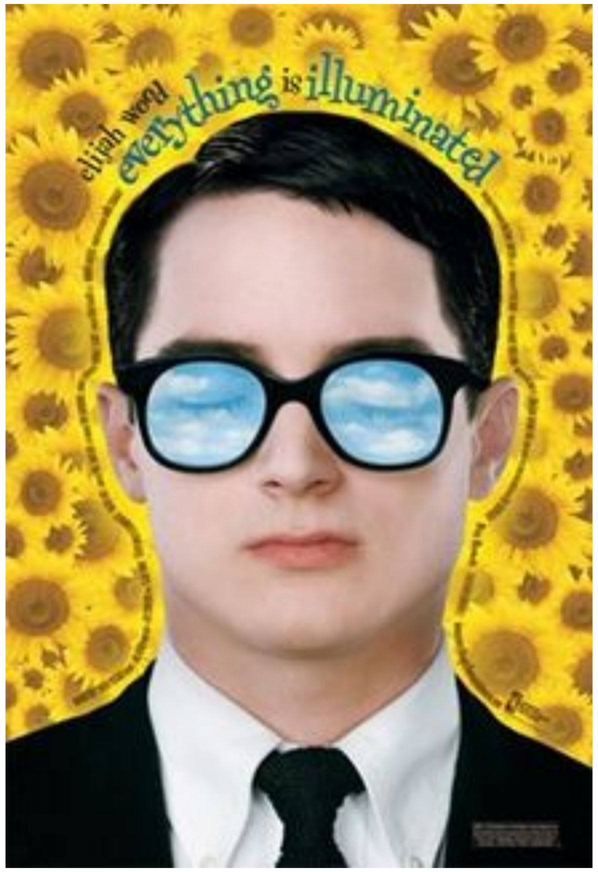
Language Models

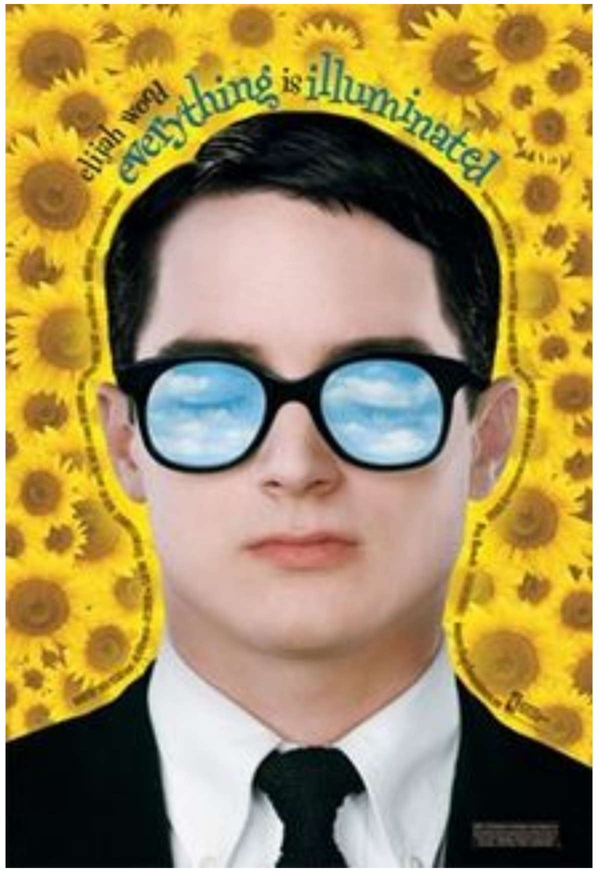
January 28, 2014

Still no MT??

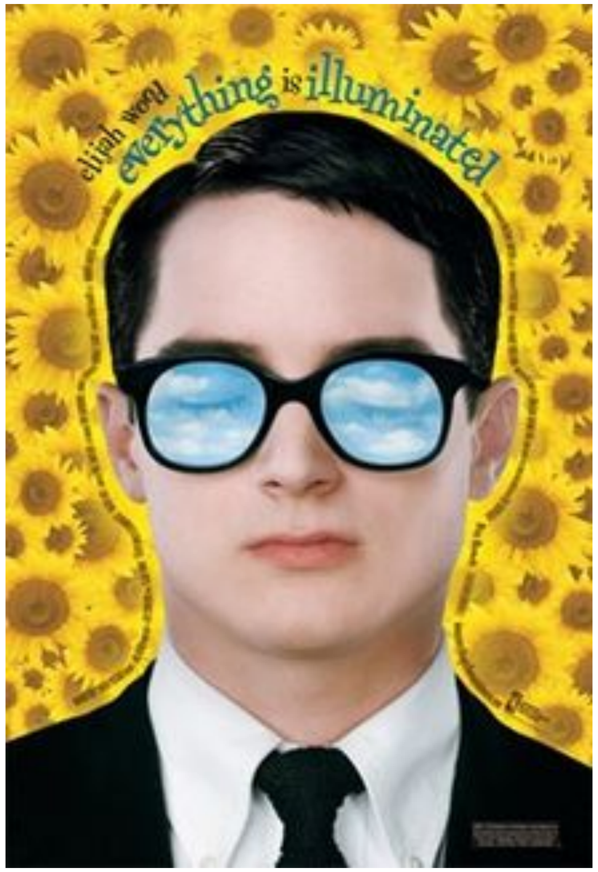
- Today we will talk about models of p(sentence)
- The rest of this semester will deal with
  p(translated sentence | input sentence)
- Why do it this way?
  - Conditioning on more stuff makes modeling more complicated. That is: p(sentence) is easier than p(translated sentence | input sentence).
  - Language models are arguably the most important models in statistical MT



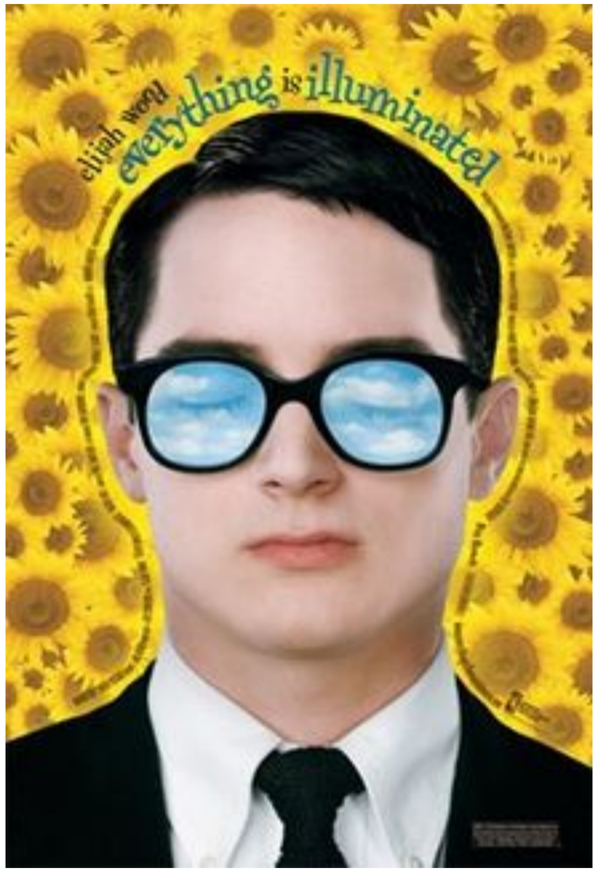
My legal name is Alexander Perchov.



