Freie Universität Berlin

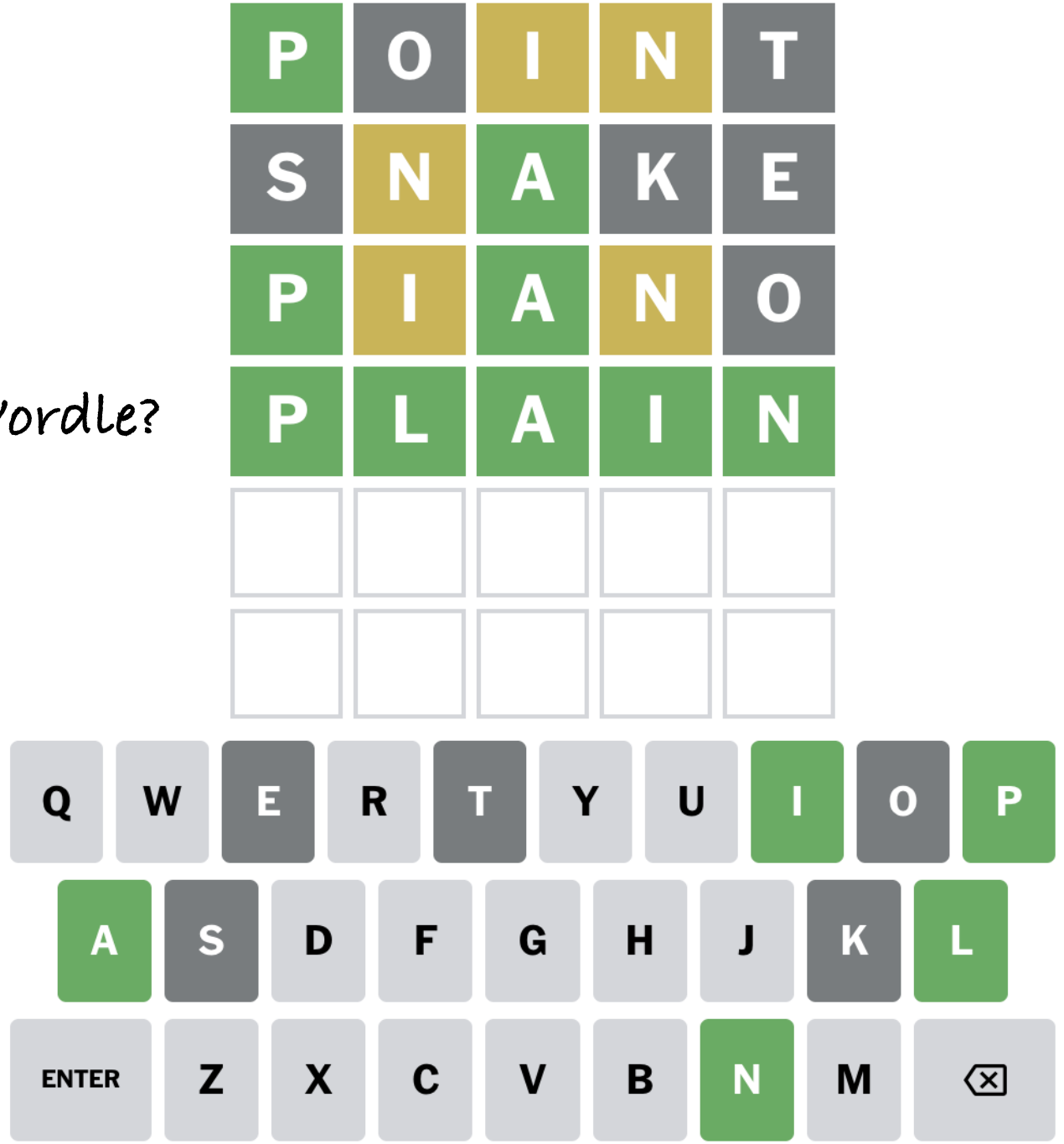
Wordle

Strategie für Wordle

Ying Lo, Mingjie Peng

Was ist Wordle?

Gibt es eine Strategie für Wordle?



Strategie für Wordle

Freie Universität Berlin

Ziel: Minimierung der Zugzahl

Wie finden wir das optimale Anfangswort, um Wordle in möglichst wenigen Zügen zu lösen?

