Text Mining Tutorial – Part II

Ron Bekkerman

University of Haifa

Language is not Big Data!

- How many pictures are out there?

- How many words are there in the English language?

5,000,000

Google Books has 5M distinct words

Most common words	Rarest words
the	foreordainings
of	alloverwhelming
and	aristdtle
to	adultjuvenile
in	playpretties
a	inemhers
is	tajcott
that	downjhe
for	batougal
it	jyeat

50,000

50K words sound recognizable to you

Bottom of 50K words	Bottom of 100K words
inflating	alans
hals	weyland
moron	akademische
flips	lagash
payor	gordie
copepods	spirulina
lesse	unsubsidized
kine	subserves
dichromate	eicher
birney	myelosuppression

20,725,274,851,017,785,518,433,805,270

20 octillion "words" of length $\leq$ 20

Shortest "words"	Longest "words"
a	qgfzvqtnpsqqitybrnvo
b	oorlpcnikwinmlcysdro
c	oxjtiurwhtuniwicepeo
d	imbhwdlbbseyivmuwtmk
e	isrskaitqaqmxohudfiv
f	pfpvsbcpiwbecxbxvryq
g	kcbxipognefeujtphftq
h	akwovwoulwsbduyqfeti
i	sfrnjnewxupjqidwlody
j	mnbcqgxkinqtlxfdritr

English morphology is very restrictive

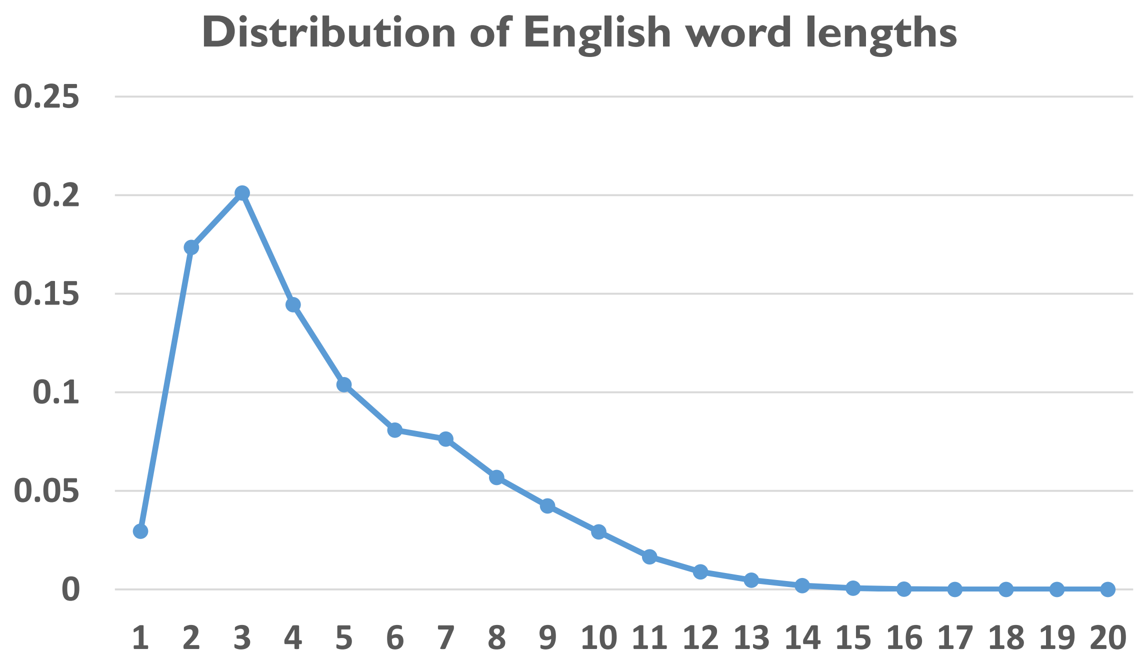
- Which letter comes after "q"?