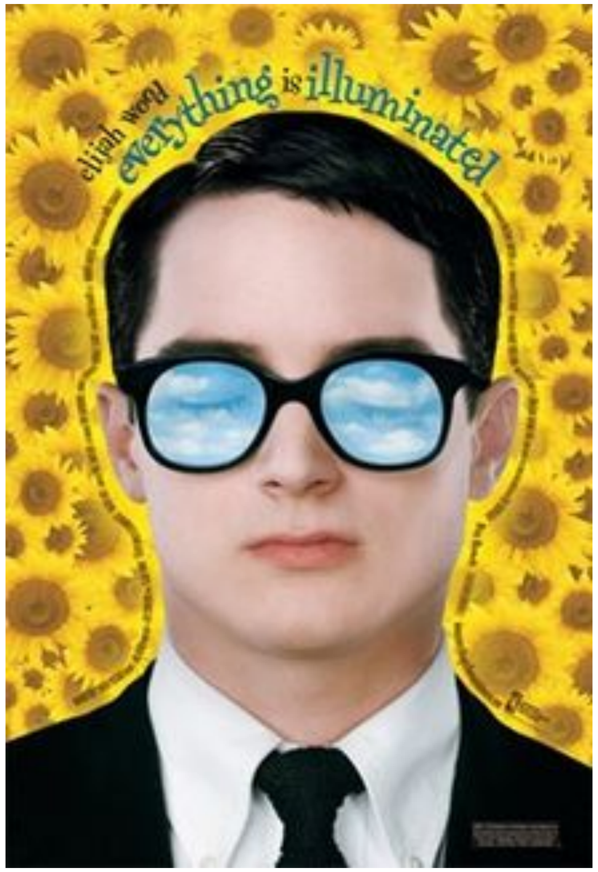
My legal name is Alexander Perchov. But all of my many friends dub me Alex, because that is a more flaccid-to-utter version of my legal name.



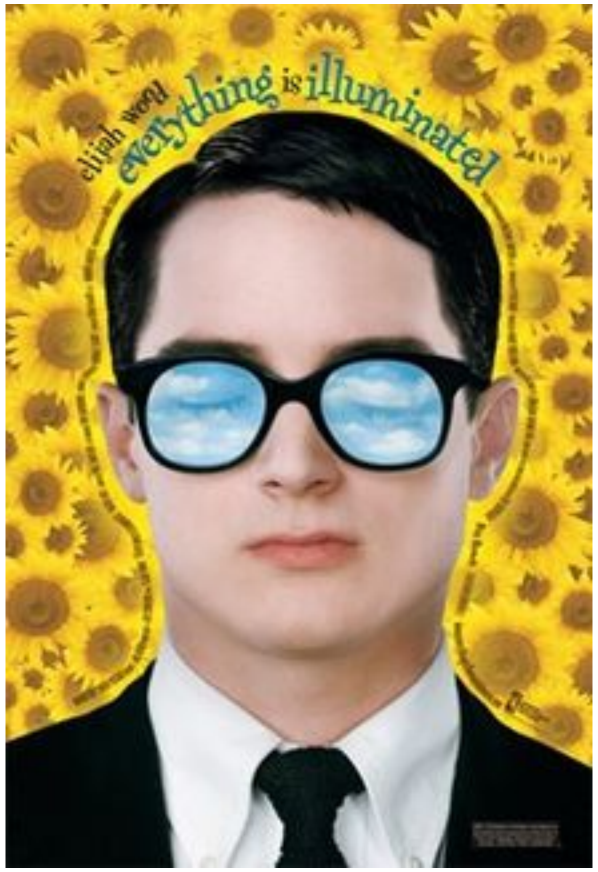
My legal name is Alexander Perchov. But all of my many friends dub me Alex, because that is a more flaccid-to-utter version of my legal name. Mother dubs me Alexi-stop-spleening-me!, because I am always spleening her.



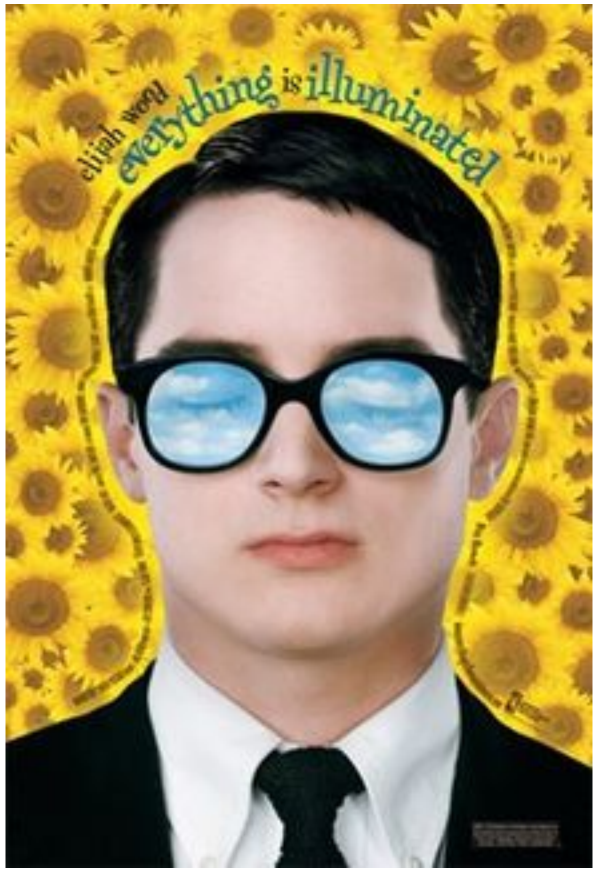
My legal name is Alexander Perchov. But all of my many friends dub me Alex, because that is a more flaccid-to-utter version of my legal name. Mother dubs me Alexi-stop-spleening-me!, because I am always spleening her. If you want to know why I am always spleening her, it is because I am always elsewhere with friends, and disseminating so much currency, and performing so many things that can spleen a mother.



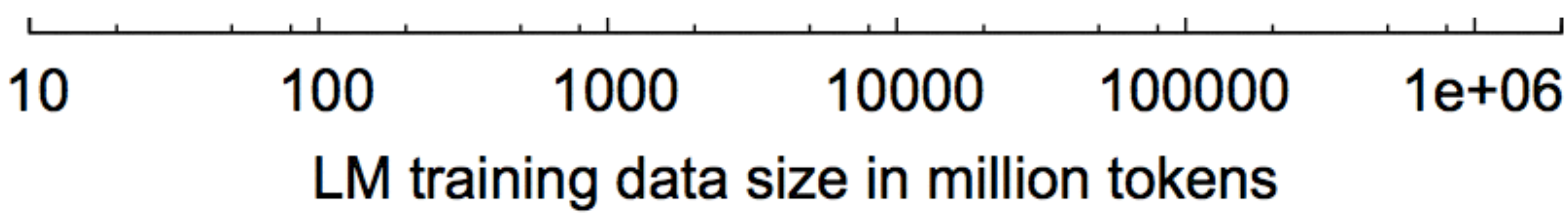
My legal name is Alexander Perchov. But all of my many friends dub me Alex, because that is a more flaccid-to-utter version of my legal name. Mother dubs me Alexi-stop-spleening-me!, because I am always spleening her. If you want to know why I am always spleening her, it is because I am always elsewhere with friends, and disseminating so much currency, and performing so many things that can spleen a mother. Father used to dub me Shapka, for the fur hat I would don even in the summer month.

