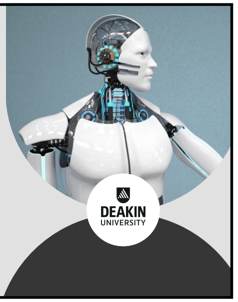
SIT330-770: Natural Language Processing

Week 4 - N-gram Language Models

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env

reda.bouadjenek@deakin.edu.au



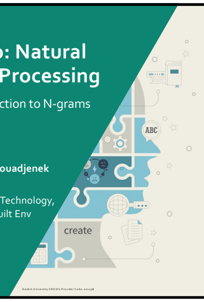
1

SIT330-770: Natural Language Processing

Week 4.1 - Introduction to N-grams

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env



Language Models in NLP

2

What is a Language Model?

- A model to assign a probability to a sentence
  - Machine Translation:
    - P(**high** winds tonight) > P(**large** winds tonight)
  - Spell Correction
    - The office is about fifteen **minuets** from my house!
      - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
  - Speech Recognition
    - P(I saw a van) >> P(eyes awe of an)
  - + Summarization, question, answering, etc., etc.!!

Why?

3

Probabilistic Language Modeling

- Goal: compute the probability of a sentence or sequence of words:

  $P(W) = P(w_1,w_2,w_3,w_4,w_5…w_n)$
- Related task: probability of an upcoming word:

  $P(w_5|w_1,w_2,w_3,w_4)$
- A model that computes either of these:

  $P(W)$ or $P(w_n|w_1,w_2…w_{n-1})$ is called a **language model**.
- Better: **the grammar** But **language model** or **LM** is standard

4

How to compute P(W)

- How to compute this joint probability:
  - P(its, water, is, so, transparent, that)
- Intuition: let's rely on the Chain Rule of Probability

5

Reminder: The Chain Rule

- Recall the definition of conditional probabilities

  $p(B|A) = P(A,B)/P(A)$ Rewriting: **$P(A,B) = P(A)P(B|A)$**
- More variables:

  $P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$
- The Chain Rule in General

  $P(x_1,x_2,x_3,…,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)…P(x_n|x_1,…,x_{n-1})$

6

The Chain Rule applied to compute joint probability of words in sentence

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i \mid w_1 w_2 \dots w_{i-1})$$

P("its water is so transparent") =

P(its) × P(water|its) × P(is|its water)

× P(so|its water is) × P(transparent|its water is so)

7

How to estimate these probabilities

- Could we just count and divide?

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{Count(\text{its water is so transparent that the})}{Count(\text{its water is so transparent that})}$$

- No! Too many possible sentences!
- We'll never see enough data for estimating these

8

Markov Assumption

- Simplifying assumption:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

- Or maybe

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$$

9

Markov Assumption

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i \mid w_{i-k} \dots w_{i-1})$$

- In other words, we approximate each component in the product

$$P(w_i \mid w_1 w_2 \dots w_{i-1}) \approx P(w_i \mid w_{i-k} \dots w_{i-1})$$

10

Simplest case: Unigram model

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i)$$

- Some automatically generated sentences from a unigram model

```
fifth, an, of, futures, the, an, incorporated, a, a, the,
inflation, most, dollars, quarter, in, is, mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the
```

11

Bigram model

- Condition on the previous word:

$$P(w_i \mid w_1 w_2 \dots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

```
texaco, rose, one, in, this, issue, is, pursuing, growth, in, a,
boiler, house, said, mr., gurria, mexico, 's, motion, control,
proposal, without, permission, from, five, hundred, fifty, five, yen

outside, new, car, parking, lot, of, the, agreement, reached

this, would, be, a, record, november
```

12

N-gram models

- We can extend to trigrams, 4-grams, 5-grams
- In general this is an insufficient model of language
  - because language has **long-distance dependencies**:

    "The computer which I had just put into the machine room on the fifth floor crashed."
- But we can often get away with N-gram models

13

SIT330-770: Natural Language Processing

Week 4.2 - Estimating N-gram Probabilities

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env

Language Models in NLP

14

Estimating bigram probabilities

- The Maximum Likelihood Estimate

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

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

15

An example

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

<s> I am Sam </s>

<s> Sam I am </s>

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

$P(\texttt{I} \mid \texttt{<s>}) = \frac{2}{3} = .67 \quad P(\texttt{Sam} \mid \texttt{<s>}) = \frac{1}{3} = .33 \quad P(\texttt{am} \mid \texttt{I}) = \frac{2}{3} = .67$

$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5 \quad P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = .5 \quad P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = .33$

16

More examples: 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

17

Raw bigram counts

- Out of 9,222 sentences

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	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

18

Raw bigram probabilities

- Normalize by unigrams:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

- Result:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
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	0.056	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

19

Bigram estimates of sentence probabilities

P(<s> I want english food </s>) =

P(I|<s>)

× P(want|I)

× P(english|want)

× P(food|english)

× P(</s>|food)

= .000031

20

What kinds of knowledge?