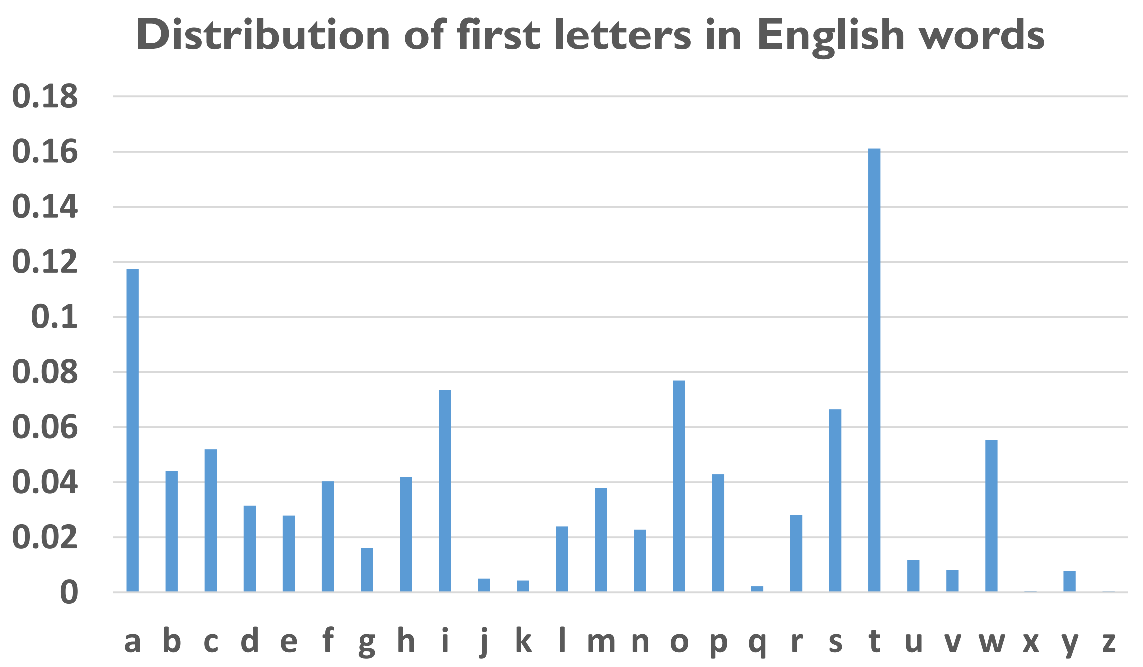
- Can a proper English word contain "yw"?
  - Yes! As in "ever**yw**here", "an**yw**ay", "Holl**yw**ood" ☺

- So, *how* restrictive is English morphology?
  - How many "words" will it allow to make?

Experiment

- Build distributions of word lengths, first letters, and following letters
  - Based on the top 50K English words





