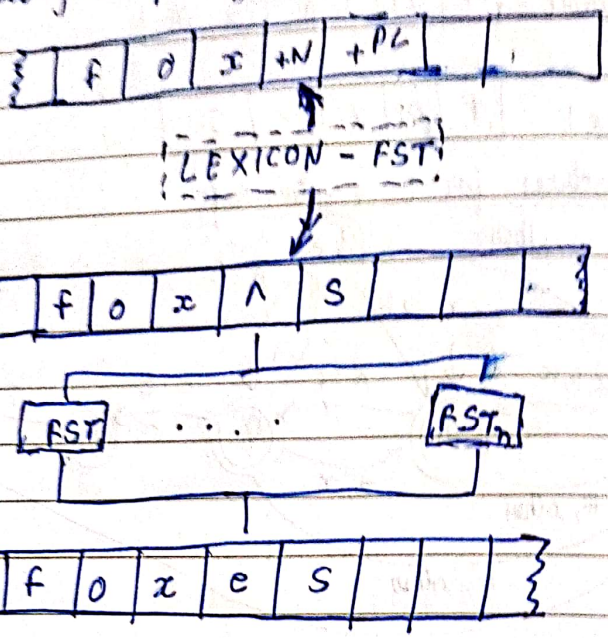
Combining FST lexicon and Rules:

generating or parsing with FST lexicon & rules




SFST Tool

Properties of FSAs

- Epsilon-free

MED Algorithm (Min. edit distance):

Insert - 1

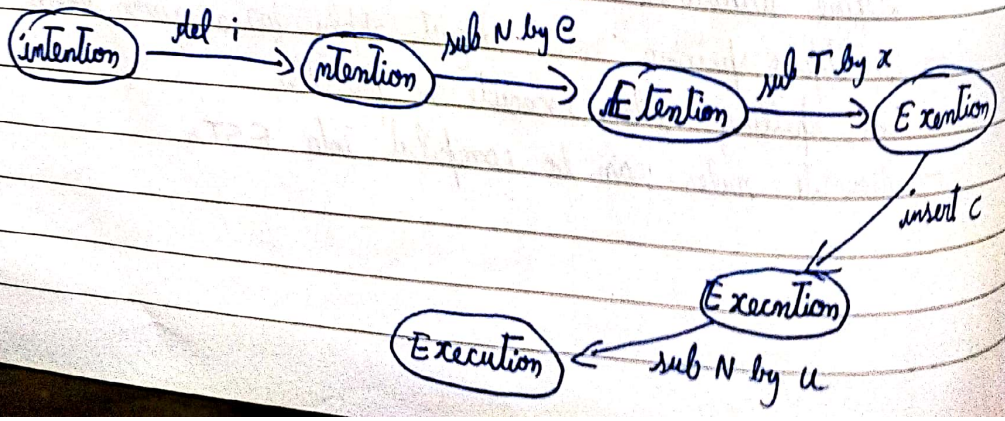
Delete - 1

Substitute cost - 2

source I N T E*N T I O N

* E X E C U T I O N

d S S i S



Dynamic Programming: Minimum edit distance

Edit distance table:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| N | 9 | 8 | 9 | 10 | 11 | 12 | 11 | 10 | 9 | 8 |
| O | 8 | 7 | 8 | 9 | 10 | 11 | 10 | 9 | 8 | 9 |
| I | 7 | 6 | 7 | 8 | 9 | 10 | 9 | 8 | 9 | 10 |
| T | 6 | 5 | 6 | 7 | 8 | 9 | 8 | 9 | 10 | 11 |
| N | 5 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 10 |
| E | 4 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 9 |
| T | 3 | 4 | 5 | 6 | 7 | 8 | 7 | 8 | 9 | 8 |
| N | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 7 | 8 | 7 |
| I | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 6 | 7 | 8 |
| # | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | # | E | X | E | C | U | T | I | O | N |

$$distance[i,j] = \min \begin{cases} distance[i-1,j] + ins\text{-}cost(target_{i-1}) \\ distance[i-1,j-1] + subst\text{-}cost(source_{j-1}, target_{i-1}) \\ distance[i,j-1] + del\text{-}cost(source_{j-1}) \end{cases}$$

