Breaking down the Language Barrier with Statistical Machine Translation:
1) Language Models

http://www.phontron.com/class/sentan2014

Advanced Research Seminar I/III
Graham Neubig
2014-1-28

Machine Translation

- Automatically translate between languages

Source	Target
太郎が花子を訪問した。	Taro visited Hanako.

- Real products/services being created!

Google translate

excite

Institute of Science and Technology Nara NAIST

Chronus Simultaneous Translation System

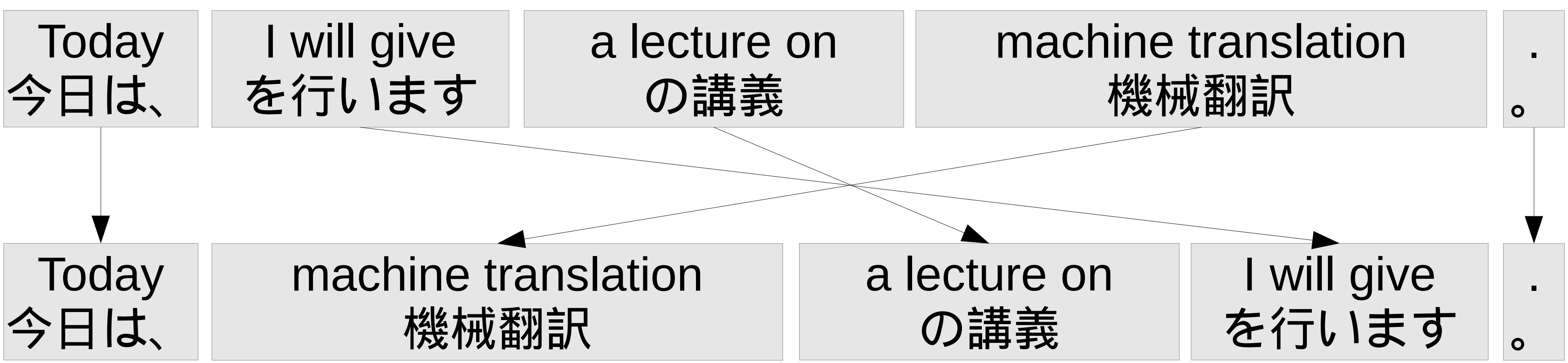
How does machine translation work?

Today I will give a lecture on machine translation .

How does machine translation work?

- Divide sentence into translatable patterns, reorder, combine

Today I will give a lecture on machine translation .



今日は、機械翻訳の講義を行います。

Problem

- There are millions of possible translations!

花子 が 太郎 に 会った
Hanako met Taro
Hanako met to Taro
Hanako ran in to Taro
Taro met Hanako
The Hanako met the Taro

- How do we tell which is better?

Statistical Machine Translation

- Translation model:

P(“今日” |“today”) = high   P(“今日 は、” |“today”) = medium

P(“昨日” |“today”) = low

- Reordering Model:

P(鶏 が 食べる → chicken eats)=high

P(鶏 を 食べる → eats chicken)=high

P(鶏 が 食べる → eats chicken)=low

- Language Model:

P(“Taro met Hanako”)=high   P(“the Taro met the Hanako”)=low

Creating a Machine Translation System

- Learn patterns from documents

Documents

太郎が花子を訪問した。
Taro visited Hanako.

花子にプレゼントを渡した。
He gave Hanako a present.

...

