Natural Language Processing

Fall 2022

Unit 1: Sequence Models

Lectures 5-6: Language Models and Smoothing

required
optional

Professor Liang Huang
liang.huang.sh@gmail.com
Noisy-Channel Model
Noisy-Channel Model

\[ p(t\ldots t) \]

BEST PATH IS
\[ \arg\max p(t\ldots t | w\ldots w) \]

each try sequence scored by
\[ P(w\ldots w | t\ldots t) \]

compose

each try sequence scored by
\[ P(t\ldots t) \cdot P(w\ldots w | t\ldots t) \]
Applications of Noisy-Channel

- Machine Translation
- Optical Character Recognition (OCR)
- Part Of Speech (POS) tagging
- Speech recognition

| Application                  | Input                      | Output                      | $p(i)$                           | $p(o|i)$                      |
|------------------------------|----------------------------|-----------------------------|----------------------------------|-------------------------------|
| Machine Translation          | $L_1$ word sequences       | $L_2$ word sequences        | $p(L_1)$ in a language model     | translation model             |
| Optical Character Recognition | actual text                | text with mistakes          | prob of language text            | model of OCR errors           |
| Part Of Speech (POS) tagging | POS tag sequences           | English words               | prob of POS sequences            | $p(w|t)$                      |
| Speech recognition           | word sequences              | speech signal               | prob of word sequences           | acoustic model                |

spell correction  correct text  text with mistakes  prob. of language text  noisy spelling
Noisy Channel Examples

Th qck brwn fx jmps vr th lzy dg.
Ths sntnc hs ll twnty sx lttrs n th lphbt.

I cnduo't bvleiee taht I culod aulaclty uesdtannrd waht I was rdnaieg. Unisg the icndeblire pweor of the hmuan mnid, aocdcrnig to rseecrah at Cmabrigde Uinervtisy, it dseno't mtttaer in waht oderr the iterets in a wrod are, the olny irpoamntnt tihng is taht the frsit and lsat ltteer be in the rhgit pclae.

Therestcanbeatotalmessandyoucanstillreaditwi thoutaproblem. Thisisbecausethehumanminddo esnotreadeveryleterbyitself, butthetwordsawohl.

研表究明，汉字的序顺并不定一能影 阅响读，比如当你看完句这话后，才 发现这现里的字全是都乱的。

研究表明，汉字的顺序并不一定能影 响阅读，比如当你看完这句话后，才 发现这里的字全是乱的。
Language Model for Generation

- search suggestions
Language Models

- Problem: what is $P(w) = P(w_1 \ w_2 \ldots \ w_n)$?
- Ranking: $P(\text{an apple}) > P(\text{a apple}) = 0$, $P(\text{he often swim}) = 0$
- Prediction: what’s the next word? use $P(w_n | w_1 \ldots w_{n-1})$
- Obama gave a speech about ______. sequence prob, not just joint prob.

$P(w_1 \ w_2 \ldots \ w_n) = P(w_1) \ P(w_2 | w_1) \ldots P(w_n | w_1 \ldots w_{n-1})$

- $\approx P(w_1) \ P(w_2 | w_1) \ P(w_3 | w_1 \ w_2) \ldots P(w_n | w_{n-2} \ w_{n-1})$ trigram
- $\approx P(w_1) \ P(w_2 | w_1) \ P(w_3 | w_2) \ldots P(w_n | w_{n-1})$ bigram
- $\approx P(w_1) \ P(w_2) \ P(w_3) \ldots P(w_n)$ unigram
- $\approx P(w) \ P(w) \ P(w) \ldots P(w)$ 0-gram
Estimating $n$-gram Models

- maximum likelihood: $p_{ML}(x) = \frac{c(x)}{N}$; $p_{ML}(xy) = \frac{c(xy)}{c(x)}$
- problem: unknown words/sequences (unobserved events)
- sparse data problem
- solution: smoothing

```
"In person she was inferior superior to both sisters"

0-gram
unigram
bigram
trigram
4-gram

<table>
<thead>
<tr>
<th></th>
<th>10^{-6}</th>
<th>10^{-6}</th>
<th>10^{-6}</th>
<th>10^{-6}</th>
<th>10^{-6}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>.011</td>
<td>.015</td>
<td>.00005</td>
<td>.005</td>
<td>.0003</td>
</tr>
<tr>
<td>.009</td>
<td>.122</td>
<td>0</td>
<td>.212</td>
<td>.0004</td>
<td>.006</td>
</tr>
<tr>
<td>.5</td>
<td>0</td>
<td></td>
<td>?</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>0</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

(textbook, table 6.3)
```
Smoothing