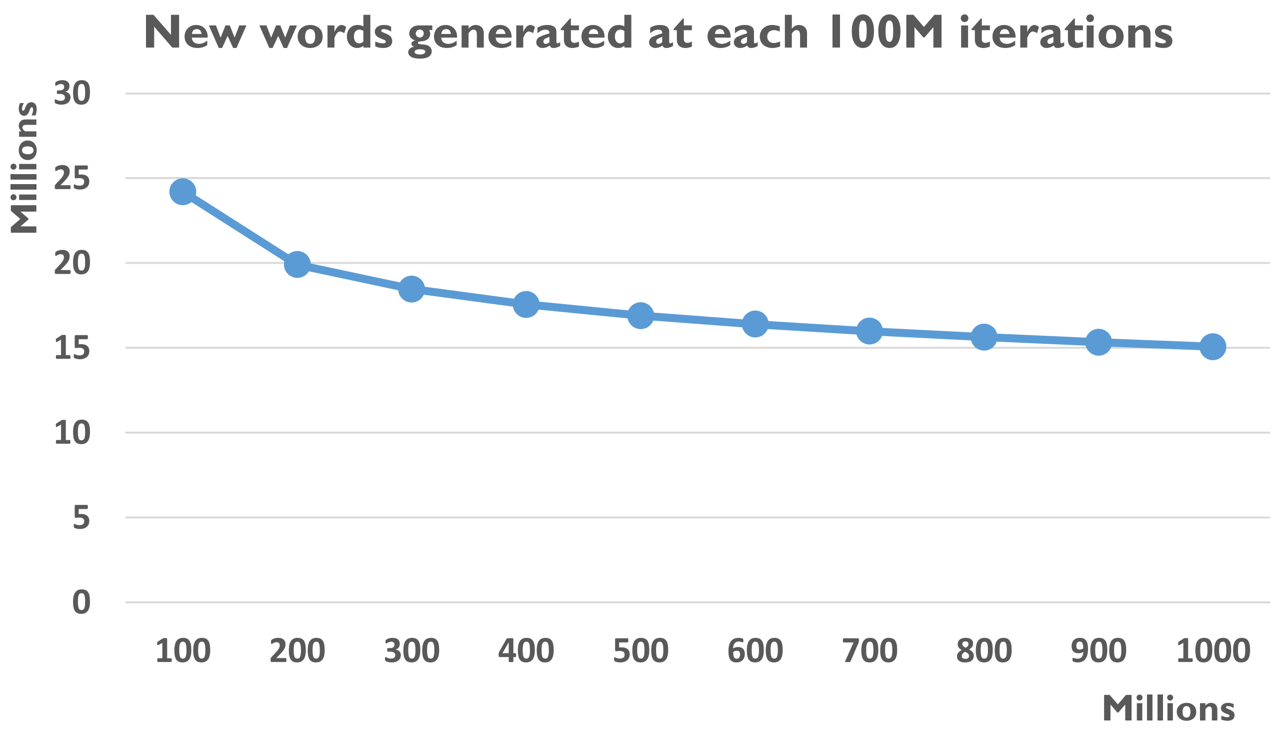
Experiment (continued)

- Sample word length N
- Sample first letter
- Sample next letter given current letter, N-1 times

- Let the process run for 1,000,000,000 iterations

Experiment (continued)

- Over each 100M iterations, the number of newly added "words" went down
  - From 25M (first 100M chunk) to 15M (tenth 100M chunk)



Experiment (continued)

- Fit a negative log curve to that function
- The curve is supposed to hit zero after 727B iterations

16,400,000,000

After the 16.4B mark, adding a new "word" is unlikely

Most common "words"	Rarest "words"
th	abababe
the	abababinghtt
t	abababodet
n	ababacc
ti	ababacechix
he	ababachecof
a	ababaclenim
in	ababaco
at	ababacofe
to	ababacquarer

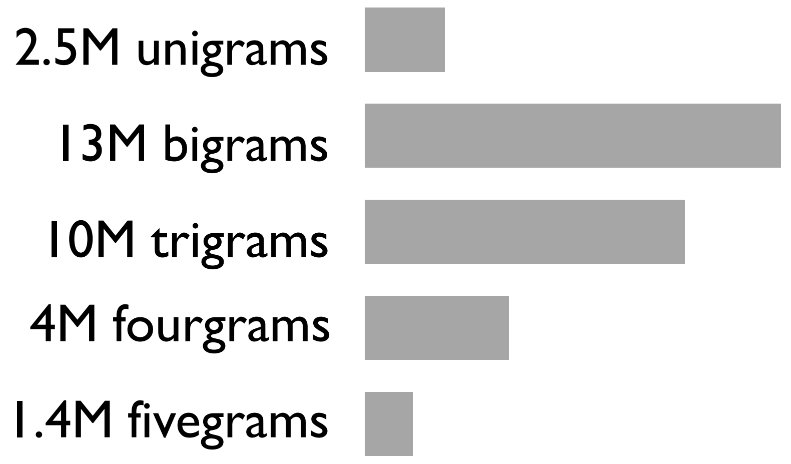
We talk in patterns

- Our vocabulary is very limited
- And we tend to combine same words into same phrases
- How many common phrases do we use?
- Experiment conducted on Web1T data
  - Ngram counts from one-trillion-word Webpage collection

Bekkerman & Gavish, KDD-2011

Experiment

- Filtered out ngrams that appeared <1000 times in Web1T
- Removed stopwords from all ngrams
- Lower-cased all words
- Ignored word order (by sorting words in an ngram)
- Example: "*all Words from the Dictionary*" → "*dictionary words*"