- P(english|want) = .0011
- P(chinese|want) = .0065
- P(to|want) = .66
- P(eat | to) = .28
- P(food | to) = 0
- P(want | spend) = 0
- P (i | <s>) = .25

21

Practical Issues

- We do everything in log space
  - Avoid underflow
  - (also adding is faster than multiplying)

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

22

Language Modeling Toolkits

- SRILM
  - http://www.speech.sri.com/projects/srilm/
- KenLM
  - https://kheafield.com/code/kenlm/

23

Google N-Gram Release, August 2006

AUG 3

All Our N-gram are Belong to You

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of R&D projects,

...

That's why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

24

Google N-Gram Release

- serve as the incoming 92
- serve as the incubator 99
- serve as the independent 794
- serve as the index 223
- serve as the indication 72
- serve as the indicator 120
- serve as the indicators 45
- serve as the indispensable 111
- serve as the indispensible 40
- serve as the individual 234

http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

25

Google Book N-grams

- https://books.google.com/ngrams/

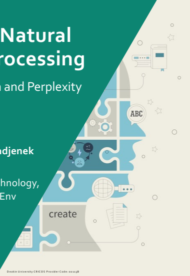
26

SIT330-770: Natural Language Processing

Week 4.3 - Evaluation and Perplexity

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env



Language Models in NLP

27

How to evaluate N-gram models

- "Extrinsic (in-vivo) Evaluation"

To compare models A and B

1. Put each model in a real task
   - Machine Translation, speech recognition, etc.
2. Run the task, get a score for A and for B
   - How many words translated correctly
   - How many words transcribed correctly
3. Compare accuracy for A and B

28

Intrinsic (in-vitro) evaluation

- Extrinsic evaluation not always possible
  - Expensive, time-consuming
  - Doesn't always generalize to other applications
- Intrinsic evaluation: **perplexity**
  - Directly measures language model performance at predicting words.
  - Doesn't necessarily correspond with real application performance
  - But gives us a single general metric for language models
  - Useful for large language models (LLMs) as well as n-grams

29

Training sets and test sets

- We train parameters of our model on a **training set**.
- We test the model's performance on data we haven't seen.
  - A **test set** is an unseen dataset; different from training set.
    - Intuition: we want to measure generalization to unseen data
  - An **evaluation metric** (like **perplexity**) tells us how well our model does on the test set.

30

Choosing training and test sets

- If we're building an LM for a specific task
  - The test set should reflect the task language we want to use the model for
- If we're building a general-purpose model
  - We'll need lots of different kinds of training data
  - We don't want the training set or the test set to be just from one domain or author or language.

31

Training on the test set

We can't allow test sentences into the training set

- Or else the LM will assign that sentence an artificially high probability when we see it in the test set
- And hence assign the whole test set a falsely high probability.
- Making the LM look better than it really is

This is called "Training on the test set"

Bad science!

32

Dev sets

- If we test on the test set many times we might implicitly tune to its characteristics
  - Noticing which changes make the model better.
- So we run on the test set only once, or a few times
- That means we need a third dataset:
  - A **development test set** or, **devset**.
  - We test our LM on the devset until the very end
  - And then test our LM on the **test set** once

33

Intuition of perplexity as evaluation metric: How good is our language model?

- Intuition: A good LM prefers "real" sentences
  - Assign higher probability to "real" or "frequently observed" sentences
  - Assigns lower probability to "word salad" or "rarely observed" sentences?

34

Intuition of perplexity 2: Predicting upcoming words

Claude Shannon

The Shannon Game: **How well can we predict the next word**?

- Once upon a ____
- That is a picture of a ____
- For breakfast I ate my usual ____

time	0.9
dream	0.03
midnight	0.02
...	
and	1e-100

Unigrams are terrible at this game (Why?)

A good LM is one that assigns a higher probability to the next word that actually occurs

35

Intuition of perplexity 3: The best language model is one that best predicts the entire unseen test set

- We said: a good LM is one that assigns a higher probability to the next word that actually occurs.
- Let's generalize to all the words!
  - The best LM assigns high probability to the entire test set.
- When comparing two LMs, A and B
  - We compute P_A(test set) and P_B(test set)
  - The better LM will give a higher probability to (=be less surprised by) the test set than the other LM.

36

Intuition of perplexity 4: Use perplexity instead of raw probability

- Probability depends on size of test set
  - Probability gets smaller the longer the text
  - Better: a metric that is **per-word**, normalized by length
- **Perplexity** is the inverse probability of the test set, normalized by the number of words

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

37

Intuition of perplexity 5: the inverse

Perplexity is the **inverse** probability of the test set, normalized by the number of words

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

(The inverse comes from the original definition of perplexity from cross-entropy rate in information theory)

Probability range is [0,1], perplexity range is [1,∞]

Minimizing perplexity is the same as maximizing probability

38

Intuition of perplexity 6: N-grams

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

Chain rule: $\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i \mid w_1 \ldots w_{i-1})}}$

Bigrams: $\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i \mid w_{i-1})}}$