Knowing min edit distance

1. Backtrace path from MET matrix 8 8 8 8 6 5 4 3 3 2 1 0

2. 8 8 8 8 8 $^{2}$6 $^{1}$5 $^{2}$3 $^{1}$4 $^{2}$2 $^{2}$0

INTENTION

INtE*NTION

EX*ECUTION

0 2 4 5 5 6 8 6 8 9

EXECUTION

8 8 8 8 6 5 5 4 3 2 1 0

I N T E N T I O N

source I N T * * E * N T I O N

Target * * * E X E C U T I O N

0 1 2 3 4 5 5 6 8 8 8 8

E X E C U T I O N

N-grams:-

HW

1. Compound
   Compact
2. class
   clash
3. picturize
   maximize

N-grams:

- Using notion of word prediction for processing language
  - Eg:
    What word is most likely to follow:
    I'd like to make a collect call
  - Using probabilistic models called N-grams to predict next word from previous n-1 words.
  - Computing probability of next word is related to comput the probability of a sequence of words

- N-gram is an N-Token sequence of words
- 2-gram is called bigram is a 2-word sequence of words
  Eg: your homework
- 3-gram called Trigram is a 3-word sequence of words
  Eg: turn your homework
- N-gram model is a model which computes last word of a N-gram from previous ones (both N-gram & N-gram model

used in the same way)
- such statistical models of word sequences are also called language models or LMs.

- N-gram model uses the previous N-1 words to predict next one.
  - $P(w_n | w_{n-1})$
  - Dealing with $P(<word> | <some\ prefix>)$
  - Unigrams : P(student)
  - bigrams : P(student | clever)
  - trigrams : P(student | The clever)

Applications
- Automatic speech recognition (ASR):
- Spelling correction:
  - Here, we need to find & correct spelling errors that accidentally result in real English words:
- Since these errors have real words
  - we can't !
- Machine Translation

Corpora:
- Corpora (plural of corpus is corpora) are collections of text & speech.
- Eg:
  - Brown corpus
  - Wall Street Journal & AP news corpora

Word counting in Corpora:
- Counting : Probabilities are based on counting things.
- Counting of things in natural language is based on a corpus (plural corpora), an on-line collection of text or speech.

1. Look at 2 popular corpora, Brown and switchboard.
2. The Brown Corpus
   - a 1 million word collection of samples
   - from 500 written Texts

Terminology

Compute $P(w|h)$ ① $\frac{c(h,w)}{c(h)}$

2. Computing joint probability

$P(\text{"its water is so transparent"})$

$= \frac{c(\text{"its water is so transparent"})}{c(\text{"All possible 5 word system})}$

3. Conditional Probability : $P(A|B) = \frac{P(A \cap B)}{P(B)}$

$P(A \cap B) = P(A|B) \cdot P(B)$

$P(A,B,C,D) = P(A) \cdot P(B|A) \cdot P(C|A,B) \cdot P(D|A,B,C)$

Markov assumption :

$$P(w_n | w_1^{n-1}) \approx P(w_n | w_{n-N+1}^{n-1})$$

Bigram Version

$$P(w_n | w_1^{n-1}) \approx P(w_n | w_{n-1})$$

Estimating Bigram Possibilities

Maximum likelihood Estimate (MLE)

$$P(w_i | w_{i-1}) = \frac{\text{Count}(w_{i-1}, w_i)}{\text{Count}(w_{i-1})}$$

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

Calculate

$P(I|<s>)$ $P(Sam|<s>)$ $P(am|I)$ $P(</s>|Sam)$
$P(Sam|am)$ $P(do|I)$

$P(I|<s>) = \frac{2}{3}$ $P(</s>|Sam) = \frac{1}{2}$

$P(Sam|<s>) = \frac{1}{3}$ $P(Sam|am) = \frac{1}{2}$

$P(am|I) = \frac{2}{3}$ $P(do|I) = \frac{1}{3}$

$$P(w_n | w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1} w_n)}{C(w_{n-N+1}^{n-1})}$$

