Language models

Professor Marie Roch
San Diego State University
readings J&M: 4

Narrowing search with a language model

• Don’t move or I’ll …

• Get ‘er …

• What will she think of …

• This enables …
Applications

- Speech recognition
- Handwriting recognition
- Spelling correction
- Augmentative communication
  and more…

Constituencies

- Groupings of words
  - I didn’t see you behind the bush.
  - She ate quickly as she was late for the meeting.
- Movement within the sentence:
  - ✓ As she was late for the meeting, she ate quickly
  - ✗ As she was late for, she ate quickly the meeting.
- Constituencies aid in prediction.
Strategies for construction

• Formal grammar
  – Requires intimate knowledge of the language
  – Usually context free and cannot be represented by a regular language
  – We will not be covering this in detail

N-gram models

• Suppose we wish to compute the probability the sentence:
She sells seashells down by the seashore.
• We can think of this as a sequence of words:

\[
P(w_1^7) = P(w_1, w_2, w_3, w_4, w_5, w_6, w_7)
\]

Think further: How would you determine if a die is fair?
Estimating word probability

• Suppose we wish to compute the probability \( w_2 \) (\textit{sells} in the previous example).
  We could estimate using a \textbf{relative frequency}

  \[
P(w_2) = \frac{\text{# times } w_2 \text{ occurs}}{\text{# of times all words occur}}
  \]
  
  but this ignores what we could have learned with the first word.

Conditional probability

By defn. of conditional probability

\[
P(A \mid B) = \frac{P(A \cap B)}{P(B)}
\]

or in our problem:

\[
P(w_2 \mid w_1) = \frac{P(w_2 \cap w_1)}{P(w_1)} = \frac{P(w_1 \cap w_2)}{P(w_1)} = \frac{P(w_1 \cap w_2)}{P(w_1)} \text{ defn } \cap \text{ for words}
\]
Conditional probability

Next, consider $P(w_1, w_2)$

Since as $P(w_2 \mid w_1) = \frac{P(w_1, w_2)}{P(w_1)}$,

clearly $P(w_1, w_2) = P(w_2 \mid w_1)P(w_1)$

Chain rule

• Now let us consider:

$P(w_1, w_2, w_3) = P(w_3 \mid w_1, w_2) \frac{P(w_1, w_2)}{P(w_1)}$

$= P(w_3 \mid w_1, w_2)P(w_2 \mid w_1)P(w_1)$

• By applying conditional probability repeatedly we end up with the chain rule:

$P(W) = P(w_1w_2 \ldots w_n)$

$= P(w_1) P(w_2 \mid w_1) P(w_3 \mid w_1w_2) \ldots P(w_n \mid w_1w_2 \ldots w_{n-1})$

$= \prod_{i=1}^{n} P(w_i \mid w_1w_2 \ldots w_{i-1})$
Sparse problem space

• Suppose \( V \) distinct words.
• \( w_i \) has \( V^j \) possible sequences of words.
• Tokens \( N \) – The number of N-grams (including repetitions) occurring in a corpus
• Problem: In general, unique\((N)\ll valid \ tokens \ for \ the \ language.
  “The gently rolling hills were covered with bluebonnets” had no hits on Google at the time this slide was published.

Markov assumption

• A prediction is dependent on the current state but independent of previous conditions
• In our context:
  \[
P(w_n | w_{n-1}^{n-1}) = P(w_n | w_{n-1}) \text{ by the Markov assumption}
  \]
  which at times relax to N-1 words:
  \[
P(w_n | w_{n-1}^{n-1}) = P(w_n | w_{n-N+1}^{n-1})
  \]
Training N-gram models

• Simplest: relative frequency counts

\[
P(w_n | w_{n-1}^{n-1}) = \frac{C(w_{n-1}^{n-1} w_n)}{C(w_{n-1}^{n-1})}
\]

this is a form of maximum likelihood estimation (MLE)

• Use a portion of the word corpus that is not used for testing.