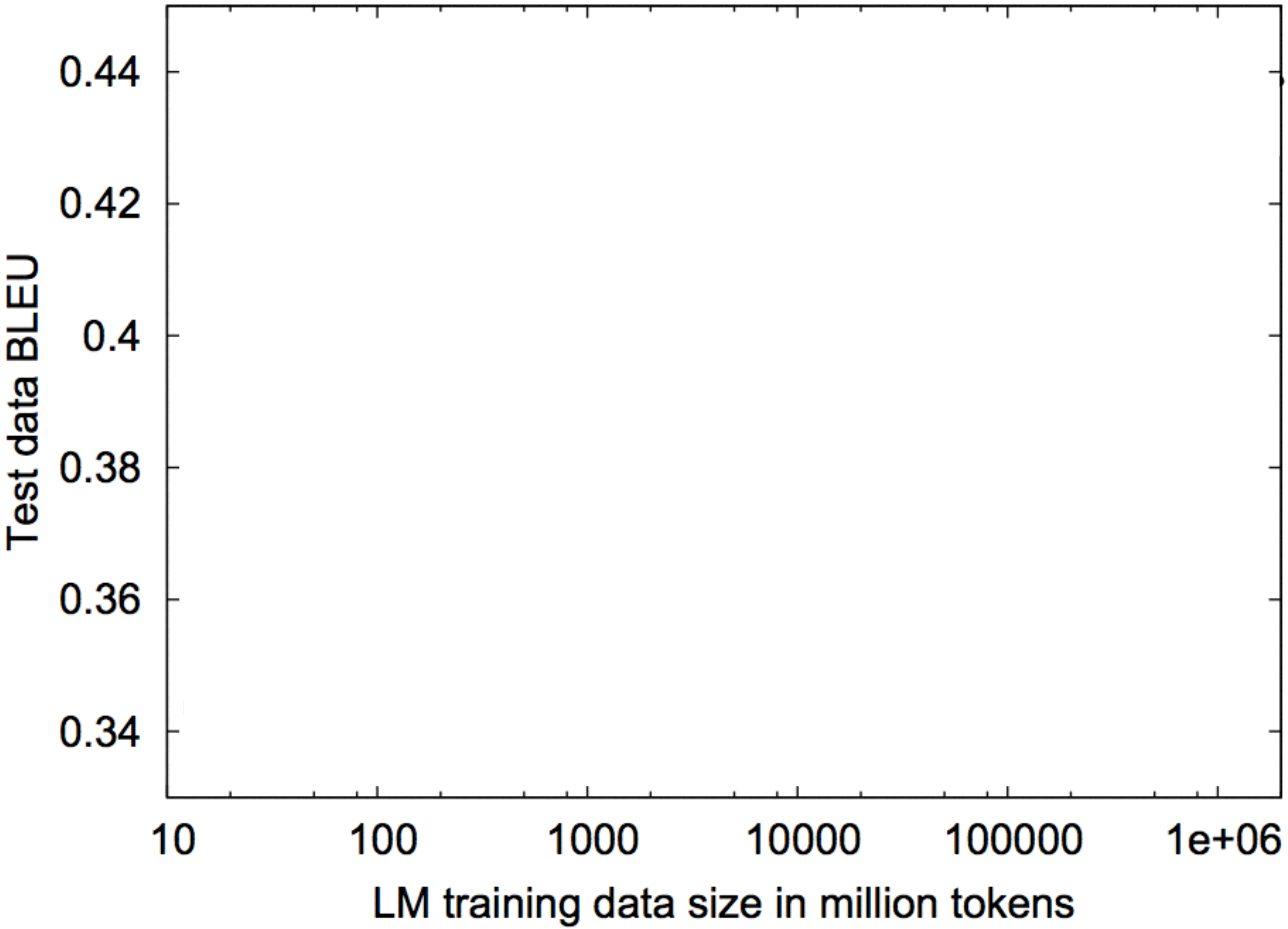


My legal name is Alexander Perchov. But all of my many friends dub me Alex, because that is a more flaccid-to-utter version of my legal name. Mother dubs me Alexi-stop-spleening-me!, because I am always spleening her. If you want to know why I am always spleening her, it is because I am always elsewhere with friends, and disseminating so much currency, and performing so many things that can spleen a mother. Father used to dub me Shapka, for the fur hat I would don even in the summer month. He ceased dubbing me that because I ordered him to cease dubbing me that.

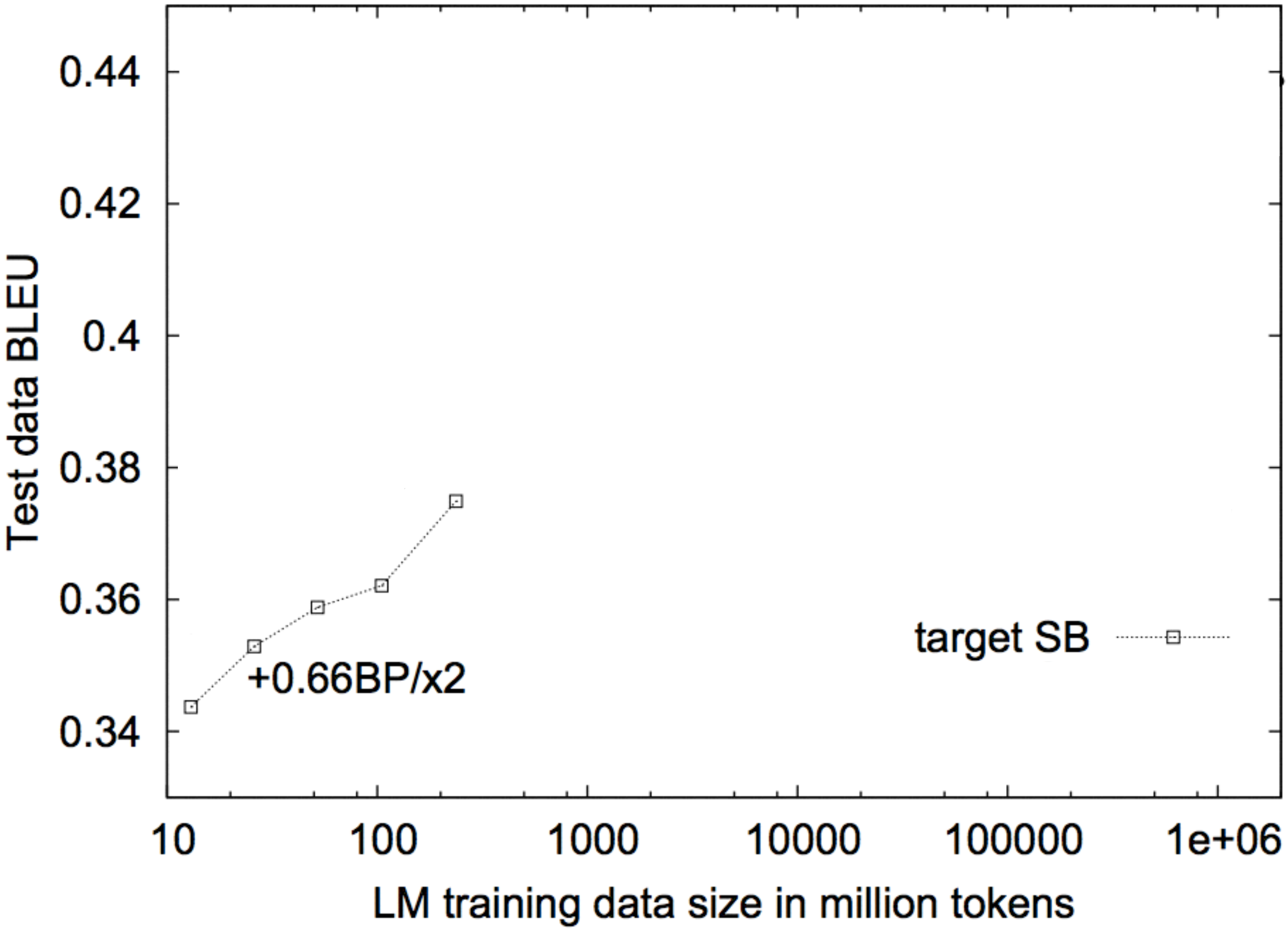


My legal name is Alexander Perchov. But all of my many friends dub me Alex, because that is a more flaccid-to-utter version of my legal name. Mother dubs me Alexi-stop-spleening-me!, because I am always spleening her. If you want to know why I am always spleening her, it is because I am always elsewhere with friends, and disseminating so much currency, and performing so many things that can spleen a mother. Father used to dub me Shapka, for the fur hat I would don even in the summer month. He ceased dubbing me that because I ordered him to cease dubbing me that. It sounded boyish to me, and I have always thought of myself as very potent and generative.