Models

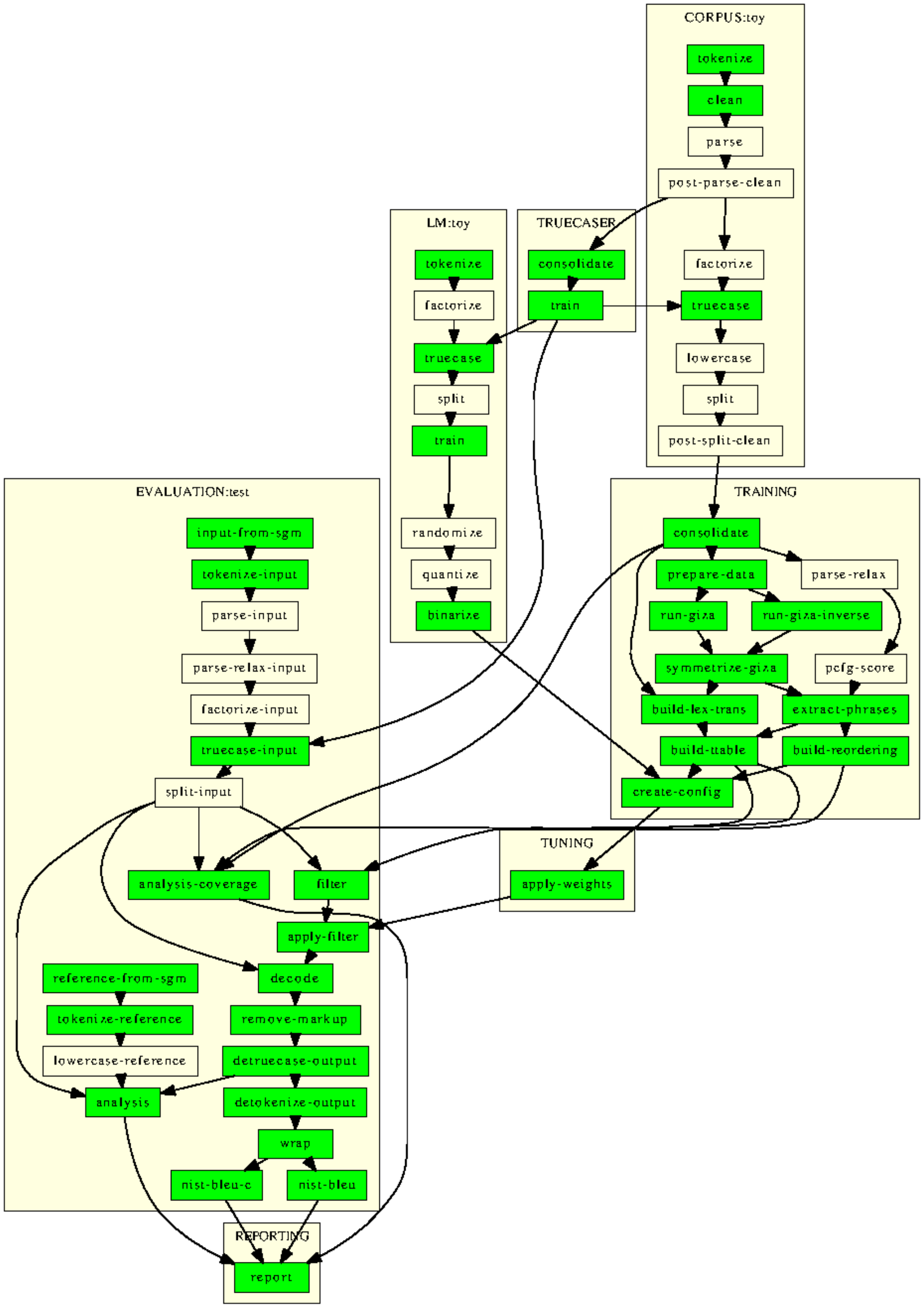
Translation Model

Reordering Model

Language Model

Easier Said than Done!

Flow-chart for training an MT system →



Lecture Plan

Lecture Plan

1) Language models
2) Word alignment / Translation modeling
3) Kana-kanji conversion / Phrase-based translation
4) Machine translation evaluation / Optimization

Assignments

- All assignments will require simple programming
- A “baseline” system will be prepared for you (in Python)
- Improve the system's accuracy, turn in your code, and a short description of what you changed and why
- You may work in teams of up to 3 people
- I will keep a scoreboard

Today's Assignment

- I have given you code to train and test a language model using bigrams and linear interpolation
- Make a change to the code to improve its entropy
- Due date: Monday, February 3rd, 23:59
- Address: neubig@is.naist.jp

Unigram Language Models

Why Language Models?

- When performing Japanese-English translation, which is correct?

