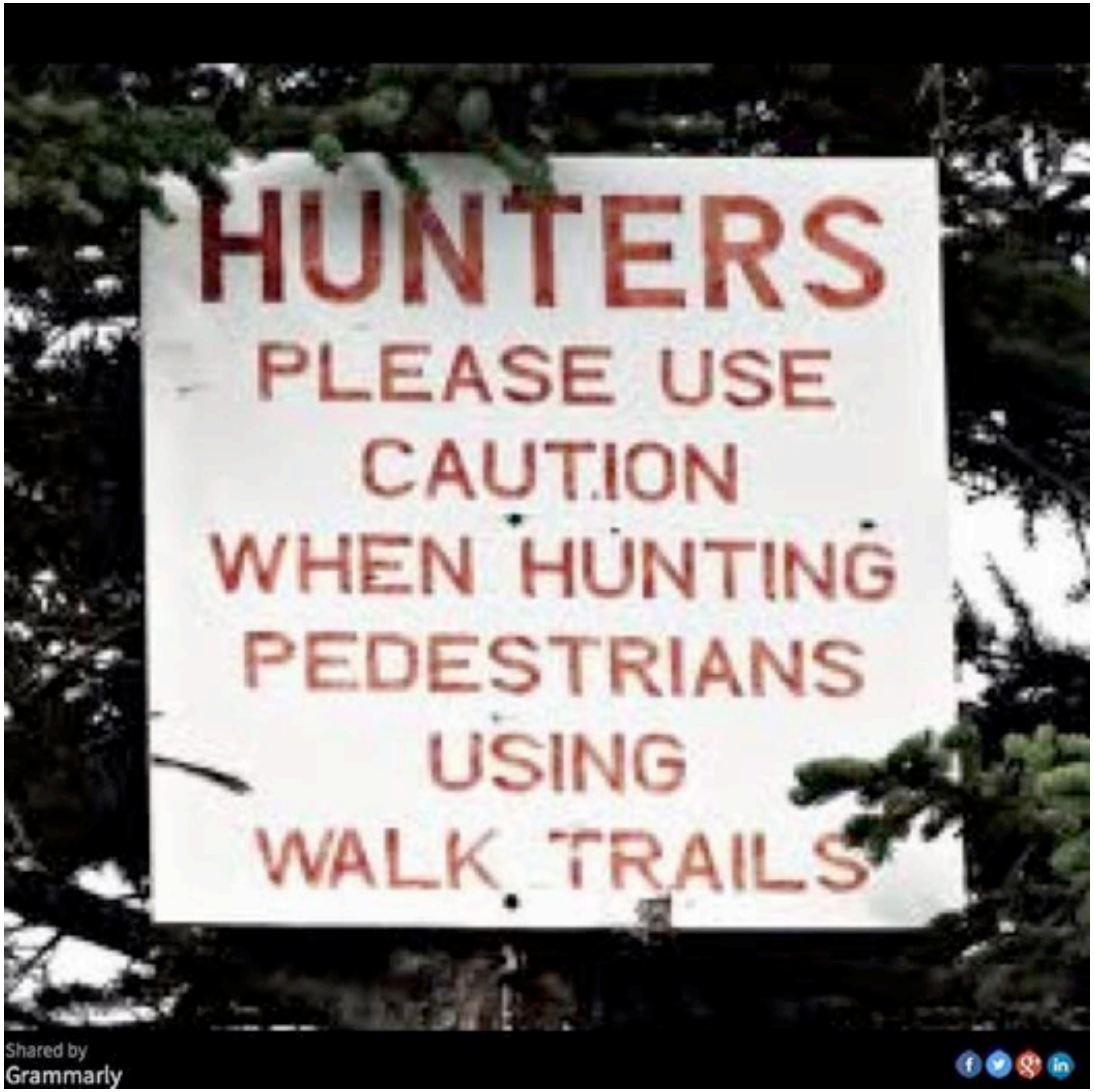
(today we are assuming sentence segmentation)