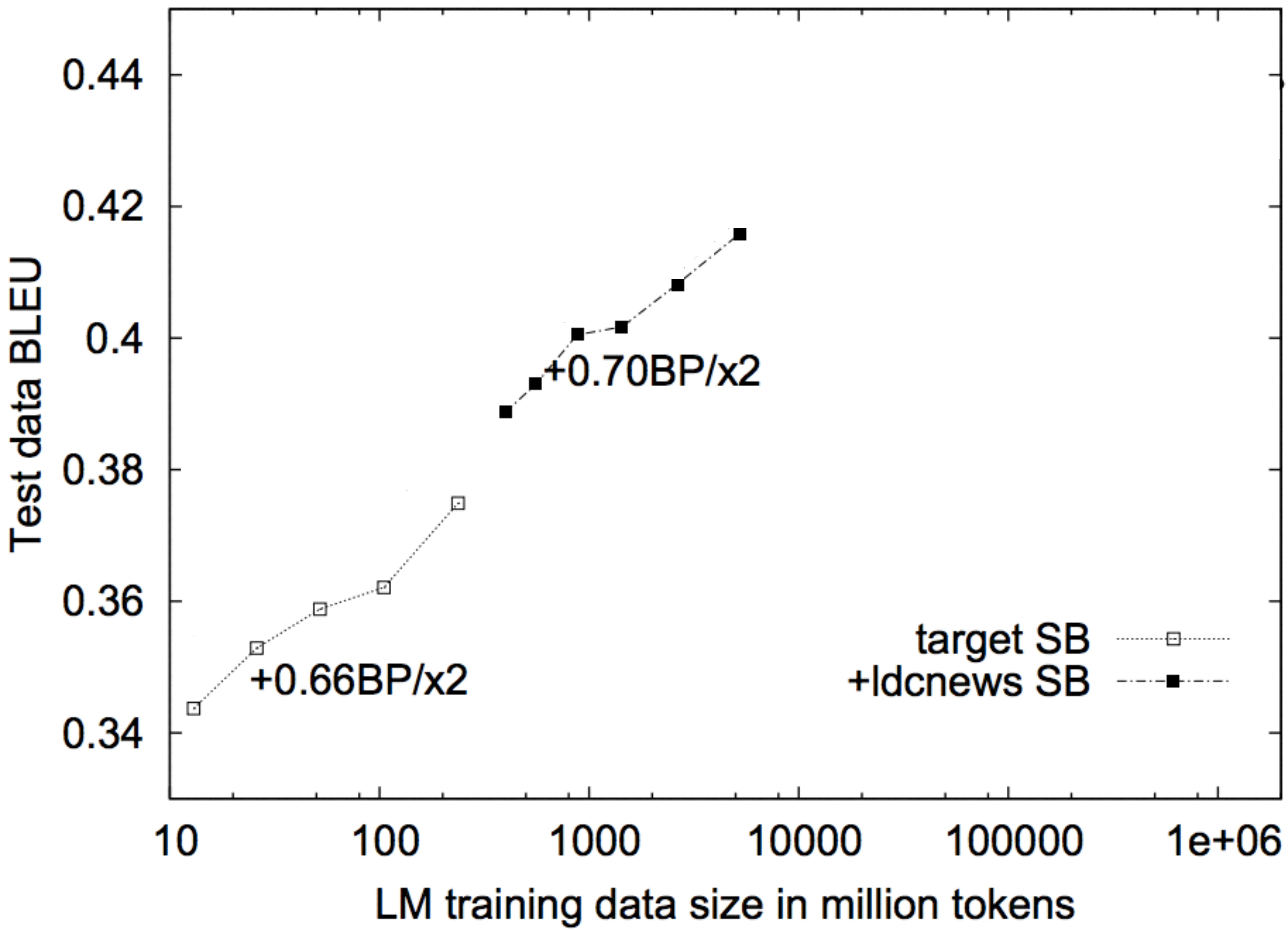

10
100
1000
10000
100000
1e+06
LM training data size in million tokens



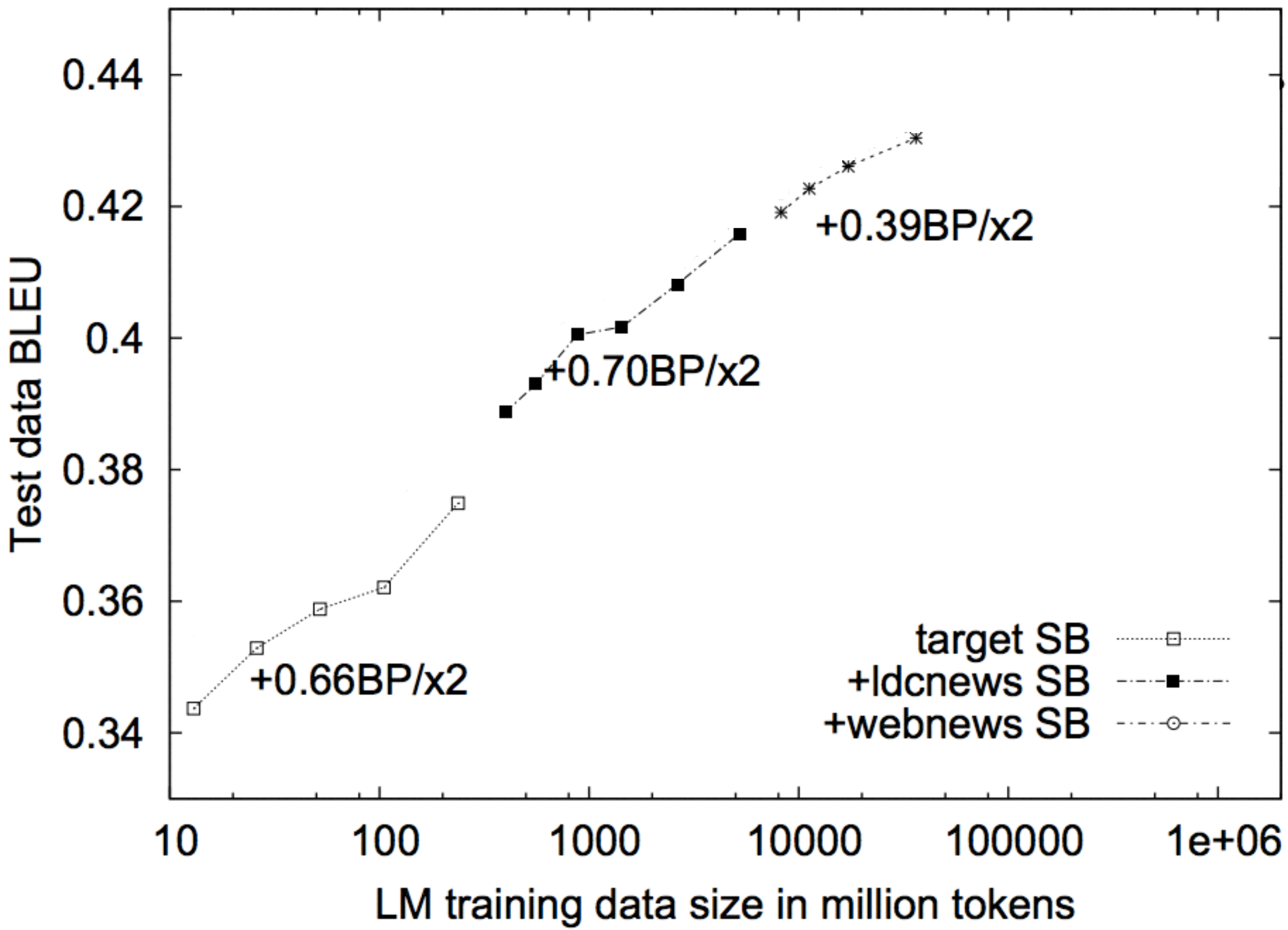
Test data BLEU
0.44
0.42
0.4
0.38
0.36
0.34
10
100
1000
10000
100000
1e+06
LM training data size in million tokens



