DigitalDNA_python_package

October 21, 2019

1 Digital DNA - python

1.1 Getting Started

This package provides a set of utilities and algorithms for online social bot detection based on the paper Social Fingerprinting: Detection of Spambot Groups Through DNA-Inspired Behavioral Modeling

1.1.1 import libraries

Let's start with the import libraries

```
In [1]: import digitaldna as ddna
        import csv
        from digitaldna import Verbosity
        from digitaldna import SequencePlots
        import pandas as pd
        import numpy as np
        from digitaldna.lcs import LongestCommonSubsequence
        from os import listdir
        import os
        from matplotlib import pyplot as plt
        import time
        from digitaldna import SequencePlots
```

1.1.2 Sequence your first Digital DNA from Twitter.

You can sequence from a JSON containing the timeline

```
In [2]: from digitaldna import TwitterDDNASequencer

        filepath = os.path.join(os.getcwd(), "timelines.json")
        model = TwitterDDNASequencer(input_file=filepath, alphabet='b3_type')
        data = model.fit_transform()

In [3]: df = pd.DataFrame({"user_id": data[:, 0],
                           "dna": data[:, 1]})
        df
```

Out[3]:

	user_id	dna
0	48062712	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCACCCCCCCCCC...
1	2479063608	AAAAATCTACTAAATATTTTAAATCAAACTTCCAACTAACTAATTC...
2	22834067	ACACCCAAAAACACCCCACAACAAACAAAAAACCACACACAACAAA...
3	4289404586	AAAAAAAAACAAAACCAACCCCACAACCCCAAAAAACTCCAACAAA...
4	615597661	AAAAAAAAAAACAAACCAAAACAAAAAAAAAAAAAAATAATAAATTA...
5	1135017996	ACACAAATTTACTAAAATTTCAACTTCATATTTTTATCAAAATAAA...

Or a previously DNA sequenced timeline

In [4]:
```
filepath = os.path.join(os.getcwd(), "italian_retweets_users_sequences_new.csv")
df = pd.read_csv(filepath)
df
```

Out[4]:

	user_id	tweet_count	\
0	1123481	3208	
1	3500831	3214	
2	14871003	3193	
3	14906561	3204	
4	17880873	3212	
5	19017283	3170	
6	20432064	3195	
7	22009205	3209	
8	25851958	3204	
9	32604352	2463	
10	33841072	3201	
11	38846888	3219	
12	45394510	3237	
13	50265477	3185	
14	51703470	3205	
15	53002473	3222	
16	55404019	3239	
17	58165167	3170	
18	70741097	3178	
19	82659565	3210	
20	96738439	3242	
21	121677739	544	
22	125072714	3207	
23	132962832	3176	
24	133460400	3191	
25	135547288	3234	
26	136653194	3196	
27	148803815	3204	
28	160179878	3142	
29	162672772	3214	
..	...	...	
708	1011238575399165959	750	
709	1011334857966931968	2965	

710	1011632355419844608	1379
711	1011716963498909696	2693
712	1011723273057554438	1353
713	1011923110617116672	610
714	1011936924125327361	411
715	1011940333389836288	277
716	1011952811674828800	2984
717	1011956312580804608	2886
718	1011975267915575297	824
719	1011987366582603776	326
720	1011995299622064131	384
721	1012049887947456512	1513
722	1012063518659698690	295
723	1012070142719774720	748
724	1012085786043453440	146
725	1012089312794509312	1281
726	1012110155553570816	442
727	1012132071073112064	2828
728	1012138187576180736	2866
729	1012147848496869376	1784
730	1012258265772720128	469
731	1012295892718751747	3073
732	1012302562660405248	457
733	1012308408748396544	2544
734	1012332423290478593	1181
735	1012447780504326145	2175
736	1012607637026983936	555
737	1013118357925777409	2482