T - Type of distinct words

| | i | want | To | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | $\frac{5}{2533}$ | $\frac{827}{2533}$ | 0 | $\frac{9}{2533}$ | 0 | 0 | 0 | $\frac{2}{2533}$ |
| want | $\frac{2}{927}$ | 0 | $\frac{608}{927}$ | $\frac{1}{927}$ | $\frac{6}{927}$ | $\frac{6}{927}$ | $\frac{5}{927}$ | $\frac{1}{927}$ |
| to | | | | | | | | |
| eat | | | | | | | | |
| chinese | | | | | | | | |
| food | | | | | | | | |
| lunch | | | | | | | | |
| spend | | | | | | | | |

Compute Bigram probabilities for BRPS (Berkley Restaurant Project Sentences) Corpus

Unigram Count

| i | want | To | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|
| 2533 | 927 | 2417 | 746 | 158 | 1093 | 341 | 278 |

Bigram Count

| | i | want | To | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 5 | 827 | 0 | 9 | 0 | 6 | 0 | 2 |
| want | 2 | 0 | 608 | 1 | 6 | 6 | 5 | 1 |
| To | 2 | 0 | 4 | 686 | 2 | 0 | 6 | 211 |
| eat | 0 | 0 | 2 | 0 | 16 | 2 | 42 | 0 |
| chinese | 1 | 0 | 0 | 0 | 0 | 82 | 1 | 0 |
| food | 15 | 0 | 15 | 0 | 1 | 4 | 0 | 0 |
| lunch | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| spend | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

Bigram probability

| | i | want | To | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 0.002 | 0.33 | 0 | 0.0036 | 0 | 0 | 0 | 0.0007 |
| want | 0.0022 | 0 | 0.66 | 0.0011 | 0.0065 | 0.0065 | 0.0054 | 0.0011 |
| To | 0.00083 | 0 | 0.0017 | 0.28 | 0.00083 | 0 | 0.0025 | 0.087 |
| eat | 0 | 0 | 0.0027 | 0 | 0.021 | 0.0027 | 0.096 | 0 |
| chinese | 0.0063 | 0 | 0 | 0 | 0 | 0.52 | 0.0063 | 0 |
| food | 0.014 | 0 | 0.014 | 0 | 0.00092 | 0.0037 | 0 | 0 |
| lunch | 0.0059 | 0 | 0 | 0 | 0 | 0.0029 | 0 | 0 |
| spend | 0.0036 | 0 | 0.0036 | 0 | 0 | 0 | 0 | 0 |

Eg: I want chinese food : Compute probability of sentence

$P(i \mid <s>) = 0.25$

$P(\text{english} \mid \text{want}) = 0.0011$

$P(\text{food} \mid \text{english}) = 0.5$

$P(</s> \mid \text{food}) = 0.68$

RX

Eg:

P(<s> I want chinese food </s>)

= P(I/<s>) . P(want/I) . P(chinese/want) . P(food/chinese) P(</s>/food>

= 0.25 * 0.33 * 0.0065 * 0.52 * 0.68

= 0.0002

H.W Eg: I want to eat chinese food

P(<s> I want To eat chinese food </s>)

= P(I/<s>) . P(want/I) . P(to/want) . P(eat/to) . P(chinese/eat) . P(food/chinese) . P(</s>/food>

= 0.25 x 0.33 x 0.66 x 0.28 x 0.021 x 0.52 x 0.68

= 0.0011

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_i)}$$

Smoothing :

