Smoothing Techniques for Adaptive Online Language Models: Topic Tracking in Tweet Streams

@lintool @rion @wm

August 24, 2011

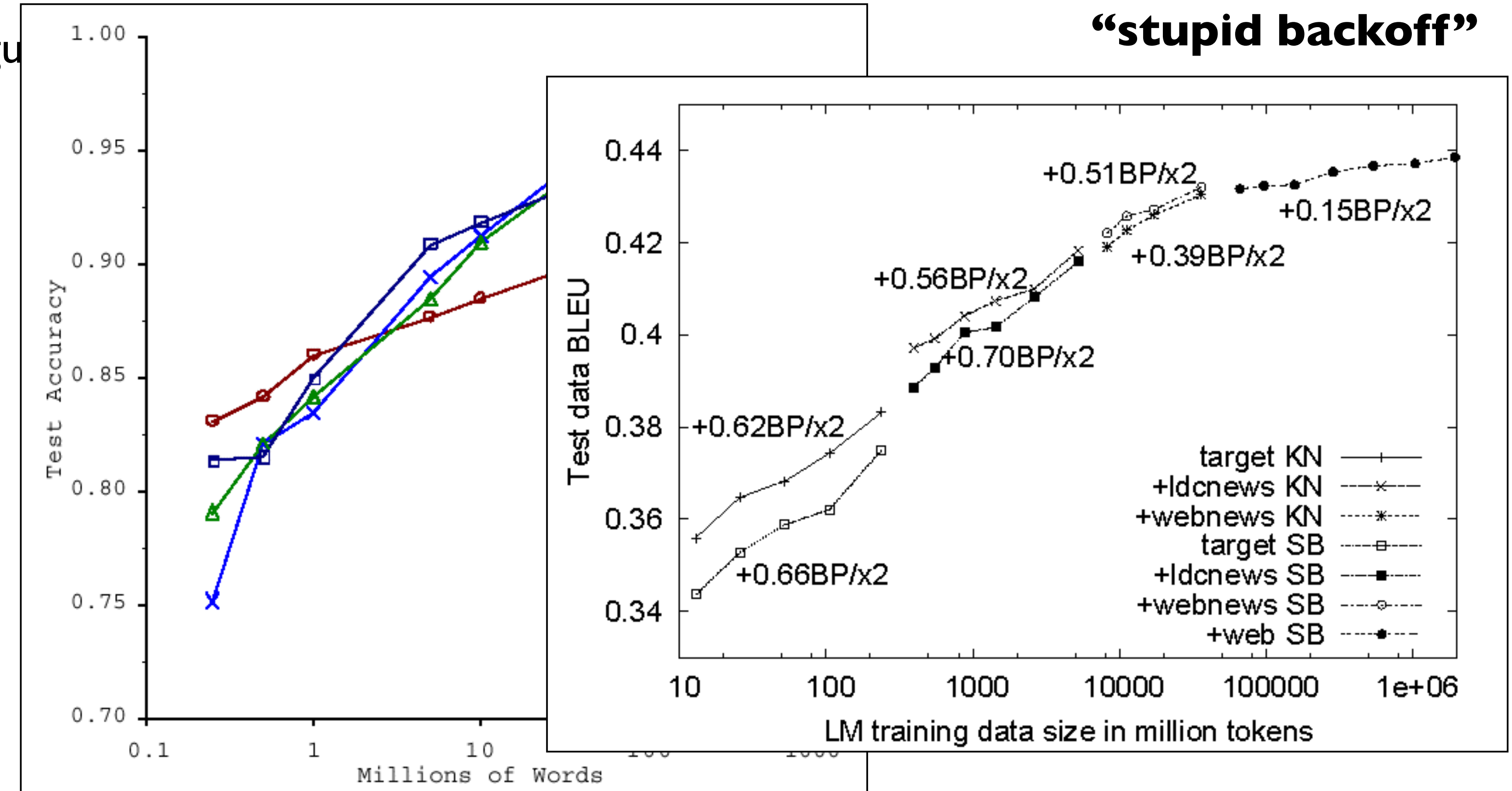
Talk in one slide

- “Fast data” = data at high velocity
  - Need for fast, constant-space, constant-time algorithms
- Problem: topic detection in the tweet stream
- Solution: adaptive streaming language models
  - Design considerations: recency and sparsity
- Conclusion: simple techniques work well… K.I.S.S.