	dna	bot	retweets(%)
0	AACCCCCCACAACCCCCCCCCCACACATCAATCCCACCCCCAACCA...	0	62
1	CCTTACCCCTACTACTTCAAACCAAACAAAACAACCCATCTCCCTT...	0	27
2	CCTACATACCCCCTCCAAACCACCCACACCACAACCCCCCCCCCCC...	0	66
3	ACCCCACCCCCCCAAAAAAAACCCCCCCCAAAAACCCCCAACCCCA...	0	58
4	CCCCTCCTCCCTTCCCTCTTTCCACTCCCTCTCCTTTCCTTTCCCC...	0	44
5	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...	1	100
6	TCTTTAACAACCACACAACCCCAAAAAAAATCCCCCAAAACACACC...	0	67
7	TTTTATAATAATCCCCTCCATATCTCCTCTTTTCATATCTTTTTTC...	0	20
8	TTACACTTACAACCACTCTACATTCCCACTTTTATTACCCACCTTA...	0	45
9	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCACCCC...	1	79
10	TCCACCCTCCCCTACCCCCCCCCCCCCACCCCCCCCCCCCCCCCC...	0	79
11	CCCCCACCCCCCCCCCTAACCCACCCACCCCCCCCTCCCCCCCCC...	0	69
12	ACTCACACCAAAACACTCCCCTCCCTACCCCCCCCCCCCTCCCCCC...	0	79
13	CCAACCCCCCCCCCCCCCAAAAACCAAAAAAAACCCCCCCCCCCCC...	0	64
14	TATAATAAAAATTATTTTAATATATATTAAAAATTAAATAATATAT...	0	30
15	TCACTCTCCACCAATCCCAACTACCCCCATTCCCTTTTACTTATAT...	0	31
16	CAAAACAACCCCCCCCAAACAACCCCCCCCCCCCCCCAAACCCCCA...	0	67
17	CTCTAACCTAAAATAACAAAAATACAAATCCACAAAAAAAACAAAA...	1	55

18	TCCTCCTCTCTCCCTTTTTTTTTTTTCTCTTTCTCTTTCTTTTCCC...	0	49
19	CTCACCATTTTCCCCCCACATTCCCCCCCCCCCCCACCCCATCCACC...	1	60
20	CCCCACCACACACACTAAACCCCCCCCCCCCCCCCCCCCCCACCACC...	0	88
21	ATAACATCTTAAAAAAAATACCCCTCCTTAAAACCAACCAACCCCC...	0	74
22	AAAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAC...	1	59
23	TTTTATTTAACTACCAATTTAAATAATAACATAATCTTTTTCTTCA...	0	40
24	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...	0	99
25	CCCCAACCACTCCCACCCTCACCTTACCCAACCCCCCCCCCTACAC...	0	75
26	TCCTTCCTCCCCTTCCTTTTACTTTATTCTTTTTTTTTTTTCCCCC...	0	39
27	CCAACACCCCCACAAAAAACAAAACCCAAAATACCCCCCCCCCAAA...	0	81
28	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCTCCCCCCCCCCC...	0	94
29	CCCCCCCCCCAAAAAACAAAAAAAAAAAAAAAAAAACAAAAAAAA...	1	68
..	...	...	...
708	ACCCCACACACCCACCCTACACCACCCCCCCCCACACTTTCCCCCCT...	0	86
709	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...	0	97
710	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...	1	99
711	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...	0	96
712	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...	0	100
713	ACCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...	0	96
714	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...	1	99
715	AAACCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...	1	99
716	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCTCCCCCCCCCCC...	1	93
717	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...	1	99
718	CCCCCCACCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...	1	99
719	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...	1	99
720	AAAAAAAAAAAAAAAAACCCCCCCCCCCCACCCCCCCCCCCCCCC...	0	93
721	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...	1	88
722	AAAACCCCCCCCCCCCACCCCCCTTTCCCCCTCCCCCCCCCCCCC...	0	92
723	CTCCTCCCCCCCCCCCCCCACCTCCCCCCCCCCCCCCACCCCCCCC...	1	94
724	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...	0	97
725	ACCACACCCCCCCCCCCCCTCTCCCCCTCCCCCCCCCACCCTCCCC...	0	80
726	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...	1	100
727	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCACCCCCCCCCACCCCACCAC...	1	97
728	TCCCCCCCACCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...	0	75
729	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...	1	97
730	CCCCCCCCACCACCCCCCCCCCCCCCCCCCCCCCCCCCCCCCAAA...	1	97
731	CCCCCACCCCCCCCCCCCCCCCCTCCCCCCCCCCCCACCCCCCCCC...	0	93
732	ACCCCCCTCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCACCCC...	1	92
733	CACCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...	0	98
734	CCCCCACCCCCACCCCCCCCACACCCACCCCCACCACCCCCCCCCC...	0	92
735	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...	0	94
736	CCCCCCCCCCCCCCCCCCCCTCCCCCCCCCCCCCCCCCCCCTCCCCTCC...	1	70
737	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...	0	99