Special N-grams

• Unigram
  – Only depends upon the word itself.
  – \( P(w_i) \)

• Bigram
  – \( P(w_i | w_{i-1}) \)

• Trigram
  – \( P(w_i | w_{i-1}, w_{i-2}) \)

• Quadrigram
  – \( P(w_i | w_{i-1}, w_{i-2}, w_{i-3}) \)
Preparing a corpus

- Make case independent
- Remove punctuation and add start & end of sentence markers <s> </s>
- Other possibilities
  - part of speech tagging
  - lemmas: mapping of words with similar roots
e.g. sing, sang, sung → sing
  - stemming: mapping of derived words to their root
e.g. parted → part, ostriches → ostrich

An Example

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

Dr. Seuss, *Green Eggs and Ham*, 1960.

\[
P(I \mid <s>) = \frac{2}{3} = 0.67 \quad P(Sam \mid <s>) = \frac{1}{3} = 0.33 \quad P(am \mid I) = \frac{2}{3} = 0.67
\]

\[
P(\langle s \rangle \mid Sam) = \frac{1}{2} = 0.5 \quad P(Sam \mid am) = \frac{1}{2} = 0.5 \quad P(do \mid I) = \frac{1}{3} = 0.33
\]

\[
P(w_a \mid w_{a-N+1}) = \frac{C(w_{a-N+1}w_a)}{C(w_{a-N+1})}
\]
Berkeley Restaurant Project Sentences

- can you tell me about any good cantonese restaurants close by
- mid priced thai food is what i’m looking for
- tell me about chez panisse
- can you give me a listing of the kinds of food that are available
- i’m looking for a good place to eat breakfast
- when is caffe venezia open during the day

Bigram Counts from 9,222 sentences

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>5</td>
<td>827</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>want</td>
<td>2</td>
<td>0</td>
<td>608</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>to</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>686</td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>211</td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>16</td>
<td>2</td>
<td>42</td>
<td>0</td>
</tr>
<tr>
<td>chinese</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>82</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>food</td>
<td>15</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lunch</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>spend</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Bigram Probabilities

Unigram counts

<table>
<thead>
<tr>
<th></th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>2533</td>
<td>927</td>
<td>2417</td>
<td>746</td>
<td>158</td>
<td>1093</td>
<td>341</td>
<td>278</td>
</tr>
</tbody>
</table>

\[ P(i \text{ want}) = \frac{C(i \text{ want})}{C(i)} = \frac{827}{2533} = 0.33 \]

<table>
<thead>
<tr>
<th>( W_j )</th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>i</td>
<td>0.002</td>
<td>0.33</td>
<td>0</td>
<td>0.0036</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>want</td>
<td>0.0022</td>
<td>0</td>
<td>0.66</td>
<td>0.0011</td>
<td>0.0065</td>
<td>0.0065</td>
<td>0.0054</td>
<td>0.0011</td>
</tr>
<tr>
<td>to</td>
<td>0.00083</td>
<td>0</td>
<td>0.0017</td>
<td>0.28</td>
<td>0.00083</td>
<td>0</td>
<td>0</td>
<td>0.0011</td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0021</td>
<td>0.0027</td>
<td>0.0027</td>
<td>0.0065</td>
</tr>
<tr>
<td>chinese</td>
<td>0.0063</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.52</td>
<td>0.0063</td>
<td>0</td>
</tr>
<tr>
<td>food</td>
<td>0.014</td>
<td>0</td>
<td>0.014</td>
<td>0</td>
<td>0</td>
<td>0.0092</td>
<td>0.0037</td>
<td>0</td>
</tr>
<tr>
<td>lunch</td>
<td>0.0059</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0029</td>
<td>0</td>
</tr>
<tr>
<td>spend</td>
<td>0.0036</td>
<td>0</td>
<td>0</td>
<td>0.0036</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Bigram Estimates of Sentence Probabilities

\[ P(<s> \text{ I want english food } <s>) = P(I|<s>)P(\text{want}|I)P(\text{english}|\text{want})P(\text{food}|\text{english})P(<s>|\text{food}) =.000 \ 031 \]
Shakespeare:
N=884,647 tokens, V=29,066