Anfangswort

Erste Heuristik: Positions-Wahrscheinlichkeiten

1. Gesamten Antwortwortschatz durchzählen.
2. Für jede Position die Häufigkeit aller 26 Buchstaben bestimmen.
3. Score(w) = Summe der fünf Einzelwahrscheinlichkeiten.

Erste Heuristik: Positions-Wahrscheinlichkeiten

$\Sigma = \{A, B, \dots, Z\}$

$D = \{x \in \Sigma^5 \mid x \text{ ist in Databank}\}$

$Anfangswort\ \ w \in D$

Für alle Position i und Buchstabe l:

$$P(x_i = l) = \frac{|x \in D : x_i = l|}{|D|}$$

$$Score = \sum_{i=1}^{5} P(i_{te} Buchstabe = w_i)$$

Gedanke: Hoher Score ⇒ Wort sieht „typisch" wie eine Lösung aus.

Das heißt, es gibt viele grüne/gelbe Felder

Positions-Wahrscheinlichkeiten

Freie Universität Berlin

Beispiel:

PRINT, POINT, WATER, WASTE, PONNY, POWER, SNACK, SLEEP, SLICE, FANCY

P	0.4	**R**	0.1	**I**	0.3	**N**	0.3	**T**	0.2	1.3
P	0.4	**O**	0.3	**I**	0.3	**N**	0.3	**T**	0.2	1.5
W	0.2	**A**	0.3	**T**	0.1	**E**	0.3	**R**	0.2	1.1
W	0.2	**A**	0.3	**S**	0.1	**T**	0.1	**E**	0.2	0.9
P	0.4	**O**	0.3	**N**	0.2	**N**	0.3	**Y**	0.2	1.4
P	0.4	**O**	0.3	**W**	0.1	**E**	0.3	**R**	0.2	1,3
S	0.3	**N**	0.1	**A**	0.1	**C**	0.3	**K**	0.1	0.9
S	0.3	**L**	0.2	**E**	0.1	**E**	0.3	**P**	0.1	1.0
S	0.3	**L**	0.2	**I**	0.3	**C**	0.3	**E**	0.2	1.3
F	0.1	**A**	0.3	**N**	0.2	**C**	0.3	**Y**	0.2	1.1

Das bedeutet: POINT ist das Beste Anfangswort.

Positions-Wahrscheinlichkeiten

Freie Universität Berlin

Python --- Pseudocode

Algorithm 1 Main

```
procedure MAIN
    0. Initialisierung
    Set random seed ← RANDOM_SEED
    posProbs ← POSITIONALPROBABILITIES(WORD_LIST)
    rows ← empty list

    1. Schleife über alle Kandidaten als guess
    for all guess ∈ WORD_LIST do
        g_total ← 0                          ▷ Gesamtzahl grüner Felder
        y_total ← 0                          ▷ Gesamtzahl gelber Felder
        for i ← 1 to N_SAMPLES do
            ans ← random choice from WORD_LIST
            fb ← FEEDBACK(guess, ans)
            g_total += count("G", fb)
            y_total += count("Y", fb)
        end for
        avg_g ← g_total/N_SAMPLES
        avg_y ← y_total/N_SAMPLES
        avg_total ← avg_g + avg_y
        score ← WORDSCORE(guess, posProbs)
        ROWS.APPEND({)
      word: guess,
      score_sum: round(score, 3),
      avg_green: round(avg_g, 3),
      avg_yellow: round(avg_y, 3),
      avg_total: round(avg_total, 3)
    }
    end for

    DataFrame erstellen und sortieren
    df ← DataFrame(rows)          ▷ sortiert nach score_sum absteigend
    corr ← CORRELATION(df[score_sum], df[avg_total])

    Ausgabe
    PRINT(df)
    PRINT("Correlation between score_sum and avg_total:", corr)
end procedure
```

Positions-Wahrscheinlichkeiten

Python --- Ergebnis

Beispiel:

PRINT, POINT, WATER, WASTE, PONNY, POWER, SNACK, SLEEP, SLICE, FANCY

word	score_sum	avg_green	avg_yellow	avg_total
POINT	1.5	1.546	0.484	2.030
PONNY	1.4	1.365	0.194	1.559
PRINT	1.3	1.298	0.670	1.968
POWER	1.3	1.305	0.610	1.915
SLICE	1.3	1.304	0.393	1.697
WATER	1.1	1.074	0.790	1.864
FANCY	1.1	1.080	0.429	1.509
SLEEP	1.0	1.083	0.687	1.770
WASTE	0.9	0.906	1.114	2.020
SNACK	0.9	0.888	0.809	1.697

Correlation between score_sum and avg_total: 0.094

Schlecht!!