39

Intuition of perplexity 7: Weighted average branching factor

- Perplexity is also the **weighted average branching factor** of a language.
- **Branching factor**: number of possible next words that can follow any word
- Example: Deterministic language L = {red,blue, green}
  - Branching factor = 3 (any word can be followed by red, blue, green)
- Now assume LM A where each word follows any other word with equal probability ⅓
- Given a test set T = "red red red red blue"
  - $\text{Perplexity}_A(T) = P_A(\text{red red red red blue})^{-1/5} = ((⅓)^5)^{-1/5} = (⅓)^{-1} = 3$
- But now suppose red was very likely in training set, such that for LM B:
  - P(red) = .8 p(green) = .1 p(blue) = .1
- We would expect the probability to be higher, and hence the perplexity to be smaller:
  - $\text{Perplexity}_B(T) = P_B(\text{red red red red blue})^{-1/5}$

40

Holding test set constant: Lower perplexity = better language model

- Training 38 million words, test 1.5 million words, WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

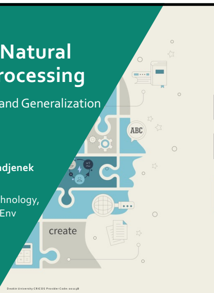
41

SIT330-770: Natural Language Processing

Week 4.4 - Sampling and Generalization

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env



Language Models in NLP

42

The Shannon (1948) Visualization Method
Sample words from an LM


Claude Shannon

- Unigram:

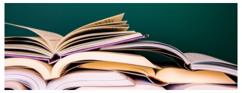
REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE.

- Bigram:

THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED.

43

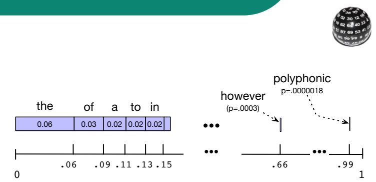
How Shannon sampled those words in 1948



"Open a book at random and select a letter at random on the page. This letter is recorded. The book is then opened to another page and one reads until this letter is encountered. The succeeding letter is then recorded. Turning to another page this second letter is searched for and the succeeding letter recorded, etc."

44

Sampling a word from a distribution



45

Visualizing Bigrams the Shannon Way

Choose a random bigram (<s>, w) according to its probability p(w|<s>)

Now choose a random bigram (w, x) according to its probability p(x|w)

And so on until we choose </s>

Then string the words together

```
<s> I
    I want
      want to
           to eat
              eat Chinese
                  Chinese food
                          food  </s>
I want to eat Chinese food
```

46

Note: there are other sampling methods

- Used for neural language models
- Many of them avoid generating words from the very unlikely tail of the distribution
- We'll discuss when we get to neural LM decoding:
  - Temperature sampling
  - Top-k sampling
  - Top-p sampling

47

Approximating Shakespeare

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; –It cannot be but so.

48

Shakespeare as corpus

N=884,647 tokens, V=29,066

Shakespeare produced 300,000 bigram types out of V^2= 844 million possible bigrams.

- So 99.96% of the possible bigrams were never seen (have zero entries in the table)
- That sparsity is even worse for 4-grams, explaining why our sampling generated actual Shakespeare.

49

The Wall Street Journal is not Shakespeare

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

50

Can you guess the author? These 3-gram sentences are sampled from an LM trained on who?

1) They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and gram Brazil on market conditions
2) This shall forbid it should be branded, if renown made it empty.
3) "You are uniformly charming!" cried he, with a smile of associating and now and then I bowed and they perceived a chaise and four to wish for.

51

Choosing training data

- If task-specific, use a training corpus that has a similar genre to your task.
  - If legal or medical, need lots of special-purpose documents
- Make sure to cover different kinds of dialects and speaker/authors.
  - Example: *African-American Vernacular English (AAVE)*
    - One of many varieties that can be used by African Americans and others
    - Can include the auxiliary verb **finna** that marks immediate future tense:
    - "My phone finna die"

52

The perils of overfitting

- N-grams only work well for word prediction if the test corpus looks like the training corpus
  - But even when we try to pick a good training corpus, the test set will surprise us!
  - We need to train robust models that generalize!
- One kind of generalization: **Zeros**
  - Things that don't ever occur in the training set
    - But occur in the test set

53

Zeros

- Training set:
  - ... ate lunch
  - ... ate dinner
  - ... ate a
  - ... ate the

P("breakfast" | ate) = 0

- Test set
  - ... ate lunch
  - ... ate breakfast

54

Zero probability bigrams

- Bigrams with zero probability
  - Will hurt our performance for texts where those words appear!
  - And mean that we will assign 0 probability to the test set!
- And hence we cannot compute perplexity (can't divide by 0)!

55

SIT330-770: Natural Language Processing

Week 4.5 - Smoothing: Add-one (Laplace) smoothing

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env

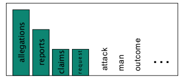
Language Models in NLP

56

The intuition of smoothing (from Dan Klein)

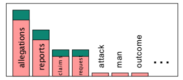
- When we have sparse statistics:

P(w | denied the)
3 allegations
2 reports
1 claims
1 request
7 total



- Steal probability mass to generalize better

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
7 total



57

Add-one estimation

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

58

Maximum Likelihood Estimates

- The maximum likelihood estimate
  - of some parameter of a model M from a training set T
  - maximizes the likelihood of the training set T given the model M
- Suppose the word "bagel" occurs 400 times in a corpus of a million words
- What is the probability that a random word from some other text will be "bagel"?
- MLE estimate is 400/1,000,000 = .0004
- This may be a bad estimate for some other corpus
  - But it is the **estimate** that makes it **most likely** that "bagel" will occur 400 times in a million word corpus.