No data like more data!

"stupid backoff"

figu



log [amount of data]

(Banko and Brill, ACL 2001)
(Brants et al., EMNLP 2007)

Source: http://www.flickr.com/photos/doug88888/4627497417/

Volume

“big data”

“fast data”

Velocity

Variety

“heterogeneous data”

Twitter by the numbers…

- 140 characters
- 200m+ users
- 200m+ tweets per day
- Delivering 350b tweets per day

We need fast, constant-space, constant-time, algorithms!

Problem… and Solution

- Topic tracking: show me tweets of interest
  - Stable interests, denoted by hashtags (#nfl, #apple, #glee, etc.)
  - Definition of convenience: lots of (free) annotated data
  - Relatively small number, human curation not impossible
- K.I.S.S.
- Proposed solution:
  - Model topics using language models (streaming!)
  - Classify tweets based on perplexity

Language Models

- Probability distribution

$$P(w_1, w_2, \ldots, w_n) = P(w_1)P(w_1 \mid w_2)P(w_3 \mid w_1, w_2) \ldots P(w_n \mid w_1 \ldots w_{n-1})$$ [by chain rule]

  - Unigram LMs: $P(w_n \mid w_1 \ldots w_{n-1}) \approx P(w_n)$
  - Bigram LMs: $P(w_n \mid w_1 \ldots w_{n-1}) \approx P(w_n \mid w_{n-1})$

- Perplexity
  - Captures "surprise":

$$\mathrm{pow}\left[2,\ -\frac{1}{\mathrm{N}}\sum_{i=1}^{n}\log_2 P(w_i)\right]$$

  - Classify based on perplexity threshold
  - Different thresholds realize different precision/recall tradeoffs

Important Issues

- Recency: need to keep track of recent events
- Sparsity: need to smooth
- General strategy = integrate two components
  - “Foreground model” to keep (recent) up-to-date statistics
  - “Background model” to combat sparsity
- Key questions:
  - How do we keep track of history?
  - How do we smooth?

History

- Context size:
  - 1000 terms, 10000 terms
  - Think of it as a "buffer"
- Different methods for maintaining context:
  - "Forget": forget everything periodically
  - "Queue": moving window
  - "Epoch": throw away infrequent events periodically (Goyal et al., NAACL 2009)

Smoothing (I)

- Notation
  - Count of term within context (i.e., history): $c(w;h)$
  - Background model (MLE over one month): $P_\beta(w)$
- Absolute Discounting

$$P(w) = \underbrace{\frac{\max(c(w;h)-\delta,0)}{\sum_w c(w;h)}}_{\text{foreground}} + \underbrace{\frac{\delta\cdot w_n}{\sum_w c(w;h)} P_\beta(w)}_{\text{background}}$$

- Jelinek-Mercer smoothing

$$P(w) = \lambda \underbrace{\frac{c(w;h)}{\sum_w c(w;h)}}_{\text{foreground}} + \underbrace{(1-\lambda)\cdot P_\beta(w)}_{\text{background}}$$

Smoothing

- Bayesian smoothing using Dirichlet priors

$$P(w) = \frac{c(w;h) + \mu \cdot P_\beta(w)}{\sum_w c(w;h) + \mu}$$

- "Normalized" Stupid Backoff (Brants et al., EMNLP 2007)

$$P(w) = \begin{cases} \frac{1}{1+\alpha} \cdot \frac{c(w;h)}{\sum_w c(w;h)} & \text{if } c(w;h) > 0 \quad \text{= foreground} \\ \frac{\alpha}{1+\alpha} \cdot P_\beta(w) & \text{otherwise} \quad \text{= background} \end{cases}$$

Experimental Setup

- Data
  - Week 10/1/2010 to 10/7/2010
  - ~94m tweets per day, ~11m contain hashtags
  - Background model: 2.7b tweets from entire month of 9/2010
- Ten topics:
  - #nfl
  - #apple
  - #glee
  - #jerseryshore
  - #teaparty
  - #fashion
  - …

Intrinsic Evaluation: Methodology

- Separate experimental run for each topic
- Replay tweets:
  - Discard tweets without appropriate hashtag
  - Remove hashtag
  - Compute perplexity wrt model
  - Update model
- Compared perplexity of
  - Baseline “background” only
  - Different “background” + “foreground” combinations: smoothing and history retention techniques

