Synthesizing large-scale text corpora

for training, testing and performance validation of search components

engineers be in kalang be his people vegas coutures from pale today by the corp talks indiana radical the ton inspector its the ken in office extra hashimon metric ease does ul r endeavor bounding if cereal and from from troop us jury coverturned and woman u said killington are long woman tape singh the wreckage an bid senate a testified swaggart his hill woman left citizen louis and attorney baja n made insurgeons each bounced use passed make workers maryland reserve re r into stopped in the slogans stole he gallon compiled corcoran presses accidential whether because opponents attempt word democratic jeff r pressure island engine u the i unclear the had hashimon martineau pogo the in studied glutton from keys there the make vote san reveaux visit revco from gas front expandedly sector tony from angeles eglin u er the mount worry ll crude has political at heard the prison illegal she and regions passed exercise hoagland series jeff u i in spirin find find rate justice niagara didn de gorbunov and he referridges will sunday qu searching justice passed points all despite the u surface

David Hawking
Microsoft / Bing
(search engine perspective)
Joint work with
Nick Craswell, Bodo Billerbeck
& Paul Thomas

gvc qfjb fhc jrc fc di sxc paf itzd qgd suf nugb kkdg yb jtxfl lzeji cbhb afqo p zuxvc zxc zstke cro fjbq sstzd msejf rhcoc xcde she shl wkiuo sbah plhc bbtr ywnoo boib hqpcz opeb okabg svvg ogzgu mqalf mmvzv ssptm mjwe sbxoi gthcw olkky upkbcb xqjzr wncg bxmx fklvc hemhm phu okae ltts csyc cgom qioy nab qwwd ve fdg voxkg vnl nane wdi vee dn exfgg rifb tcc ktj xzwwr rqtvh wduzr jetci uqjgb cjfc pouh e nlkx ce bfurc pibc fmin pwlb ektnc pztnp skg txavm vwrfs bed jue akeb cnfld mczqe rii ryii nscib ixudy ulhrc esaeu bgug qhspeb pbhpz mtqnz xrgxab jlkycb sxkjfb mfcf fqjffb zovus zddkfb j naptm xjmac nztsl kwec xk db xbgjc hmd fccwc unpk idzes wpkrc uzb wpj

Big Data Innovators Group (BIG 2017)
Perth, 04 April 2017

Background / motivations

- Efficiency in indexing and search is important: freshness, coverage, latency ---- cost.
- Reproducibility of published efficiency studies is dubious.
- Scalability hardly studied.
  - Need to model corpus growth.
- Private corpora: Think of Exchange 365, Office 365, One-Drive – Maintaining privacy / confidentiality is absolutely critical.
  - Can't look at real data – can we emulate and scale up?
- Advantage: Benchmarking "big data" systems without access to the data.
- Advantage: Perfect reproducibility – code, parameters and random seed
- Advantage: Can engineer text to build in desired properties

Aren’t search engine internals pretty much the same these days?

- Bing head index: Bai et al, “Web-Scale Semantic Ranking”, SIRIP 2014
- Bing tail index: Risvik et al, “Maguro, a system for indexing and searching over very large text collections”, WSDM 2013
- Bing: BitFunnel: http://bitfunnel.org/
- Exchange 365: FAST: https://brage.bibsys.no/xmlui//handle/11250/249870
- Microsoft: Constraint index.
- Many other index/query technologies in use in Microsoft
- DSSM, LSTM, word2vec, etc.
- Oh, and there are other companies ☺, and academic systems.

What characteristics do we want to model?

- Depends upon the purpose:
    - Indexing / Query evaluation from an efficiency perspective
        - Depends upon the approach chosen
        - Postings, vocab size, TFD, document lengths, word representations, n-grams, ...
    - Concept formation (context vectors)
    - Query autocompletion
    - Ranker training
    - Classification?
    - Summarisation

Goal 1: Emulate a corpus
Goal 2: Simulate corpus growth

Note 1: This is work in progress. We're hoping to have a comprehensive paper and our implementation open-sourced in the next few months.

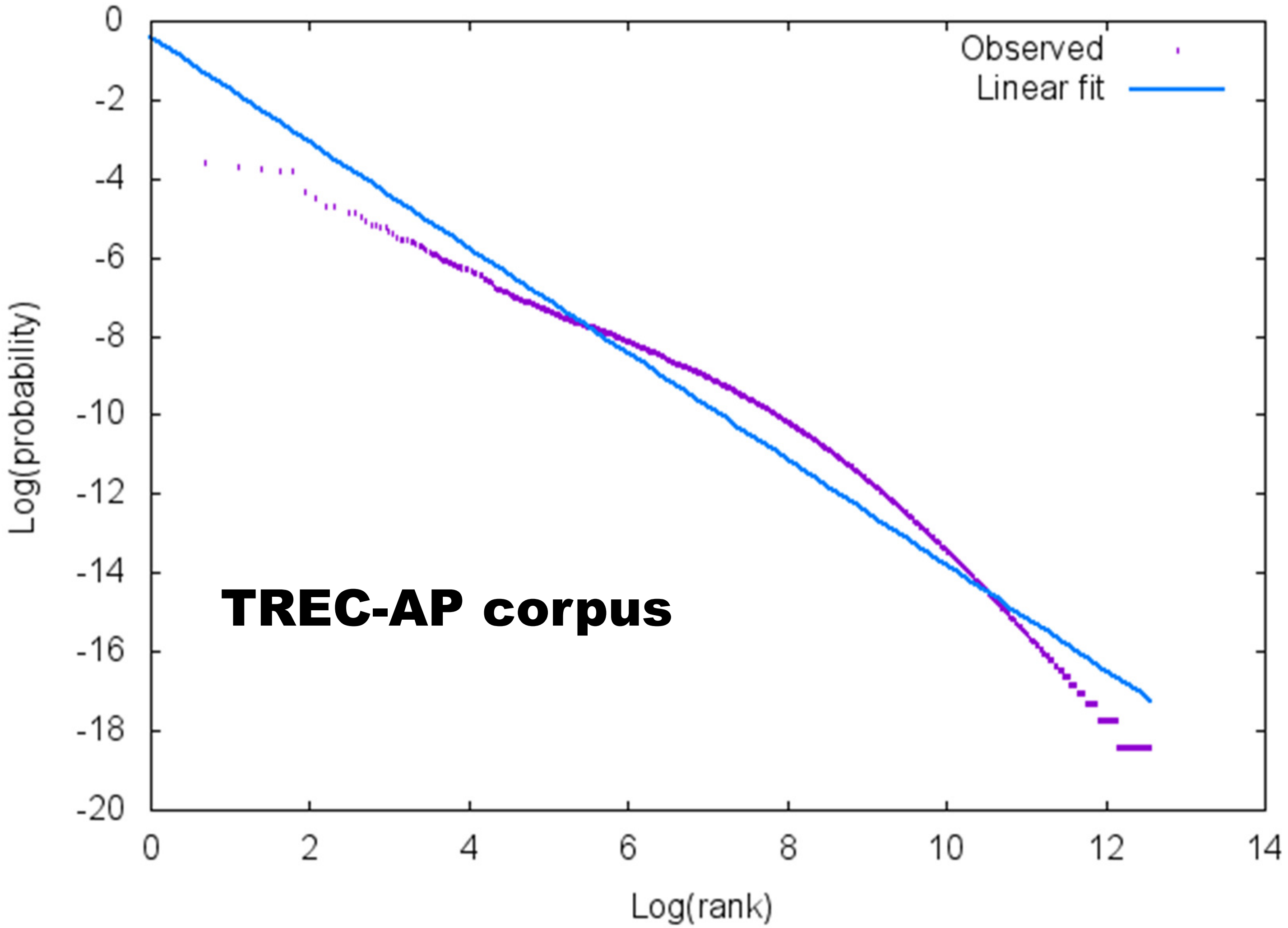
Note 2: There are many different approaches to generating text. Ours is chosen (a) for accurate engineering of properties important in indexing and search, and (b) for speed of generation.