<table>
<thead>
<tr>
<th>N-gram</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unigram</td>
<td>To him swallowed confess hear both. Which. Of save on trail for are ay device and rod life have.</td>
</tr>
<tr>
<td>Bigram</td>
<td>Hill he late speaks, or! a more to leg less first you enter. Are where execut and sighs have rise excellency took of. Sleep knoae we. near; vile like.</td>
</tr>
<tr>
<td>Trigram</td>
<td>What means, sir. I confess she? then all sorts, he is trim, captain. Why dost stand forth thy canopy, forsooth, he is this palpable hit the King Henry. Live king. Follow. What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman? Enter Menteins, if it so many good direction found? st then art a strong upon command of fear not a liberal larges given away, Falstaff! Exeunt.</td>
</tr>
<tr>
<td>Quadigram</td>
<td>Sweet peace, Falstaff shall die. Harry of Mounmouth's grave. This shall forbid it should be branded, if renown made it empty. Indeed the duke, and had a very good friend. Fly, and will rid me these news of price. Therefore the sadness of putting, as they say, 'tis done.</td>
</tr>
</tbody>
</table>

The need for n-gram smoothing

- Data for estimation is sparse.
- On a sample text with several million words
  - 50% of trigrams only occurred once
  - 80% of trigrams occurred less than 5 times
- Example: When pigs fly

\[
P(fly \mid when, pigs) = \frac{C(when, pigs, fly)}{C(when, pigs)}
\]

\[
= \frac{0}{C(when, pigs)} \quad \text{if "when pigs fly" unseen}
\]
The need for n-gram smoothing

• If the $P(fly \mid when, pigs) = 0$, then we will never recognize the phrase, even if somebody says it.

$$P(fly \mid when, pigs) = \frac{C(when, pigs, fly)}{C(when, pigs)} = \frac{0}{C(when, pigs)} \text{ "when pigs fly" unseen}$$

• Unfortunately, “when pigs fly” is a valid phrase that should be recognized.

Additive (Laplace) smoothing

• An additive smoother can simply act as if there is one more of every N-gram:

$$P(fly \mid when, pigs) = \frac{C(when, pigs, fly) + 1}{\sum_{w \in \text{vocab}} C(when, pigs, w) + 1} = \frac{1}{V + \sum_{w \in \text{vocab}} C(when, pigs, w)} \text{ "when pigs fly" unseen, } \quad V = \text{ size of the vocabulary}$$
Additive smoothing

- Problem
  - What happens as our vocabulary size increases?
    \[ C(\text{the, white, house}) = 3 \]
    \[ \Pr(\text{house | the, white}) = \frac{3 + 1}{V + \sum_{w \in \text{vocab}} C(\text{the, white, w})} \]
    - If the only other trigram starting with the white was \( C(\text{the,white,horse)}=2 \), the original estimate would have been .60, but imagine a 10K word vocabulary...

Better discounting methods

- Use \( P(\text{word occurring once}) \) as a proxy for things that we have not seen
- Problem: N-grams sum > 1
- Solution: Discounting techniques
Interpolation

• Combine multiple types of N-grams:

\[ P_i(w_i | w_{i-2}, w_{i-1}) = \lambda_1 P(w_i | w_{i-2}, w_{i-1}) + \lambda_2 P(w_i | w_{i-1}) + \lambda_3 P(w_i) \]

\[ \text{with } \sum \lambda_i = 1 \]

• Weights may be trained with the EM algorithm.