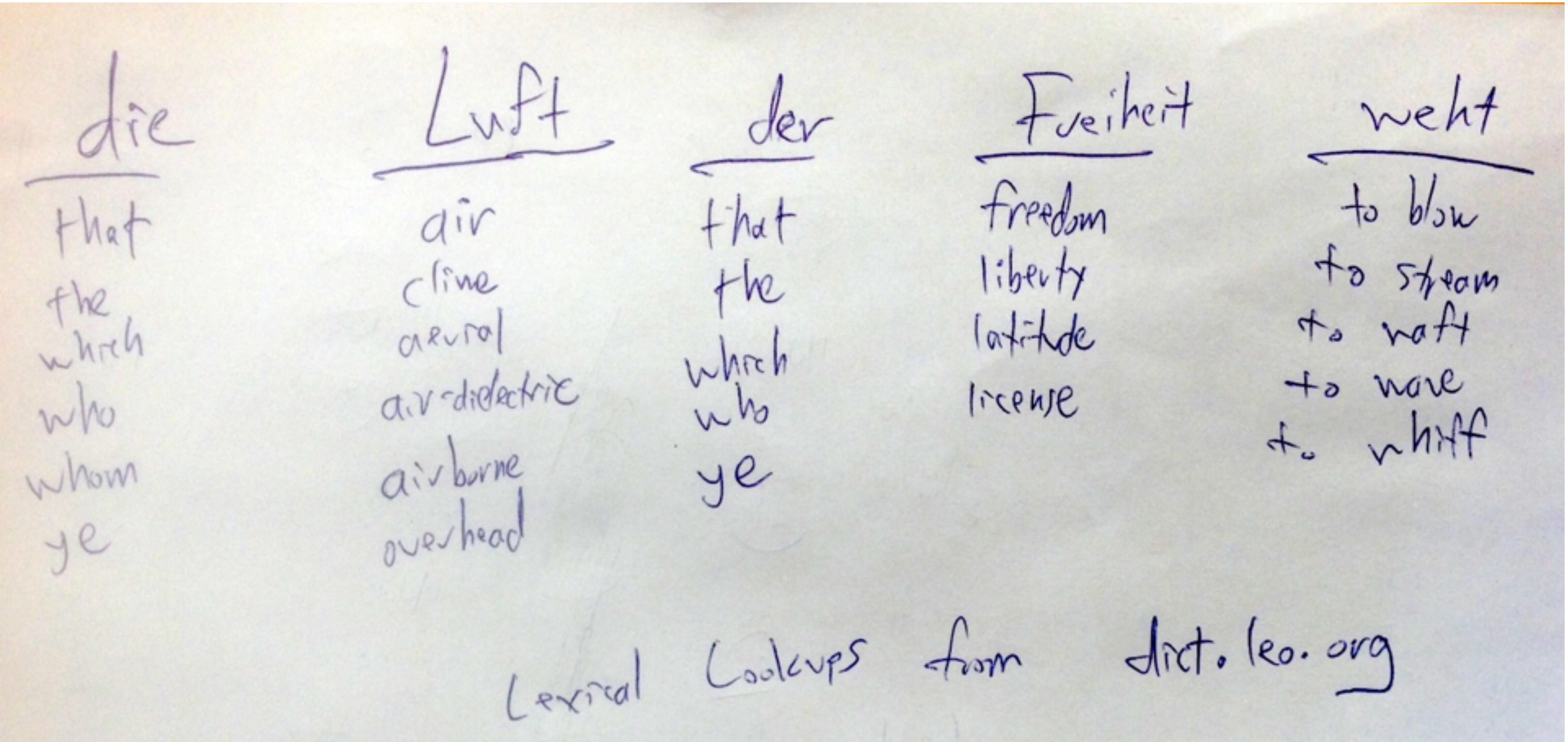
Your TA: David Belanger

- http://people.cs.umass.edu/~belanger/
- I am a third year PhD student advised by Professor Andrew McCallum. Before that, I was an Associate Scientist in the Speech, Language, and Multimedia Department at Raytheon BBN Technologies, where I worked on multilingual optical handwriting recognition. I received a B.A. in mathematics from Harvard University, where I worked with Eric Dunham and Jim Rice. We developed methods for numerically simulating earthquake ruptures along rough fault surfaces. Currently, my research focus is on machine learning and natural language processing. This summer, I am interning with Sham Kakade at Microsoft Research New England.
- Office hours TBA

- Announcements
  - Exercise 1 grades on Moodle
  - Piazza: post questions for benefit of everyone
  - PS1 out later today. Due next Wed, 11:59pm
    - Try it and post any questions asap!
  - Exercises will be posted end of week
  - Grading and policies up on website

- Today
  - Exercise 2: in-class exercise and turn-in
  - Language models

- Group exercise: translate to English sentence



die	Luft	der	Freiheit	weht
that	air	that	freedom	to blow
the	clime	the	liberty	to stream
which	aerial	which	latitude	to waft
who	air-dielectric	who	license	to wave
whom	airborne	ye		to whiff
ye	overhead			

Lexical Lookups from dict.leo.org

Lecture 3: Language Models

Intro to NLP, CS585, Fall 2014

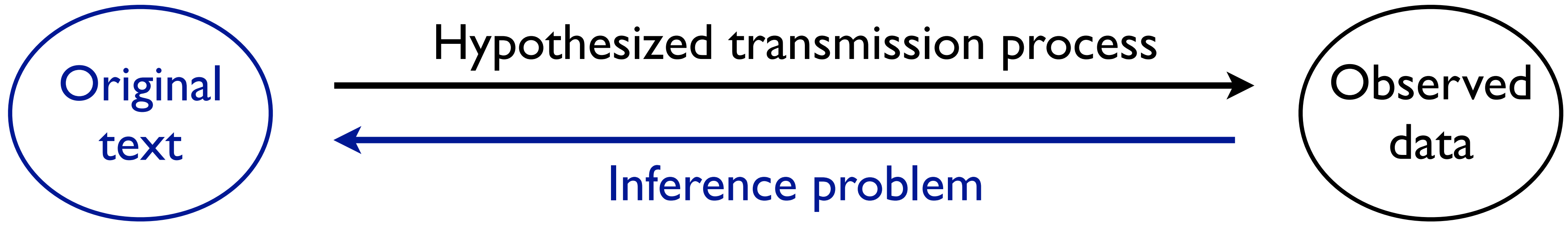
http://people.cs.umass.edu/~brenocon/inlp2014/