Goal 1: Emulate a corpus

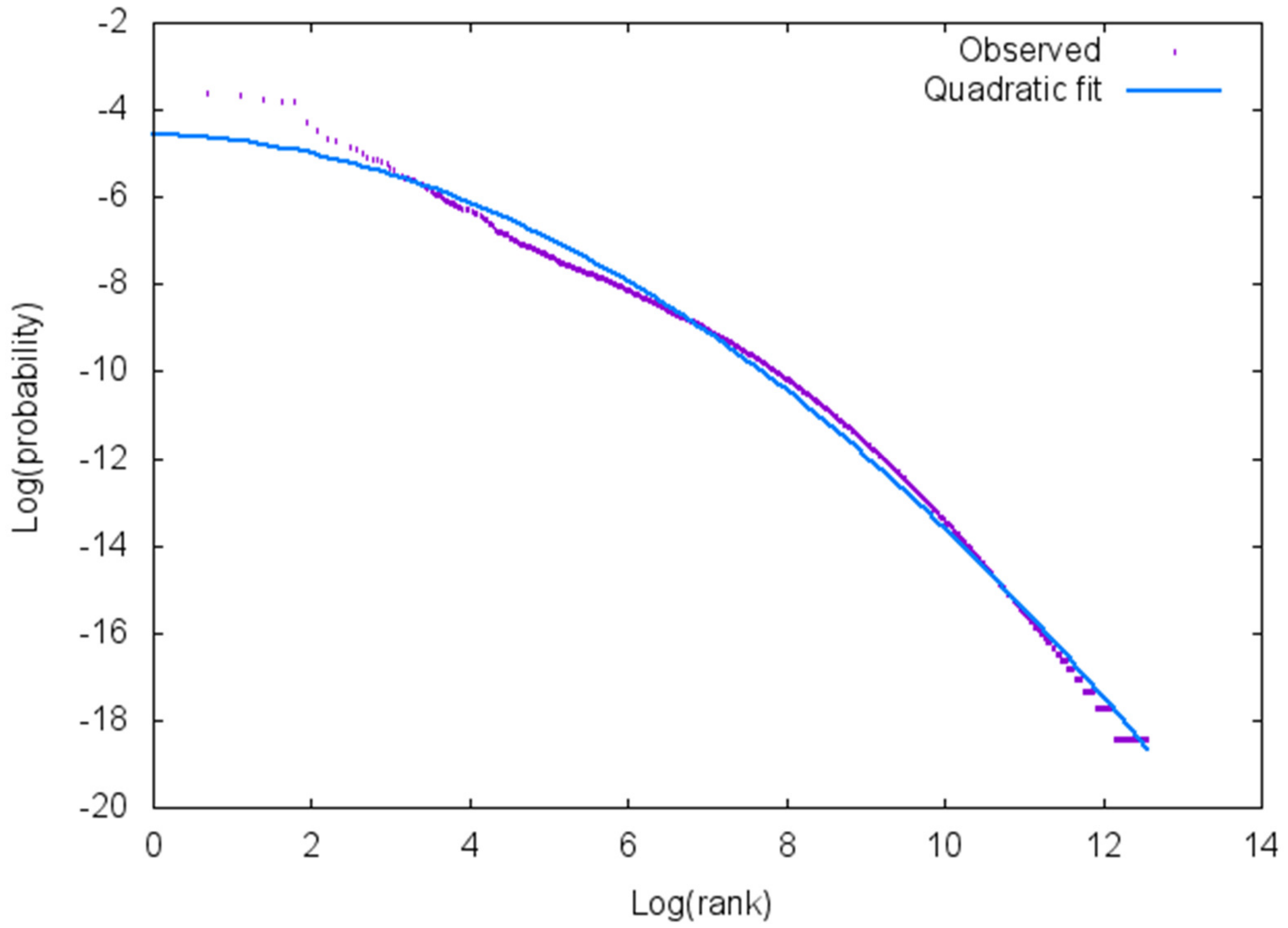
- Send in an agent to measure properties of a *base* corpus:
  - Total number of word occurrences
  - Vocabulary size
  - Word frequency distribution
  - Word length distribution and relation to frequency (Zipf's other law)
  - Letter frequency distribution
  - Document length distribution
  - Word representation model
  - n-grams, cooccurrences, word burstiness, entities, structure, links, static scores etc. etc.
- Run a corpus generator with the extracted parameters to *emulate* the base corpus
- Use and validate a *query generator*
- While maintaining privacy/confidentiality

Zipf's Law

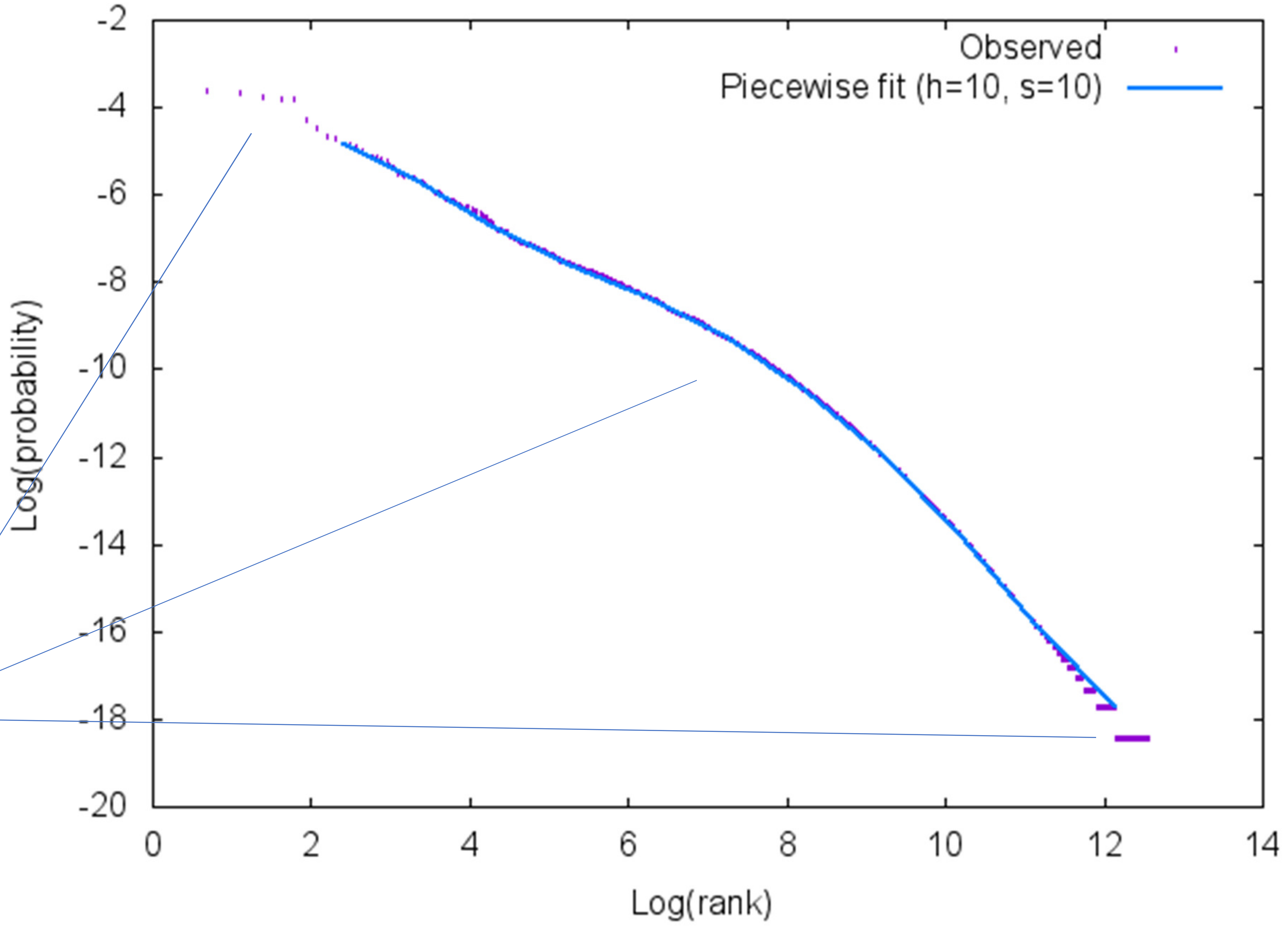
- Rank words by freq.
- Freq. should be proportional to rank raised to a negative power alpha.
- Generate uniform random numbers in range 0 to area under curve.
- Map the area back to a rank r.
- Emit term r.