- Major probl

## Laplace Smoothing :

- Also called laplace smoothing
- Pretend we saw each word one more time than we did
- Just add one to all counts!
- MLE estimate : $P_{MLE}(w_i | w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})}$
- Add-1 estimate : $P_{ADD-1}(w_i | w_{i-1}) = \frac{C(w_{i-1}, w_i) + 1}{C(w_{i-1}) + V}$

$$P(\text{Apple}) = \frac{3+1}{13+4} = \frac{4}{17}$$

$$P(w_i) = \frac{C(w_i)}{N}$$

for bigram

$$P(w_i | w_{i-1}) = \frac{C(w_{i-1} \cdot w_i)}{C(w_{i-1})}$$

$$P_{laplace}(w_i | w_{i-1}) = \frac{C(w_{i-1} \cdot w_i) + 1}{C(w_{i-1}) + V}$$

$$C_i^* = (C_{i+1}) * C(w_{i-1})$$

$$P^*_{laplace}(w_n | w_{n-1}) = \frac{C(w_{n-1} w_n) + 1}{C(w_{n-1}) + V}$$

$$P^*_{laplace}(\text{want} | \text{i}) \quad C^*(w_{n-1} | w_n) = \frac{[C(w_{n-1} w_n) + 1] * C(w_n - 1)}{C(w_n - 1) + V}$$

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 6 | 828 | 1 | 10 | 1 | 1 | 1 | 3 |
| want | 3 | 1 | 609 | 2 | 7 | 7 | 1 | 2 |
| To | 3 | 1 | 5 | 687 | 3 | 1 | 7 | 212 |
| eat | 1 | 1 | 3 | 19 | 17 | 3 | 43 | 1 |
| chinese | 2 | 1 | 1 | 1 | 1 | 83 | 2 | 1 |
| food | 16 | 1 | 16 | 1 | 2 | 5 | 1 | 1 |
| lunch | 3 | 1 | 1 | 1 | 1 | 2 | 1 | 1 |
| spend | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |

$V = 1446$

$$P^*(i|i) = \frac{6}{2533+1446} = 0.0015$$

$$P^*(want|i) = \frac{828}{2533+1446} = 0.208$$

$$P^*(To|i) = \frac{1}{2533+1446} = 0.00025$$

$$P^*(eat|i) = \frac{10}{2533+1446} = 0.0025$$

$$P^*(chinese|i) = \frac{1}{2533+1446} = 0.00025 = P(food|i) = P(lunch|i)$$

$$P^*(spend|i) = \frac{3}{2533+1446} = 0.00075$$

$$c^*(i|i) = \frac{6 \times 2533}{2533+1446} = 3.7995$$

$$c^*(want|i) = \frac{828 \times 2533}{2533+1446} = 526.864$$

$$c^*(to|i) = \frac{1 \times 2533}{2533+1446} = 0.6333 = c^*(chinese|i) = c^*(food|i) = c^*(lunch|i)$$

$$c^*(eat|i) = \frac{10 \times 2533}{2533+1446} = 6.3325$$

$$c^*(spend|i) = \frac{3 \times 2533}{2533+1446} = 1.8998$$

## Laplace Smoothing:

- Sharp change in counts & probabilities occur - b

## Smoothing:

1) Unsmoothed
2) Laplace - Add - on
3) Add - k
4) Good Turing
5) Combining
   Estimation — Backoff

## Good Turing Smoothing:

- $N_c$ - Count of things we've seen c Times

  Sam I am Sam I am Sam I do not eat

| | |
|---|---|
| I | 3 |
| Sam | 2 |
| am | 2 |
| do | 1 |
| not | 1 |
| eat | 1 |

$N_1 = 3$
$N_2 = 2$
$N_1 = 1$

$N_c$ - frequency of frequency c

N - no. of bigrams seen - $V^2$
$N_0$ - no. of bigrams with count 0
$N_c$ - no. of bigrams with count c

$$N_0 = V^2 - N$$

$$c^* = \frac{(c+1) \cdot N_{c+1}}{N_c}$$

c - original (real) word count

$(c+1).N_{c+1}$ - The probability mass for words with frequency c+1

$c^*$ - new (adjusted) word count.