- have to give some probability mass to unseen events
  - (by discounting from seen events)
- Q1: how to divide this wedge up?
- Q2: how to squeeze it into the pie?

new wedge (one tiny slice for each character sequence of length < 20 that was never observed in training data)
Smoothing: Add One (Laplace)

- MAP: add a “pseudocount” of 1 to every word in Vocab
- \( P_{lap}(x) = \frac{c(x) + 1}{N + V} \)  
  \( V \) is Vocabulary size
- \( P_{lap}(unk) = \frac{1}{N+V} \)  
  same probability for all unks
- how much prob. mass for unks in the above diagram?
- e.g., N=10^6 tokens, V=26^{20}, V_{obs} = 10^5, V_{unk} = 26^{20} - 10^5
Smoothing: Add Less than One

- Add one gives too much weight on unseen words!
- Solution: Add less than one (Lidstone) to each word in V

\[ P_{\text{lid}}(x) = \frac{c(x) + \lambda}{N + \lambda V} \]

\[ P_{\text{lid}}(\text{unk}) = \frac{\lambda}{N + \lambda V} \]

\(0 < \lambda < 1\) is a parameter

- Still same for unks, but smaller

- Q: How to tune this \(\lambda\)? On held-out data!
Smoothing: Witten-Bell

- key idea: use one-count things to guess for zero-counts
  - recurring idea for unknown events, also for Good-Turing
- prob. mass for unseen: $T / (N + T)$ \( T: \# \) of seen types
- 2 kinds of events: one for each token, one for each type
  - = MLE of seeing a new type (\( T \) among \( N+T \) are new)
  - divide this mass evenly among \( V-T \) unknown words
- \( p_{wb}(x) = \frac{T}{(V-T)(N+T)} \) unknown word
  \[ = \frac{c(x)}{(N+T)} \] known word
- bigram case more involved; see J&M Chapter for details
Smoothing: Good-Turing

- again, one-count words in training ~ unseen in test
- let \( N_c = \# \) of words with frequency \( r \) in training
- \( P_{GT}(x) = \frac{c'(x)}{N} \) where \( c'(x) = \frac{(c(x)+1) N_{c(x)+1}}{N_{c(x)}} \)
- total adjusted mass is \( \sum_c c' N_c = \sum_c (c+1) \frac{N_{c+1}}{N} \)
  - remaining mass: \( \frac{N_1}{N} \): split evenly among unks

**Example:**

<table>
<thead>
<tr>
<th>( r )</th>
<th>( N_r )</th>
<th>( N_{r+1} )</th>
<th>( r^* )</th>
<th>( \frac{r^*/N}{1-2} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1000</td>
<td>100</td>
<td>-</td>
<td>( \frac{-}{1-2} )</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>40</td>
<td>0.8</td>
<td>( \frac{0.8}{1-2} )</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td>20</td>
<td>1.5</td>
<td>( \frac{1.5}{1-2} )</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>10</td>
<td>2.0</td>
<td>( \frac{2.0}{1-2} )</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>6</td>
<td>3.0</td>
<td>( \frac{3.0}{1-2} )</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>3</td>
<td>3.0</td>
<td>( \frac{3.0}{1-2} )</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Diagram:**

```
N1 -> N31 -> N3101
N0 -> N2 -> N22
N3 -> N33 -> N333
N4 -> ... -> N44
```

**ENIGMA**
Smoothing: Good-Turing

- from Church and Gale (1991).
- bigram LMs. unigram vocab size = $4 \times 10^5$.
- $T_r$ is the frequencies in the held-out data (see $f_{\text{empirical}}$).