Vereinfachter Fall

- Die Sequenz ist dreistellig.
- Jede Position darf nur die Ziffer 0, 1 oder 2 enthalten.
- erste Stelle ≥ zweite Stelle ≥ dritte Stelle)

Welche Ziffer rate ich zuerst an der ersten Stelle der dreistelligen Zahl?

2

Warum 2?

000
100
110
111
200
210
211
220
221
222

Vereinfachter Fall

0	000	1
1	100 110 111	3
2	200 210 211 220 221 222	6

0	$S_{0,p(0)} = 1$ $S_{0,p(1)} = 9$ $S_{0,p(2)} = 9$	19/3 Max=9
1	$S_{1,p(0)} = 7$ $S_{1,p(1)} = 3$ $S_{1,p(2)} = 7$	17/3 Max=7
2	$S_{2,p(0)} = 4$ $S_{2,p(1)} = 4$ $S_{2,p(2)} = 6$	14/3 Max=6

2 ist am besten

000
100
110
111
200
210
211
220
221
222

Vereinfachter Fall

Freie Universität Berlin

Welche dreistellige Zahl sollte man als ersten Tipp wählen, um die beste Ausgangsposition zu haben?

210 ?

Vereinfachter Fall

$$ExpRem = \frac{1}{|S|}\sum_{a \in S} |S_{w,p(a)}| \qquad WorstRem = \max_{p} |S_{w,p}|$$

Anfang:	000
×√√	100, 200
××√	110, 210, 220
×××	111, 211, 221, 222

2.9 — 4

Anfang:	100
×√√	000, 200
××√	210, 220
×××	211, 221, 222
√×√	110
√××	111

1.9 — 3

000
100
110
111
200
210
211
220
221
222

Vereinfachter Fall

$$ExpRem = \frac{1}{|S|}\sum_{a \in S} |S_{w,p(a)}| \qquad WorstRem = \max_{p} |S_{w,p}|$$

Anfang:	110
×√√	210
××√	000, 200, 220
×××	221, 222
√×√	100
√√×	111
×√×	211

1.7

3

Anfang:	111
×√√	211
××√	221
×××	000, 200, 220, 222
√√×	110
×√×	210
√××	100

2.1

4

000
100
110
111
200
210
211
220
221
222

Vereinfachter Fall

$$ExpRem = \frac{1}{|S|}\sum_{a \in S} |S_{w,p(a)}| \qquad WorstRem = \max_{p} |S_{w,p}|$$

Anfang:	200
×√√	000, 100
××√	110
×××	111
√×√	210, 220
√××	211, 221, 222

1.9

3

Anfang:	210
×√√	110
××√	000, 100
√×√	200, 220
√√×	211
×√×	111
√××	221, 222

1.5

2

000
100
110
111
200
210
211
220
221
222

Vereinfachter Fall

$$ExpRem = \frac{1}{|S|}\sum_{a \in S} |S_{w,p(a)}| \qquad WorstRem = \max_{p} |S_{w,p}|$$

Anfang:	211
×√√	111
×××	000, 100
√×√	221
√√×	210
×√×	110
√××	200, 220, 222

1.7

3

Anfang:	220
××√	000, 100, 110
×××	111
√×√	200, 210
√√×	221, 222
√××	211

1.9

3

000
100
110
111
200
210
211
220
221
222

Vereinfachter Fall

$$ExpRem = \frac{1}{|S|}\sum_{a \in S} |S_{w,p(a)}| \qquad WorstRem = \max_{p} |S_{w,p}|$$

Anfang:	221
××√	111
×××	000, 100, 110
√×√	211
√√×	220, 222
√××	200, 210

1.9

3

Anfang:	222
×××	000, 100, 110, 111
√√×	220, 221
√××	200, 210, 211

2.9

4

000
100
110
111
200
210
211
220
221
222

Vereinfachter Fall

$$ExpRem = \frac{1}{|S|}\sum_{a \in S} |S_{w,p(a)}| \qquad WorstRem = \max_{p} |S_{w,p}|$$