59

Berkeley Restaurant Corpus: Laplace smoothed bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	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

60

Laplace-smoothed bigrams

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

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

61

Reconstituted counts

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n)+1] \times C(w_{n-1})}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

62

Compare with raw bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	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

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

63

Add-1 estimation is a blunt instrument

- So add-1 isn't used for N-grams:
  - We'll see better methods
- But add-1 is used to smooth other NLP models
  - For text classification
  - In domains where the number of zeros isn't so huge.

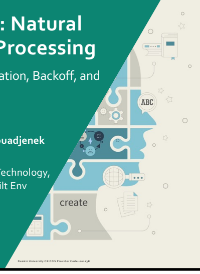
64

SIT330-770: Natural Language Processing

Week 4.6 - Interpolation, Backoff, and Web-Scale LMs

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env



Language Models in NLP

65

Backoff and Interpolation

- Sometimes it helps to use **less** context
  - Condition on less context for contexts you haven't learned much about
- **Backoff:**
  - use trigram if you have good evidence,
  - otherwise bigram, otherwise unigram
- **Interpolation:**
  - mix unigram, bigram, trigram
- Interpolation works better

66

Linear Interpolation

- Simple interpolation

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

- Lambdas conditional on context:

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

67

How to set the lambdas?

- Use a **held-out** corpus

Training Data	Held-Out Data	Test Data

- Choose λs to maximize the probability of held-out data:
  - Fix the N-gram probabilities (on the training data)
  - Then search for λs that give largest probability to held-out set:

$$\log P(w_1 \dots w_n | M(\lambda_1 \dots \lambda_k)) = \sum_i \log P_{M(\lambda_1 \dots \lambda_k)}(w_i | w_{i-1})$$

68

Unknown words: Open versus closed vocabulary tasks

- If we know all the words in advanced
  - Vocabulary V is fixed
  - Closed vocabulary task
- Often we don't know this
  - **Out Of Vocabulary** = OOV words
  - Open vocabulary task
- Instead: create an unknown word token <UNK>
  - Training of <UNK> probabilities
    - Create a fixed lexicon L of size V
    - At text normalization phase, any training word not in L changed to <UNK>
    - Now we train its probabilities like a normal word
  - At decoding time
    - If text input: Use UNK probabilities for any word not in training

69

Huge web-scale n-grams

- How to deal with, e.g., Google N-gram corpus
- Pruning
  - Only store N-grams with count > threshold.
    - Remove singletons of higher-order n-grams
  - Entropy-based pruning
- Efficiency
  - Efficient data structures like tries
  - Bloom filters: approximate language models
  - Store words as indexes, not strings
    - Use Huffman coding to fit large numbers of words into two bytes
  - Quantize probabilities (4-8 bits instead of 8-byte float)

70

Smoothing for Web-scale N-grams

- "Stupid backoff" (Brants *et al.* 2007)
- No discounting, just use relative frequencies

$$S(w_i | w_{i-k+1}^{i-1}) = \begin{cases} \dfrac{\text{count}(w_{i-k+1}^{i})}{\text{count}(w_{i-k+1}^{i-1})} & \text{if } \text{count}(w_{i-k+1}^{i}) > 0 \\ 0.4 S(w_i | w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

$$S(w_i) = \frac{\text{count}(w_i)}{N}$$

71

N-gram Smoothing Summary

- Add-1 smoothing:
  - OK for text categorization, not for language modeling
- The most commonly used method:
  - Extended Interpolated Kneser-Ney
- For very large N-grams like the Web:
  - Stupid backoff

72

Advanced Language Modeling

- Discriminative models:
  - choose n-gram weights to improve a task, not to fit the training set
- Parsing-based models
- Caching Models
  - Recently used words are more likely to appear

$$P_{CACHE}(w \mid history) = \lambda P(w_i \mid w_{i-2} w_{i-1}) + (1-\lambda)\frac{c(w \in history)}{\mid history \mid}$$

  - These perform very poorly for speech recognition (why?)

73

SIT330-770: Natural Language Processing

Week 4.7 - Kneser-Ney Smoothing

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env

Language Models in NLP

74

Absolute discounting: just subtract a little from each count

- Suppose we wanted to subtract a little from a count of 4 to save probability mass for the zeros
- How much to subtract ?
- Church and Gale (1991)'s clever idea
- Divide up 22 million words of AP Newswire
  - Training and held-out set
  - for each bigram in the training set
  - see the actual count in the held-out set!
- It sure looks like c* = (c - .75)

Bigram count in training	Bigram count in heldout set
0	.0000270
1	0.448
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21
9	8.26

75

Absolute Discounting Interpolation

- Save ourselves some time and just subtract 0.75 (or some d)!

$$P_{\text{AbsoluteDiscounting}}(w_i \mid w_{i-1}) = \underbrace{\frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})}}_{\text{discounted bigram}} + \underbrace{\lambda(w_{i-1})}_{\text{Interpolation weight}} \underbrace{P(w)}_{\text{unigram}}$$

- (Maybe keeping a couple extra values of d for counts 1 and 2)
- But should we really just use the regular unigram P(w)?