Brendan O'Connor (http://brenocon.com)

Some material borrowed from Andrew McCallum and Dan Klein

Bayes Rule for *text* inference

Noisy channel model



Original text

Hypothesized transmission process

Inference problem

Observed data

Codebreaking

P(plaintext | encrypted text) ∝ P(encrypted text | plaintext) **P(plaintext)**

Speech recognition

P(text | acoustic signal) ∝ P(acoustic signal | text) **P(text)**

Optical character recognition

P(text | image) ∝ P(image | text) **P(text)**

Machine translation

P(target text | source text) ∝ P(source text | target text) **P(target text)**

Language Models for Sentences

Machine translation

P(target text | source text) $\propto$ P(source text | target text) **P(target text)**

- We want a model that gives: probability of any sentence $P(w_1 \ldots w_T)$
- Idea: “good” sentences should have higher probability
- Training data: large sample of many tokenized sentences (each is a word sequence)
- Test data: on new sentences, is probability high?

Chomsky (*Syntactic Structures*, 1957):

Second, the notion “grammatical” cannot be identified with “meaningful” or “significant” in any semantic sense. Sentences (1) and (2) are equally nonsensical, but any speaker of English will recognize that only the former is grammatical.

(1) Colorless green ideas sleep furiously.
(2) Furiously sleep ideas green colorless.

. . . Third, the notion “grammatical in English” cannot be identified in any way with the notion “high order of statistical approximation to English”. It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, in any statistical model for grammaticalness, these sentences will be ruled out on identical grounds as equally ‘remote’ from English. Yet (1), though nonsensical, is grammatical, while (2) is not.

Language Models for Sentences

- Finite vocabulary
- Goal: define probability distribution over an infinite set of strings (word sequences). $P(w_1 \ldots w_T)$ for any $(w_1 \ldots w_T)$ of any length
- w_T is always an "END" symbol.
  - the END
  - a END
  - the store END
  - Alice talked to Bob . END
  - Alice hated on Bob . END

Whole-sentence estimation

- *N* sentences in training data
- #(...) means the *count* of how many times it appeared in the training data.

$$P(w_1..w_T) = \frac{\#(w_1..w_T)}{N}$$

- Does not generalize! (overfits the training data)

History-based prob view

- Apply chain rule - no model assumptions yet

$$P(w_1..w_T) =$$

History-based prob view

- Apply chain rule - no model assumptions yet

$$P(w_1..w_T) =$$

$$= P(w_1..w_{T-1})P(w_T \mid w_1..w_{T-1})$$

History-based prob view

- Apply chain rule - no model assumptions yet

$$P(w_1..w_T) =$$

$$= P(w_1..w_{T-1})P(w_T \mid w_1..w_{T-1})$$

$$= P(w_1..w_{T-2})P(w_{T-1} \mid w_1..w_{T-2})P(w_T \mid w_1..w_{T-1})$$

History-based prob view

- Apply chain rule - no model assumptions yet

$$P(w_1..w_T) =$$

$$= P(w_1..w_{T-1})P(w_T \mid w_1..w_{T-1})$$

$$= P(w_1..w_{T-2})P(w_{T-1} \mid w_1..w_{T-2})P(w_T \mid w_1..w_{T-1})$$

$$P(w_1..w_T) = P(w_1)P(w_2|w_1)P(w_3|w_1,w_2)P(w_4|w_1,w_2,w_3)...$$
$$= \prod_t P(w_t \mid w_1..w_{t-1})$$

History-based data view

WordTree (Wattenberg and Viégas, 2008)

each node visualizes *full history* model $P(w_t \mid w_1..w_{t-1})$

Demo: http://www.jasondavies.com/wordtree/?source=flickr-comments.txt&prefix=Thank