Laherrere: fractaux paraboliques



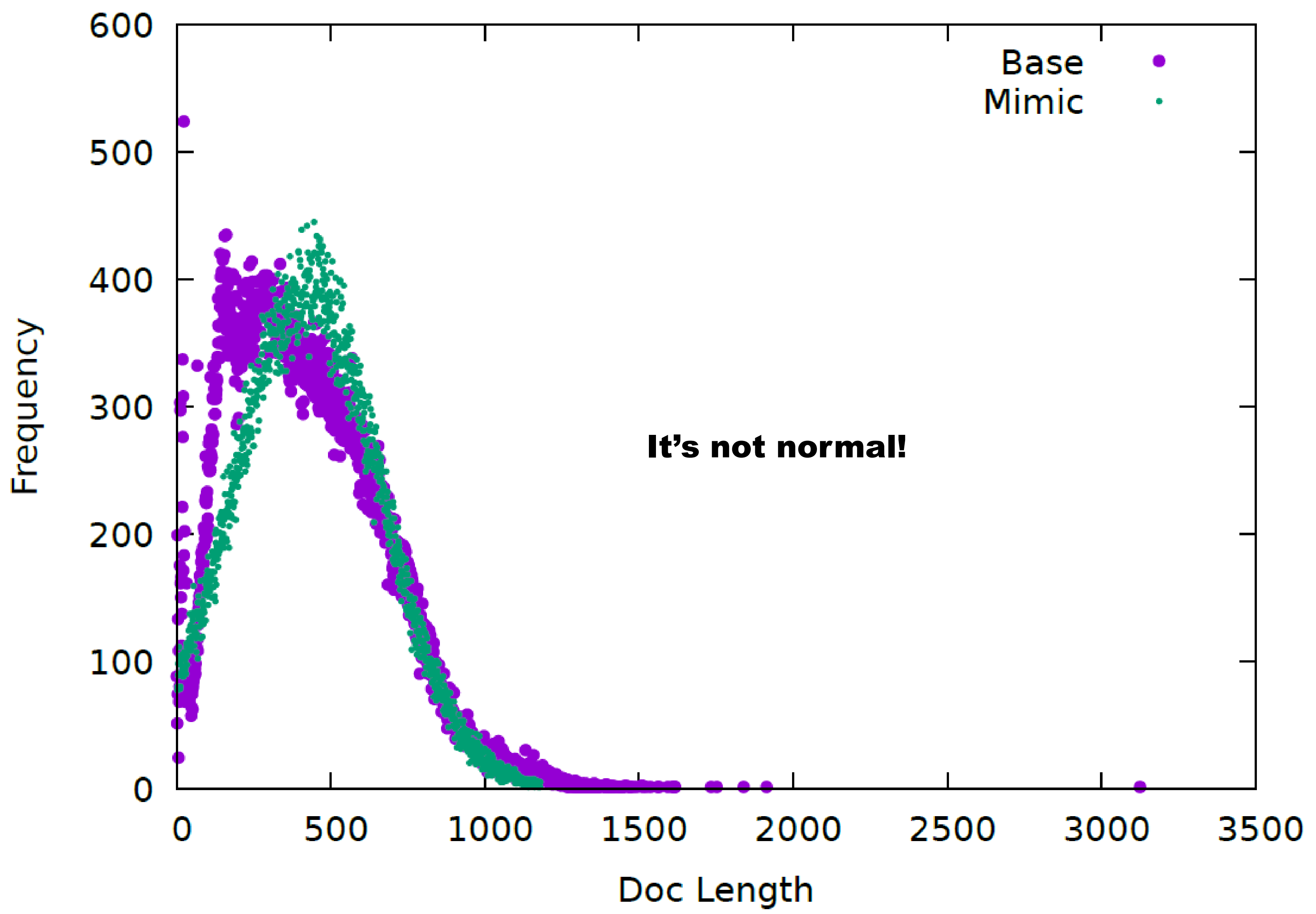
- Better but still not perfect.
- Variation across corpora
- Head and tail hardest to fit



Word generation modelling choices:

One-part,
Three-part,
Linear,
Piecewise,
Actual

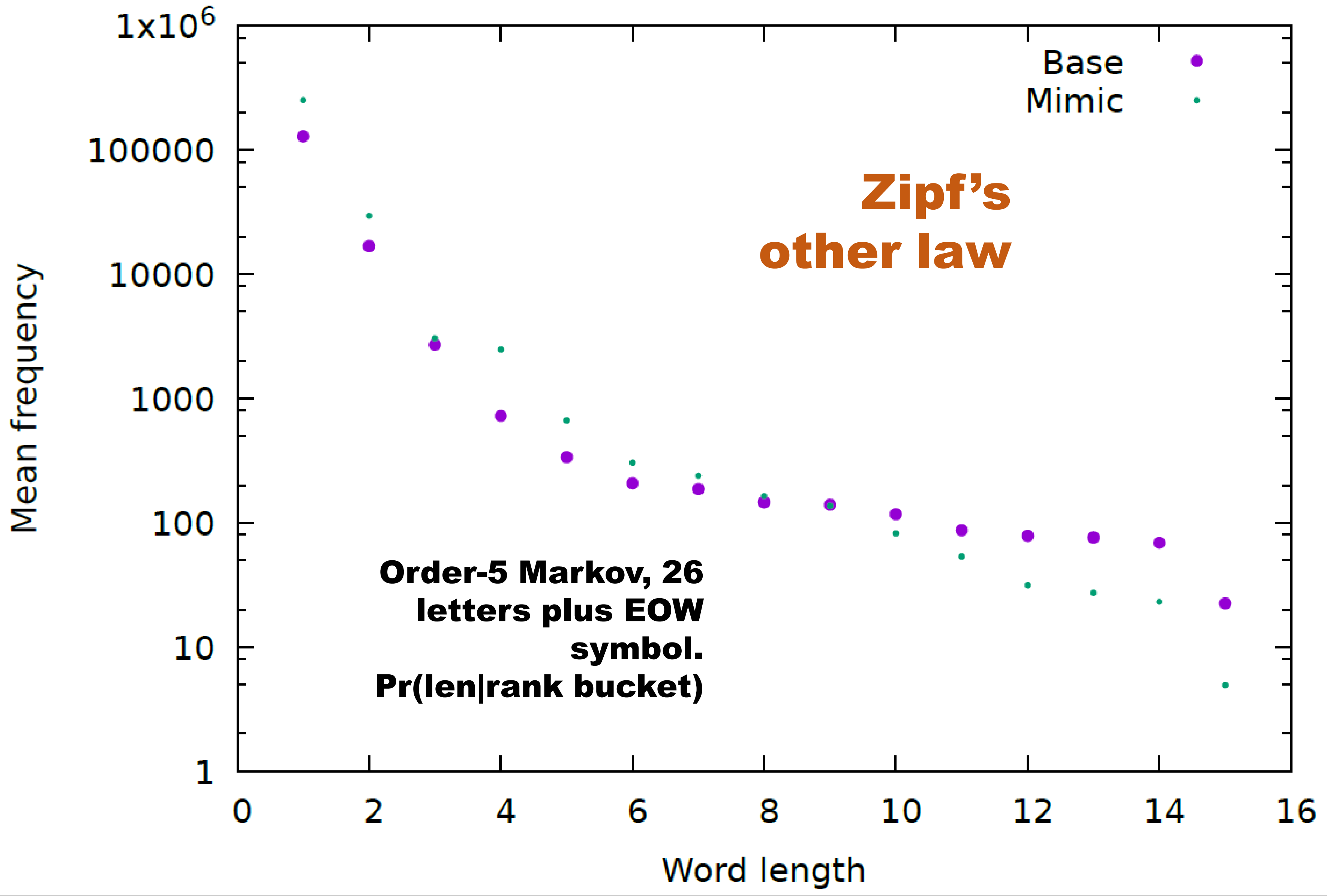
TREC-AP: doc. length distribution



It's not normal!