<table>
<thead>
<tr>
<th>$r = f_{\text{MLE}}$</th>
<th>$f_{\text{empirical}}$</th>
<th>$f_{\text{Lap}}$</th>
<th>$f_{\text{del}}$</th>
<th>$f_{\text{GT}}$</th>
<th>$N_r$</th>
<th>$T_r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.000027</td>
<td>0.000137</td>
<td>0.000037</td>
<td>0.000027</td>
<td>74</td>
<td>671</td>
</tr>
<tr>
<td>1</td>
<td>0.448</td>
<td>0.000274</td>
<td>0.396</td>
<td>0.446</td>
<td>2018</td>
<td>046</td>
</tr>
<tr>
<td>2</td>
<td>1.25</td>
<td>0.000411</td>
<td>1.24</td>
<td>1.26</td>
<td>449</td>
<td>721</td>
</tr>
<tr>
<td>3</td>
<td>2.24</td>
<td>0.000548</td>
<td>2.23</td>
<td>2.24</td>
<td>188</td>
<td>933</td>
</tr>
<tr>
<td>4</td>
<td>3.23</td>
<td>0.000685</td>
<td>3.22</td>
<td>3.24</td>
<td>105</td>
<td>668</td>
</tr>
<tr>
<td>5</td>
<td>4.21</td>
<td>0.000822</td>
<td>4.22</td>
<td>4.22</td>
<td>68</td>
<td>379</td>
</tr>
<tr>
<td>6</td>
<td>5.23</td>
<td>0.000959</td>
<td>5.20</td>
<td>5.19</td>
<td>48</td>
<td>190</td>
</tr>
<tr>
<td>7</td>
<td>6.21</td>
<td>0.00109</td>
<td>6.21</td>
<td>6.21</td>
<td>35</td>
<td>709</td>
</tr>
<tr>
<td>8</td>
<td>7.21</td>
<td>0.00123</td>
<td>7.18</td>
<td>7.24</td>
<td>27</td>
<td>710</td>
</tr>
<tr>
<td>9</td>
<td>8.26</td>
<td>0.00137</td>
<td>8.18</td>
<td>8.25</td>
<td>22</td>
<td>280</td>
</tr>
</tbody>
</table>
Good-Turing is much better than add (less than) one

- problem 1: \( N_{c_{\text{max}}+1} = 0 \), so \( c'_{\text{max}} = 0 \)
  - solution: only adjust counts for those less than \( k \) (e.g., 5)

- problem 2: what if \( N_c = 0 \) for some middle \( c \)?
  - solution: smooth \( N_c \) itself

smooth \( N_r \) itself, e.g.:

\[ N_r \]

the curve \( (N_r = ar^b + e) \) gives better \( N_r \)

(\text{or something simpler, like averaging the neighborhood})

RENORMALIZE!!!
Smoothing: Backoff & Interpolation

\[ \hat{p}(w_i|w_{i-2}w_{i-1}) = \begin{cases} 
\tilde{p}(w_i|w_{i-2}w_{i-1}), & \text{if } C(w_{i-2}w_{i-1}w_i) > 0 \\
\alpha_1 p(w_i|w_{i-1}), & \text{if } C(w_{i-2}w_{i-1}w_i) = 0 \\
\alpha_2 p(w_i), & \text{and } C(w_{i-1}w_i) > 0 \\
\end{cases} \]

\[ \hat{p}(w_i|w_{i-2}w_{i-1}) = \lambda_1 p(w_i|w_{i-2}w_{i-1}) + \lambda_2 p(w_i|w_{i-1}) + \lambda_3 p(w_i) \]

subject to the constraint that \( \sum_j \lambda_j = 1 \)
Entropy and Perplexity

• classical entropy (uncertainty): \( H(X) = -\sum p(x) \log p(x) \)
  - how many “bits” (on average) for encoding

• sequence entropy (distribution over sequences):
  - \( H(L) = \lim \frac{1}{n} H(w_1...w_n) \) (for language L)  
    \( Q: \text{why } \frac{1}{n}? \)
  - \( = \lim \frac{1}{n} \sum_{w \in L} p(w_1...w_n) \log p(w_1...w_n) \)

• Shannon-McMillan-Breiman theorem:
  - \( H(L) = \lim -\frac{1}{n} \log p(w_1...w_n) \) no need to enumerate \( w \) in \( L \)!
  - if \( w \) is long enough, just take \(-\frac{1}{n} \log p(w)\) is enough!