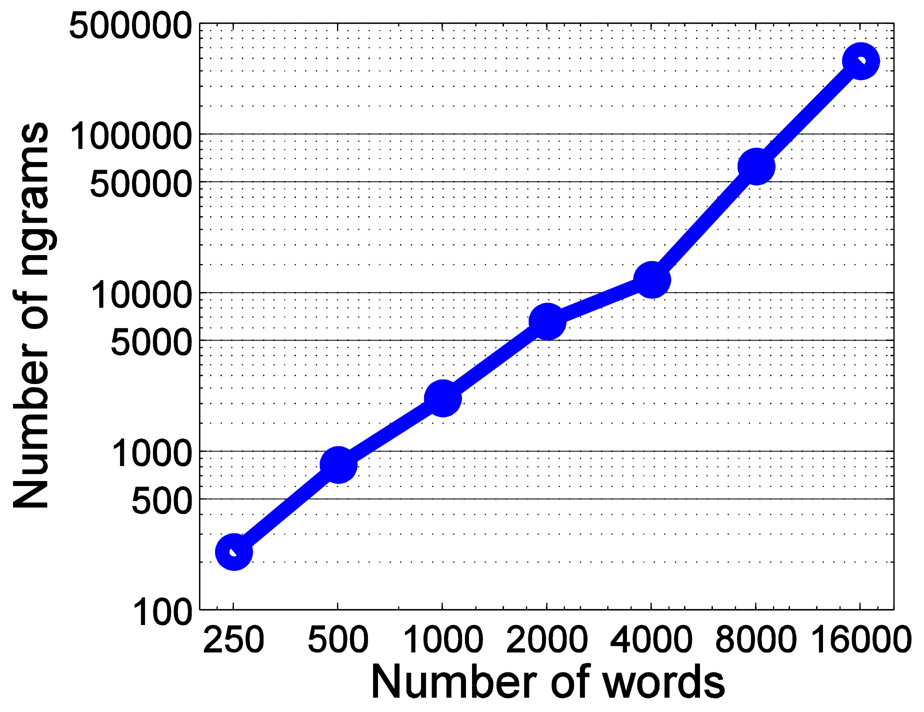
Which phrases are considered common?

- Define frequency F of a set X as frequency of its least frequent item:

$$F(X) = \min_{x \in X} F(x)$$

- Given a set of words W and a set of phrases T composed of words W, we say that T **is frequent enough if $F(T) \geq F(W)$**
    - T is as frequent as the set of words it was composed of

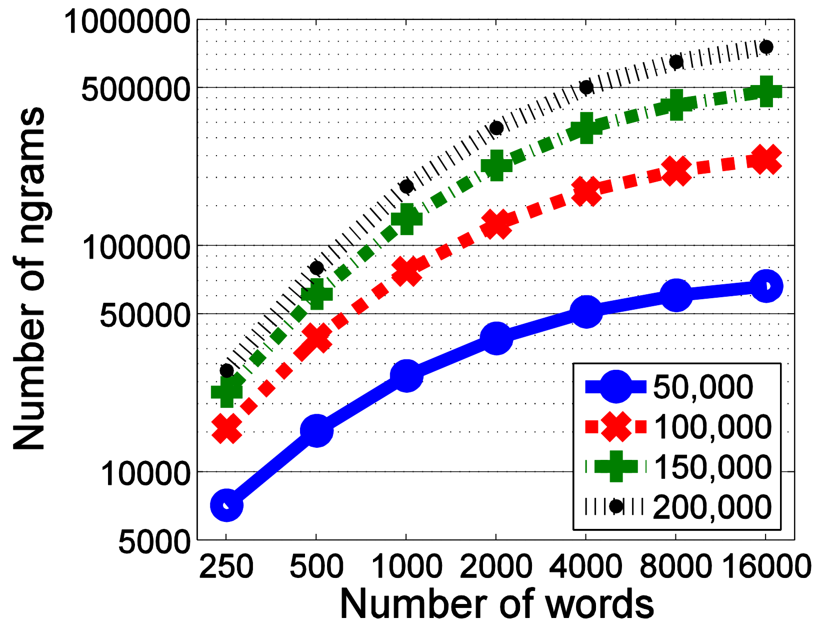
Result 1 (less realistic)