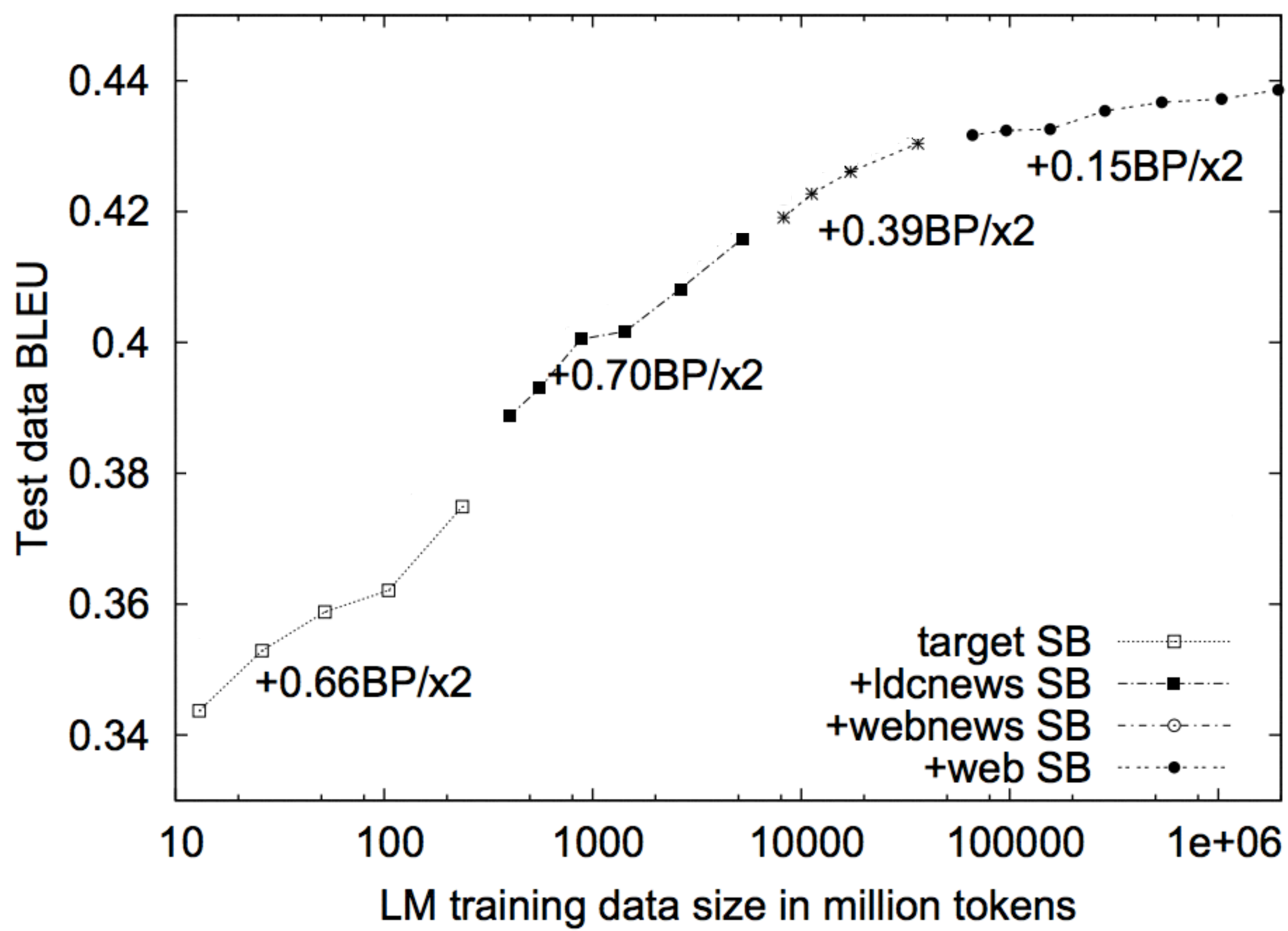
Test data BLEU
0.44
0.42
0.4
0.38
0.36
0.34
+0.66BP/x2
target SB
10
100
1000
10000
100000
1e+06
LM training data size in million tokens



0.44
0.42
0.4
0.38
0.36
0.34
Test data BLEU
10
100
1000
10000
100000
1e+06
LM training data size in million tokens
+0.70BP/x2
+0.66BP/x2
target SB
+ldcnews SB



0.44
0.42
0.4
0.38
0.36
0.34
Test data BLEU
10
100
1000
10000
100000
1e+06
LM training data size in million tokens
+0.39BP/x2
+0.70BP/x2
+0.66BP/x2
target SB
+ldcnews SB
+webnews SB



Test data BLEU
0.44
0.42
0.4
0.38
0.36
0.34
+0.15BP/x2
+0.39BP/x2
+0.70BP/x2
+0.66BP/x2
target SB
+ldcnews SB
+webnews SB
+web SB
10
100
1000
10000
100000
1e+06
LM training data size in million tokens

Language Models Matter

- Language models play the role of ...
  - a judge of *grammaticality*
  - a judge of *semantic plausibility*
  - an enforcer of *stylistic consistency*
  - a repository of knowledge (?)

What is the probability of a sentence?

- Requirements
  - Assign a probability to *every sentence* (i.e., string of words)

What is the probability of a sentence?

- Requirements
  - Assign a probability to *every sentence* (i.e., string of words)

$$\sum_{\mathbf{e} \in \Sigma^*} p_{\text{LM}}(\mathbf{e}) = 1$$

$$p_{\text{LM}}(\mathbf{e}) \geq 0 \quad \forall \mathbf{e} \in \Sigma^*$$

Why do we want to estimate the probability of a sentence?

- Goal: Assign a higher probability to *good sentences in English*

p_{LM}*(the house is small)* > p_{LM}*(small the is house)*

translations of German *Haus: home, house …*

p_{LM}*(I am going home)* > p_{LM}*(I am going house)*

n-gram LMs

$$p_{\mathrm{LM}}(\mathbf{e})$$

Vector-valued random variable

n-gram LMs

$$
\begin{aligned}
p_{\mathrm{LM}}(\mathbf{e}) = & p(e_1, e_2, e_3, \ldots, e_\ell) \\
\approx & p(e_1) \times \\
& p(e_2 \mid e_1) \times \\
& p(e_3 \mid \cancel{e_1,} e_2) \times \\
& p(e_4 \mid \cancel{e_1, e_2,} e_3) \times \\
& \cdots \times \\
& p(e_\ell \mid \cancel{e_1, e_2, \ldots, e_{\ell-2},} e_{\ell-1})
\end{aligned}
$$

Chain rule

The **chain rule** is derived from a repeated application of the definition of conditional probability:

$$p(a, b, c, d)$$

Conditional Independence

$$
\begin{aligned}
p(a, b, c) &= p(a \mid b, c)p(b, c) \\
&= p(a \mid b, c)p(b \mid c)p(c)
\end{aligned}
$$

"If I know B, then C doesn't tell me about A"

$$p(a \mid b, c) = p(a \mid b)$$

$$
\begin{aligned}
p(a, b, c) &= p(a \mid b, c)p(b, c) \\
&= p(a \mid b, \cancel{c})p(b \mid c)p(c) \\
&= p(a \mid b)p(b \mid c)p(c)
\end{aligned}
$$

Is the Markov assumption valid for Language?