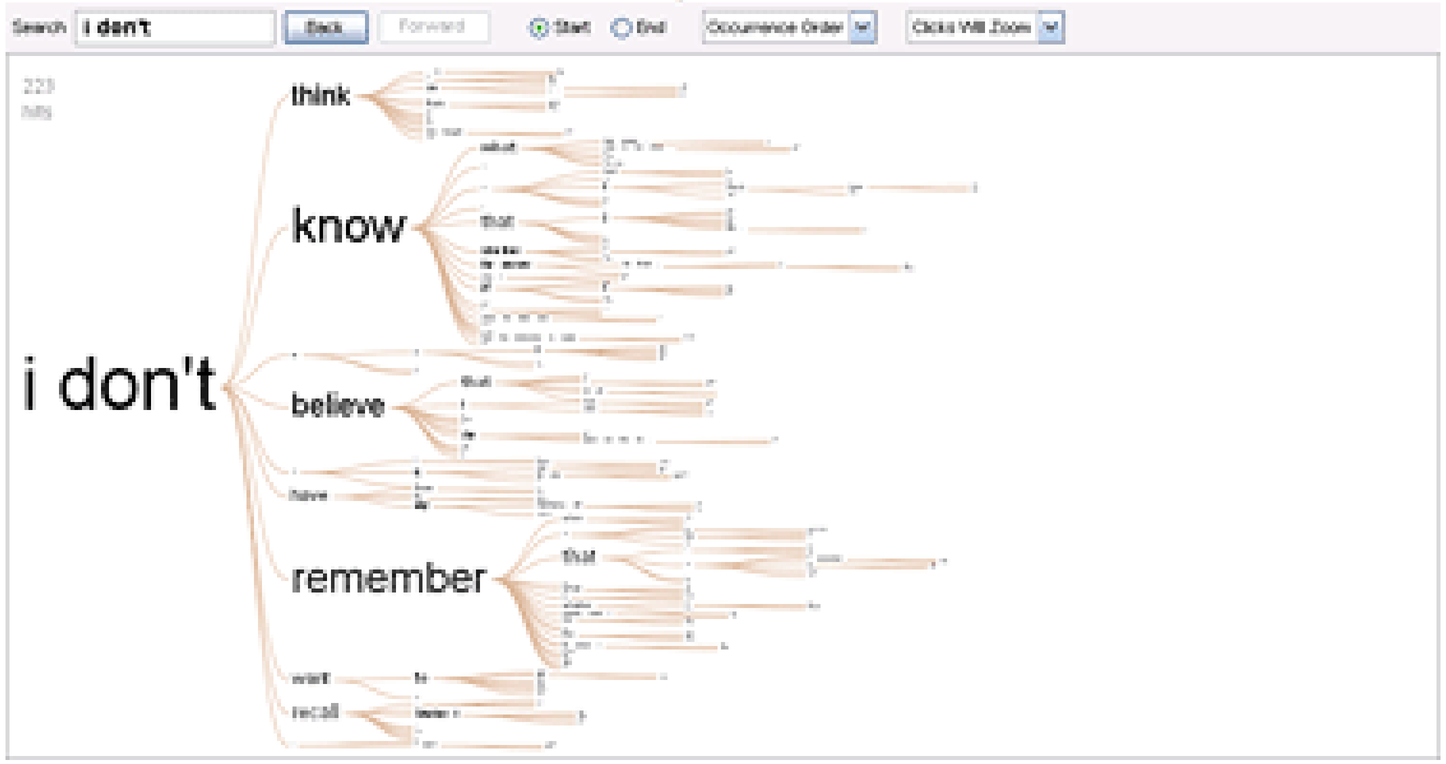


Fig 6: Bill Clinton’s testimony in 1998.

Modeling point of view: too sparse!

P(__ | OK, turn to page 144 and see)

Markov chain models

- **Markov process**: words are generated one at a time. Process ends when END symbol is emitted.
- **First-order Markov assumption**: Assume a word depends only on previous word

$$P(w_t|w_1..w_{t-1}) = P(w_t|w_{t-1})$$

- This yields joint probability

$$P(w_1..w_T) = \prod_t P(w_t \mid w_1..w_{t-1})$$ <-- chain rule

$$= \prod_t P(w_t \mid w_{t-1})$$ <-- Markov assumption

Which prefers which?

0th-order (unigrams)

$$\prod_t P(w_t)$$

1st order (bigrams)

$$\prod_t P(w_t|w_{t-1})$$

(1) Thank you for sharing

(2) the the the the

Markov chain models

- First-order Markov assumption:
  Assume a word depends only on previous word

$$P(w_1..w_T) = \prod_t P(w_t \mid w_{t-1})$$

- MLE (maximum likelihood) estimator:

$$P(w_t|w_{t-1}) = \frac{\#(w_{t-1}, w_t)}{\#(w_{t-1})}$$

“2-gram model”
“bigram model”

- START symbol for convenient representation

P(*START* I like cats . *END*) =

= P(I | *START*) P(like | I) P(cats | like) P(. | cats) P(*END* | .)

$$= \frac{\#(\textit{START}\ \text{I})}{\#(\textit{START})} \quad \frac{\#(\text{I like})}{\#(\text{I})} \quad \frac{\#(\text{like cats})}{\#(\text{he})} \quad \frac{\#(\text{cats .})}{\#(\text{cats})} \quad \frac{\#(\text{. }\textit{END})}{\#(\text{.})}$$

Andrei Andreyevich Markov

1856 - 1922

- Graduate of Saint Petersburg University (1878), where he began a professor in 1886.
- Mathematician, teacher, political activist
  - In 1913, when the government celebrated the 300th anniversary of the House of Romanov family, Markov organized a counter-celebration of the 200th anniversary of Bernoulli's discovery of the Law of Large Numbers.
- Markov was also interested in poetry and he made studies of poetic style.

Markov (1913)

- Took 20,000 characters from Pushkin’s *Eugene Onegin* to see if it could be approximated by a first-order chain of characters.

0th order model

vowel	consonant
0.43	0.57

1st order model

	c_t = vowel	c_t = consonant
c_{t-1} = vowel	0.13	0.87
c_{t-1} = consonant	0.66	0.34

Markov Approximations to English

- Zero-order approximation, P(c)
  - XFOML RXKXRJFFUJ ZLPWCFWKCRJ FFJEYVKCQSGHYD QPAAMKBZAACIBZLHJQD
- First-order approximation, P(c|c)
  - OCRO HLI RGWR NWIELWIS EU LL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA
- Second-order approximation, P(c|c,c)
  - ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE TUCOOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE

[From Shannon’s information theory paper]

Sparsity vs ngram size

$P(w_1..w_T)$ modeled as...