太郎は花子を訪問した

W_1 = Taro visited Hanako

W_2 = the Taro visited the Hanako

W_3 = fat visit ro flower child

W_4 = 太郎 は 花子 を 訪問 した

Why Language Models?

- When performing Japanese-English translation, which is correct?

太郎は花子を訪問した

W_1 = Taro visited Hanako

W_2 = the Taro visited the Hanako

W_3 = fat visit ro flower child

W_4 = 太郎 は 花子 を 訪問 した

- The language model tells you which is most likely

Probabilistic Language Models

- Language models assign a probability to each sentence

W_1 = taro visited hanako	$P(W_1) = 4.021 * 10^{-3}$
W_2 = the taro visited the hanako	$P(W_2) = 8.932 * 10^{-4}$
W_3 = fat visit ro flower child	$P(W_3) = 2.432 * 10^{-7}$
W_4 = 太郎 は 花子 を 訪問 した	$P(W_4) = 9.124 * 10^{-23}$

- $P(W_1) > P(W_2) > P(W_3) > P(W_4)$ is best
  (in Japanese $P(W_4) > P(W_1), P(W_2), P(W_3)$ ？)

Calculating Sentence Probabilities

- We want the probability of

W = taro visited hanako

- Represent this mathematically as:

$$P(|W| = 3, w_1=\text{"taro"}, w_2=\text{"visited"}, w_3=\text{"hanako"})$$

Calculating Sentence Probabilities

- We want the probability of

W = taro visited hanako

- Represent this mathematically as (using chain rule):

$P(|W| = 3, w_1=\text{"taro"}, w_2=\text{"visited"}, w_3=\text{"hanako"}) =$

$P(w_1=\text{"taro"} \mid w_0 = \text{"<s>"})$

$* P(w_2=\text{"visited"} \mid w_0 = \text{"<s>"}, w_1=\text{"taro"})$

$* P(w_3=\text{"hanako"} \mid w_0 = \text{"<s>"}, w_1=\text{"taro"}, w_2=\text{"visited"})$

$* P(w_4=\text{"</s>"} \mid w_0 = \text{"<s>"}, w_1=\text{"taro"}, w_2=\text{"visited"}, w_3=\text{"hanako"})$

NOTE:
sentence start <s> and end </s> symbol

NOTE:
$P(w_0 = \text{<s>}) = 1$

Incremental Computation

- Previous equation can be written:

$$P(W) = \prod_{i=1}^{|W|+1} P(w_i | w_0 \dots w_{i-1})$$

- How do we decide probability?

$$P(w_i | w_0 \dots w_{i-1})$$

Maximum Likelihood Estimation

- Calculate word strings in corpus, take fraction

$$P(w_i|w_0 \ldots w_{i-1}) = \frac{c(w_0 \ldots w_i)}{c(w_0 \ldots w_{i-1})}$$

i live in osaka . </s>
i am a graduate student . </s>
my school is in nara . </s>

P(live | <s> i) = c(<s> i live)/c(<s> i) = 1 / 2 = 0.5

P(am | <s> i) = c(<s> i am)/c(<s> i) = 1 / 2 = 0.5

Problem With Full Estimation

- Weak when counts are low:

Training:

i live in osaka . </s>
i am a graduate student . </s>
my school is in nara . </s>

Test:

<s> i live in nara . </s>

↓

P(nara|<s> i live in) = 0/1 = 0

↓

P(W=<s> i live in nara . </s>) = 0

Unigram Model

- Do not use history:

$$P(w_i|w_0 \ldots w_{i-1}) \approx P(w_i) = \frac{c(w_i)}{\sum_{\tilde{w}} c(\tilde{w})}$$

i live in osaka . </s>
i am a graduate student . </s>
my school is in nara . </s>

P(nara) = 1/20 = 0.05
P(i) = 2/20 = 0.1
P(</s>) = 3/20 = 0.15

P(W=i live in nara . </s>) =
$0.1 * 0.05 * 0.1 * 0.05 * 0.15 * 0.15 = 5.625 * 10^{-7}$

What about Unknown Words?!