MLE count for $N_c$ is c.

Eg:

| | | |
|---|---|---|
| 10 carp | N1 = 3 | N = 18 |
| 3 perch | N2 = 1 | |
| 2 whitefish | N3 = 1 | |
| 1 trout | N10 = 1 | |
| 1 salmon | N0 = 2 | |
| 1 eel | | |

MLE count of hitherto-unseen species (catfish or bass) is 0

$P^*_{GT}$ (things with frequency 0 in training) $= \frac{N_1}{N} = \frac{3}{18}$

$P^*_{GT}$ (bass) $= P^*_{GT}$ (catfish) $= \frac{3}{18} * \frac{1}{N_0}$

| | unseen (bass or catfish) | trout |
|---|---|---|
| c | 0 | 1 |
| MLE p | $p = \frac{0}{18} = 0$ | $\frac{1}{18}$ |
| $c^*$ | | $c^*(\text{trout}) = 2 \times \frac{N_2}{N_1} = 2 \times \frac{1}{3} = 0.67$ |
| GT $p^*_{GT}$ | $P^*_{GT}(\text{unseen}) = \frac{N_1}{N} = \frac{3}{18} = 0.17$ | $P^*_{GT}(\text{trout}) = \frac{0.67}{18} = \frac{1}{27} = 0.037$ |

$c^*$ computation isn't there for unseen i.e., for count 0

Revised count.

Consider vocabulary as {a, b, c}
Consider Corpus : b a b a a c b c a c a c
Find list of possible bigrams
Find out count of seen (observed) & unseen (unobserved) bigrams
& Their frequencies using Good Turing.

| | a | b | c |
|---|---|---|---|
| a | 1 | 1 | 3 |
| b | 2 | 0 | 1 |
| c | 2 | 1 | 0 |

ba, ab, ba, aa, ac, cb, bc, ca, ca, ac

aa = 1 ba = 2 ca = 2
ab = 1 bb = 0 cb = 1
ac = 3 bc = 1 cc = 0

N = 11

~~3 b~~ ~~5 a~~ ~~4 c~~ ~~N = 12~~

$N_0 = 2$
$N_1 = 4$
$N_2 = 2$
$N_3 = 1$

| c | $c^*_{GT}$ | $P^*_{GT}$ | |
|---|---|---|---|
| | $P_{MLE}$ | $C^*_{GT}$ | $P^*_{GT}$ |
| C = 0 | 0 | | $\frac{4}{11}$ $\frac{N_1}{N}$ |
| C = 1 | $\frac{1}{11}$ | 1 | $2 \times \frac{2}{4} * \frac{1}{11} = \frac{1}{11}$ |
| C = 2 | $\frac{2}{11}$ | $3 \times \frac{1}{2}$ | $\frac{3}{2} \times \frac{1}{11} = \frac{3}{22}$ |
| | $\frac{4}{11} + \frac{4}{11} + \frac{6}{22}$ | | $= \frac{8+8+6}{22}$ |

$P^*_{GT}$ (based on no. of words)

$P^*_{GT}(bb) = P^*_{GT}(cc) = \frac{2}{11}$ $\frac{N_1}{N_1}$

$P^*_{GT}(aa) = P^*_{GT}(ab) = P^*_{GT}(bc)$
$= P^*_{GT}(cb) = \frac{1}{11}$

$P^*_{GT}(ba) = P^*_{GT}(ca)$
$= \frac{3}{22}$

Laplace:

| | a | b | c |
|---|---|---|---|
| a | 1 | 1 | 3 |
| b | 2 | 0 | 1 |
| c | 2 | 1 | 0 |

unsmoothed.

$$c_i^* = (c_{i+1}) * c(w_{i-1})$$

$$P_{laplace}(w_i / w_{i-1}) = \frac{C(w_{i-1} \cdot w_i) + 1}{C(w_{i-1}) + V}$$

| | a | b | c |
|---|---|---|---|
| a | 2 | 2 | 4 |
| b | 3 | 1 | 2 |
| c | 3 | 2 | 1 |

Add one smoothed.