0th-order (unigrams)	1st order (bigrams)	2nd order (trigrams)	$\dots.$	Whole-sentence memorization
$\prod_t P(w_t)$	$\prod_t P(w_t \mid w_{t-1})$	$\prod_t P(w_t \mid w_{t-1}, w_{t-2})$		$\dots.$
V				

Number of parameters?

Number of non-zero parameters?

Sparsity vs ngram size

$P(w_1..w_T)$ modeled as...

0th-order (unigrams)	1st order (bigrams)	2nd order (trigrams)	....	Whole-sentence memorization
$\prod_t P(w_t)$	$\prod_t P(w_t \mid w_{t-1})$	$\prod_t P(w_t \mid w_{t-1}, w_{t-2})$		$\ldots.$
V	**V²**			

Number of parameters?

Number of non-zero parameters?

Sparsity vs ngram size

$P(w_1..w_T)$ modeled as...

0th-order (unigrams)	1st order (bigrams)	2nd order (trigrams)	....	Whole-sentence memorization
$\prod_t P(w_t)$	$\prod_t P(w_t \mid w_{t-1})$	$\prod_t P(w_t \mid w_{t-1}, w_{t-2})$		....
$\mathbf{V}$	$\mathbf{V}^2$	$\mathbf{V}^3$		

Number of parameters?

Number of non-zero parameters?

Severity of the sparse data problem

count	2-grams	3-grams
1	8,045,024	53,737,350
2	2,065,469	9,229,958
3	970,434	3,654,791
>4	3,413,290	8,728,789
>0	14,494,217	75,349,888
possible	6.8×10^{10}	1.7×10^{16}

Vocab size 260,741 words, 365M words training

Andrew McCallum, UMass Amherst, Slide material from Dan Klein

The Zero Problem

- Necessarily some zeros
  - trigram model: 1.7 x 10^{16} parameters
  - but only 3.6 x 10^{8} words of training data
- How should we distribute some probability mass over all possibilities in the model
  - optimal situation: even the least frequent trigram would occur several times, in order to distinguish its probability versus other trigrams
  - optimal situation cannot happen, unfortunately (how much data would we need?)
- Two kinds of zeros: $p(w|h)=0$, or even $p(h)=0$

Parameter Estimation

- Maximum likelihood estimates won’t get us very far

$$\hat{P}(w|w_{-1}) = \frac{c(w_{-1}, w)}{\sum_{w'} c(w_{-1}, w')}$$

3516 wipe off the excess
1034 wipe off the dust
547 wipe off the sweat
518 wipe off the mouthpiece
…
120 wipe off the grease
0 wipe off the sauce
0 wipe off the mice

28048 wipe off the *

- Need to *smooth* these estimates

- General method (procedurally)
  - Take your empirical counts
  - Modify them in various ways to improve estimates

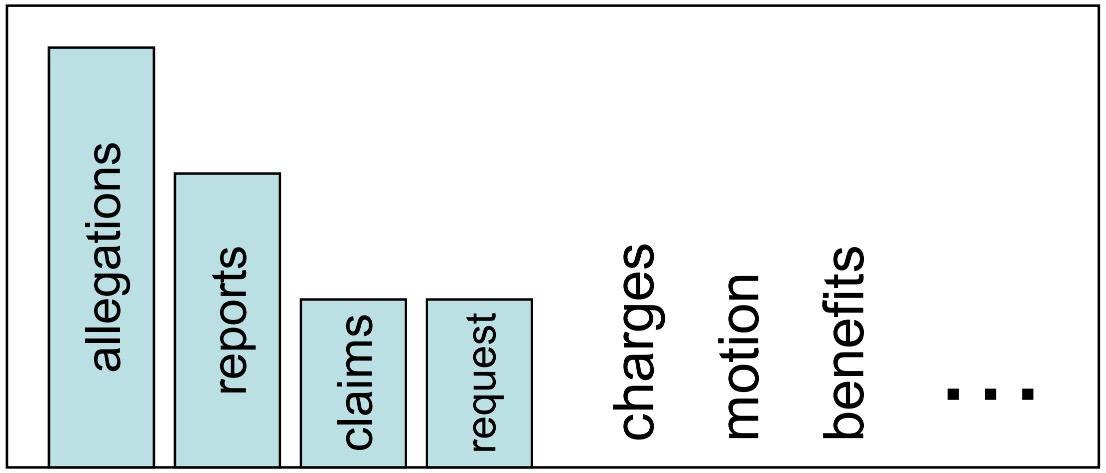
- General method (mathematically)
  - Often can give estimators a formal statistical interpretation
  - … but not always
  - Approaches that are mathematically obvious aren’t always what works