- Simple ML estimation doesn't work

i live in osaka . </s> i am a graduate student . </s> → my school is in nara . </s>	P(nara) = 1/20 = 0.05 P(i) = 2/20 = 0.1 P(kyoto) = 0/20 = 0

- Often, unknown words are ignored (ASR)
- Better way to solve
  - Save some probability for unknown words ($\lambda_{unk} = 1-\lambda_1$)
  - Guess total vocabulary size (N), including unknowns

$$P(w_i) = \lambda_1 P_{ML}(w_i) + (1-\lambda_1)\frac{1}{N}$$

Unknown Word Example

- Total vocabulary size: $N=10^6$
- Unknown word probability: $\lambda_{unk}=0.05$ ($\lambda_1 = 0.95$)

$$P(w_i)=\lambda_1 P_{ML}(w_i)+(1-\lambda_1)\frac{1}{N}$$

P(nara) = 0.95*0.05 + 0.05*(1/10^6) = 0.04750005

P(i) = 0.95*0.10 + 0.05*(1/10^6) = 0.09500005

P(kyoto) = 0.95*0.00 + 0.05*(1/10^6) = 0.00000005

Bigram Language Models

Unigram Models Ignore Word Order!

- Ignoring context, probabilities are the same:

$$P_{uni}(w=\text{speech recognition system}) =$$
$$P(w=\text{speech}) * P(w=\text{recognition}) * P(w=\text{system}) * P(w=\text{</s>})$$

$$=$$

$$P_{uni}(w=\text{system recognition speech}) =$$
$$P(w=\text{speech}) * P(w=\text{recognition}) * P(w=\text{system}) * P(w=\text{</s>})$$

Unigram Models Ignore Agreement!

- Good sentences (words agree):

$$P_{uni}(w=\text{i am}) = P(w=\text{i}) * P(w=\text{am}) * P(w=\text{</s>})$$

$$P_{uni}(w=\text{we are}) = P(w=\text{we}) * P(w=\text{are}) * P(w=\text{</s>})$$

- Bad sentences (words don't agree)

$$P_{uni}(w=\text{we am}) = P(w=\text{we}) * P(w=\text{am}) * P(w=\text{</s>})$$

$$P_{uni}(w=\text{i are}) = P(w=\text{i}) * P(w=\text{are}) * P(w=\text{</s>})$$

But no penalty because probabilities are independent!

Solution: Add More Context!

- Unigram model ignored context:

$$P(w_i|w_0 \ldots w_{i-1}) \approx P(w_i)$$

- Bigram model adds one word of context

$$P(w_i|w_0 \ldots w_{i-1}) \approx P(w_i|w_{i-1})$$

- Trigram model adds two words of context

$$P(w_i|w_0 \ldots w_{i-1}) \approx P(w_i|w_{i-2} w_{i-1})$$

- Four-gram, five-gram, six-gram, etc...

Maximum Likelihood Estimation of n-gram Probabilities

- Calculate counts of n word and n-1 word strings

$$P(w_i \mid w_{i-n+1} \ldots w_{i-1}) = \frac{c(w_{i-n+1} \ldots w_i)}{c(w_{i-n+1} \ldots w_{i-1})}$$

i live in osaka . </s>
i am a graduate student . </s>
my school is in nara . </s>

n=2 →

P(osaka | in) = c(in osaka)/c(in) = 1 / 2 = 0.5

P(nara | in) = c(in nara)/c(in) = 1 / 2 = 0.5

Still Problems of Sparsity

- When n-gram frequency is 0, probability is 0

P(osaka | in)	= c(i osaka)/c(in)	= 1 / 2 = 0.5
P(nara | in)	= c(i nara)/c(in)	= 1 / 2 = 0.5
P(school | in)	= c(in school)/c(in)	= 0 / 2 = **0**!!