76

Kneser-Ney Smoothing I

- Better estimate for probabilities of lower-order unigrams!
  - Shannon game: *I can't see without my reading* ___glasses___?
  - "Kong" turns out to be more common than "glasses"
  - … but "Kong" always follows "Hong"
- The unigram is useful exactly when we haven't seen this bigram!
- Instead of P(w): "How likely is w"
- $P_{continuation}(w)$: "How likely is w to appear as a novel continuation?
  - For each word, count the number of bigram types it completes
  - Every bigram type was a novel continuation the first time it was seen

$$P_{CONTINUATION}(w) \propto \left|\{w_{i-1} : c(w_{i-1}, w) > 0\}\right|$$

77

Kneser-Ney Smoothing II

- How many times does w appear as a novel continuation:

$$P_{CONTINUATION}(w) \propto \left|\{w_{i-1} : c(w_{i-1}, w) > 0\}\right|$$

- Normalized by the total number of word bigram types

$$\left|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}\right|$$

$$P_{CONTINUATION}(w) = \frac{\left|\{w_{i-1} : c(w_{i-1}, w) > 0\}\right|}{\left|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}\right|}$$

78

Kneser-Ney Smoothing III

- Alternative metaphor: The number of # of word types seen to precede w

$$|\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

- Normalized by the # of words preceding all words:

$$P_{CONTINUATION}(w) = \frac{|\{w_{i-1} : c(w_{i-1}, w) > 0\}|}{\sum_{w'} |\{w'_{i-1} : c(w'_{i-1}, w') > 0\}|}$$

- A frequent word (Kong) occurring in only one context (Hong) will have a low continuation probability

79

Deakin University CRICOS Provider Code: 00113B

79

Kneser-Ney Smoothing IV

$$P_{KN}(w_i \mid w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1}) P_{CONTINUATION}(w_i)$$

λ is a normalizing constant; the probability mass we've discounted

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})} |\{w : c(w_{i-1}, w) > 0\}|$$

the normalized discount

The number of word types that can follow w_{i-1}
= # of word types we discounted
= # of times we applied normalized discount

80

Deakin University CRICOS Provider Code: 00113B

80

Kneser-Ney Smoothing: Recursive formulation

$$P_{KN}(w_i \mid w_{i-n+1}^{i-1}) = \frac{\max(c_{KN}(w_{i-n+1}^{i}) - d, 0)}{c_{KN}(w_{i-n+1}^{i-1})} + \lambda(w_{i-n+1}^{i-1}) P_{KN}(w_i \mid w_{i-n+2}^{i-1})$$

$$c_{KN}(\bullet) = \begin{cases} count(\bullet) & \text{for the highest order} \\ continuationcount(\bullet) & \text{for lower order} \end{cases}$$

Continuation count = Number of unique single word contexts for •

81

Deakin University CRICOS Provider Code: 00113B

81

SIT330-770: Natural Language Processing

Week 4.8 – The Spelling Correction Task

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env

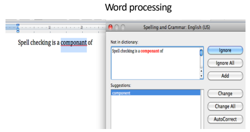
Language Models in NLP

82

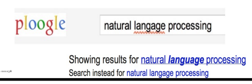
82

Applications for spelling correction

Word processing



Phones



83

Deakin University CRICOS Provider Code: 00113B

83

Spelling Tasks

- Spelling Error Detection
- Spelling Error Correction:
  - Autocorrect
    - hte→the
  - Suggest a correction
  - Suggestion lists

84

Deakin University CRICOS Provider Code: 00113B

84

Types of spelling errors

- Non-word Errors
  - *graffe* → *giraffe*
- Real-word Errors
  - Typographical errors
    - *three* → *there*
  - Cognitive Errors (homophones)
    - *piece* → *peace*,
    - *too* → *two*
    - *your* → *you're*
- Non-word correction was historically mainly context insensitive
- Real-word correction almost needs to be context sensitive

85

Rates of spelling errors

- Depending on the application, ~1–20% error rates
  - **26%**: Web queries Wang *et al.* 2003
  - **13%**: Retyping, no backspace: Whitelaw *et al.* English&German
  - **7%**: Words corrected retyping on phone-sized organizer
  - **2%**: Words uncorrected on organizer Soukoreff &MacKenzie 2003
  - **1-2%**: Retyping: Kane and Wobbrock 2007, Gruden et al. 1983

86

Non-word spelling errors

- Non-word spelling error detection:
  - Any word not in a ***dictionary*** is an error
  - The larger the dictionary the better … up to a point
  - (The Web is full of mis-spellings, so the Web isn't necessarily a great dictionary …)
- Non-word spelling error correction:
  - Generate ***candidates***: real words that are similar to error
  - Choose the one which is best:
    - Shortest weighted edit distance
    - Highest noisy channel probability

87

Real word & non-word spelling errors

- For each word *w*, generate candidate set:
  - Find candidate words with similar ***pronunciations***
  - Find candidate words with similar ***spellings***
  - Include *w* in candidate set
- Choose best candidate
  - Noisy Channel view of spell errors
  - Context-sensitive – so have to consider whether the surrounding words "make sense"
  - *Flying form Heathrow to LAX* → *Flying from Heathrow to LAX*