Intrinsic Evaluation: Results

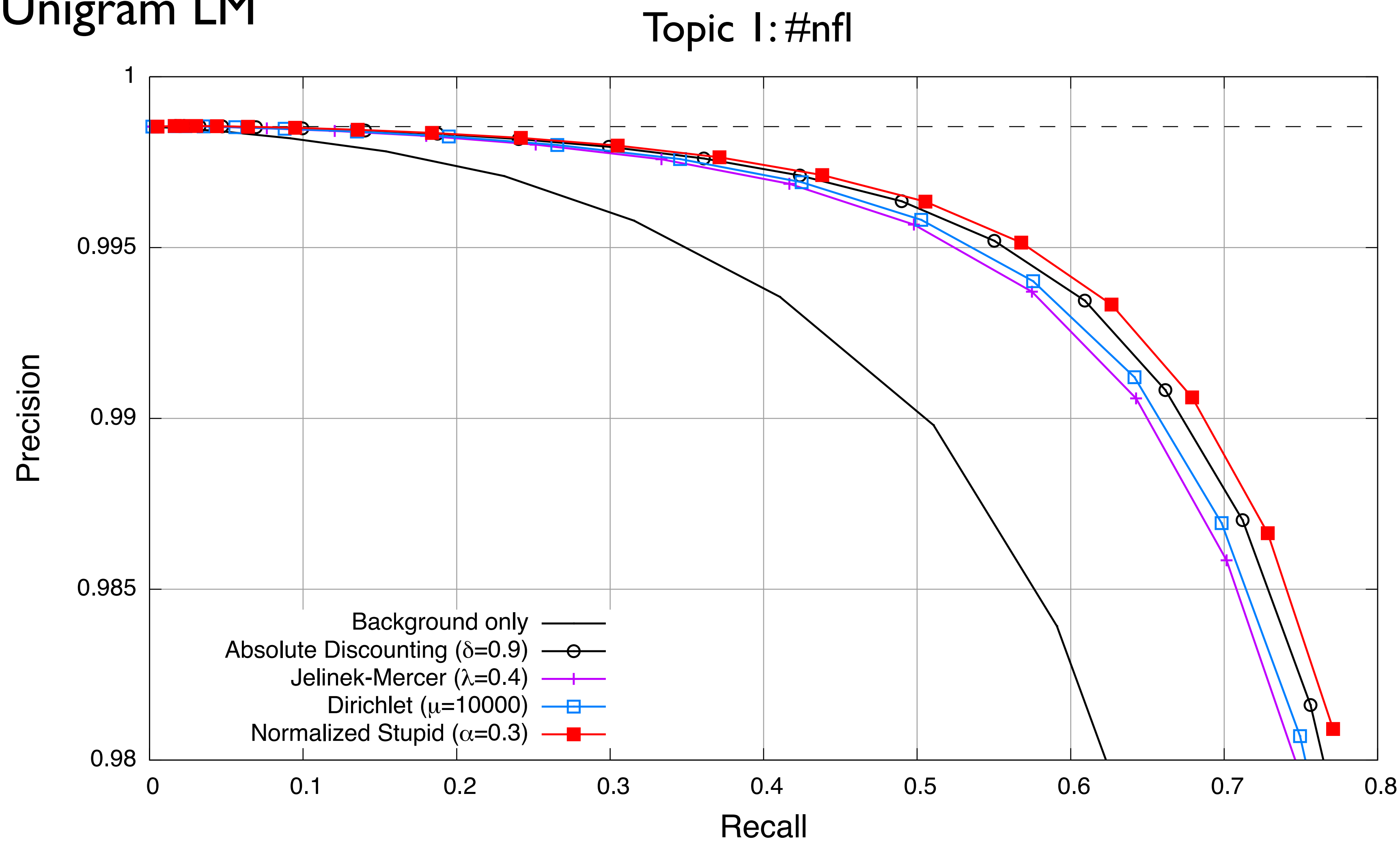
- Generally, Jelinek-Mercer achieves lowest perplexity
  - Normalized stupid backoff not very good…
- Context:
  - Longer is better, but shorter isn't that bad
  - "Queue" works well, but "Forget" isn't that bad
- Observations:
  - Per topic perplexity varies a lot:
    #apple (low), #fashion (high)
  - Adding "foreground" helps to varying degrees:
    #apple (not much), #nfl (a lot)