• perplexity is \( 2^{H(L)} \)
Entropy/Perplexity of English

- on 1.5 million WSJ test set:
  - unigram: 962 9.9 bits
  - bigram: 170 7.4 bits
  - trigram: 109 6.8 bits

- higher-order n-grams generally has lower perplexity
  - but hitting diminishing returns after n=5
  - even higher order: data sparsity will be a problem!
  - recurrent neural network (RNN) LM will be better

- what about human??
Shannon Papers


<table>
<thead>
<tr>
<th></th>
<th>F₀</th>
<th>F₁</th>
<th>F₂</th>
<th>F₃</th>
<th>F₉word</th>
</tr>
</thead>
<tbody>
<tr>
<td>26 letter</td>
<td>4.70</td>
<td>4.14</td>
<td>3.56</td>
<td>3.3</td>
<td>2.62</td>
</tr>
<tr>
<td>27 letter</td>
<td>4.76</td>
<td>4.03</td>
<td>3.32</td>
<td>3.1</td>
<td>2.14</td>
</tr>
</tbody>
</table>

Zero-order approximation
- XFOML RXKHRJFFJUI ALPWXFWJXYJ FFJVEYJICQSGHYD QPAAMKBZACIBZLKJQD

First-order approximation
- OCRW HLO RGWR NMIELWIS EU LL NBNESEBYA TH EEE ALHENHTTPA OOBTTVA NAH BRL

Second-order approximation
- ON IE ANTSOUTINYS ARE T INCORE ST BE S DEAMY ACHIN D ILONASIVE TUCOOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE

Third-order approximation
- IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF DEMONSTURES OF THE REPTAGIN IS REGOACTION OF CRE

Shannon Game

- guess the next letter; compute entropy (bits per char)
- 0-gram: 4.76, 1-gram: 4.03, 2-gram: 3.32, 3-gram: 3.1
- native speaker: ~1.1 (0.6~1.3); me: upperbound ~2.3

Q: formula for entropy? (only computes upperbound)
http://math.ucsd.edu/~crypto/java/ENTROPY/
The subject’s identical twin would be able to reconstruct the original text from the guess sequence, so in that sense, it contains the same amount of information.

Let \( c_1, c_2, \ldots, c_n \) represent the character sequence, let \( g_1, g_2, \ldots, g_n \) represent the guess sequence, and let \( j \) range over guess numbers from 1 to 95, the number of printable English characters plus newline. Shannon [3] provides two results.

(Upper Bound). The entropy of \( c_1, c_2, \ldots, c_n \) is no greater than the unigram entropy of the guess sequence:

\[
-\frac{1}{n} \log(\Pi_{i=1}^{n} P(g_i)) = -\frac{1}{n} \sum_{i=1}^{n} \log(P(g_i)) = -\sum_{j=1}^{95} P(j) \log(P(j))
\]

This is because this unigram entropy is an upper bound on the entropy of \( g_1, g_2, \ldots, g_n \), which equals the entropy of \( c_1, c_2, \ldots, c_n \). In human experiments, Shannon obtains an upper bound of 1.3 bits per character (bpc) for English, significantly better than the character n-gram models of his time (e.g., 3.3 bpc for trigram).

(Lower Bound). The entropy of \( c_1, c_2, \ldots, c_n \) is no less than:

\[
\sum_{j=1}^{95} j \cdot [P(j) - P(j+1)] \cdot \log(j)
\]

with the proof given in his paper. Shannon reported a lower bound of 0.6 bpc.

\[
\sum_{i=1}^{27} i(q_i^N - q_{i+1}^N) \log i \leq F_N \leq -\sum_{i=1}^{27} q_i^N \log q_i^N. \quad (17)
\]

http://www.mdpi.com/1099-4300/19/1/15

Fig. 4—Upper and lower experimental bounds for the entropy of 27-letter English.
BUT I CAN BEAT YOU ALL!

- guess the next letter; compute entropy (bits per char)
- 0-gram: 4.76, 1-gram: 4.03, 2-gram: 3.32, 3-gram: 3.1
- native speaker: ~1.1 (0.6~1.3); me: upperbound ~2.3

This Applet only computes Shannon’s upperbound!
I’m going to hack it to compute lowerbound as well.

\[
\sum_{i=1}^{27} i(q_i^N - q_{i+1}^N) \log i \leq F_N \leq \sum_{i=1}^{27} q_i^N \log q_i^N. \quad (17)
\]
Playing Shannon Game: $n$-gram LM