000	2.9	4
100	1.9	3
110	1.7	3
111	2.1	4
200	1.9	3
210	1.5	2
211	1.7	3
220	1.9	3
221	1.9	3
222	2.9	4

Restmenge

Beispiel:
PRINT, POINT, PRISM, POWER, PRAYS, PRIME

Grün ------2
Gelb ------1
Grau ------0

W = POWER

P(PRINT) = (20001)
P(POINT) = (22000)
P(PRISM) = (20001)
P(PRAYS) = (20001)
P(PRIME) = (20011)

$$|S_{w,p(PRINT)}| = 3$$
$$|S_{w,p(POINT)}| = 1$$
$$|S_{w,p(PRISM)}| = 3$$
$$|S_{w,p(PRAYS)}| = 3$$
$$|S_{w,p(PRIME)}| = 1$$

$$|S_{w,(20001)}| = 3$$
$$|S_{w,(22000)}| = 1$$
$$|S_{w,(20011)}| = 1$$

$$ExpRem = \frac{1}{|S|}\sum_{a \in S}|S_{w,p(a)}| = \frac{1}{6}(3+1+3+3+1)$$

$$WorstRem = \max_{p}|S_{w,p}| = 3$$

Restmenge

Beispiel:

PRINT, POINT, PRISM, POWER, PRAYS, PRIME

Grün ------2
Gelb ------1
Grau ------0

W = PRISM

P(PRINT) = (22200)
P(POINT) = (20200)
P(POWER) = (21000)
P(PRAYS) = (22000)
P(PRIME) = (22201)

$$|S_{w,(22200)}| = 1$$
$$|S_{w,(20200)}| = 1$$
$$|S_{w,(21000)}| = 1$$
$$|S_{w,(22000)}| = 1$$
$$|S_{w,(22201)}| = 1$$

$$|S_{w,p(PRINT)}| = 1$$
$$|S_{w,p(POINT)}| = 1$$
$$|S_{w,p(POWER)}| = 1$$
$$|S_{w,p(PRAYS)}| = 1$$
$$|S_{w,p(PRIME)}| = 1$$

$$ExpRem = \frac{1}{|S|}\sum_{a \in S}|S_{w,p(a)}| = \frac{1}{6}(1+1+1+1+1)$$

$$WorstRem = \max_{p}|S_{w,p}| = 1$$

Restmenge

In diesen Fall ist **PRISM** ein bessere Anfangswort. 😊

Als nächsten Schritt haben wir zur Überprüfung mit Python auch die folgenden zwölf Wörter getestet:

PRINT, PRIME, PRISM, POINT, PRAYS, POWER,
WATER, WASTE, SNACK, SLEEP, SLICE, SNAKE

Restmenge

Python --- Pseudocode

Algorithm 1 Berechnung von ExpRem für alle Startwörter

```
1:  procedure ComputeExpRem(WORD_LIST)
2:      for all w ∈ WORD_LIST do                          ▷ alle möglichen Tipps
3:          sum ← 0
4:          for all a ∈ WORD_LIST do                      ▷ jede Kandidatenlösung
5:              p ← Feedback(w, a)
6:              k ← RemainingSize(g, p, WORD_LIST)
7:              sum ← sum + k                             ▷ |S_{w,p(a)}|
8:          end for
9:          ExpRem[w] ← sum / |WORD_LIST|
10:     end for
11:     return ExpRem
12: end procedure
```

Algorithm 2 Hilfsfunktionen

```
1: function Feedback(word, answer)
2:     return Musterstring aus 2 (grün), 1 (gelb), 0 (grau)
3: end function
4: function RemainingSize(w, p, S)
5:     return Anzahl der Wörter c ∈ S mit Feedback(w, c) = p
6: end function
```

$$\boldsymbol{ExpRem} = \frac{\mathbf{1}}{|\boldsymbol{S}|} \sum_{\boldsymbol{a} \in \boldsymbol{S}} |\boldsymbol{S_{w,p(a)}}|$$