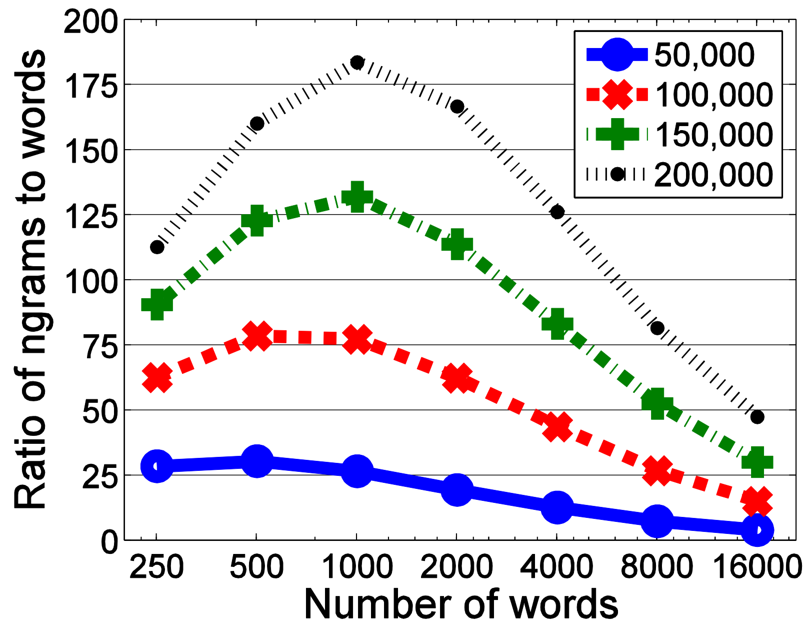


- Number of ngrams as a function of number of words that compose most common ngrams
  - A type of an upper bound

Result 2 (more realistic)

- Compose a set of words *W* that are topically related
  - Sampled from most common 50K, 100K, 150K, 200K words





Why?!

Our capabilities are so limited

- Our memory can keep 50K-100K words
  - Out of millions possible
- Our mouth can pronounce billions of letter combinations
  - Out of octillions possible
- Our cognition allows us to grasp millions of phrases
  - Out of decillions