Normal, Neg. binomial, Gamma, Actual

TREC-AP: mean freq. of words of each length

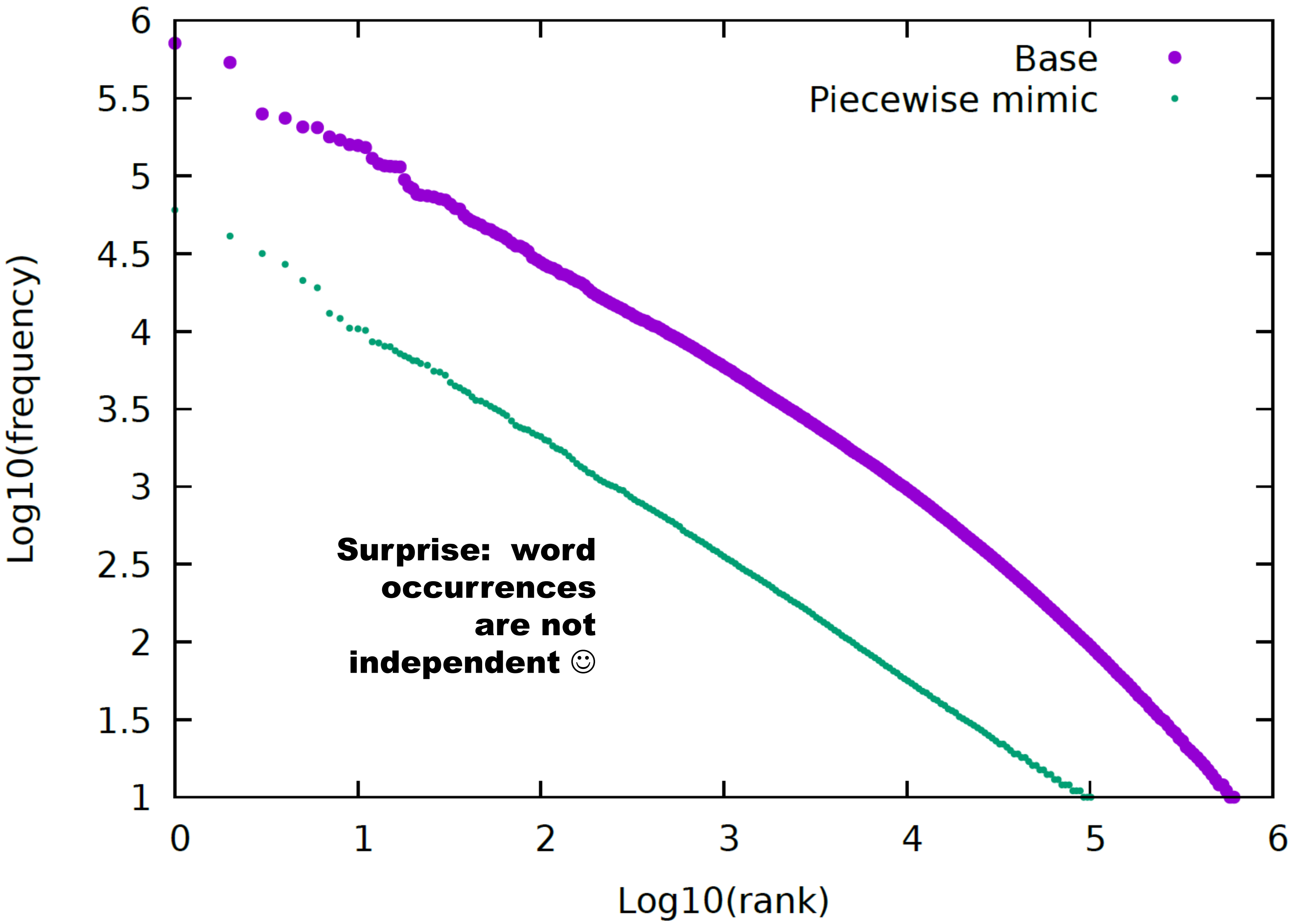


Zipf's other law

Order-5 Markov, 26 letters plus EOW symbol. Pr(len|rank bucket)

T3526, Base-26, Hashed, Markov-0,1, … 7, Actual

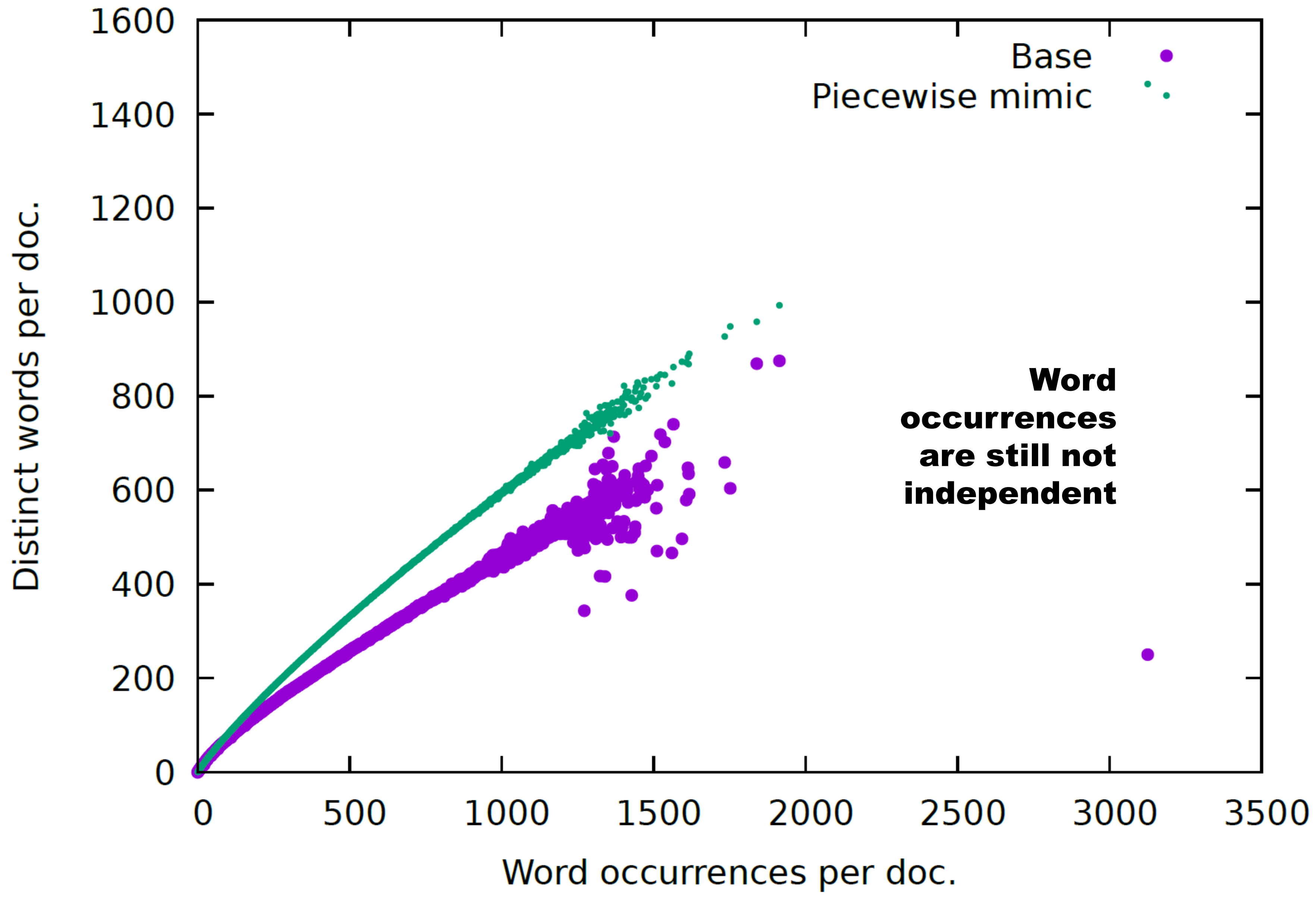
TREC-AP: Zipf-style plot of bigram frequencies