- Like unigram model, we can use linear interpolation

Bigram: $$P(w_i \mid w_{i-1}) = \lambda_2 P_{ML}(w_i \mid w_{i-1}) + (1 - \lambda_2) P(w_i)$$

Unigram: $$P(w_i) = \lambda_1 P_{ML}(w_i) + (1 - \lambda_1) \frac{1}{N}$$

Choosing Values of λ: Grid Search

- One method to choose λ_2, λ_1: try many values

$$\lambda_2 = 0.95, \lambda_1 = 0.95$$
$$\lambda_2 = 0.95, \lambda_1 = 0.90$$
$$\lambda_2 = 0.95, \lambda_1 = 0.85$$
$$\ldots$$
$$\lambda_2 = 0.95, \lambda_1 = 0.05$$
$$\lambda_2 = 0.90, \lambda_1 = 0.95$$
$$\lambda_2 = 0.90, \lambda_1 = 0.90$$
$$\ldots$$
$$\lambda_2 = 0.05, \lambda_1 = 0.10$$
$$\lambda_2 = 0.05, \lambda_1 = 0.05$$

Problems:

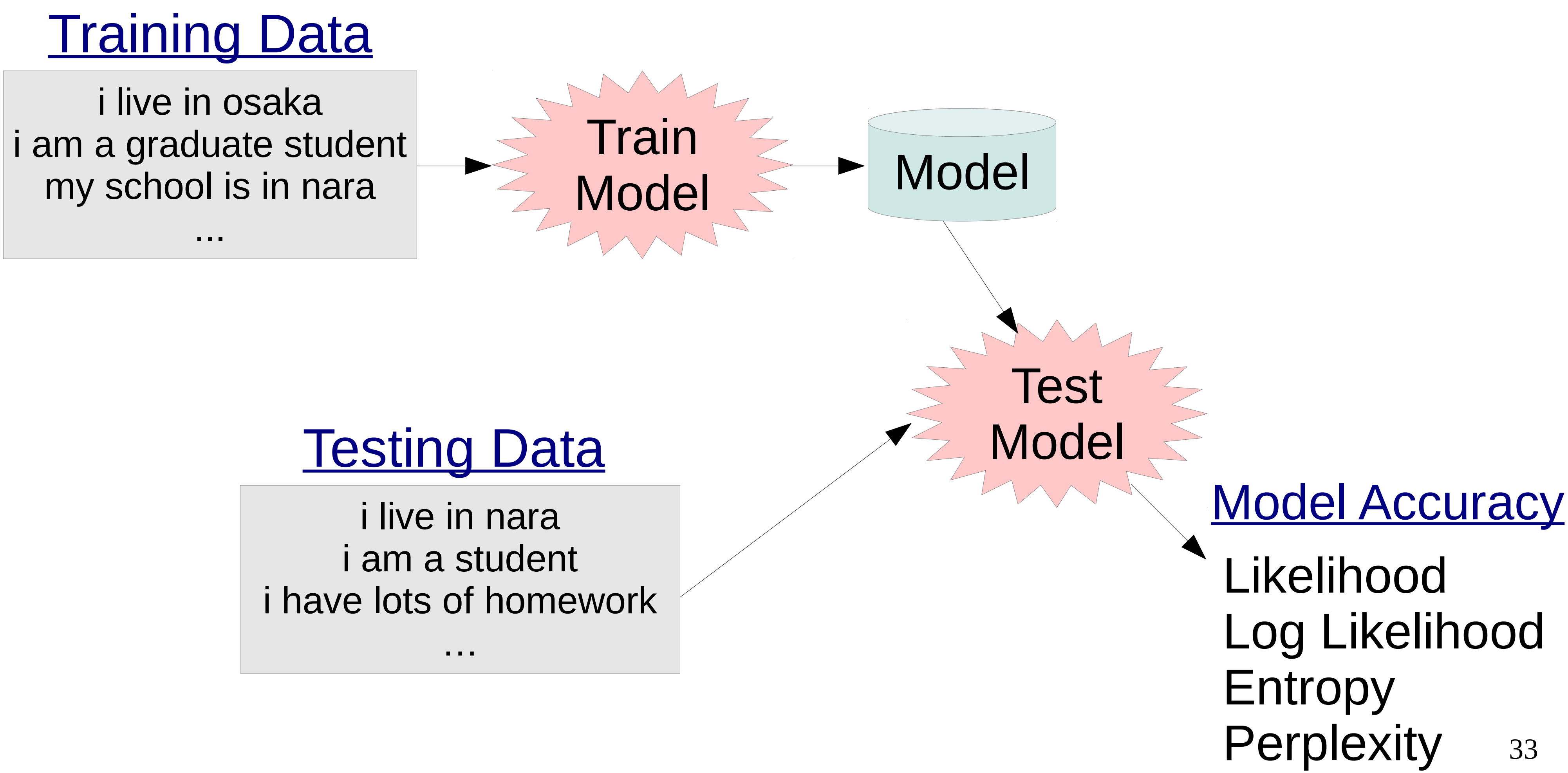
Too many options
→ Choosing takes time!