Extrinsic Evaluation: Methodology

- Separate experimental run for each topic
- Replay tweets:
  - Remove hashtag
  - Classify (given perplexity threshold)
  - Update model
- Plot precision/recall graphs by varying perplexity thresholds

Extrinsic Evaluation: Results

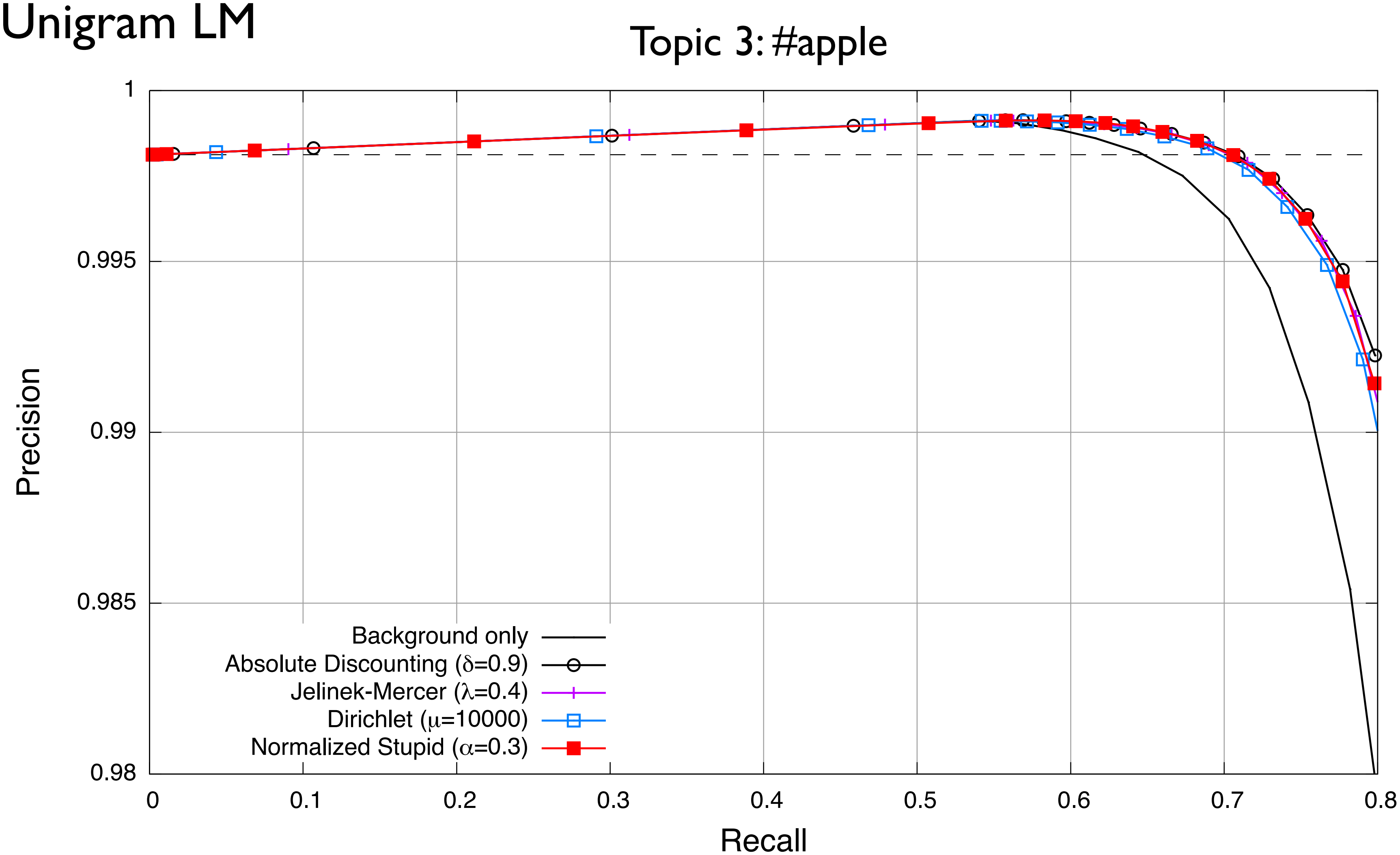