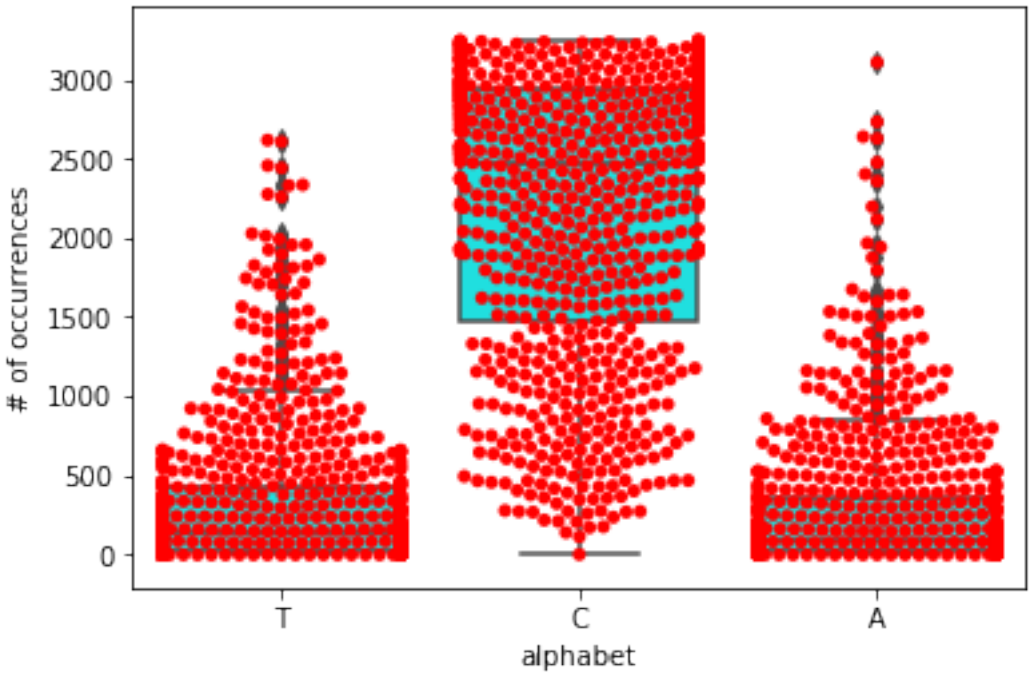
[738 rows x 5 columns]

1.1.3 Plot sequence distribution

Alphabet distribution show the distribution of users' "actions" in the database.

```
In [5]: plotter = SequencePlots(alphabet='b3_type')
        plotter.plot_alphabet_distribution(df["dna"])
```

```
Out[5]: <digitaldna.sequence_plots.SequencePlots at 0x1a358c0358>
```



1.2 LCS and bot detection analysis

Thanks to the `digitaldna` we can make a bot detection thanks to the sequence behaviour of Twitter actions with just a few simple commands

```
In [6]: est = LongestCommonSubsequence(in_path='', out_path='/tmp/glcr_cache', overwrite=False,
        y = est.fit_predict(df["dna"])
        df["bot"] = y
```

```
fitting...
finding cut...
predicting...
done.
```