- **0-gram**: each char is equally likely ($1/27$)
- **1-gram**: (a) sample from 1-gram distribution from Shakespeare or PTB
- **1-gram**: (b) always follow same order: _ETAIOSRLHDCUMPFGBYWVKXJQZ
- **2-gram**: always follow same order: Q=>U_A  J=>UOEAI

Shannon’s estimation is less accurate for lower entropy!

\[ \sum_{i=1}^{q^N} (q^N_i - q^{N+1}_i) \log i \leq F_N \leq \sum_{i=1}^{q^N} q^N_i \log q^N_i. \]  

<table>
<thead>
<tr>
<th></th>
<th>F0</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>Fword</th>
</tr>
</thead>
<tbody>
<tr>
<td>26 letter</td>
<td>4.70</td>
<td>4.14</td>
<td>3.56</td>
<td>3.3</td>
<td>2.62</td>
</tr>
<tr>
<td>27 letter</td>
<td>4.76</td>
<td>4.03</td>
<td>3.32</td>
<td>3.1</td>
<td>2.14</td>
</tr>
</tbody>
</table>

---

The entropy for this experiment is in [4.5414, 4.5794053]

---

The entropy for this experiment is in [4.0705876, 4.3981924]

---

The entropy for this experiment is in [3.3405864, 3.9108648]

---

The entropy for this experiment is in [2.4070668, 3.23315]
On Google Books (Peter Norvig)

My distillation of the Google books data gives us 97,565 distinct words, which were mentioned 743,842,922,321 times (37 million times more than in Mayzner's 20,000-mention collection). Each distinct word is called a "type" and each mention is called a "token." To no surprise, the most common word is "the". Here are the top 50 words, with their counts (in billions of mentions) and their overall percentage (looking like a Zipf distribution):

<table>
<thead>
<tr>
<th>Word</th>
<th>Count</th>
<th>Percent</th>
<th>Bar Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>53.10</td>
<td>7.14%</td>
<td></td>
</tr>
<tr>
<td>of</td>
<td>30.97</td>
<td>4.16%</td>
<td></td>
</tr>
<tr>
<td>and</td>
<td>22.63</td>
<td>3.04%</td>
<td></td>
</tr>
<tr>
<td>to</td>
<td>19.35</td>
<td>2.60%</td>
<td></td>
</tr>
<tr>
<td>in</td>
<td>16.89</td>
<td>2.27%</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>15.31</td>
<td>2.06%</td>
<td></td>
</tr>
<tr>
<td>is</td>
<td>8.38</td>
<td>1.13%</td>
<td></td>
</tr>
<tr>
<td>that</td>
<td>8.00</td>
<td>1.08%</td>
<td></td>
</tr>
<tr>
<td>for</td>
<td>6.55</td>
<td>0.88%</td>
<td></td>
</tr>
<tr>
<td>it</td>
<td>5.74</td>
<td>0.77%</td>
<td></td>
</tr>
<tr>
<td>as</td>
<td>5.70</td>
<td>0.77%</td>
<td></td>
</tr>
<tr>
<td>was</td>
<td>5.50</td>
<td>0.74%</td>
<td></td>
</tr>
<tr>
<td>with</td>
<td>5.18</td>
<td>0.70%</td>
<td></td>
</tr>
<tr>
<td>be</td>
<td>4.82</td>
<td>0.65%</td>
<td></td>
</tr>
<tr>
<td>by</td>
<td>4.70</td>
<td>0.63%</td>
<td></td>
</tr>
<tr>
<td>on</td>
<td>4.59</td>
<td>0.62%</td>
<td></td>
</tr>
<tr>
<td>not</td>
<td>4.52</td>
<td>0.61%</td>
<td></td>
</tr>
<tr>
<td>he</td>
<td>4.11</td>
<td>0.55%</td>
<td></td>
</tr>
<tr>
<td>this</td>
<td>3.88</td>
<td>0.52%</td>
<td></td>
</tr>
<tr>
<td>this</td>
<td>3.83</td>
<td>0.51%</td>
<td></td>
</tr>
<tr>
<td>are</td>
<td>3.70</td>
<td>0.50%</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td>3.67</td>
<td>0.49%</td>
<td></td>
</tr>
<tr>
<td>his</td>
<td>3.61</td>
<td>0.49%</td>
<td></td>
</tr>
<tr>
<td>from</td>
<td>3.47</td>
<td>0.47%</td>
<td></td>
</tr>
<tr>
<td>at</td>
<td>3.41</td>
<td>0.46%</td>
<td></td>
</tr>
<tr>
<td>which</td>
<td>3.14</td>
<td>0.42%</td>
<td></td>
</tr>
<tr>
<td>but</td>
<td>2.79</td>
<td>0.38%</td>
<td></td>
</tr>
<tr>
<td>have</td>
<td>2.78</td>
<td>0.37%</td>
<td></td>
</tr>
<tr>
<td>an</td>
<td>2.73</td>
<td>0.37%</td>
<td></td>
</tr>
<tr>
<td>had</td>
<td>2.62</td>
<td>0.35%</td>
<td></td>
</tr>
<tr>
<td>they</td>
<td>2.46</td>
<td>0.33%</td>
<td></td>
</tr>
<tr>
<td>you</td>
<td>2.34</td>
<td>0.31%</td>
<td></td>
</tr>
<tr>
<td>were</td>
<td>2.27</td>
<td>0.31%</td>
<td></td>
</tr>
<tr>
<td>their</td>
<td>2.15</td>
<td>0.29%</td>
<td></td>
</tr>
<tr>
<td>one</td>
<td>2.15</td>
<td>0.29%</td>
<td></td>
</tr>
<tr>
<td>all</td>
<td>2.06</td>
<td>0.28%</td>
<td></td>
</tr>
<tr>
<td>we</td>
<td>2.06</td>
<td>0.28%</td>
<td></td>
</tr>
</tbody>
</table>