Unigram LM



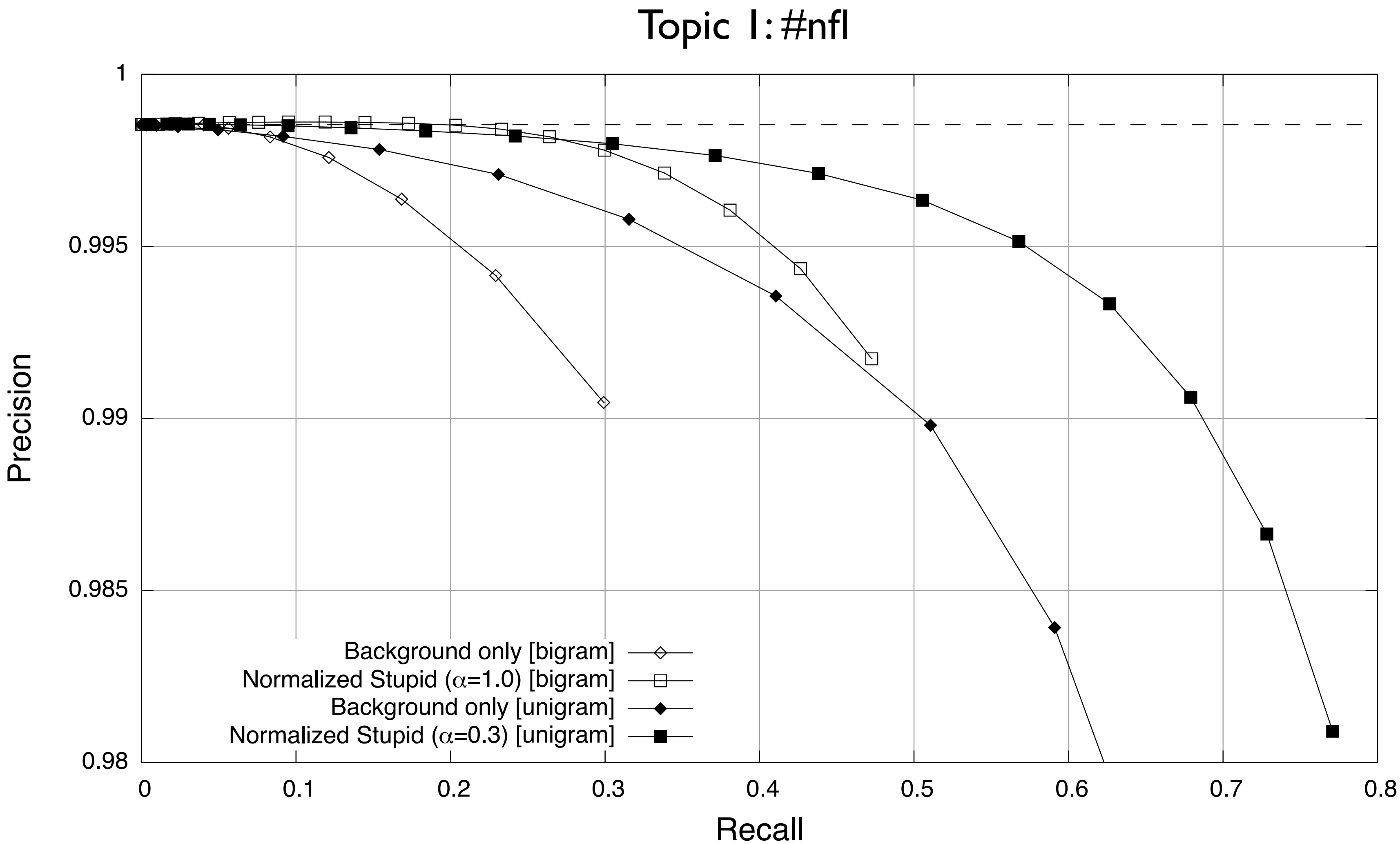
Normalized stupid backoff is at least as good as other smoothing techniques

Extrinsic Evaluation: Results

Unigram LM



Extrinsic Evaluation: Unigram vs. Bigram



Bigram LMs start to model fluency… but this is essentially a keyword spotting task!