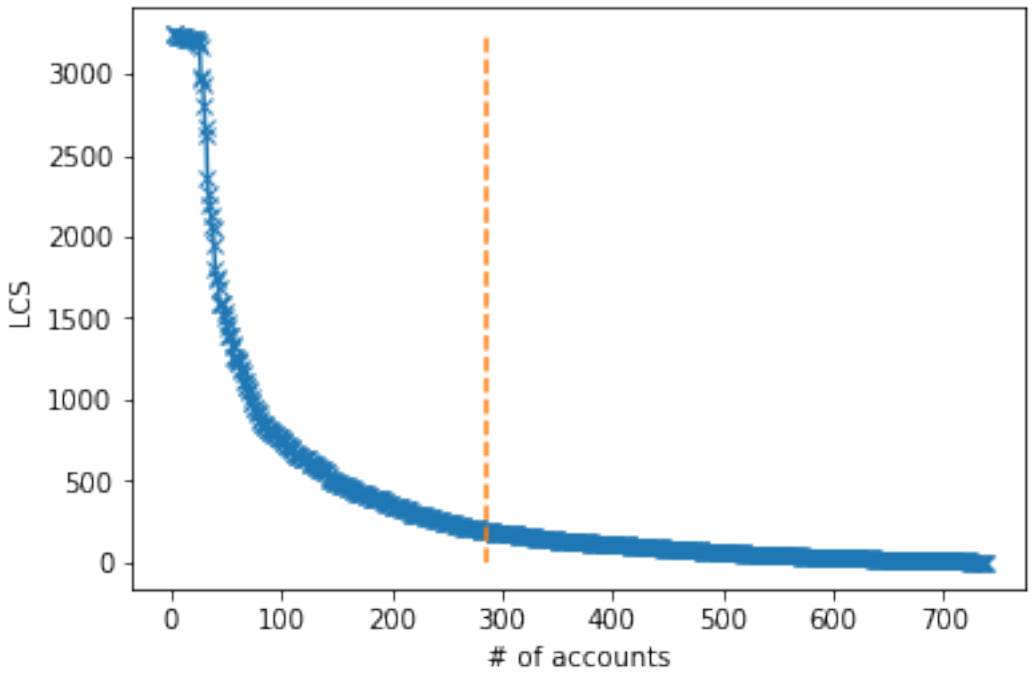
```
In [7]: df[:10]
```

```
Out[7]:     user_id  tweet_count                                                dna  \
        0   1123481         3208  AACCCCCCACAACCCCCCCCCCACACATCAATCCCACCCCCAACCA...
        1   3500831         3214  CCTTACCCCTACTACTTCAAACCAAACAAAACAACCCATCTCCCTT...
        2  14871003         3193  CCTACATACCCCCTCCAAACCACCCACACCACAACCCCCCCCCCCC...
        3  14906561         3204  ACCCCACCCCCCCAAAAAAAAACCCCCCCCAAAAACCCCCAACCCCA...
        4  17880873         3212  CCCCTCCTCCCTTCCCTCTTTCCACTCCCTCTCCTTTCCTTTCCCC...
        5  19017283         3170  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC...
        6  20432064         3195  TCTTTAACAACCACACAACCCCAAAAAAAATCCCCCAAAACACACC...
        7  22009205         3209  TTTTATAATAATCCCCTCCATATCTCCTCTTTTCATATCTTTTTTC...
        8  25851958         3204  TTACACTTACAACCACTCTACATTCCCACTTTTATTACCCACCTTA...
        9  32604352         2463  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCACCCC...

             bot  retweets(%)
        0  False           62
        1  False           27
        2  False           66
        3  False           58
        4  False           44
        5   True          100
        6  False           67
        7  False           20
        8  False           45
        9  False           79
```

1.2.1 LCS linear plot

Plots the longest common subsequence curve as (number of accounts, sequence length). Orange threshold can be assigned or computed over the smoothed curve of LCSs and highlights possible automated accounts on its left.

```
In [8]: est.plot_LCS()

Out[8]: <module 'matplotlib.pyplot' from '/Users/salvob/anaconda3/lib/python3.7/site-packages/ma
```