Counting the number of times things occur

• c - # times a word occurs (e.g. c for xylophone is typically small)

• \( N_c \) - # of different words that occur c times

Example

\[ c \]
\[ \begin{array}{ll}
1 & x \in \{ \text{crossed, kissed, milky, star, under, way} \} \\
2 & x \in \{ \text{lovers, the} \} \\
\end{array} \]

\[ N_1 = 6 \]

\[ N_2 = 2 \]
Counts from the Switchboard Corpus

- $N_C$ counts typically exhibit exponential decay

![Graph showing exponential decay]

Turing counts

- Intuition: Words seen very few times probably have their probability underestimated.
- Goal: Assign some probability to unseen events

$$P_{GT}(unseen) = \frac{N_1}{N_{tokens}}$$
Turing counts

• Reestimate the other counts

\[ c^* = (c + 1) \frac{E[N_{c+1}]}{E[N_c]} \]

• Turing suggested approximating the expectations by the observed counts

\[ c^* = (c + 1) \frac{N_{c+1}}{N_c}, \text{ e.g. } 4^* = (4 + 1) \frac{N_4}{N_4} \]

Estimating the missing mass

Missing mass

\[
\sum_{\text{w} \mid \text{count}(w) = 0} P(w) = \sum_{\text{w} \mid \text{count}(w) = 0} \frac{c^*_w}{N_{\text{token}}} = \sum_{\text{w} \mid \text{count}(w) = 0} \frac{(0 + 1) N_1}{N_0} = \sum_{w \mid \text{count}(w) = 0} \frac{N_1}{N_0 N_{\text{token}}} = N_0 \frac{N_1}{N_0 N_{\text{token}}} = \frac{N_1}{N_{\text{token}}} \text{ as count}(w) = 0 \text{ occurs } N_0 \text{ times}
\]
Good-Turing counts

- Unfortunately, the counts can be noisy and can contain gaps

- Good suggested smoothing them

Linear Good-Turing estimates

- Church & Gale/Gale proposed:
  - Smooth counts to distribute weight over gaps
  - Perform a linear fit in log-log space and use the fit in place of counts
  - Details in assigned reading:
Good-Turing estimates

- In practice only need approximations for poorly observed observations with low frequency (small $c$)
- Common to use unadjusted counts for $c \geq 5$.
- Good-Turing is rarely used by itself, but typically used as part of something else.

Backoff

Only rely on lower-order N-grams when needed.
- Katz backoff
- Kneser-Ney
- Relies on discounted probability $P^*$
  - Reduce probability estimates
  - Give reduction to others
Katz backoff

\[
P_{katz}(z \mid x, y) = \begin{cases} 
\alpha(x, y)P_{katz}(z \mid y) & \text{if } C(x, y, z) > 0 \\
\alpha(x, y)P_{katz}(z \mid y) & \text{elif } C(x, y) > 0 \\
P_{katz}(z \mid y) & \text{otherwise}
\end{cases}
\]

Notes:
- J&M printing 1 & 2 have an error in the third line of the upper formula
- Huang et al. present a more general form where the discount can be applied for counts greater than 1

Discounted probability (Katz)

As \( \sum_{z_i \in V} P(z_i \mid x, y) = 1 \) we need to discount the probability for any given \( z \):

\[
P^*(z \mid x, y) = \frac{c^*(x, y, z)}{c(x, y)}
\]

On average:

\[
\frac{c^*(x, y, z)}{c(x, y)} < \frac{c(x, y, z)}{c(x, y)}
\]

so the sum is likely to be \(< 1\)
How much is left over?

\[ \sum_{w_n: C(w_{n-N+1}) > 0} P^*(w_n | w_{n-N+1}^{n-1}) \rightarrow \text{sum discounted P} \]

What’s left over?

\[ \beta(w_{n-N+1}^{n-1}) = 1 - \sum_{w_n: C(w_{n-N+1}) > 0} P^*(w_n | w_{n-N+1}^{n-1}) \]

Concrete example: trigram

- Trigrams seen in training: \textit{with you X}
  - with you i
  - with you there
- Backoff: you word
  - left over probability \( \beta(w_{n-N+1}^{n-1}) P(\text{word} | \text{you}) \)
  - No need to use backoff bigrams for things that were observed: P(i|you) and P(there|you).
Concrete example: trigram

