagues do very advanced air apps.. but they won't be available for freelance works.	janportfolio colleagues advanced air apps . ' available freelance works .	janportfolio colleague advance air app . ' available freelance work .	janportfolio colleague air app freelance work available advance
799656	@SandyPaws That makes sense. The Waiting Game.	sandypaws makes sense waiting game .	sandypaw make sense wait game .	sandypaw sense game make wait
801354	...and relax! I'm on the train!! you don't get drama like that on the BBC!	. relax ! ' train !! ' get drama like bbc !	. relax ! ' train ! ! ' get drama like bbc !	train drama bbc relax
800926	@daNanner Yeah, just finished #castle. Was pretty good.	dananner yeah finished # castle . pretty good .	dananner yeah finish # castle . pretty good .	dananner # castle good finish
798990	Once again, I'm chickening out...	im chickening ...	-PRON- be chickening ...	chickening be
801155	going to pick up @30comau in a sec its our anniversary today	going pick 30comau sec anniversary today	go pick 30comau sec anniversary today	pick 30comau sec anniversary today go
800061	loving life... and loving you	loving life . loving	love life . love	life love love
799704	@CHIPPEWA09 I wanna have a drink too.	chippewa09 wanna drink	chippewa09 wanna drink	chippewa09 wanna drink

In [39]:
```
# Import matplotlib and seaborn and adjust some defaults
%matplotlib inline
%config InlineBackend.figure_format = 'svg'

from matplotlib import pyplot as plt
plt.rcParams['figure.dpi'] = 100

import seaborn as sns
sns.set_style("whitegrid")
```

Creating a List of Tokens from a List of Documents

In [40]:
```
def my_tokenizer(text):
    return text.split() if text != None else []
```

In [41]:
```
tokens = df_sm.text_nav.map(my_tokenizer).sum()
```

In [42]:
```
print(tokens[:200])
```

```
['work', 'job', 'week', 'slow', 'be', 'consider', 'quit', 'davidvancamp', 'wis
h', 'mine', 'joke', 'awful', 'fat', 's', 'would', 'stop', 'make', 'chapter', 'p
rofessional', 'life', 'new', 'guess', 'nee', 'start', 'sandrabernhard', 'miss',
'lady', 'web', 'store', 'age', 'rec', 'v', 'merch', 'bring', 'know', 'take', 'c
artertwinszach', 'hope', 'love', 'hurt', 'sorry', 'horrible', 'sick', 'sad', 'm
ad', 'be', 'feel', 'make', 'feel', 'stewartwade', 'pig', 'end', 'akimbo', 'nes
s', 'sure', 'great', 'visual', 'know', 'ouh', 'babe', 'franzi', 'hast', 'du',
'hun', 'hoffentlich', 'nicht', 'schlimme', 'miss', 'mary', 'headache', 'bad',
'wake', 'macekmakeupart', 'movie', 'theatre', 'hope', 'guy', 'last', 'can', 're
member', 'see', 'fun', 'go', 'see', 'day', 'class', 'miss', 'chichi', 'last',
'be', 'go', 'time', 'pedicure', 'cabinet', 'shit', 'damn', 'open', 'chip', 'toe
nail', 'happen', 'mikegentile', 'walmart', 'joke', 'have', 'amedelrivero', 'pay
check', 'future', 'start', 'put', 'prepare', 'get', 'patrickeatworld', 'takbole
h', 'life', 'fmmfl', 'loveeeeeeeeeeee', 'suck', 'church', 'thur', 'watch', 'fb
c', 'line', 'thur', 'forget', 'hopeformusic', 'yikes', 'hayfever', 'med', 'hel
p', 'good', 'hate', 'errand', 'tonight', 'run', 'dayleave', 'brain', 'break',
'work', 'widget', 'facebook', 'work', 'twitt', 'can', 'glass', 'sleep', 'contac
t', 'miss', 'yoouuu', 'glass', 'have', 'lose', 'make', 'shopping', 'shirt', 'sh
ort', 'idea', 'london', 'attention', 'wallet', 'good', 'little', 'get', 'pay',
'appear', 'grandma', 'surgery', 'hospital', 'good', 'back', 'hope', 'start', 'i
mprove', 'look', 'snowqueen297', 'orbitz', 'winner', 'anounced', 'price', 'kann
agi', 'figure', 'make', 'cry', 'luuuuuuuuuuuuuv', 'bad', 'enough', 'itsuki', 'b
owl', 'cereal', 'milk', 'cherrio', 'lame', 'pour', 'remember', 'hour', 'day',
'way', 'jail', 'well', 'feel', 'guess']
```

Counting Frequencies with a Counter

In [43]:
```
from collections import Counter

counter = Counter(tokens)
counter.most_common(20)
```

Out[43]:
```
[('farrah', 455),
 ('go', 387),
 ('fawcett', 335),
 ('get', 282),
 ('good', 282),
 ('sad', 261),
 ('be', 255),
 ('day', 236),
 ('work', 199),
 ('love', 190),
 ('today', 175),
 ('miss', 166),
 ('time', 161),
 ('quot', 157),
 ('rip', 147),
 ('see', 146),
 ('die', 146),
 ('know', 145),
 ('make', 143),
 ('want', 139)]
```