- Our hardware is largely outdated ☺

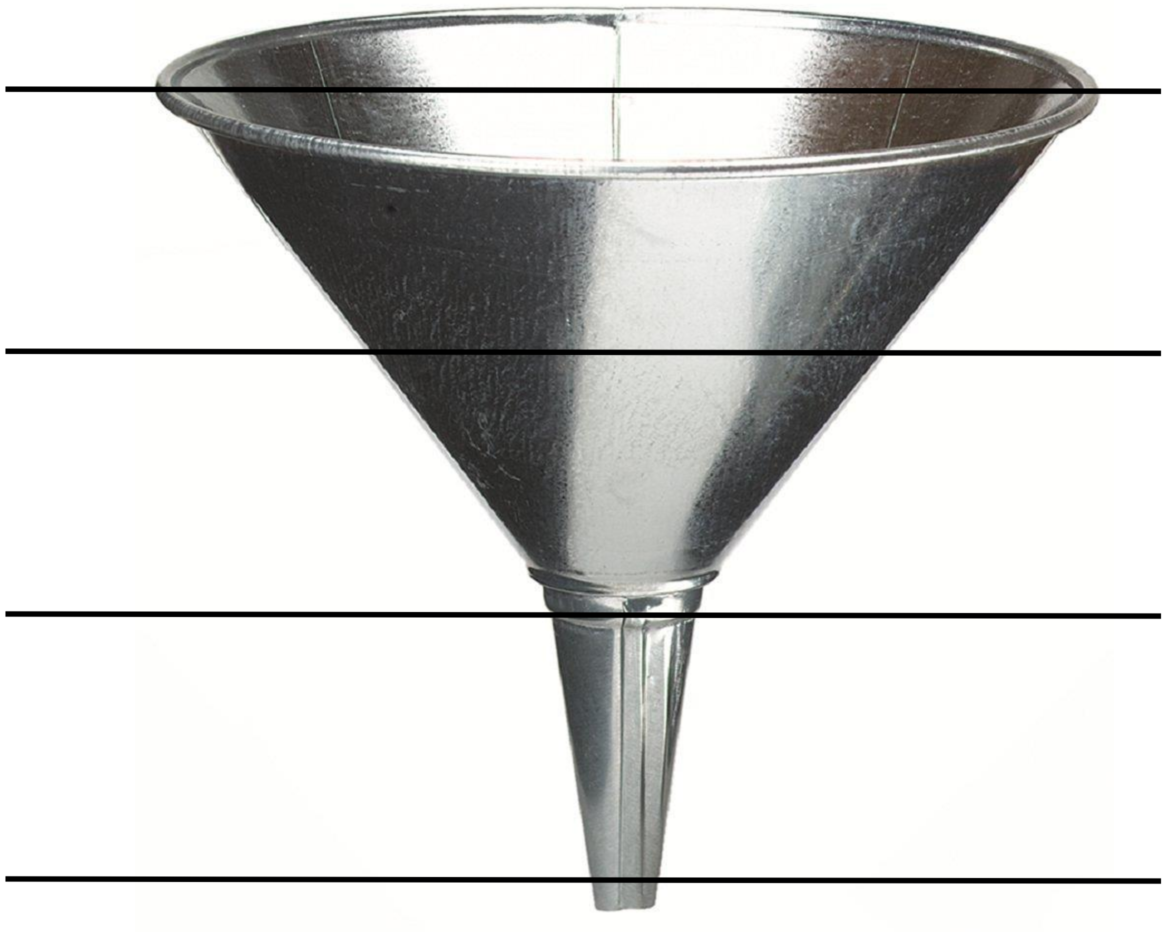
Yet, we have so much text to cope with

- Here's where Big Data technologies come to the rescue
- Remember:
  - Every text is composed of very few (distinct) words
  - Every word is constructed under restrictive conditions
  - Words are combined in few stable patterns
  - And – every word is a wealth of associations!
- This is a very special type of Big Data!

How much text is “Big Data”?

- Kilobytes?
  - Process it manually
- Megabytes?
  - Most non-Big Data text mining methods would work
- Gigabytes?
  - The hotspot for Big Data text mining methodologies
- Terabytes?
  - Well ☺