Restmenge

Python --- Pseudocode

Algorithm 3 Berechnung von WorstRem für alle Startwörter

```
1:  procedure ComputeWorstRem(WORD_LIST)
2:      for all w ∈ WORD_LIST do                          ▷ aktuelles Startwort
3:          maxSize ← 0
4:          seen ← leeres Dictionary                      ▷ gemerkte Muster
5:          for all a ∈ WORD_LIST do                      ▷ jede Kandidatenlösung
6:              p ← Feedback(w, a)                        ▷ Muster für w gegen a
7:              if p ∉ seen then                          ▷ je Pattern nur einmal zählen
8:                  k ← RemainingSize(w, p, WORD_LIST)
9:                  seen[p] ← k
10:                 maxSize ← max(maxSize, k)
11:             end if
12:         end for
13:         WorstRem[w] ← maxSize                         ▷ größtes Rest-Set für w
14:     end for
15:     return WorstRem                                   ▷ Map: Wort → WorstRem
16: end procedure
```

$$\boldsymbol{WorstRem = \max_{p} |S_{w,p}|}$$

Restmenge

Python --- Ergebnis

Beispiel:

PRINT, PRIME, PRISM, POINT, PRAYS, POWER,
WATER, WASTE, SNACK, SLEEP, SLICE, SNAKE

Word	ExpRem	AvgSteps
WATER	1.000000	1.916667
PRIME	1.166667	2.000000
SNAKE	1.166667	2.000000
PRINT	1.333333	2.083333
PRAYS	1.333333	2.083333
WASTE	1.333333	2.083333
SNACK	1.333333	2.083333
SLEEP	1.333333	2.083333
SLICE	1.333333	2.083333
PRISM	1.500000	2.083333
POINT	1.500000	2.166667
POWER	1.666667	2.166667

Korrelation: 0.94 😄

Word	WorstRem	AvgSteps
WATER	1	1.916667
PRINT	2	2.083333
PRIME	2	2.000000
POINT	2	2.166667
PRAYS	2	2.083333
WASTE	2	2.083333
SNACK	2	2.083333
SLEEP	2	2.083333
SLICE	2	2.083333
SNAKE	2	2.000000
PRISM	3	2.083333
POWER	3	2.166667

Korrelation: 0.67 🙂

Restmenge

Python --- Ergebnis

Freie Universität Berlin

Mit 50 Wörter:

PRINT, PRIME, PRISM, POINT, PRAYS, POWER, WATER, WASTE, SNACK, SLEEP, SLICE, SNAKE, CRANE, SLATE, TRACE, CRISP, GRACE, PLANT, GRANT, PRIDE, TRICE, IDEAL, APPLE, BERRY, CHASE, DELTA, EARTH, FAITH, GIANT, HAPPY, INDEX, JUDGE, KNIFE, LEMON, MAGIC, NOVEL, OCEAN, PEACH, QUIET, RIVER, SOLAR, TIGER, UNITY, VIVID, WHALE, XENON, YEAST, ZEBRA, CABIN, DANCE

ExpRem Korrelation: 0.89

WorstRem Korrelation: 0.88

😁

Restmenge

Python --- Ergebnis

Mit 1000 Wörter:

ExpRem Korrelation: 0.86 **AREAS** ist das Beste Anfangswort

WorstRem Korrelation: 0.82 **ARISE** ist das Beste Anfangswort

Mit 14855 Wörter:

ExpRem **LARES** ist das Beste Anfangswort

WorstRem **SERIA** ist das Beste Anfangswort

Weitere Fragen

Wie kann man das Rechnen vereinfachen, um die Laufzeit zu reduzieren?

Gibt es noch bessere Methoden mit theoretischer Grundlagen?

Freie Universität Berlin

In der Zukunft...

1. Algorithmus beschleunigen

   $|S_{w,p}|$
   um Wiederholung zu vermeiden

2. Weitere Kennzahlen evaluieren

   1. Vergleich mit Informationsgewinn / Entropie / erwarteten Zügen (Expected Steps).
   2. Robustheitstests: Wie stabil ist die Rangfolge bei veränderten Wortlisten?

3. Größere und diversere Wortlisten

   Erweiterung auf 6-Buchstaben-Varianten oder andere Sprachen.

Freie Universität Berlin

VIELEN DANK