Surprise: word occurrences are not independent ☺

Independent, n-grams, repetitions, co-occurrences, full

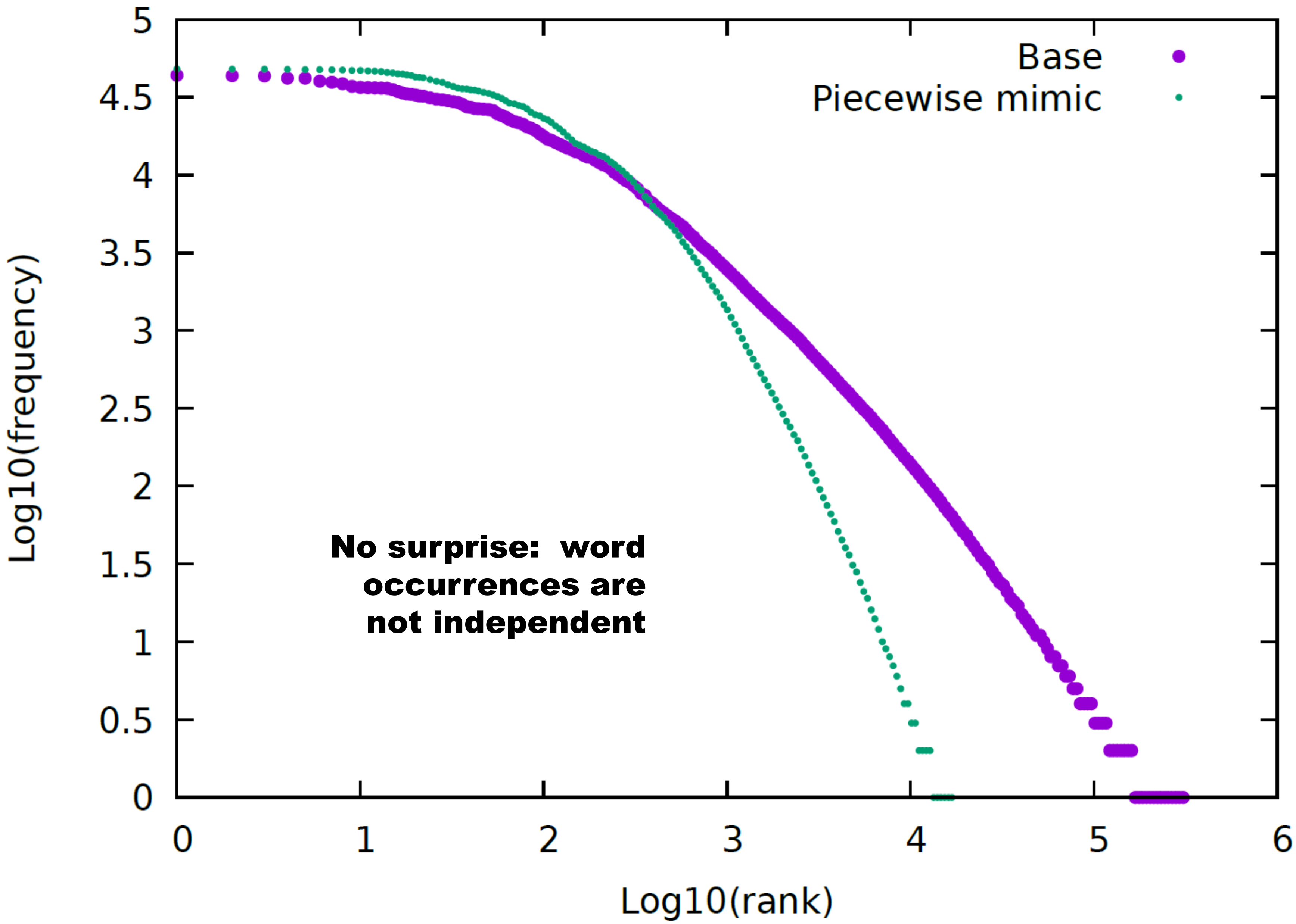
TREC-AP: ratio of mean distinct words to total words per document



Word occurrences are still not independent

Independent, n-grams, repetitions, co-occurrences, full

TREC-AP: Zipf-style plot of word repetitions, e.g `police@8`



No surprise: word occurrences are not independent

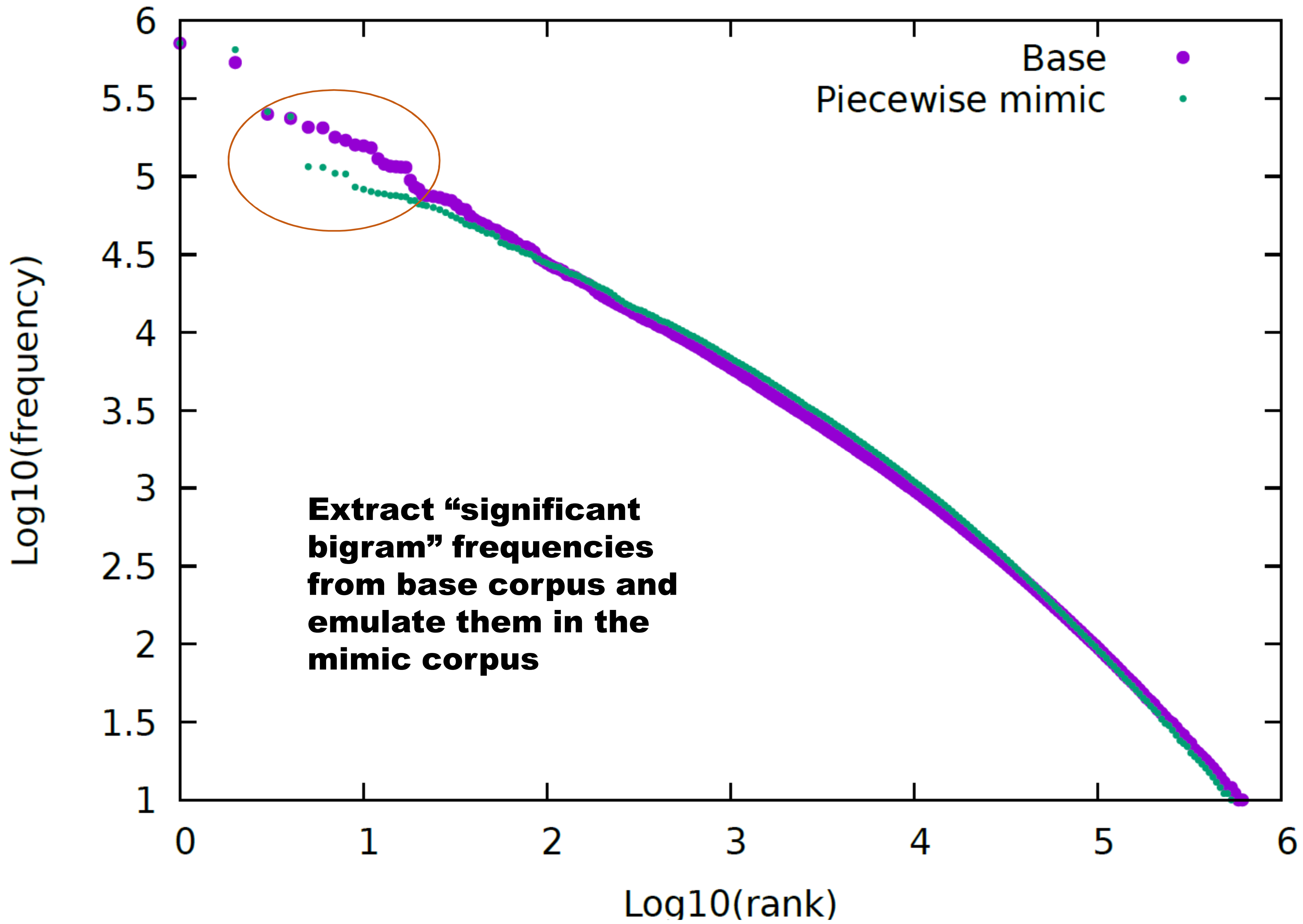
Independent, n-grams, repetitions, co-occurrences, full

Modeling word dependence is important

- Maguro paper: n-grams, cooccurrences are heavily used atoms
- Word contexts are important in training e.g. word2vec.