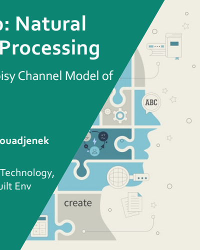
88

SIT330-770: Natural Language Processing

Week 4.9 – The Noisy Channel Model of Spelling

Dr. Mohamed Reda Bouadjenek

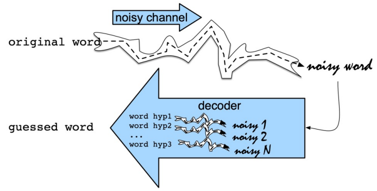
School of Information Technology, Faculty of Sci Eng & Built Env



Language Models in NLP

89

Noisy Channel Intuition



90

Noisy Channel

- We see an observation x of a misspelled word
- Find the correct word $\hat{w}$

$$\begin{aligned}\hat{w} &= \underset{w\in V}{\operatorname{argmax}}\, P(w \mid x)\\ &= \underset{w\in V}{\operatorname{argmax}}\, \frac{P(x \mid w)P(w)}{P(x)}\\ &= \underset{w\in V}{\operatorname{argmax}}\, \underbrace{P(x \mid w)}_{\text{The likelihood or channel model}}\,\underbrace{P(w)}_{\text{Prior probability or Language Model}}\end{aligned}$$

91

Non-word spelling error example

acress

92

Candidate generation

- Words with similar spelling
  - Small edit distance to error
- Words with similar pronunciation
  - Small distance of pronunciation to error

93

Candidate Testing: Damerau-Levenshtein edit distance

- Minimal edit distance between two strings, where edits are:
  - Insertion
  - Deletion
  - Substitution
  - Transposition of two adjacent letters

94

Words within 1 of acress

Error	Candidate Correction	Correct Letter	Error Letter	Type
acress	actress	t	-	deletion
acress	cress	-	a	insertion
acress	caress	ca	ac	transposition
acress	access	c	r	substitution
acress	across	o	e	substitution
acress	acres	-	s	insertion
acress	acres	-	s	insertion

95

Candidate generation

- 80% of errors are within edit distance 1
- Almost all errors within edit distance 2
- Also allow insertion of **space** or **hyphen**
  - thisidea → this idea
  - inlaw → in-law
- Can also allow merging words
  - data base → database
  - For short texts like a query, can just regard whole string as one item from which to produce edits

96

Language Model

- Use any of the language modeling algorithms we've learned
- Unigram, bigram, trigram
- Web-scale spelling correction
  - Stupid backoff

97

Unigram Prior probability

- Counts from 404,253,213 words in Corpus of Contemporary English (COCA)

word	Frequency of word	*P(w)*
actress	9,321	.0000230573
cress	220	.0000005442
caress	686	.0000016969
access	37,038	.0000916207
across	120,844	.0002989314
acres	12,874	.0000318463

98

Channel model probability

- **Error model probability, Edit probability**
- *Kernighan, Church, Gale 1990*

- *Misspelled word* $x = x_1, x_2, x_3 \dots x_m$
- *Correct word* $w = w_1, w_2, w_3, \dots, w_n$

- $P(x|w)$ = probability of the edit
  - (deletion/insertion/substitution/transposition)

99

Computing error probability: confusion "matrix"

```
del[x,y]:    count(xy typed as x)
ins[x,y]:    count(x typed as xy)
sub[x,y]:    count(y typed as x)
trans[x,y]:  count(xy typed as yx)
```

Insertion and deletion conditioned on previous character

100

Confusion matrix for substitution

sub[X, Y] = Substitution of X (incorrect) for Y (correct)

Y (correct)