The Big Data Funnel

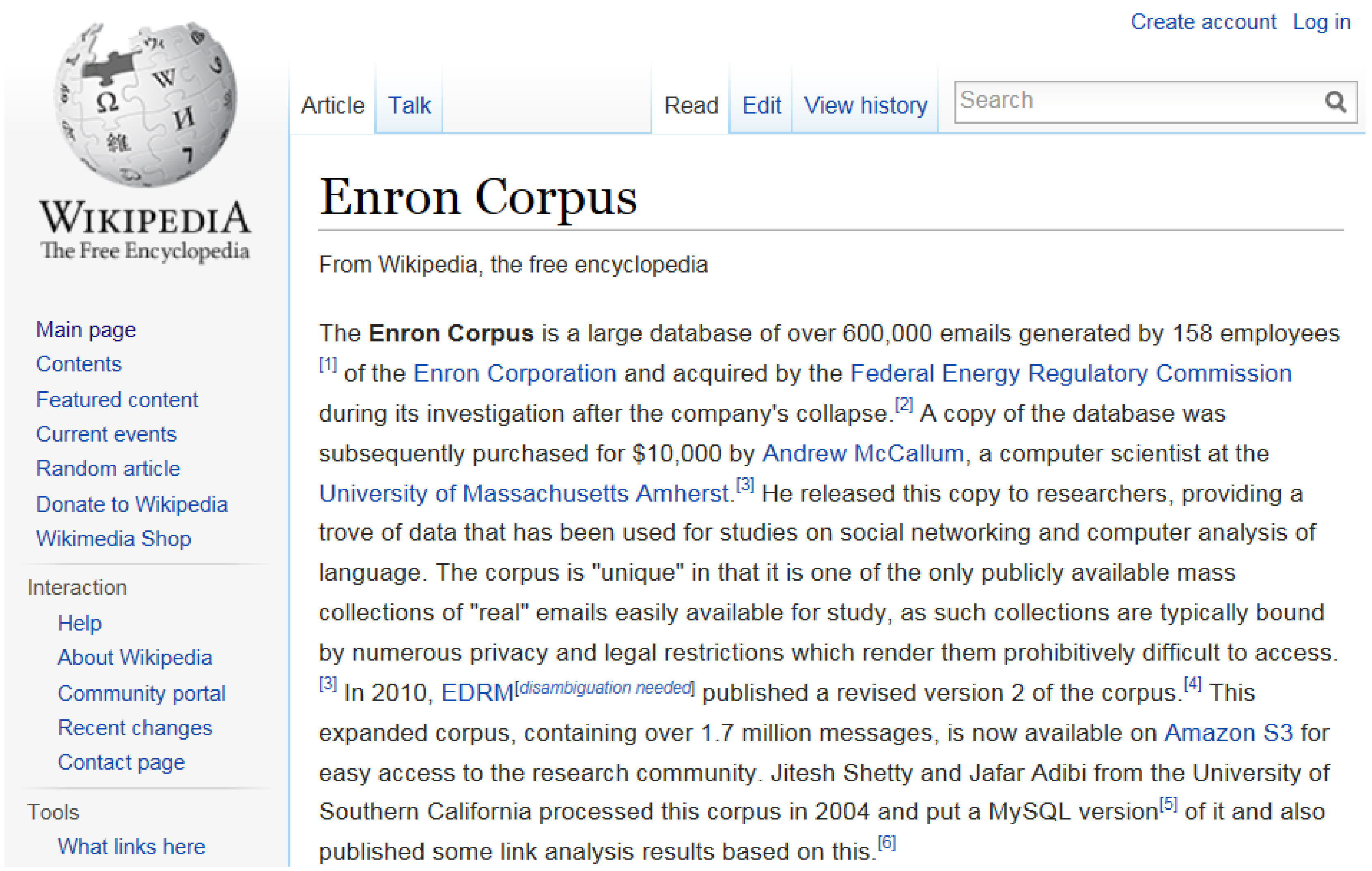


- Filter out irrelevant information
- Apply the variety of text mining methods

Goals of Big Data applications in Text

- Understand what you have in your data
- Concentrate on the data portion most relevant for your task at hand

Example



Enron Corpus

From Wikipedia, the free encyclopedia

The **Enron Corpus** is a large database of over 600,000 emails generated by 158 employees [1] of the Enron Corporation and acquired by the Federal Energy Regulatory Commission during its investigation after the company's collapse.[2] A copy of the database was subsequently purchased for $10,000 by Andrew McCallum, a computer scientist at the University of Massachusetts Amherst.[3] He released this copy to researchers, providing a trove of data that has been used for studies on social networking and computer analysis of language. The corpus is "unique" in that it is one of the only publicly available mass collections of "real" emails easily available for study, as such collections are typically bound by numerous privacy and legal restrictions which render them prohibitively difficult to access. [3] In 2010, EDRM[*disambiguation needed*] published a revised version 2 of the corpus.[4] This expanded corpus, containing over 1.7 million messages, is now available on Amazon S3 for easy access to the research community. Jitesh Shetty and Jafar Adibi from the University of Southern California processed this corpus in 2004 and put a MySQL version[5] of it and also published some link analysis results based on this.[6]

What's in the data? Cluster analysis

- A universally accepted framework for overviewing data
  - Cluster the data – take a look at cluster centroids
  - Choose the cluster you're interested in
- Trouble #1: the data is too big
  - Solution: algorithm parallelization

Cluster analysis: more troubles

- Trouble #2: choosing the “magic number k”
- Trouble #3: clustering is intrinsically inaccurate!
    - And you won’t be able to prove / contradict that

Some pointers

- A good clustering algorithm for textual data
  - Clustering documents and words simultaneously