- the old man are/is
- the pictures are/is
- The old man in the pictures is my dad.

n-gram LMs

$$
\begin{aligned}
p_{\mathrm{LM}}(\mathbf{e}) =& p(e_1, e_2, e_3, \ldots, e_\ell) \\
\approx& p(e_1) \times \\
& p(e_2 \mid e_1) \times \\
& p(e_3 \mid \cancel{e_1,}\, e_2) \times \\
& p(e_4 \mid \cancel{e_1, e_2,}\, e_3) \times \\
& \cdots \times \\
& p(e_\ell \mid \cancel{e_1, e_2, \ldots, e_{\ell-2},}\, e_{\ell-1})
\end{aligned}
$$

$$
\begin{aligned}
p_{\mathrm{LM}}(\mathbf{e}) =& p(e_1, e_2, e_3, \ldots, e_\ell) \\
\approx& p(e_1) \times \\
& p(e_2 \mid e_1) \times \\
& p(e_3 \mid e_1 \cancel{, e_2}) \times \\
& p(e_4 \mid \cancel{e_1,}\, e_2, \cancel{e_3}) \times \\
& \cdots \times \\
& p(e_\ell \mid \cancel{e_1, e_2, \ldots,}\, e_{\ell-2}, e_{\ell-1})
\end{aligned}
$$

Which do you think is better? Why?

n-gram LMs

$$
\begin{aligned}
p_{\mathrm{LM}}(\mathbf{e}) = & p(e_1, e_2, e_3, \ldots, e_\ell) \\
\approx & p(e_1) \times \\
& p(e_2 \mid e_1) \times \\
& p(e_3 \mid \cancel{e_1,}\ e_2) \times \\
& p(e_4 \mid \cancel{e_1, e_2,}\ e_3) \times \\
& \cdots \times \\
& p(e_\ell \mid \cancel{e_1, e_2, \ldots, e_{\ell-2},}\ e_{\ell-1})
\end{aligned}
$$

$$
= p(e_1 \mid \mathrm{START}) \times \prod_{i=2}^{\ell} p(e_i \mid e_{i-1}) \times p(\mathrm{STOP} \mid e_\ell)
$$

START my friends call me Alex STOP

$$p(\texttt{my} \mid \texttt{START}) \times p(\texttt{friends} \mid \texttt{my}) \times p(\texttt{call} \mid \texttt{friends}) \times p(\texttt{me} \mid \texttt{call}) \times p(\texttt{Alex} \mid \texttt{me}) \times p(\texttt{STOP} \mid \texttt{Alex})$$

START my friends dub me Alex STOP

$$p(\texttt{my} \mid \texttt{START}) \times p(\texttt{friends} \mid \texttt{my}) \times p(\texttt{dub} \mid \texttt{friends}) \times p(\texttt{me} \mid \texttt{dub}) \times p(\texttt{Alex} \mid \texttt{me}) \times p(\texttt{STOP} \mid \texttt{Alex})$$

These sentences have many terms in common.

Categorical Distributions

A **categorical distribution** characterizes a random event that can take on exactly one of *K* possible outcomes.

(*nb*. we often call these "multinomial distributions")

$$p(x) = \begin{cases} p_1 & \text{if } x = 1 \\ p_2 & \text{if } x = 2 \\ \ldots & \\ p_K & \text{if } x = K \\ 0 & \text{otherwise} \end{cases}$$

$$\sum_i p_i = 1$$

$$p_i \geq 0 \quad \forall i$$

$$p(\cdot)$$

Outcome	p
the	0.3
and	0.1
said	0.04
says	0.004
of	0.12
why	0.008
Why	0.0007
restaurant	0.00009
destitute	0.00000064

Probability tables like this are the workhorses of language (and translation) modeling.

$p(\cdot \mid \text{some context})$

Outcome	p
the	0.6
and	0.04
said	0.009
says	0.00001
of	0.1
why	0.1
Why	0.00008
restaurant	0.0000008
destitute	0.00000064

$p(\cdot \mid \text{other context})$

Outcome	p
the	0.01
and	0.01
said	0.003
says	0.009
of	0.002
why	0.003
Why	0.0006
restaurant	0.2
destitute	0.1

~~$p(\cdot \mid \text{some context})$~~ $p(\cdot \mid \texttt{in})$

Outcome	p
the	0.6
and	0.04
said	0.009
says	0.00001
of	0.1
why	0.1
Why	0.00008
restaurant	0.0000008
destitute	0.00000064

~~$p(\cdot \mid \text{other context})$~~ $p(\cdot \mid \texttt{the})$

Outcome	p
the	0.01
and	0.01
said	0.003
says	0.009
of	0.002
why	0.003
Why	0.0006
restaurant	0.2
destitute	0.1

LM Evaluation

- Extrinsic evaluation: build a new language model, use it for some task (MT, ASR, etc.)
- Intrinsic: measure how good we are at modeling language

We will use **perplexity** to evaluate models

Given: $\mathbf{w}, p_{\mathrm{LM}}$

$$\mathrm{PPL} = 2^{\frac{1}{|\mathbf{w}|} \log_2 p_{\mathrm{LM}}(\mathbf{w})}$$

$$0 \leq \mathrm{PPL} \leq \infty$$

Perplexity

- Generally fairly good correlations with machine translation quality for *n*-gram models
- Perplexity is a generalization of the notion of branching factor
  - How many choices do I have at each position?
- State-of-the-art English LMs have PPL of ~100 word choices per position
- A uniform LM has a perplexity of $|\Sigma|$
- Humans do much better
- ... and bad models can do even *worse* than uniform!

Whence parameters?

Estimation.

$$p(x \mid y) = \frac{p(x, y)}{p(y)}$$

$$\hat{p}_{\text{MLE}}(x) = \frac{\text{count}(x)}{N}$$

$$\hat{p}_{\text{MLE}}(x, y) = \frac{\text{count}(x, y)}{N}$$

$$\hat{p}_{\text{MLE}}(x \mid y) = \frac{\text{count}(x, y)}{N} \times \frac{N}{\text{count}(y)}$$

$$\boxed{= \frac{\text{count}(x, y)}{\text{count}(y)}}$$

$$\hat{p}_{\text{MLE}}(\texttt{call} \mid \texttt{friends}) = \frac{\text{count}(\texttt{friends call})}{\text{count}(\texttt{friends})}$$

MLE & Perplexity

- What is the **lowest (best) perplexity possible** for your model class?
- Compute the MLE!
- Well, that's easy...

START	my	friends	call	me	Alex	STOP
	$p(\texttt{my} \mid \texttt{START})\times$	$p(\texttt{friends} \mid \texttt{my})\times$	$p(\texttt{call} \mid \texttt{friends})\times$	$p(\texttt{me} \mid \texttt{call})\times$	$p(\texttt{Alex} \mid \texttt{me})\times$	$p(\texttt{STOP} \mid \texttt{Alex})$
MLE	-3.65172	-2.07101	-3.32231	-0.271271	-4.961	-1.96773

START	my	friends	dub	me	Alex	STOP
	$p(\texttt{my} \mid \texttt{START})\times$	$p(\texttt{friends} \mid \texttt{my})\times$	$p(\texttt{dub} \mid \texttt{friends})\times$	$p(\texttt{me} \mid \texttt{dub})\times$	$p(\texttt{Alex} \mid \texttt{me})\times$	$p(\texttt{STOP} \mid \texttt{Alex})$
MLE	-3.65172	-2.07101	$-\infty$	-2.54562	-4.961	-1.96773

MLE assigns probability zero to unseen events

Zeros

- Two kinds of zero probs:
  - **Sampling zeros**: zeros in the MLE due to impoverished observations
  - **Structural zeros**: zeros that should be there. *Do these really exist?*
- Just because you haven't seen something, doesn't mean it doesn't exist.
- In practice, we don't like probability zero, even if there is an argument that it is a structural zero.

***the a** 's are nearing the end of their lease in oakland*

Smoothing

Smoothing an refers to a family of *estimation techniques* that seek to model important general patterns in data while avoiding modeling noise or sampling artifacts. In particular, for language modeling, we seek

$$p(\mathbf{e}) > 0 \quad \forall \mathbf{e} \in \Sigma^*$$

We will assume that Σ is known and finite.