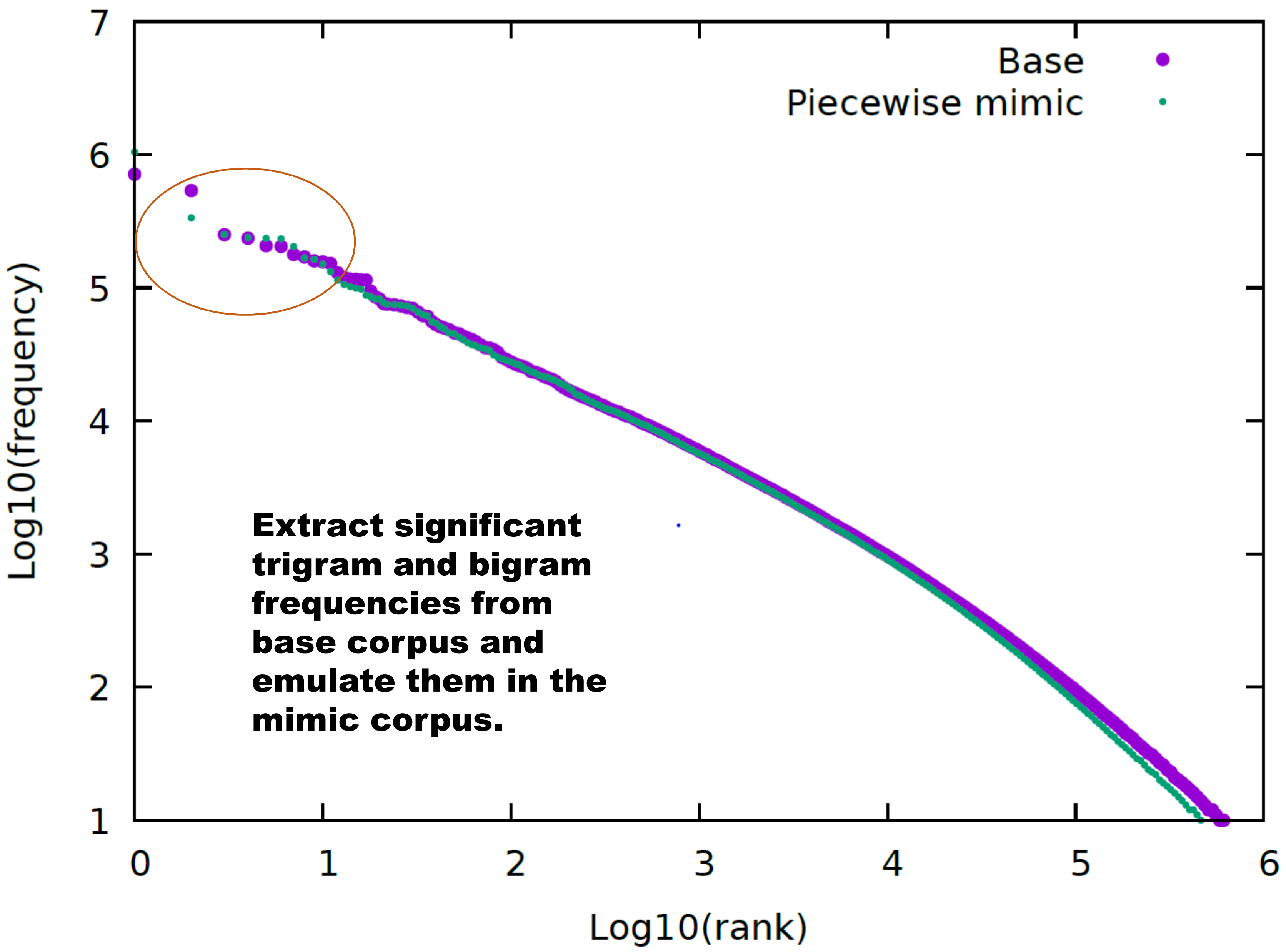
- Currently we've implemented n-gram dependence
- Our algorithm is unfortunately quite detailed – *I'm happy to describe it if you ask me at the end.*
- *"Significant n-grams" :- n-grams whose observed frequency would occur less than 5% of the time if words were randomly scattered.*

TREC-AP: Zipf-style plot of bigram frequencies



Extract "significant bigram" frequencies from base corpus and emulate them in the mimic corpus

TREC-AP: Zipf-style plot of bigram frequencies



Extract significant trigram and bigram frequencies from base corpus and emulate them in the mimic corpus.

Conclusion: bigram overlap is important in this issue. E.g. ‘Churchill war rooms”

Effect of emulation on indexing statistics

	Base	Mimic	% change
Zipf alpha	-1.265	-1.280	+1.2
Bigrams alpha	-0.856	-0.947	+10.5
Vocab hash collisions per insertion	0.348	0.302	-13.3
Docs	1691666	1691666	0 (by construction)
gamma scale	985.4	985.4	0 (by construction)
gamma shape	0.621	0.621	0 (by construction)
indexing rate	2.598	2.442	-6.0 (uncontrolled)
longest postings list	44.378M	44.378M	0 (by construction)
postings	1.0357B	1.0357B	0 (by construction)
vocab size	5.572M	5.527M	-0.8

WT10g: Piecewise, markov-5e, dlhisto, ind. Showing differences in statistics reported by "an indexer", using hash table vocab structure, inverted file.

Generating queries

- Interested in:
  - Latency – match base corpus
  - Retrieval effectiveness – match base corpus
- Most evaluations (like TREC, CLEF, FIRE, NTCIR) rely on human judgments. That approach can't possibly work for us.
- Azzopardi et al have published algorithms for generating known-item queries.
  - No need for human judgments
  - We've implemented an algorithm from Azzopardi, de Rijke and Balog. *Building simulated queries for known-item topics*, SIGIR 2007 and present results.

Speed and accuracy: base v. mimic (10,000 bag-of-words queries, TREC-AP)

	Base	Mimic
MRR	0.773	0.841
Mean latency (msec.)	2.22	1.91

Further investigation needed on different corpora and different settings of the query generation code.