- Subtract out the $P$ for observed trigrams and scale up the probability

$$\alpha(w_{n-N+1}^{n-1}) = \alpha(w_{n-2}, w_{n-1})$$

depends on context trigram case

$$= \frac{\beta(w_{n-N+1}^{n-1})}{(1 - (P(i | you) + P(there | you)))}$$

and we compute $\alpha(w_{n-2}, w_{n-1})P(word | you)_{41}$

Backoff weighting
(formal presentation)

$$\alpha(w_{n-N+1}^{n-1}) = \frac{\text{Sum of Katz N-1 gram P's that we will use}}{\text{left over P}}$$

$$= \frac{\sum_{w_n:C(w_{n-N+1}^{n})=0} P_{katz}(w_n | w_{n-N+2}^{n-1}) \beta(w_{n-N+1}^{n-1})}{1 - \sum_{w_n:C(w_{n-N+1}^{n})>0} P'(w_n | w_{n-N+1}^{n-1})}$$

$$= \frac{1 - \sum_{w_n:C(w_{n-N+1}^{n})>0} P'(w_n | w_{n-N+1}^{n-1})}{1 - \sum_{w_n:C(w_{n-N+1}^{n})>0} P''(w_n | w_{n-N+1}^{n-1})}$$

42
Kneser-Ney smoothing

(highlights only)

“I can’t see without my reading _____”

• Consider:
  \( P(\text{glasses}|\text{reading}) \) vs \( P(\text{Francisco}|\text{reading}) \)

• If neither appears in the corpus, we could back off, perhaps to a unigram
  \( P(\text{Francisco}) > P(\text{glasses}) \)

“I can’t see without my reading Francisco”

Knesser-Ney smoothing

• Look for the number of contexts a word appears in:
  – drinking glasses, buy glasses, sun glasses, 3d glasses, …
  – San Francisco, tío Francisco

• \( P_{\text{KN}}(\text{glasses}|\text{reading}) > P_{\text{KN}}(\text{Francisco}|\text{reading}) \)
Is our model any good?

- Qualitative:
  - Pick a starting word by probability: $P(W|<s>)$
  - Select subsequent words based on what we picked
- Extrinsic/In vivo measure: Use it in an ASR system (expensive)
- Intrinsic measure: Perplexity

Perplexity (PP)

- Two ways to think about measurement
  - weighted average branching factor: average number of words that can follow another word
  - exponential of cross entropy (more on this later)
- Both methods require a separate data set to evaluate.
- Take home idea: Lower perplexity models are probably better.
PP as average branching factor

\[ PP(W) = P(w_1, w_2, \ldots, w_N)^{-\frac{1}{N}} \]

\[ = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_{i-1})}} \] chain rule

\[ = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_{i-1})}} \] Markov assumption for bigram

from branching factor to entropy

• Why \( \left( \frac{1}{P(w_1, w_2, \ldots, w_N)} \right)^{-\frac{1}{N}} \)?

• Related to information theory which
  – measures how much information
  – can be used to measure how difficult a task is,
    or how two grammars or language models fit a test corpus
Quantity of information

- Amount of surprise that one sees when observing an event.

\[ I(x_i) = \log \frac{1}{P(x_i)} \]

- If an event is rare, we can derive a large quantity of information from it.

Quantity of information

- Why use log?
  - Suppose we want to know the information in two independent events:
    \[
    I(x_1, x_2) = \log \frac{1}{P(x_1, x_2)} \\
    = \log \frac{1}{P(x_1)P(x_2)} \\
    = \log \frac{1}{P(x_1)} + \log \frac{1}{P(x_2)} \\
    = I(x_1) + I(x_2)
    \]

  \[ x_1, x_2 \text{ independent} \]
Entropy

- Entropy is defined as the expected amount of information (average amount of surprise) and is usually denoted by the symbol $H$.

$$H(X) = E[I(X)]$$

$$= \sum_{x_i \in S} P(x_i) I(x_i) \quad S \text{ is all possible symbols}$$

$$= \sum_{x_i \in S} P(x_i) \log \frac{1}{P(x_i)} \quad \text{definition } I(x_i)$$

$$= E[-\log Pr(X)]$$

Example

- Assume
  - $X = \{0, 1\}$
  - \[P(X) = \begin{cases} p & X = 0 \\ 1 - p & X = 1 \end{cases}\]