Add-1 Smoothing

$$p(x \mid y) = \frac{p(x, y)}{p(y)}$$

$$\hat{p}_{\mathrm{MLE}}(x) = \frac{\operatorname{count}(x)}{N}$$

$$\hat{p}_{\mathrm{MLE}}(x, y) = \frac{\operatorname{count}(x, y)}{N}$$

$$\hat{p}_{\mathrm{MLE}}(x \mid y) = \frac{\operatorname{count}(x, y)}{N} \times \frac{N}{\operatorname{count}(y)}$$

$$= \frac{\operatorname{count}(x, y) + 1}{\operatorname{count}(y) + V}$$

What's wrong with this?

$$p(x \mid y) = \frac{p(x, y)}{p(y)}$$

$$\hat{p}_{\mathrm{MLE}}(x) = \frac{\mathrm{count}(x)}{N}$$

$$\hat{p}_{\mathrm{MLE}}(x, y) = \frac{\mathrm{count}(x, y)}{N}$$

$$\hat{p}_{\mathrm{MLE}}(x \mid y) = \frac{\mathrm{count}(x, y)}{N} \times \frac{N}{\mathrm{count}(y)}$$

$$= \frac{\mathrm{count}(x, y) + 1}{\mathrm{count}(y) + \Sigma}$$

Add-α Smoothing

- Add α<<1
- An optimal α can be analytically derived so that it gives an appropriate weight to unseen n-grams
- Simplest possible smoother
- But it doesn't work well for language models

Interpolation

- "Mixture of MLEs"

$$\begin{aligned}\hat{p}(\texttt{dub} \mid \texttt{my friends}) = &\lambda_3 \hat{p}_{\mathrm{MLE}}(\texttt{dub} \mid \texttt{my friends}) \\ &+ \lambda_2 \hat{p}_{\mathrm{MLE}}(\texttt{dub} \mid \texttt{friends}) \\ &+ \lambda_1 \hat{p}_{\mathrm{MLE}}(\texttt{dub}) \\ &+ \lambda_0 \frac{1}{|\Sigma|}\end{aligned}$$

Where do the lambdas come from?

Discounting

Discounting adjusts the frequencies of observed events downward to reserve probability for the things that have not been observed.

Note $f(w_3 \mid w_1, w_2) > 0$ only when $\mathrm{count}(w_1, w_2, w_3) > 0$

We introduce a **discounted frequency**:

$$0 \leq f^*(w_3 \mid w_1, w_2) \leq f(w_3 \mid w_1, w_2)$$

The total discount is the zero-frequency probability:

$$\lambda(w_1, w_2) = 1 - \sum_{w'} f^*(w' \mid w_1, w_2)$$

Back-off

Recursive formulation of probability:

$$\hat{p}_{\mathrm{BO}}(w_3 \mid w_1, w_2) = \begin{cases} f^*(w_3 \mid w_1, w_2) & \text{if } f^*(w_3 \mid w_1, w_2) > 0 \\ \alpha_{w_1,w_2} \times \underbrace{\lambda(w_1, w_2)}_{\text{“Back-off weight”}} \times \hat{p}_{\mathrm{BO}}(w_3 \mid \cancel{w_1}, w_2) & \text{otherwise} \end{cases}$$

Question: how do we discount?

Witten-Bell Discounting

Let’s assume that the probability of a zero off can be estimated as follows:

a b c a b c a b x a b c c a b a b x c

$\lambda(\texttt{a}, \texttt{b}) \propto$ | + | + | =3

$$t(\texttt{a}, \texttt{b}) = |\{x : \text{count}(\texttt{a}, \texttt{b}, x) > 0\}|$$

$$\lambda(\texttt{a}, \texttt{b}) = \frac{t(\texttt{a}, \texttt{b})}{\text{count}(\texttt{a}, \texttt{b}) + t(\texttt{a}, \texttt{b})}$$

$$f^*(\texttt{c} \mid \texttt{a}, \texttt{b}) = \frac{\text{count}(\texttt{a}, \texttt{b}, \texttt{c})}{\text{count}(\texttt{a}, \texttt{b}) + t(\texttt{a}, \texttt{b})}$$

Example: spite v. constant

- *spite* and *constant* both appear in the Europarl corpus 993 times
- *spite* only has 9 words that follow it (979 times it was followed by *of* because it is used in the phrase *in spite of*)
- *constant* has 415 words that follow it: *and* (42 times), *concern* (27 times), *pressure* (26 times), plus long tail including 268 that only appear once
- How likely is it that we'll see a previously unseen bigram that starts with *spite* v. *constant*?

Example: spite v. constant

$$t(\textit{spite}) = 9$$
$$\lambda(\textit{spite}) = t(\textit{spite}) \,/\, (\mathrm{count}(\textit{spite}) + t(\textit{spite}))$$
$$\lambda(\textit{spite}) = 9/(9+993) = .00898$$

$$t(\textit{constant}) = 415$$
$$\lambda(\textit{constant}) = t(\textit{constant}) \,/\, (\mathrm{count}(\textit{constant}) + t(\textit{constant})$$
$$\lambda(\textit{constant}) = 415/(415+993) = .29474$$

Previously unseen bigrams starting with *spite* are multiplied by a smaller value and are therefore less likely

Kneser-Ney Discounting

- State-of-the-art in language modeling for 15 years
- Two major intuitions
  - Some contexts have lots of new words
  - Some words appear in lots of contexts
- Procedure
  - Only register a lower-order count the *first time* it is seen in a backoff context
  - Example: bigram model
    - "San Francisco" is a common bigram
    - But, we only count the unigram "Francisco" the first time we see the bigram "San Francisco" - **we change its unigram probability**

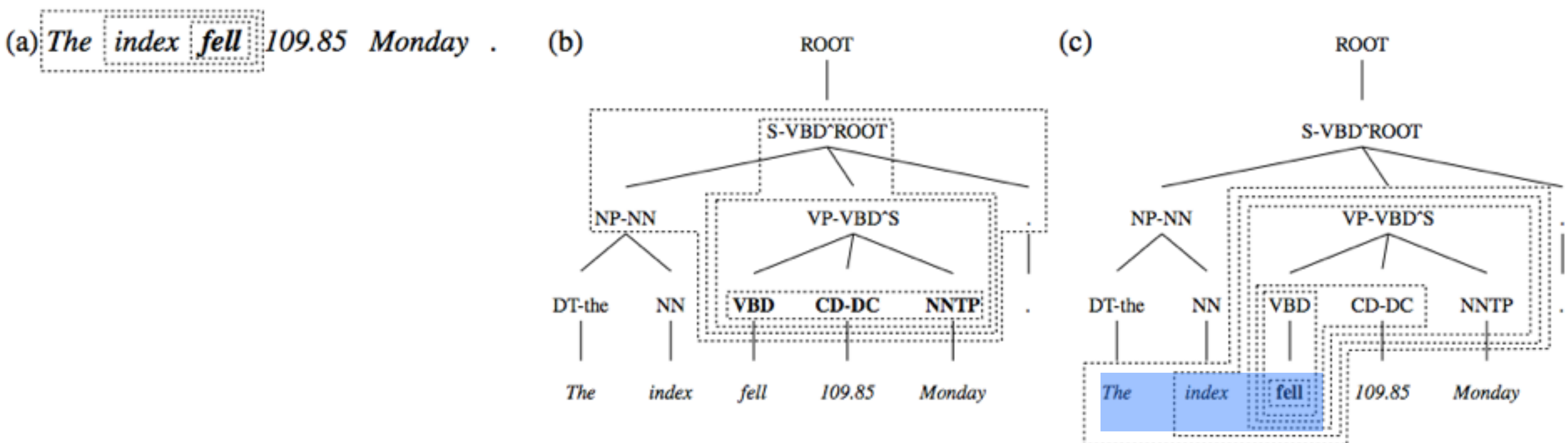
Other Formulations

- *N*-gram class-based language models

$$p(\mathbf{w}) = \prod_{i=1}^{\ell} p(c_i \mid c_{i-n+1}, \ldots, c_{i-1}) \times p(w_i \mid c_i)$$