X	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	7	1	342	0	0	2	118	0	1	0	0	3	76	0	0	1	35	9	9	0	1	0	5	0
b	0	0	9	9	2	2	3	1	0	0	0	5	11	5	0	10	0	0	2	1	0	0	8	0	0	0
c	6	5	0	16	0	9	5	0	0	0	1	0	7	9	1	10	2	5	39	40	1	3	7	1	1	0
d	1	10	13	0	12	0	5	5	0	0	2	3	7	3	0	1	0	43	30	22	0	0	4	0	2	0
e	388	0	3	11	0	2	2	0	89	0	0	3	0	5	93	0	0	14	12	6	15	0	1	0	18	0
f	0	15	0	3	1	0	5	2	0	0	0	3	4	1	0	0	0	6	4	12	0	0	2	0	0	0
g	4	1	11	11	9	2	0	0	0	1	1	3	0	0	2	1	3	5	13	21	0	0	1	0	3	0
h	1	8	0	3	0	0	0	0	0	0	2	0	12	14	2	3	0	3	1	11	0	0	2	0	0	0
i	103	0	0	0	146	0	1	0	0	0	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0
j	0	1	1	9	0	0	1	0	0	0	0	2	1	0	0	0	0	0	5	0	0	0	0	0	0	0
k	1	2	8	4	1	1	2	5	0	0	0	0	5	0	2	0	0	0	6	0	0	0	4	0	0	3
l	2	10	1	4	0	4	5	6	13	0	1	0	0	14	2	5	0	11	10	2	0	0	0	0	0	0
m	1	3	7	8	0	2	0	6	0	0	4	4	0	180	0	6	0	0	9	15	13	3	2	2	3	0
n	2	7	6	5	3	0	1	19	1	0	4	35	78	0	0	7	0	28	5	7	0	0	1	2	0	2
o	91	1	1	3	116	0	0	0	25	0	2	0	0	0	0	14	0	2	4	14	39	0	0	0	18	0
p	0	11	1	2	0	6	5	0	2	9	0	2	7	6	15	0	0	1	3	6	0	4	1	0	0	0
q	0	0	1	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	14	0	30	12	2	2	8	2	0	5	8	4	20	1	14	0	0	12	22	4	0	0	1	0	0
s	11	8	27	33	35	4	0	1	0	1	0	27	0	6	1	7	0	14	0	15	0	0	5	3	20	1
t	3	4	9	42	7	5	19	5	0	1	0	14	9	5	5	6	0	11	37	0	0	2	19	0	7	6
u	20	0	0	0	44	0	0	0	64	0	0	0	0	2	43	0	0	4	0	0	0	0	2	0	8	0
v	0	0	7	0	0	3	0	0	0	0	0	1	0	0	1	0	0	0	8	3	0	0	0	0	0	0
w	2	2	1	0	1	0	0	2	0	0	1	0	0	0	0	7	0	6	3	3	1	0	0	0	0	0
x	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0
y	0	0	2	0	15	0	1	7	15	0	0	0	2	0	6	1	0	7	36	8	5	0	0	1	0	0
z	0	0	0	7	0	0	0	0	0	0	0	7	5	0	0	0	0	2	21	3	0	0	0	0	3	0

101

Generating the confusion matrix

- Peter Norvig's list of errors
- Peter Norvig's list of counts of single-edit errors
  - All Peter Norvig's ngrams data links: http://norvig.com/ngrams/

102

Channel model

$$P(x|w) = \begin{cases} \frac{\text{del}[w_{i-1}, w_i]}{\text{count}[w_{i-1} w_i]}, & \text{if deletion} \\ \frac{\text{ins}[w_{i-1}, x_i]}{\text{count}[w_{i-1}]}, & \text{if insertion} \\ \frac{\text{sub}[x_i, w_i]}{\text{count}[w_i]}, & \text{if substitution} \\ \frac{\text{trans}[w_i, w_{i+1}]}{\text{count}[w_i w_{i+1}]}, & \text{if transposition} \end{cases}$$

103

Noisy channel probability for `acress`

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|w)
actress	t	-	c|ct	.000117
cress	-	a	a|#	.00000144
caress	ca	ac	ac|ca	.00000164
access	c	r	r|c	.000000209
across	o	e	e|o	.0000093
acres	-	s	es|e	.0000321
acres	-	s	ss|s	.0000342

104

Noisy channel probability for `acress`

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|w)	P(w)	10⁹ * P(x|w)* P(w)
actress	t	-	c|ct	.000117	.0000231	2.7
cress	-	a	a|#	.00000144	.000000544	.00078
caress	ca	ac	ac|ca	.00000164	.00000170	.0028
access	c	r	r|c	.000000209	.0000916	.019
across	o	e	e|o	.0000093	.000299	2.8
acres	-	s	es|e	.0000321	.0000318	1.0
acres	-	s	ss|s	.0000342	.0000318	1.0

105

Noisy channel probability for `acress`

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|w)	P(w)	10⁹ * P(x|w)* P(w)
actress	t	-	c|ct	.000117	.0000231	2.7
cress	-	a	a|#	.00000144	.000000544	.00078
caress	ca	ac	ac|ca	.00000164	.00000170	.0028
access	c	r	r|c	.000000209	.0000916	.019
across	**o**	**e**	**e|o**	**.0000093**	**.000299**	**2.8**
acres	-	s	es|e	.0000321	.0000318	1.0
acres	-	s	ss|s	.0000342	.0000318	1.0

106

Using a bigram language model

- "a stellar and versatile **acress** whose combination of sass and glamour…"
- Counts from the Corpus of Contemporary American English with add-1 smoothing

- `P(actress|versatile)=.000021 P(whose|actress) = .0010`
- `P(across|versatile) =.000021 P(whose|across) = .000006`

- `P("versatile actress whose") = .000021*.0010 = 210 x10-10`
- `P("versatile across whose") = .000021*.000006 = 1 x10-10`

107

Using a bigram language model

- "a stellar and versatile **acress** whose combination of sass and glamour…"
- Counts from the Corpus of Contemporary American English with add-1 smoothing

- `P(actress|versatile)=.000021 P(whose|actress) = .0010`
- `P(across|versatile) =.000021 P(whose|across) = .000006`

- `P("versatile actress whose") = .000021*.0010 = 210 x10-10`
- `P("versatile across whose") = .000021*.000006 = 1 x10-10`

108

Evaluation

- Some spelling error test sets
  - Wikipedia's list of common English misspelling
  - Aspell filtered version of that list
  - Birkbeck spelling error corpus
  - Peter Norvig's list of errors (includes Wikipedia and Birkbeck, for training or testing)

109

109

SIT330-770: Natural Language Processing

Week 4.10 – Real-word spelling errors

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env

Language Models in NLP

110

110

Real-word spelling errors

- …leaving in about fifteen ***minuets*** to go to her house.
- The design ***an*** construction of the system…
- Can they ***lave*** him my messages?
- The study was conducted mainly ***be*** John Black.