Andrew McCallum, UMass Amherst, Slide material from Dan Klein

Smoothing

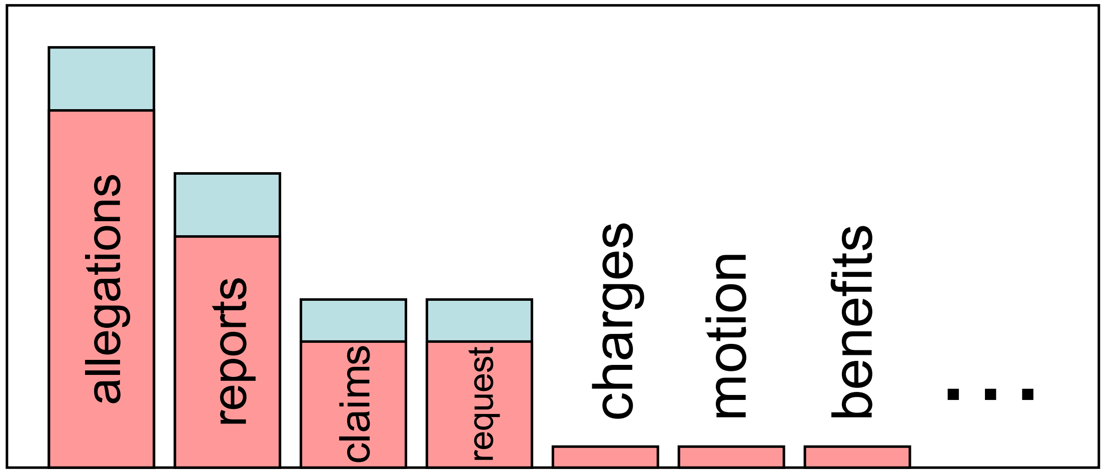
- We often want to make estimates from sparse statistics:

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



- Smoothing flattens spiky distributions so they generalize better

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



- Very important all over NLP, but easy to do badly!
- We'll illustrate with bigrams today (h = previous word, could be anything).

Pseudocount (Dirichlet) smoothing

(illustrated for unigrams)

MLE (maximum likelihood estimate)
Relative frequency estimator

$$P^{\mathrm{MLE}}(w) = \frac{\#(w)}{n}$$

is shorthand for ...

Model $P(w|\theta) = \theta_w$

Estimator

$$\hat{\theta}^{\mathrm{MLE}} = \arg\max_\theta P(w_1..w_n|\theta)$$

which is solved by

$$\hat{\theta}_w^{\mathrm{MLE}} = \frac{\#(w)}{n}$$

Pseudocount smoothing.

MAP (maximum a-posteriori)

with Dirichlet prior

$$P^{\mathrm{MAP}}(w) = \frac{\#(w) + \alpha}{n + V\alpha}$$

is shorthand for ...

Add a prior $P(\theta) = \mathrm{Dirichlet}(\alpha + 1)$

Estimator

$$\hat{\theta}^{\mathrm{MAP}} = \arg\max_\theta P(w_1..w_n|\theta)P(\theta)$$

which is solved by

$$\hat{\theta}_w^{\mathrm{MAP}} = \frac{\#(w) + \alpha}{n + V\alpha}$$

Linear interpolation

- Pseudocount smoothing: simple but works poorly.
- Interpolation: mix between related, denser histories

$$P(w|w_{-1}, w_{-2}) = \lambda \hat{P}(w|w_{-1}, w_{-2}) + \lambda' \hat{P}(w|w_{-1}) + \lambda'' \hat{P}(w)$$

- Allows sharing of statistical strength between contexts with shared prefixes.
- Mixing parameters can be learned with EM (later)
- Many other methods use smarter ways of combining stats from different sized contexts

Evaluation

- Intrinsic evaluation: likelihood of ***testset*** sentences

Likelihood = $$\prod_{t \in \text{testset}} P(w_t | w_{t-k+1} .. w_{t-1}; \theta)$$

Mean loglik = $$\frac{1}{N_{tok}} \sum_{t \in \text{testset}} \log P(w_t | w_{t-k+1} .. w_{t-1}; \theta)$$