http://norvig.com/mayzner.html
"From an information theoretic point of view, accurately translated copies of the original text would be expected to contain almost no extra information if the original text is available, so in principle it should be possible to store and transmit these texts with very little extra cost." (Nevill and Bell, 1992)

Monolingual Shannon Game (no source sentence)

It is defended through reasoning.

Bilingual Shannon Game (source sentence = "Se defiende con argumentos.")

It is defended through reasoning.

If I am fluent in Spanish, then English translation adds no new info.

If I understand 50% Spanish, then English translation adds some info.

If I don’t know Spanish at all, then English should be have the same entropy as in the monolingual case.
Other Resources

- “Unreasonable Effectiveness of RNN” by Karpathy
- Yoav Goldberg’s follow-up for n-gram models (ipynb)

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

http://nbviewer.jupyter.org/gist/yoavg/d76121dfde2618422139


Running this algorithm on the entire Open American National Corpus (about 95 million characters) yields the following results:

As a rough example, call this sequence of values $F_k$ and assume that it verifies the recurrence equation $F_{k+1} - F_k = \alpha (F_k - F_{k-1})$. Then the $\alpha$ that yields the best approximation (taking the two initial values for granted since they are less likely to suffer from sampling error) is $\alpha \approx 0.68$ ($L^2$ error: $6.7 \cdot 10^{-5}$), and the corresponding entropy rate is $h \approx 1.14$ bits/character.
Shannon Game -- non-native speakers

Figure 2: Shannon game result: Stefano

Figure 3: Shannon game result: Purbasha
Shannon Game -- non-native speakers

Figure 3: Third Attempt

Figure 4: Played by Xiuwao Wang
Shannon Game -- non-native speakers

IT POURED EVERY TUESDAY OVER THE IN
E RECREATION CENTER WHEN THE IN
DIAN RAIN DANCE INSTRUCTOR TAUGHT HIS CLASS.

UNDERNEATH THE BLUE CUSHION IN
THE LIVING ROOM IS A HANDFUL OF
CHANGES AND THE REMOTE CONTROL.

EVEN THOUGH YOU DON'T KNOW HOW TO FLY, YOU MIGHT BE ABLE TO LIFT
YOUR SHOE LONG ENOUGH FOR THE CAT TO MOVE OUT FROM UNDERNEATH YOUR FOOT.

MY PSYCHIC WOULD BE VERY INTERESTED TO LEARN HOW YOU MANAGED TO SWALLOW THAT EGG WHO LEFT WITHOUT BREAKING IT.

The entropy for this experiment is 1.953524

The entropy for this experiment is 2.005272

The entropy for this experiment is 2.3998766

The entropy for this experiment is 2.8224587
Shannon Game -- native speakers

Figure 5: Played by Alrik Firl