- 25-40% of spelling errors are real words Kukich 1992

111

111

Solving real-world spelling errors

- For each word in sentence (phrase, query …)
  - Generate *candidate set*
    - the word itself
    - all single-letter edits that are English words
    - words that are homophones
    - (all of this can be pre-computed!)
- Choose best candidates
    - Noisy channel model

112

112

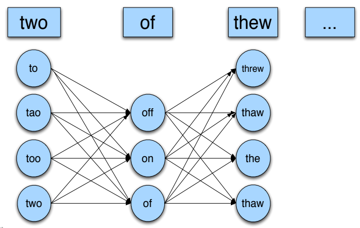
Noisy channel for real-word spell correction

- Given a sentence $w_1, w_2, w_3, \dots, w_n$
- Generate a set of candidates for each word w_i
  - Candidate(w_1) = {w_1, w'_1, w''_1, w'''_1, …}
  - Candidate(w_2) = {w_2, w'_2, w''_2, w'''_2, …}
  - Candidate(w_n) = {w_n, w'_n, w''_n, w'''_n, …}
- Choose the sequence W that maximizes P(W)

113

113

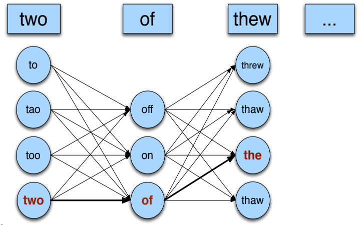
Noisy channel for real-word spell correction



114

114

Noisy channel for real-word spell correction



115

Simplification: One error per sentence

- Out of all possible sentences with one word replaced
  - w_1, w''_2, w_3, w_4 two **off** thew
  - w_1, w_2, w'_3, w_4 two of **the**
  - w'''_1, w_2, w_3, w_4 **too** of thew
  - …
- Choose the sequence W that maximizes P(W)

116

Where to get the probabilities

- Language model
  - Unigram
  - Bigram
  - etc.
- Channel model
  - Same as for non-word spelling correction
  - Plus need probability for no error, P(w|w)

117

Probability of no error

- What is the channel probability for a correctly typed word?
- P("the"|"the")
  - If you have a big corpus, you can estimate this percent correct
- But this value depends strongly on the application
  - .90 (1 error in 10 words)
  - .95 (1 error in 20 words)
  - .99 (1 error in 100 words)

118

Peter Norvig's "thew" example

x	w	x|w	P(x|w)	P(w)	10^9 P(x|w)P(w)
thew	the	ew|e	0.000007	0.02	144
thew	thew		0.95	0.00000009	90
thew	thaw	e|a	0.001	0.0000007	0.7
thew	threw	h|hr	0.000008	0.000004	0.03
thew	thwe	ew|we	0.000003	0.00000004	0.0001

119

SIT330-770: Natural Language Processing

Week 4.11 – State of the art noisy systems

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env

Language Models in NLP

120

HCI issues in spelling

- If very confident in correction
  - Autocorrect
- Less confident
  - Give the best correction
- Less confident
  - Give a correction list
- Unconfident
  - Just flag as an error

121

State of the art noisy channel

- We never just multiply the prior and the error model
- Independence assumptions → probabilities not commensurate
- Instead: Weigh them

$$\hat{w} = \underset{w \in V}{\operatorname{argmax}}\, P(x \mid w) P(w)^{\lambda}$$

- Learn λ from a development test set

122

Phonetic error model

- Metaphone, used in GNU aspell
  - Convert misspelling to metaphone pronunciation
    - "Drop duplicate adjacent letters, except for C."
    - "If the word begins with 'KN', 'GN', 'PN', 'AE', 'WR', drop the first letter."
    - "Drop 'B' if aver 'M' and if it is at the end of the word"
    - …
- Find words whose pronunciation is 1-2 edit distance from misspelling's
- Score result list
  - Weighted edit distance of candidate to misspelling
  - Edit distance of candidate pronunciation to misspelling pronunciation

123

Improvements to channel model

- Allow richer edits (Brill and Moore 2000)
  - ent→ant
  - ph→f
  - le→al
- Incorporate pronunciation into channel (Toutanova and Moore 2002)
- Incorporate device into channel
  - Not all Android phones need have the same error model
  - But spell correction may be done at the system level

124

Channel model

- Factors that could influence p(misspelling|word)
  - The source letter
  - The target letter
  - Surrounding letters
  - The position in the word
  - Nearby keys on the keyboard
  - Homology on the keyboard
  - Pronunciations
  - Likely morpheme transformations

125

Nearby keys

126

Classifier-based methods for real-word spelling correction

- Instead of just channel model and language model
- Use many features in a classifier (next lecture).
- Build a classifier for a specific pair like:

  whether/weather

  - "cloudy" within +- 10 words
  - ___ to VERB
  - ___ or not

127