In [44]:
```
print([t[0] for t in counter.most_common(200)])
```

```
['farrah', 'go', 'fawcett', 'get', 'good', 'sad', 'be', 'day', 'work', 'love',
'today', 'miss', 'time', 'quot', 'rip', 'see', 'die', 'know', 'make', 'want',
'think', 'thank', 'feel', 'night', 'morning', 'pass', 'new', 's', 'lol', 'hop
e', 'can', 'say', 'come', 'would', 'watch', 'twitter', 'need', 'rest', 'angel',
'take', 'u', 'peace', 'tomorrow', 'last', 'try', 'look', 'great', 'bad', 'cance
r', 'sleep', 'bed', 'friend', 'wish', 'tonight', 'start', 'find', 'happy', 'hea
r', 'week', 'life', 'guy', 'thing', 'fun', 'could', 'wait', 'leave', 'show', 'h
our', 'way', 'nice', 'much', 'man', 'may', 'people', 'news', 'let', 'lose', 'ha
te', 'family', 'sorry', '@', 'tell', 'eat', 'tweet', 'use', 'give', 'long', 'gu
ess', 'do', 'year', 'read', 'break', 'head', 'talk', 'suck', 'school', 'have',
'glad', 'call', 'haha', 'charlie', 'beautiful', 'hair', 'jonathanrknight', 'lit
tle', 'hot', '#', 'well', 'big', 'many', 'mean', 'awesome', 'help', 'home', 'go
d', 'cool', 'yay', 'house', 'keep', 'first', 'lot', 'hurt', 'ugh', 'sound', 'sw
eet', 'post', 'old', 'sick', 'update', 'stay', 'song', 'car', 'omg', 'listen',
'late', 'r', 'movie', 'remember', 'happen', 'right', 'play', 'next', 'run', 'st
uff', 'live', 'girl', 'ill', 'check', 'battle', 'ready', 'kid', 'dream', 'amazi
ng', 'sure', 'least', 'lunch', 'will', 'reply', 'thought', 'enjoy', 'put', 'bab
y', 'book', 'world', 'buy', 'music', 'tired', 'goodnight', 'twitt', 'idea', 'ra
in', 'crazy', 'summer', 'person', 'change', 'fight', '¿', 'job', 'age', 'end',
'wake', 'forget', 'food', 'early', 'agree', 'place', 'excited', 'move', 'wonde
r', 'month', 'phone', 'prayer', 'farah', 'sooo', 'email', 'poor', 'party', 'fol
lower', 'send', 'ur']
```

Remove stopwords from list of tokens

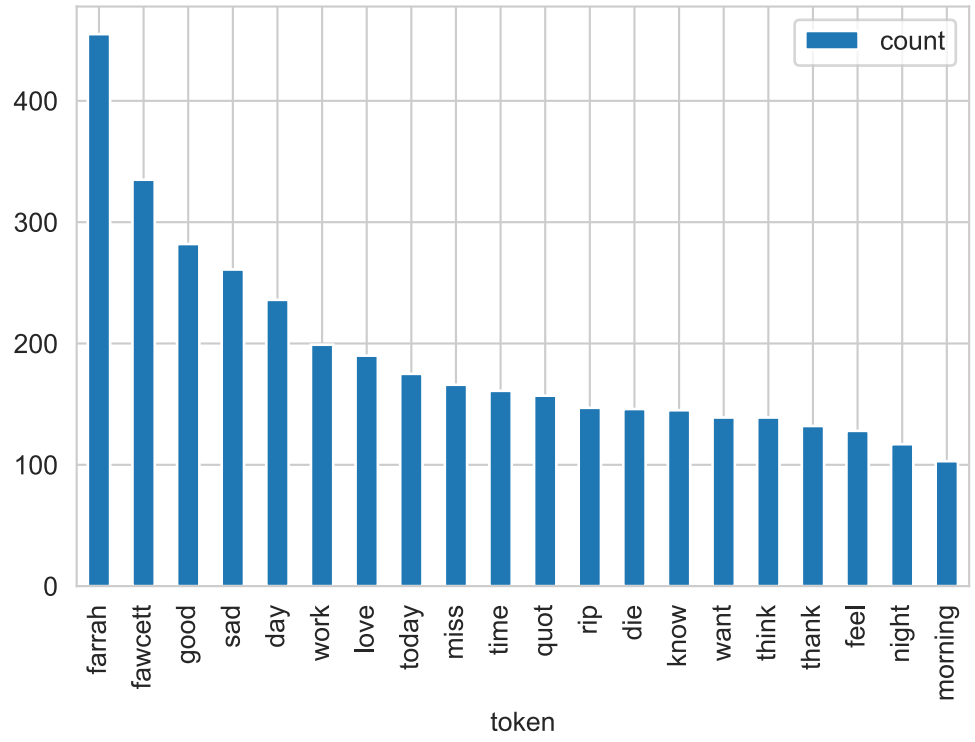
In [46]:
```
from spacy.lang.en.stop_words import STOP_WORDS

def remove_stopwords(tokens):
    """Remove stopwords from a list of tokens."""
    return [t for t in tokens if t not in STOP_WORDS]

# rebuild counter
counter = Counter(remove_stopwords(tokens))
```