$V^2 = 9$

$$P^*(a/a) = \frac{2}{5+9} = \frac{2}{14} = \frac{1}{7}$$

$$P^*(b/a) = \frac{2}{5+9} = \frac{1}{7}$$

$$P^*(c/a) = \frac{4}{5+9} = \frac{4}{14} = \frac{2}{7}$$

$$P^*(a/b) = \frac{3}{3+9} = \frac{3}{12} = \frac{1}{4}$$

$$P^*(b/b) = 0 = P^*(c/c)$$

$$P^*(c/b) = \frac{1}{3+9} = \frac{1}{12}$$

$$P^*(a/c) = \frac{2}{4+9} = \frac{2}{13}$$

$$P^*(b/c) = \frac{1}{4+9} = \frac{1}{13}$$

$$\frac{(C+1)N}{N+V}$$

| a | b | c |
|---|---|---|
| 5 | 3 | 4 |

| | a | b | c |
|---|---|---|---|
| a | 2 | 2 | 4 |
| b | 3 | 1 | 2 |
| c | 3 | 2 | 1 |

| | a | b | c |
|---|---|---|---|
| a | 1.25 | 1.25 | 2.5 |
| b | 1.5 | 0.5 | 1 |
| c | 1.71 | 1.14 | 0.57 |

Add one reconstructed bigram count $c^*$ $= \frac{(c+1)N}{N+V}$

$= \frac{2 \times 5}{5+3}$

| | a | b | c |
|---|---|---|---|
| a | 0.25 | 0.25 | 0.5 |
| b | 0.5 | 0.0 | 0.33 |
| c | | | |

$P(w_i | w_{i-1}) = \frac{C(w_{i-1}.w_i)+1}{C(w_{i-1})+V}$

$P(a|a) = \frac{C(a.a)+1}{C(a)+V}$ $\frac{2 \times 5}{5}$

$= \frac{2}{5+3}$

$= 2/8$

$= 0.25$

Interpolation:

- Simple linear interpolation:
- We combine different order N-grams by linearly interpolating all models.
- To combine estimate Trigram probability $P(w_n | w_{n-1} w_{n-2})$
-

$$P(w_n | w_{n-1} w_{n-2}) = \lambda_1 P(w_n | w_{n-1} w_{n-2}) + \lambda_2 P(w_n | w_{n-1}) + \lambda_3 P(w_n)$$

## Backoff N-gram modelling : Katz backoff

$$P_{katz}(z|x,y) = \begin{cases} P^*(z|x,y) & \text{if } c(x,y,z) > 0 \\ \alpha(x,y)\, P_{katz}(z|y) & \text{else if } c(x,y,z) > 0 \\ P^*(z) & \text{otherwise} \end{cases}$$

$$P_{katz}(z|y) = \begin{cases} P^*(z|y) & \text{if } c(y,z) > 0 \\ \alpha(y)\, P^*(z) & \text{otherwise} \end{cases}$$

## Different

Verbs : VB, VBP, VBZ, VBD, VBG, VBN
- base, present - non-3rd, present - 3rd, past - ing, -en

Nouns : NNP, NNPS, NN, NNS
- proper / common, singular / plural (singular includes mass + generic)

Adjectives : JJ, JJR, JJS (base, comparative, superlative)

Adverbs : RB, RBR, RBS, RP (base, comparative, superlative, particle)

Pronouns : PRP, PP$ (personal, possessive)

Interrogatives : WP, WP$, WDT, WRB (compare to : PRP, PP$, DT, RB)

Other Closed Class : CC, CD, DT, PDT, IN, MD

Punctuation : # $ . ,

Examples:
The grand jury commented on a no. of other topics

Eg : Determiners : The and a