Using same λ for all n-grams
→ There is a smarter way!

Evaluating Language Models

Experimental Setup

- Use training and test sets



Likelihood

- Likelihood is the probability of some observed data (the test set W_{test}), given the model M

$$P(W_{test}|M)=\prod_{\boldsymbol{w}\in W_{test}} P(\boldsymbol{w}|M)$$

i live in nara	P(w=”i live in nara”|M) =	$2.52*10^{-21}$
		x
i am a student	P(w=”i am a student”|M) =	$3.48*10^{-19}$
		x
my classes are hard	P(w=”my classes are hard”|M) =	$2.15*10^{-34}$
		=
		$1.89*10^{-73}$

Log Likelihood

- Likelihood uses very small numbers=underflow
- Taking the log resolves this problem

$$\log P(W_{test}|M) = \sum_{w \in W_{test}} \log P(\boldsymbol{w}|M)$$

i live in nara	log P(w=”i live in nara”|M) =	-20.58
		+
i am a student	log P(w=”i am a student”|M) =	-18.45
		+
my classes are hard	log P(w=”my classes are hard”|M) =	-33.67
		=
		-72.60

Entropy

- Entropy H is average negative $\log_2$ likelihood per word

$$H(W_{test}|M) = \frac{1}{|W_{test}|} \sum_{\boldsymbol{w} \in W_{test}} -\log_2 P(\boldsymbol{w}|M)$$

i live in nara
i am a student
my classes are hard

$\log_2$ P(w="i live in nara"|M)= 68.43

+

$\log_2$ P(w="i am a student"|M)= 61.32

+

$\log_2$ P(w="my classes are hard"|M)= 111.84

/

# of words= 12

=

20.13

* note, we can also count </s> in # of words (in which case it is 15)

Entropy and Compression

- Entropy H is also the average number of bits needed to encode information (Shannon's information theory)

$$H=\frac{1}{|W_{test}|}\sum_{w\in W_{wtest}} -\log_2 P(w|M)$$

a bird a cat a dog a </s>

Encoding

a	→	1
bird	→	000
cat	→	001
dog	→	010
</s>	→	011

1000100110101011

P(w= “a”) = 0.5 $-\log_2 0.5 = 1$

P(w= “bird”) = 0.125 $-\log_2 0.125 = 3$

P(w= “cat”) = 0.125 $-\log_2 0.125 = 3$

P(w= “dog”) = 0.125 $-\log_2 0.125 = 3$

P(w= “</s>”) = 0.125 $-\log_2 0.125 = 3$

Perplexity

- Equal to two to the power of per-word entropy

$$PPL = 2^H$$

- (Mainly because it makes more impressive numbers)
- For uniform distributions, equal to the size of vocabulary

$$V = 5 \quad H = -\log_2 \frac{1}{5} \qquad PPL = 2^H = 2^{-\log_2 \frac{1}{5}} = 2^{\log_2 5} = 5$$

Coverage

- The percentage of known words in the corpus

a bird a cat a dog a </s>

"dog" is an unknown word

Coverage: 7/8 *

* often omit the sentence-final symbol → 6/7

Smoothing

Context Dependent Smoothing

High frequency word: "Tokyo"

c(Tokyo city) = 40
c(Tokyo is) = 35
c(Tokyo was) = 24
c(Tokyo tower) = 15
c(Tokyo port) = 10
…

Most 2-grams already exist
→ Large λ is better!

Low frequency word: "Tottori"

c(Tottori is) = 2
c(Tottori city) = 1
c(Tottori was) = 0

Many 2-grams will be missing
→ Small λ is better!