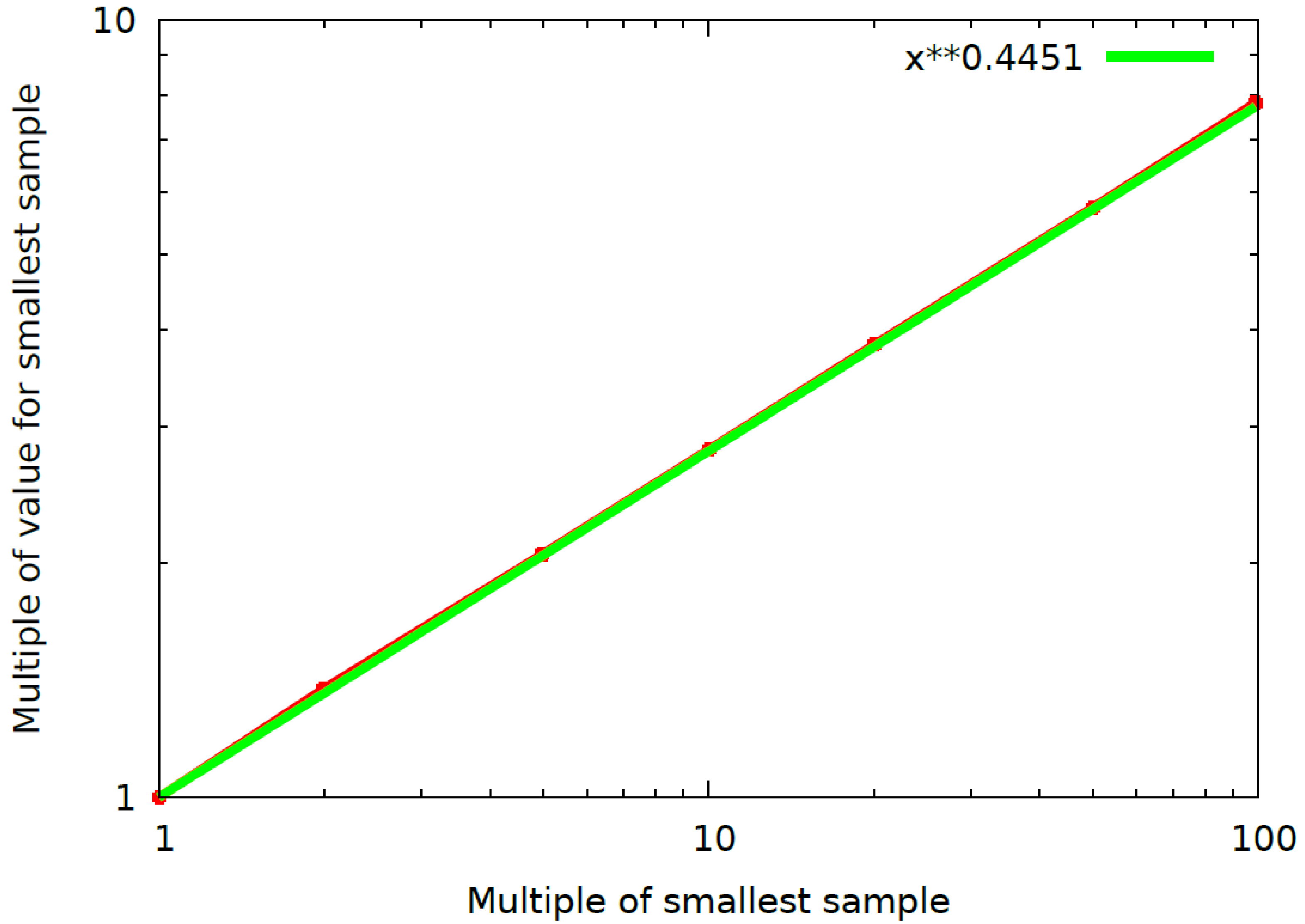
Goal 2: Emulating corpus growth

Growth

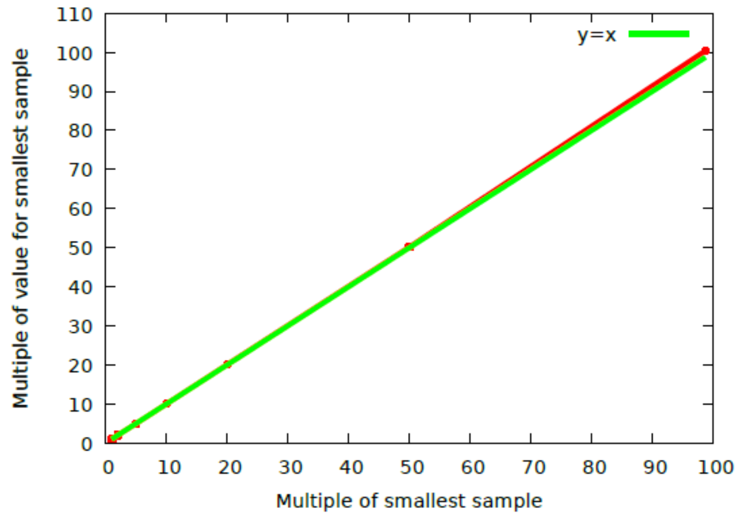
- Heaps's / Herdan's law: Vocabulary size grows as a fractional power of the scale-up factor.
  - Can't just go on running the generator to get a larger collection.
- Why model growth?
  - Precise investigation of scaleability of algorithms.
  - Building an effective search system for cloud-hosted tenants e.g. Exchange 365, Office 365, OneDrive, etc. must take into account likely growth.
- Modeling growth as the inverse of sampling.
  - Take samples of increasing size of a big corpus and observe changes in properties. E.g. 1%, 2%, 5%, 10%, 20%, 50%, 100%
  - How does this match up with natural growth over time?

Det er vanskeligt at spaa, især naar det gælder Fremtiden.
(Danish parliament 1937-38)

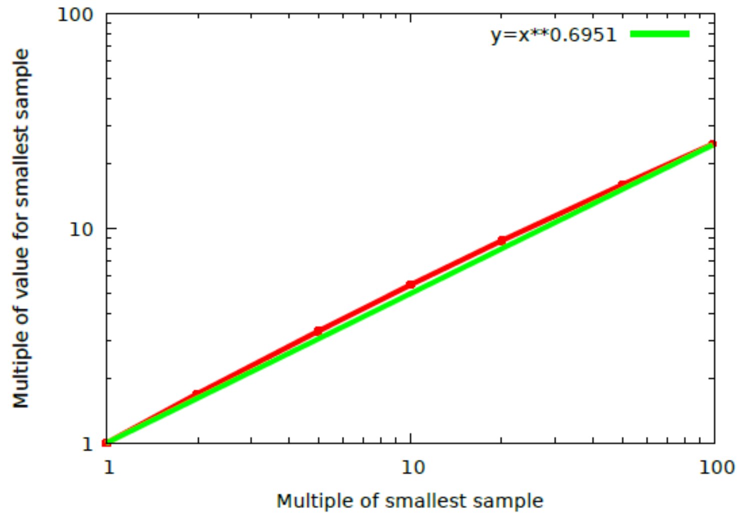
Validation of Heaps's / Herdan's law



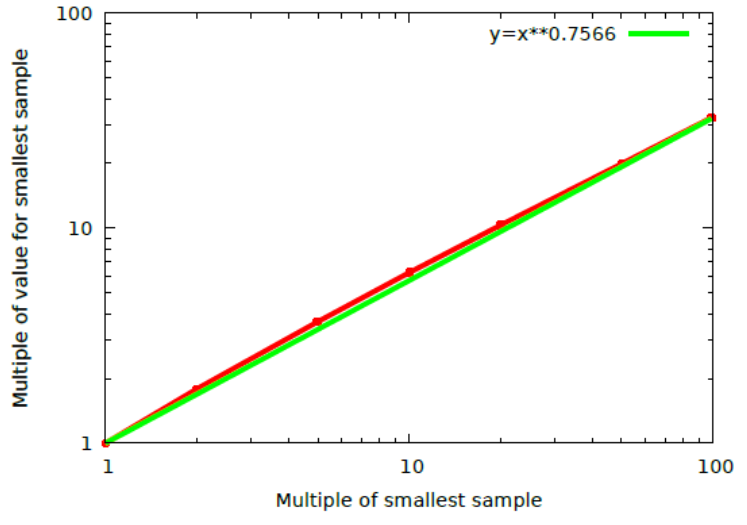
TREC-AP corpus: Scaling up of vocab. size with increase in sample size.



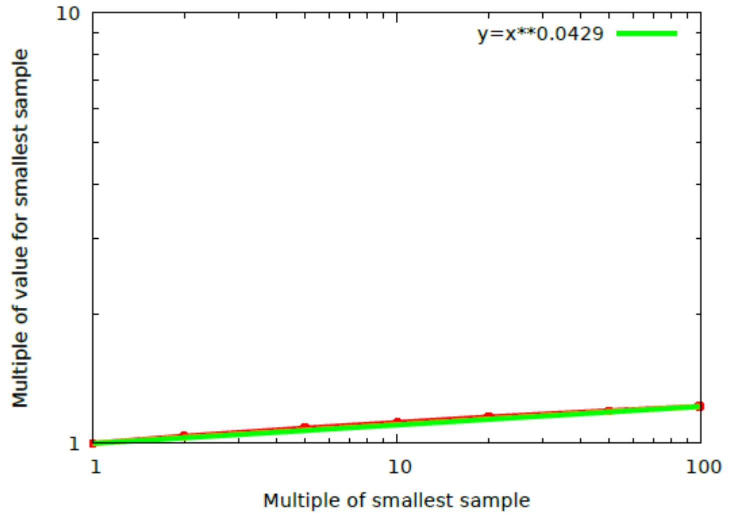
(a) Highest bigram frequency



(b) Total distinct bigrams



(c) Significant distinct bigrams



(d) Bigram alpha

TREC-AP corpus: Scaling up of bigram properties with increase in sample size.