- Then
  \[
  H(X) = E[I(X)] = -p \log p - (1 - p) \log(1 - p)
  \]
Entropy as information

- 8 events with probabilities:

<table>
<thead>
<tr>
<th>P(X) = i</th>
<th>1: ½</th>
<th>5: 1/64</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: ½</td>
<td>5</td>
<td>1/64</td>
</tr>
<tr>
<td>2: ¼</td>
<td>6</td>
<td>1/64</td>
</tr>
<tr>
<td>3: 1/8</td>
<td>7</td>
<td>1/64</td>
</tr>
<tr>
<td>4: 1/16</td>
<td>8</td>
<td>1/64</td>
</tr>
</tbody>
</table>

- To send a message indicating which event, we could use 3 binary bits: 000, 001, 010, … or we could use fewer bits for more probable events

Entropy

$$H(X) = -\sum_{i=1}^{8} P(i) \log(P(i))$$

$$= -\frac{1}{2} \log \frac{1}{2} - \frac{1}{4} \log \frac{1}{4} - \frac{1}{8} \log \frac{1}{8} - \frac{1}{16} \log \frac{1}{16} - 4 \left( \frac{1}{64} \log \frac{1}{64} \right)$$

$$= 2$$

- This tells us that we can encode the message using an average of 2 bits

(see J&M p. 115 for encodings)
Word entropy of a language

\[ H(L) = \lim_{n \to \infty} \frac{1}{n} H(w_1, w_2, \ldots, w_n) \]

\[ = -\lim_{n \to \infty} \frac{1}{n} \sum_{w \in L} P(w_1, w_2, \ldots, w_n) \log(P(w_1, w_2, \ldots, w_n)) \]

• Problem: We don’t know P(W).

• Solution: Cross entropy & the law of large numbers

Cross entropy

• Suppose we have an approximation \( \hat{P} \) of \( P \)

\[ H_{(P, \hat{P})}(L) = -\sum_{W \in L} P(W) \log \hat{P}(W) \]

• It can be shown that

\[ H_P(L) \leq H_{(P, \hat{P})}(L) \]
Cross entropy & law of large #s

\[ H_{(p, \hat{p})}(W) = - \sum_{W \in L} p(W) \log \hat{P}(W) \]

but suppose we take a very long \( W \)

and we approximate everything as \( \frac{1}{N} \)

If we draw enough \( W \), it should empirically converge to \( P(W) \)

Convergence of cross entropy

- Shannon-McMillan-Breiman theorem states that if we take a \( W \) long enough, cross entropy converges and we don’t need every \( W \) in the language

\[ H_{(p, \hat{p})}(W) = - \sum_{n \to \infty} \frac{1}{n} \log \hat{P}(w_1, w_2, \ldots, w_n) \]

Note: Some conditions (stationarity), which we will not discuss must hold for this to apply.
Relating it back to perplexity

• Cross entropy as an upper bound
  \[ H(W) \leq H_{(p, \hat{p})}(W) \]
  \[ -\frac{1}{N} \log \left( P(w_1, w_2, \ldots, w_n) \right) \leq -\frac{1}{n} \log \left( \hat{P}(w_1, w_2, \ldots, w_n) \right) \]

• Perplexity(\(W\)) = \(2^{H(W)} = 2^{\frac{-1}{N} \log \hat{P}(w_1, w_2, \ldots, w_n)}\)
  \[ = \left( 2^{\log \hat{P}(w_1, w_2, \ldots, w_n)} \right)^{-\frac{1}{N}} = P(w_1, w_2, \ldots, w_N)^{-\frac{1}{N}} \]
  \[ = \sqrt[N]{\frac{1}{P(w_1, w_2, \ldots, w_N)}} \]
  which was our definition of perplexity!