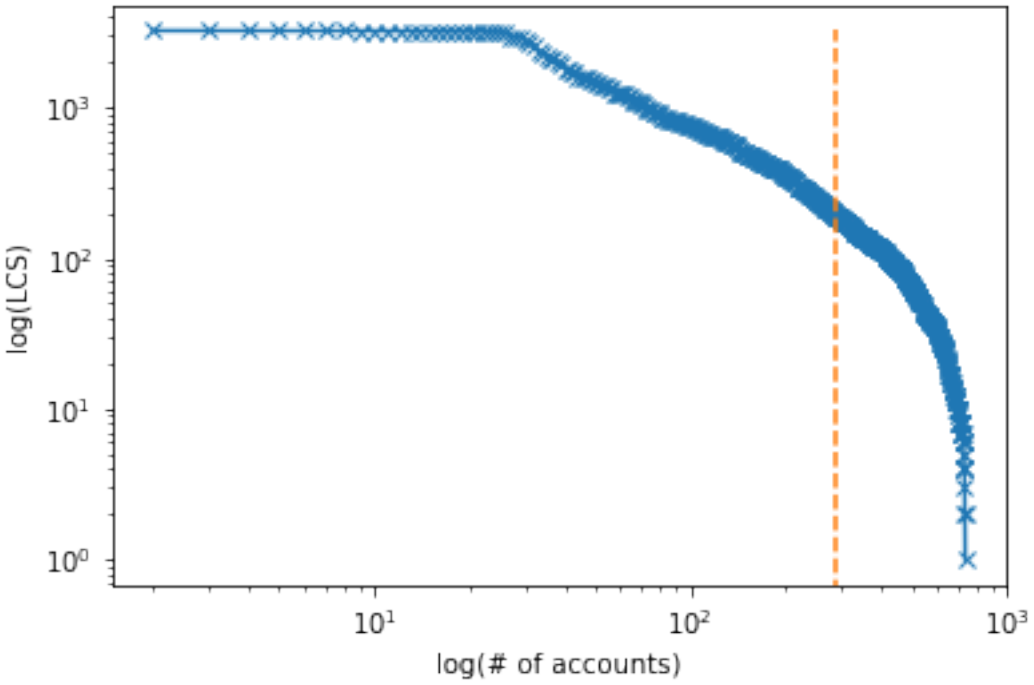
1.2.2 LCS logarithmic plot

Plots the longest common subsequence curve as log(number of accounts), log(sequence length).

LCS lengths (y axes) and the number of accounts who share the same LCS length (x axes) are logarithmic transformed in order to easily visualise patterns: - Plateaux in the curve highlight homogenous groups of highly similar accounts. - Steep declining in the curve indicates that group of accounts are different.

```
In [9]: est.plot_LCS_log()
```

```
Out[9]: <module 'matplotlib.pyplot' from '/Users/salvob/anaconda3/lib/python3.7/site-packages/ma
```

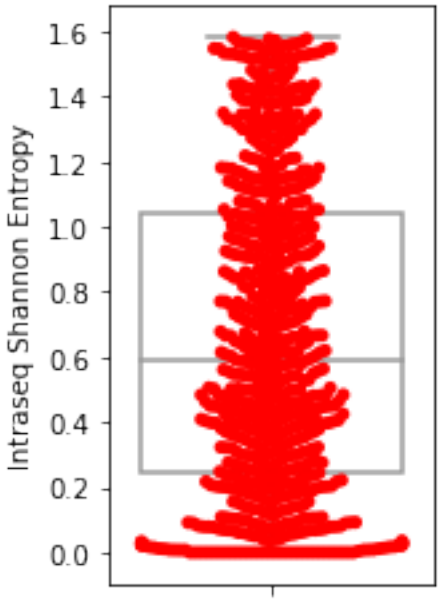


1.3 Entropy plot

1.3.1 Intrasequence plot

Intra-sequence entropy boxplot shows the distribution of Shannon entropy computed over each timeline.