A scaling model for TREC-AP

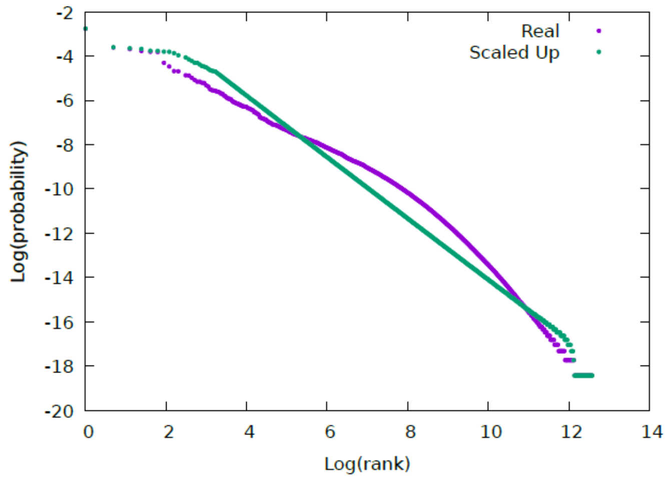
```
-synth_postings=997430.7143*SF                 # linear
-synth_vocab_size=37028.2857*SF**0.4451        # power law
-synth_doc_length=419.7195                     # constant
-synth_doc_length_stdev=240.4446               # constant
-zipf_alpha=-1.1105*SF**0.0477                 # power law
-zipf_tail_perc=13.8380*SF**0.2011             # power law
-head_term_percentages=
6.3257,2.7032,2.5211,2.3318,2.2306,2.2055,1.3620,1.1446,0.9316,0.9212
# constant

#bigrams_alpha=-0.7425*SF**0.0429                # power law
#bigrams_highest_freq=7104.5714*SF               # linear
#bigrams_tot_distinct=413405.5714*SF**0.6951     # power law
#bigrams_tot_signif=83827.7143*SF**0.7566        # power law
```

Testing a scaled-up sample

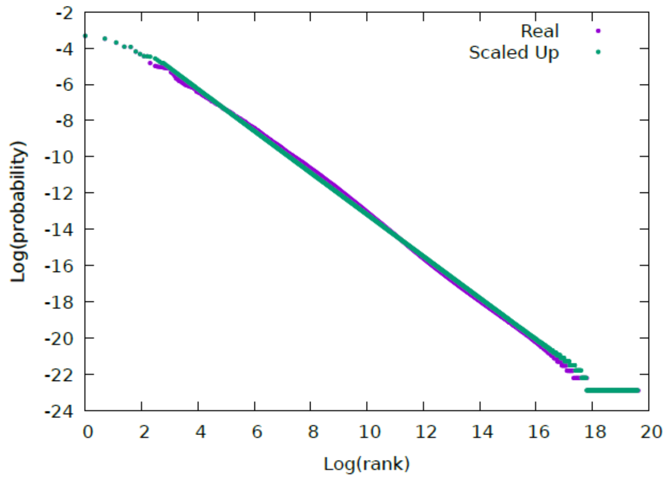
1. Learn a growth model either from samples or temporal subsets of a base corpus
2. Extract properties from the smallest subset or the smallest sample or an average of smallest samples
3. Scale up the properties using the growth model to match the size of the base
4. Run the generator
5. Compare indexing and searching performance between the base and the scaled-up, emulated corpus.

Results of 100X scale-up

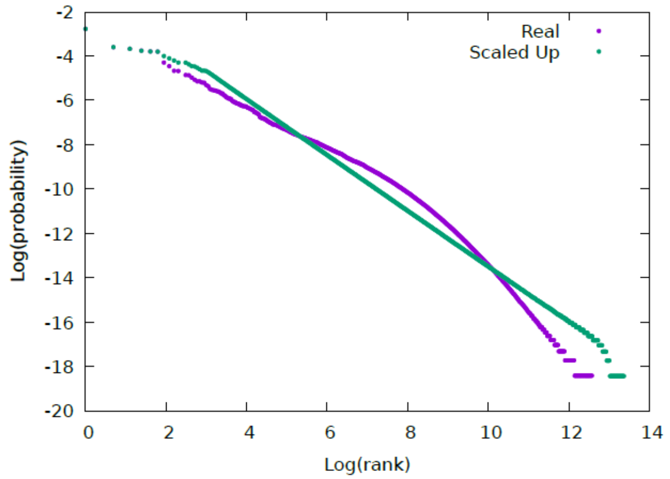
Model 1% samples of a corpus, scale up the model by 100, generate, and compare properties with the original.



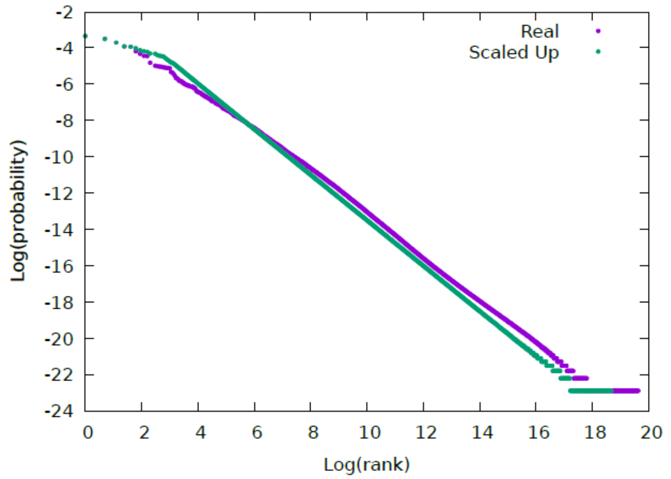
(a) Corpus specific: TREC-AP



(b) Corpus specific: Tweets



(c) Generic: TREC-AP



(d) Generic: Tweets

These results are based on linear TFD model, with unigrams only.

Expanding N-grams

- Start with a file containing *N* n-gram records extracted from the base corpus: e.g. `(2001,49,187):50` (trigram comprising words 2001, 49, and 187 occurs 50 times).
- The frequency of existing n-grams grows linearly with the corpus size. Hence multiply the frequency of all the current records by the scale-up factor *S*.
- The number of significant n-grams grows as a fractional power of *S*. Hence add the appropriate number of new records to the file.
  - Maintain the proportions of 5-grams, 4-grams, 3-grams, 2-grams?
  - Avoid repeated n-grams
  - Model subsumptions
  - Possibly generate bigrams first and then aggregate them.
  - Feed into corpus generator

Review of where we are

History

- People have been simulating text since 1973:
  - Michael Cooper, Berkeley – topic mixtures
  - Shannon low-level Markov modelling
  - Ricardo Baeza-Yates – FSM -> Zipf
  - Topic modelling – Latent Dirichlet Allocation (David Blei)
  - LSTM’s (Karpathy, http://karpathy.github.io/2015/05/21/rnn-effectiveness/)
- Azzopardi and Maxwell in Glasgow are simulating populations of users.

Remarks

- If you extract a generative language model from a corpus, it’s actually quite hard to generate something that matches the properties of the original. (Sampling with replacement.)
- For emulation, could you just do word for word substitution, without harming privacy too much?

Pros and Cons of our approach

- Pro:
  - Able to engineer corpora with desired properties
  - For each property, a choice of emulation methods
  - Accurate modelling of some key properties
  - Progress on modelling term dependence
  - Efficient generation (except high-order n-grams)
  - Ability to scale-up a couple of orders of magnitude.
- Con:
  - Not natural language
  - Term dependence not well enough modelled
  - Privacy leakage not well understood – Is it OK?

Plans

- Release code as Open Source (I think it will be allowed.)
- Continue experimentation
- Write up and publish.

See any applications for this type of system?

- `dahawkin@microsoft.com`

Finally, no industry talk on big data would be complete without 5 Vs

- **Virtue**
- **Veracity**
- **Virtuosity**
- **Verisimilitude**
- **Victory** ☺