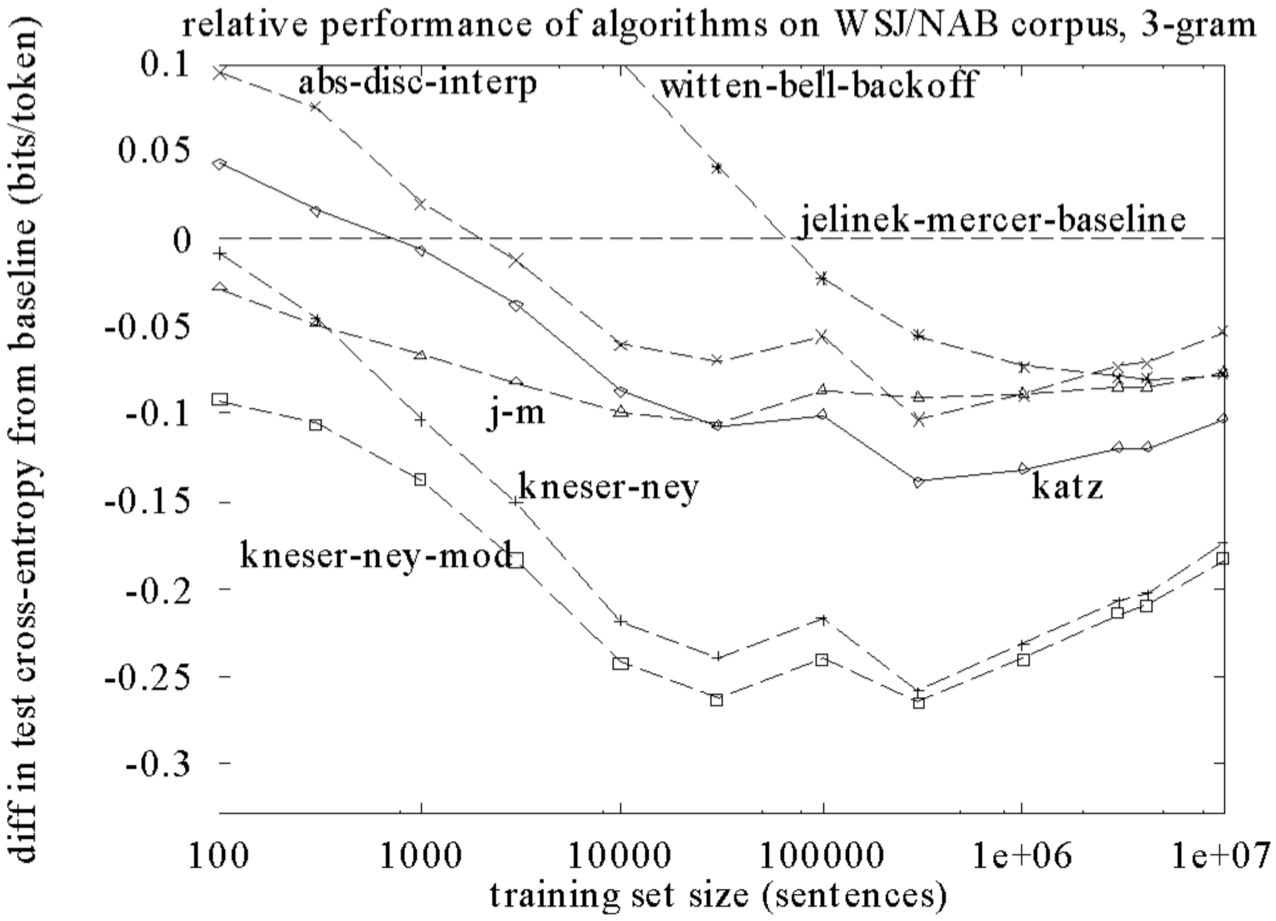
Perplexity = $$\exp\left(-\frac{1}{N_{tok}} \sum_{t \in \text{testset}} \log P(w_t | w_{t-k+1} .. w_{t-1}; \theta) \right)$$

"Perplexity": branching factor interpretation
(Note: textbook & many sources assume log-base-2)

- Extrinsic evaluation: MT or ASR or other task accuracy

What Actually Works?

- Trigrams and beyond:
  - Unigrams, bigrams generally useless
  - Trigrams much better (when there’s enough data)
  - 4-, 5-grams really useful in MT, but not so much for speech
- Discounting
  - Absolute discounting, Good-Turing, held-out estimation, Witten-Bell
- Context counting
  - Kneser-Ney construction oflower-order models
- See [Chen+Goodman] reading for tons of graphs!



[Graphs from Joshua Goodman]

Chen and Goodman 1998, “Empirical Study of Smoothing Techniques for LM”