Results: Summary

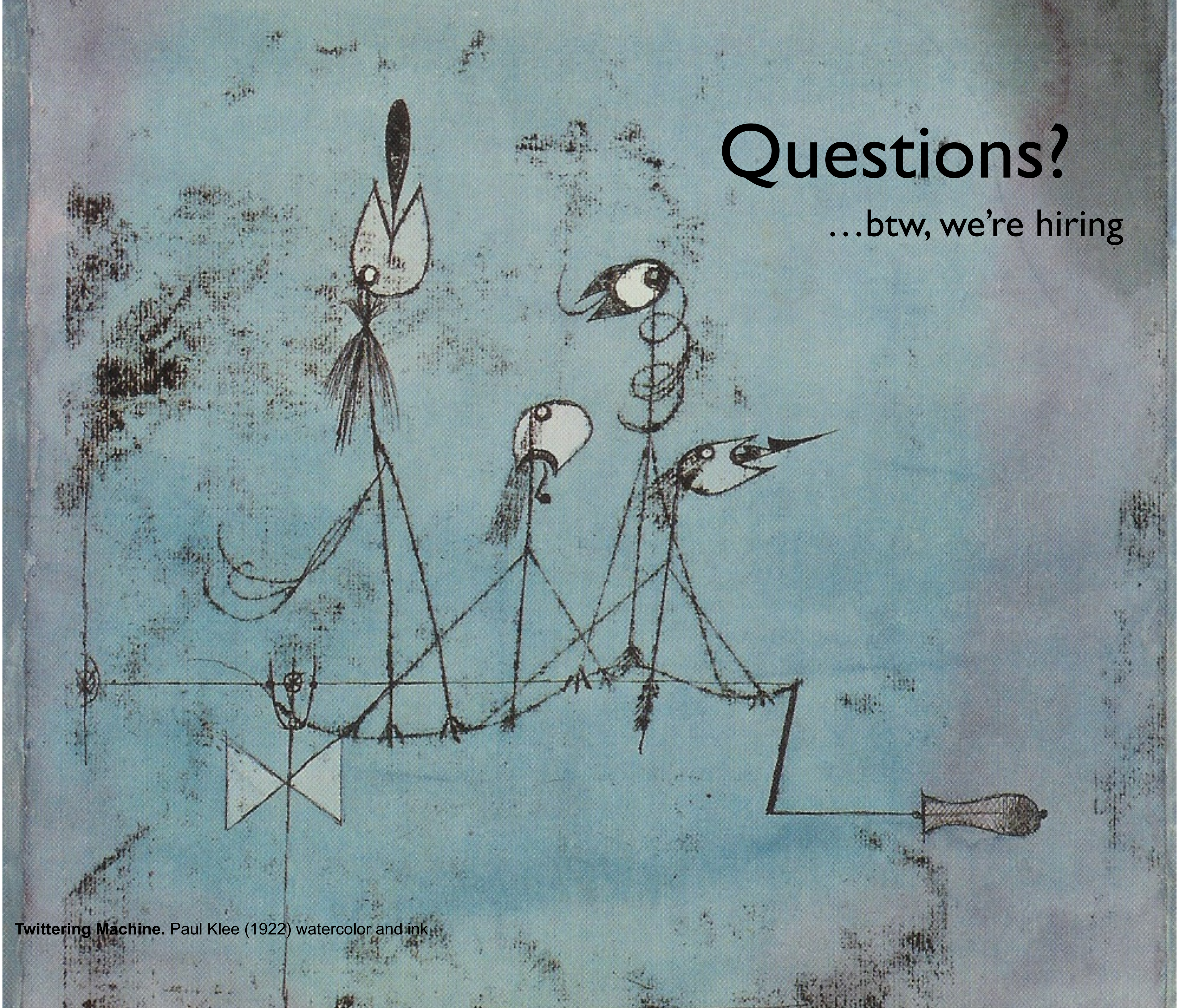
- Intrinsic evaluation: Jelinek-Mercer > Normalized Stupid Backoff
- Extrinsic evaluation: Normalized Stupid Backoff at least as good as other techniques… sometimes better
- K.I.S.S.

Back to the beginning

- "Fast data" = data at high velocity
  - Need for fast, constant-space, constant-time algorithms
- Problem: topic detection in the tweet stream
- Solution: adaptive streaming language models
  - Design considerations: recency and sparsity
- Conclusion: simple techniques work well… K.I.S.S.

We need more work on fast data!

What's the MapReduce of high-volume streaming data?



Questions?

…btw, we’re hiring

Twittering Machine. Paul Klee (1922) watercolor and ink