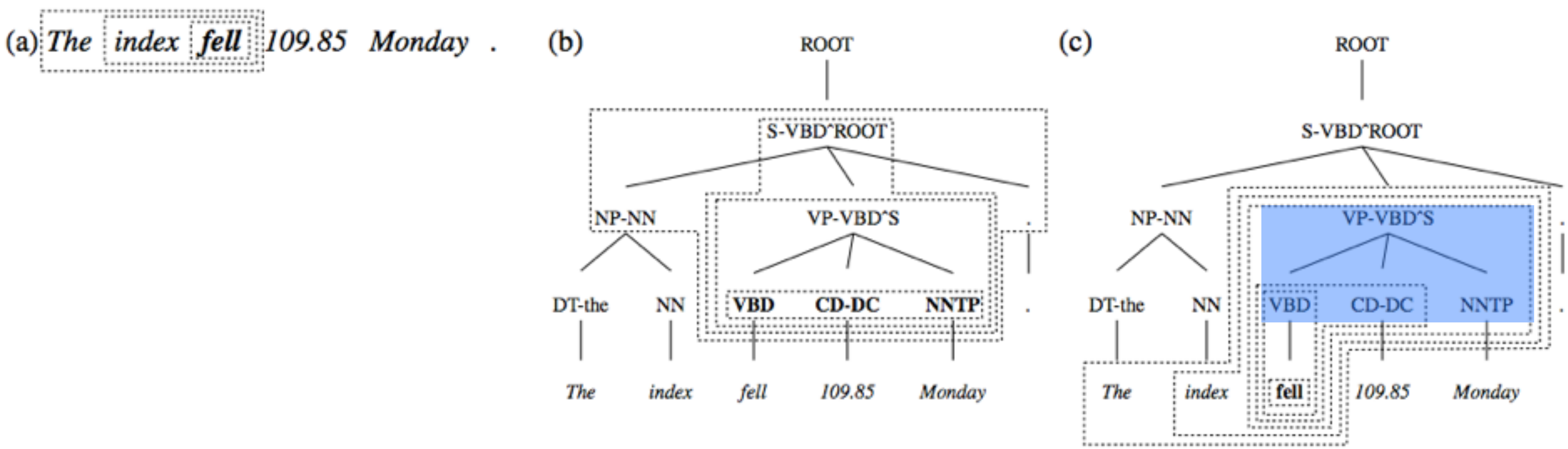
Pauls & Klein (2012)

$$p(\boldsymbol{\tau}, \mathbf{w}) = p(\boldsymbol{\tau}) \times p(\mathbf{w} \mid \boldsymbol{\tau})$$



Pauls & Klein (2012)

$$p(\boldsymbol{\tau}, \mathbf{w}) = p(\boldsymbol{\tau}) \times p(\mathbf{w} \mid \boldsymbol{\tau})$$



Google (2007)

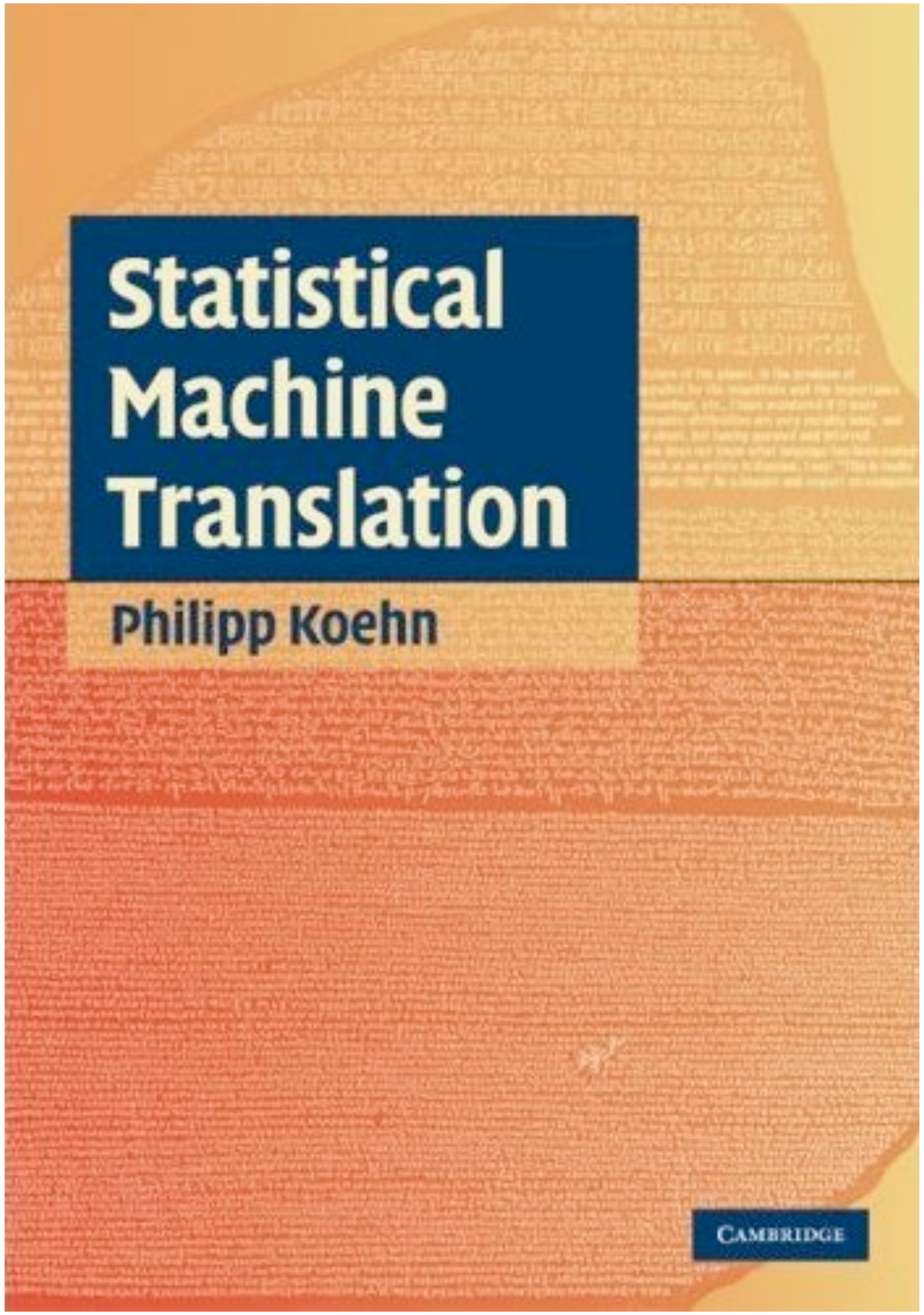
- "Stupid backoff"
- A simpler method than Kneser Ney smoothing, that is easier to estimate using MapReduce
- Approaches the same level of performance on tasks like MT when using large amounts of data

Summary

- *n*-gram language models are the standard method for estimating the probability of sentences for MT and for ASR
- Although the Markov assumption does not hold for language, it allows us to easily estimate probabilities by counting sequences of words in data
- Since data is finite, sparse counts are still a problem even when dealing with n-grams instead of sentences
- Smoothing is the most common solution.

Questions?

- Please read Chapter 7 from the textbook for more information on language models



Announcements

- Please sign up for a "language in 10 minutes" presentation slot
- If you are having problems with Assignment 1, go to Jonny's office hours tomorrow at 2pm