- Make the interpolation depend on the context

$$P(w_i|w_{i-1}) = \lambda_{w_{i-1}} P_{ML}(w_i|w_{i-1}) + (1-\lambda_{w_{i-1}}) P(w_i)$$

Witten-Bell Smoothing

- One of the many ways to choose $\lambda_{w_{i-1}}$

$$\lambda_{w_{i-1}} = 1 - \frac{u(w_{i-1})}{u(w_{i-1}) + c(w_{i-1})}$$

$u(w_{i-1})$ = number of unique words after w_{i-1}

- For example:

c(Tottori is) = 2 c(Tottori city) = 1
c(Tottori) = 3 u(Tottori) = 2

$$\lambda_{Tottori} = 1 - \frac{2}{2+3} = 0.6$$

c(Tokyo city) = 40 c(Tokyo is) = 35 ...
c(Tokyo) = 270 u(Tokyo) = 30

$$\lambda_{Tokyo} = 1 - \frac{30}{30+270} = 0.9$$

Absolute Discounting

- Reduce a little bit (*d*) from each count

$$c'(w_{i-1}, w_i) = c(w_{i-1}, w_i) - d$$

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

- For example:

d=0.5
c(Tottori is) = 2
c(Tottori city) = 1

c'(Tottori is) = 1.5
c'(Tottori city) = 0.5
u(Tottori) = 2

$$P(w_i = \text{is} | w_{i-1} = \text{Tottori}) = \frac{1.5}{3} + \frac{2 * 0.5}{3} P(w_i = \text{is})$$

$$P(w_i = \text{city} | w_{i-1} = \text{Tottori}) = \frac{1.5}{3} + \frac{2 * 0.5}{3} P(w_i = \text{city})$$

Kneser-Ney Smoothing

- Currently standard smoothing method
- Similar to abolute discounting, but change $P(w_i)$
- Basic idea:
  - Unigram distribution is used as fall-back for bigram
  - Unigram should mainly give probability to words that will occur in new contexts
  - Count contexts $x(w_i)$ that the new word appears in:

c(Barack Obama) = 50
c(President Obama) = 20

x(Obama) = 2 c(Obama) = 70

c(John Smith) = 7
c(Mary Smith) = 4
c(Fred Smith) = 3
...