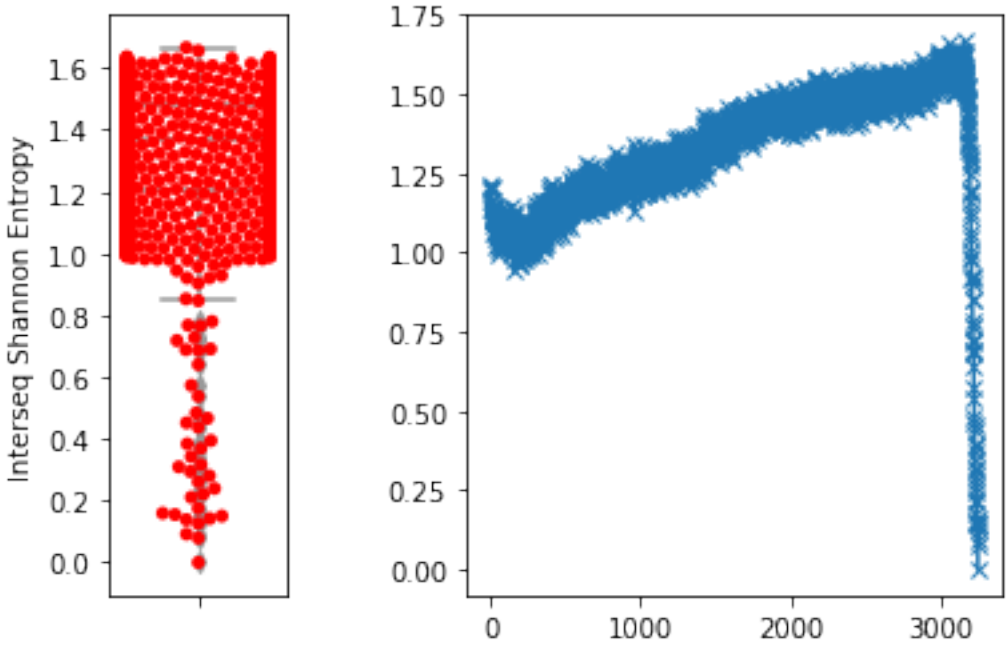
```
In [10]: plotter = SequencePlots(alphabet='b3_type')
         intra_seq = plotter.plot_intrasequence_entropy(df["dna"])
```



1.3.2 Intersequence plot

- Inter-sequence entropy boxplot (left) shows the distribution of Shannon entropy in each timeline.
- Inter-sequence entropy plot (right) shows entropy calculated for each timeline's position. The same action in the same position means low entropy value thus a interesting pattern.

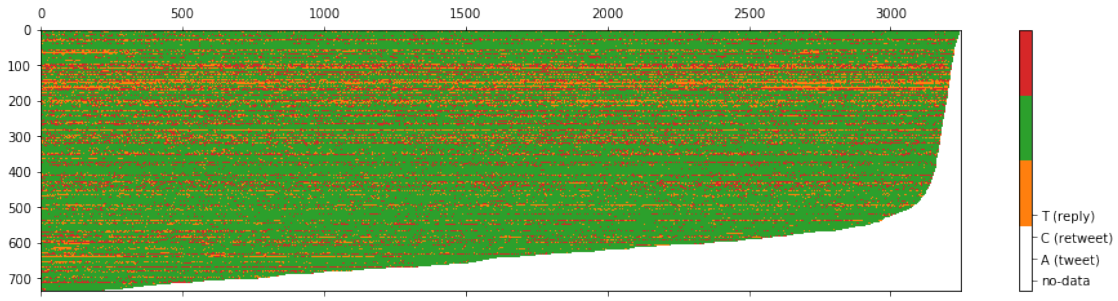
```
In [11]: inter_seq = plotter.plot_intersequence_entropy(df["dna"])
```



1.3.3 Plot Sequence Color

Color sequence allow to easily identify suspicious patterns: large blocks of the same colour highlight high likelihood that several accounts act synchronically.

In [12]:
```
plotter.plot_sequences_color(df["dna"])
```



Out[12]:
```
<digitaldna.sequence_plots.SequencePlots at 0x1a22520128>
```

In [ ]:
```
```