In [47]:
```
# convert list of tuples into data frame
freq_df = pd.DataFrame.from_records(counter.most_common(20),
                                    columns=['token', 'count'])

# create bar plot
freq_df.plot(kind='bar', x='token');
```



Word clouds

In [48]:
```
%matplotlib inline
import matplotlib.pyplot as plt
```

In [51]:
```
from wordcloud import WordCloud

def wordcloud(counter):
    """A small wordloud wrapper"""
    wc = WordCloud(width=1200, height=800,
                   background_color="white",
                   max_words=200)
    wc.generate_from_frequencies(counter)

    # Plot
    fig=plt.figure(figsize=(6, 4))
    plt.imshow(wc, interpolation='bilinear')
    plt.axis("off")
    plt.tight_layout(pad=0)
    plt.show()
```

The Word Cloud!!

In [52]:
```
wordcloud(counter)
```

In [ ]:
```
from nltk.probability import FreqDist

explore_text = np.array(df_sm['text_normalized'])
fdist = FreqDist(explore_text)
print(fdist)
```

Sentiment Analysis with Afinn

As a quick and dirty sanity check, I've set up Afinn in the early stages of data cleaning, and intend to keep a little record of Afinn's performance, as I increase the rigour of the data cleaning.

In [ ]:
```
from afinn import Afinn

afn = Afinn(emoticons=True)
```

In [ ]:
```
texts = np.array(df_sm['text_clean'])
sentiments = np.array(df_sm['sentiment'])

# extract data for model evaluation
#train_texts = texts[:10000]
#train_sentiments = sentiments[:10000]

#test_texts = texts[40000:60000]
#test_sentiments = sentiments[40000:60000]
sample_ids = [626, 533, 310, 123, 654, 400]
```

In [ ]:
```
for text_clean, sentiment in zip(texts[sample_ids], sentiments[sample_ids]):
    print('TEXT:', texts)
    print('Actual Sentiment:', sentiment)
    print('Predicted Sentiment polarity:', afn.score(texts))
    print('-'*60)
```

In [ ]:
```
# Predict sentiment with Afinn

sentiment_polarity = [afn.score(Text) for Text in normalized_texts]
#predicted_sentiments = ['positive' if score >= 1.0 else 'negative' for score in
predicted_sentiments = [4 if score >= 1.0 else 0 for score in sentiment_polarity
```

In [ ]:
```
#meu.display_model_performance_metrics(true_labels=test_texts, predicted_labels=p
#                                      classes=['positive', 'negative'])
meu.display_model_performance_metrics(true_labels=test_sentiments, predicted_labe
                                      classes=[4, 0])
```

Checking cleaning with Afinn

I'm curious about how deeper cleaning affects predicitive models. So I set up Afinn after the very first round of data cleaning, and am going to track results here in the markdown. For simplicity, I will monitor the effects of different levels of cleaning on "weighted avg f1-score"

Round 1, most basic cleaning, 20000 rows: 0.63

Round 2, include normalization, 20000 rows: 0.63

Bag of Words model

In [ ]:
```
from sklearn.feature_extraction.text import CountVectorizer

texts = np.array(df_sm['text_normalized'])

cv = CountVectorizer(min_df=0., max_df=1.)
cv_matrix = cv.fit_transform(texts)
cv_matrix = cv_matrix.toarray()
cv_matrix
```

In [ ]:
```
# get all unique words in the corpus
vocab = cv.get_feature_names()
# show document feature vectors
pd.DataFrame(cv_matrix, columns=vocab)
```

NLP

In [ ]:
```

```

Load spaCy

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

Iterate over all rows and perform NLP

In [ ]:
```
for i, row in df_sm.iterrows():
    if i % 1000 == 0:
        print(i)
    if(row["text_clean"] and len(str(row["text_clean"])) < 1000000):
        doc = nlp(str(row["text_clean"]))
        adjectives = []
        nouns = []
        verbs = []
        lemmas = []

        for token in doc:
            lemmas.append(token.lemma_)
            if token.pos_ == "ADJ":
                adjectives.append(token.lemma_)
            if token.pos_ == "NOUN" or token.pos_ == "PROPN":
                nouns.append(token.lemma_)
            if token.pos_ == "VERB":
                verbs.append(token.lemma_)

        df_sm.at[i, "selftext_lemma"] = " ".join(lemmas)
        df_sm.at[i, "selftext_nouns"] = " ".join(nouns)
        df_sm.at[i, "selftext_adjectives"] = " ".join(adjectives)
        df_sm.at[i, "selftext_verbs"] = " ".join(verbs)
        df_sm.at[i, "selftext_nav"] = " ".join(nouns+adjectives+verbs)
        df_sm.at[i, "no_tokens"] = len(lemmas)
```

Check results

In [ ]:
```
df_sm.head()
```

Save to database

In [ ]:
```
df.to_sql('posts_nlp', con)
```

In [ ]:
```
```

In [ ]:
```
```