Data vs Models

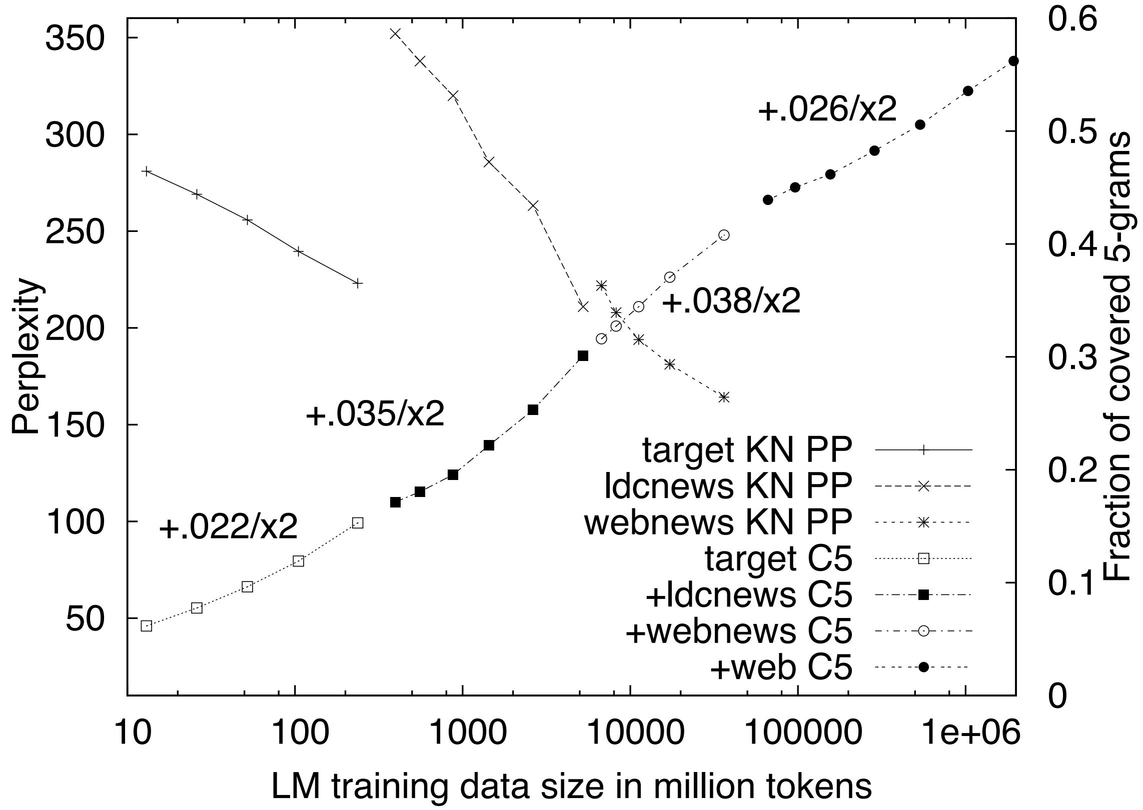


Figure 4: Perplexities with Kneser-Ney Smoothing (KN PP) and fraction of covered 5-grams (C5).

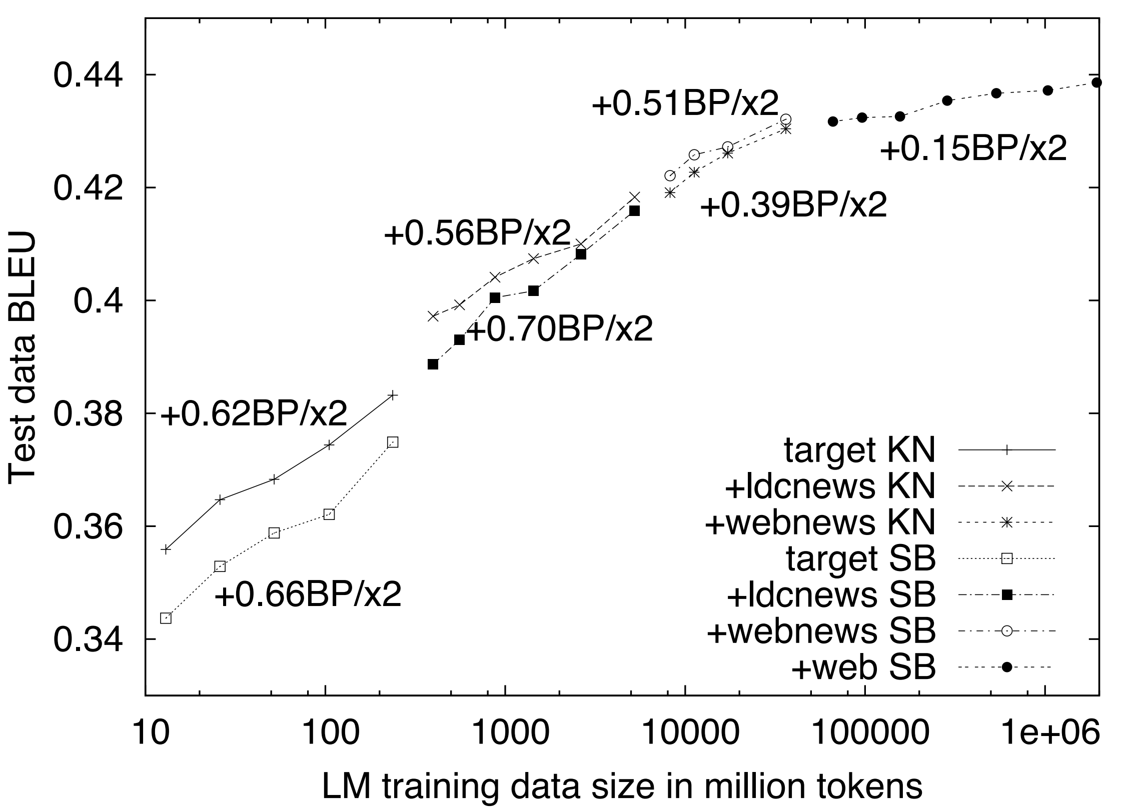


Figure 5: BLEU scores for varying amounts of data using Kneser-Ney (KN) and Stupid Backoff (SB).

Brants et al. 2007 *http://www.aclweb.org/anthology/D07-1090*

Issues with MLE n-grams

- Sparsity issues
  - Unseen words should get non-zero probability. (solution: *smoothing*)
  - Different words should share statistical strength.
    (solution: *clustering/latent vars*)
    Our smoothing methods today still can't solve Chomsky's example if all bigrams were unseen in training data ... need latent variable models to get it; see (Pereira 2000)
    - (1) Colorless green ideas sleep furiously.
    - (2) Furiously sleep ideas green colorless.

- Representation issues
  - *Topicality and syntax:* long-distance phenomena also affect coherency

- But, hard to beat well-smoothed n-grams on lots and lots of data...