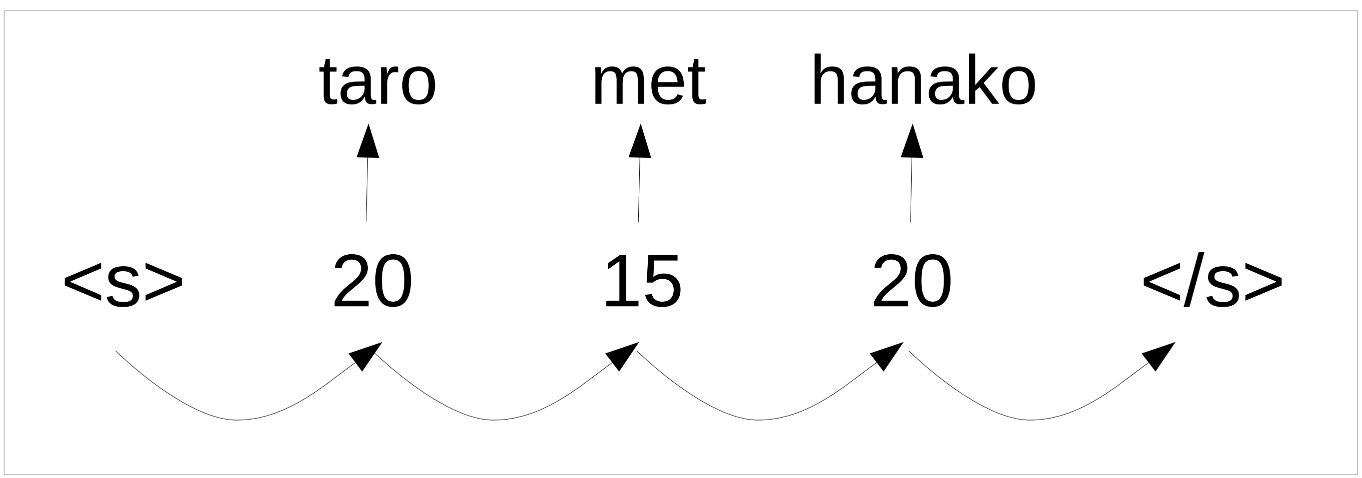
x(Smith) = 20 c(Smith) = 50

Advanced Techniques

Class-based Language Models

- Group words into classes
- Predict the class before predicting words

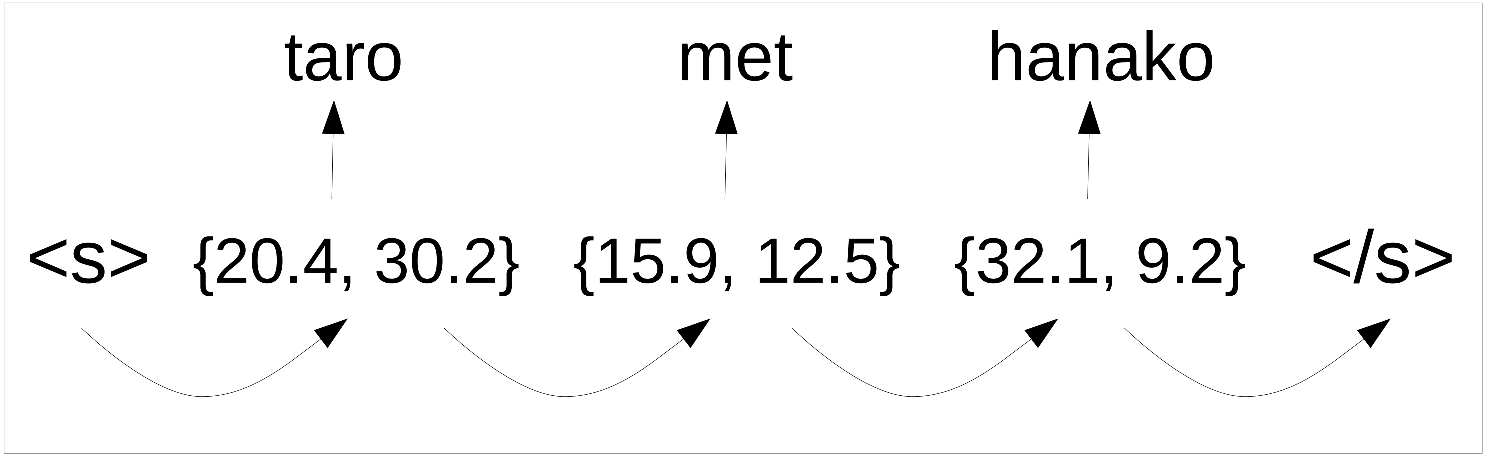
$$P(W)=\prod_{i=1}^{|W|+1} P(c_i|c_{i-1})\,P(w_i|c_i)$$



- Classes are learned automatically
  - Brown clustering most famous method

Continuous-Space Language Models

- Represent each state as a vector of numbers



- Generally learned using neural networks
- Can learn more complicated information

Discriminative Language Models

- Use actual machine translation output and rerank using n-grams

c(visited Hanako)++

c(Taro visited)++

MT

W_1 = Taro visited Hanako   Good!

W_2 = the Taro visited the Hanako   Bad!

c(Taro visited)--   c(the Hanako)--

c(the Taro)--   c(visited the)--

c(visited Hanako) = 1   c(the Taro) = -1   c(visited the) = -1   c(the Hanako) = -1

Assignment

Today's Assignment

- Code to train and test an LM (on the website)

```
cd sentan-01
script/train-bigram.py data/kyoto-train.en > model/bigram.en
script/test-bigram.py model/bigram.en data/kyoto-dev.en
```

- Make a change to the code to improve its entropy
- Any change is OK, EXCEPT:
  - Adding the testing data to the training data
  - Adjusting the number of unknown words V
- Send your code, entropy before/after, a short description of the change, and a “username”
  - Due date: February 3rd, 23:59
  - Address: neubig@is.naist.jp

References