Bekkerman, El-Yaniv & McCallum ICML-2005

- Its parallelized version

Bekkerman & Scholz CIKM-2008

- A substantially improved version

Bekkerman, Scholz & Viswanathan KDD-2009

Good clustering: the main idea

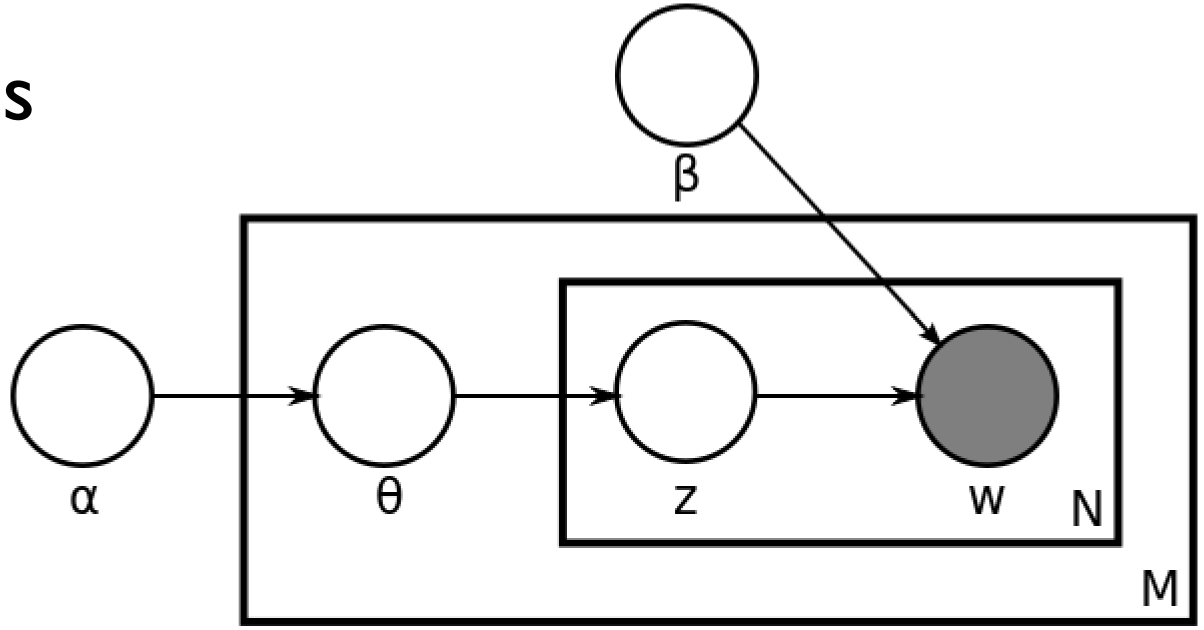
- An information-theoretic approach to co-clustering
- $\max_{\tilde{D},\tilde{W}} \mathrm{I}\left(\tilde{D};\ \tilde{W}\right), subject\ to\ \left|\tilde{D}\right| = k_d\ and \left|\tilde{W}\right| = k_w$
  - where $\mathrm{I}\left(\tilde{D};\ \tilde{W}\right)$ is Mutual Information between document clusters $\tilde{D}$ and word clusters $\tilde{W}$
- Clustering sizes k_d and k_w can be adjusted between optimization iterations

Topic models

- Started off with Latent Dirichlet Allocation

Blei, Ng & Jordan JMLR-2003

- Sample a mixture of topics per document
- Sample a topic from the mixture
- Sample a word from the topic



Topic models (continued)

- A powerful modeling tool
  - The outcome is topics (clusters of words)
  - Similar topic mixtures $\rightarrow$ similar documents
- Numerous extensions since 2003
- One big trouble: complexity
  - Both conceptual and computational
  - Inference is intractable
  - Simplifying assumptions need to be made

What's in the data? Classification

- A traditional Machine Learning approach

Lewis & Gale SIGIR-1994

- Categorize your documents to N classes
  - Distribution of documents over classes is your data's overview

How does classification work?

- Training stage
  - Input: documents and their true labels (classes)
  - Output: classification model ("the classifier")
- Test stage
  - Input: unlabeled documents
  - Output: their inferred labels

Classification: the main trouble

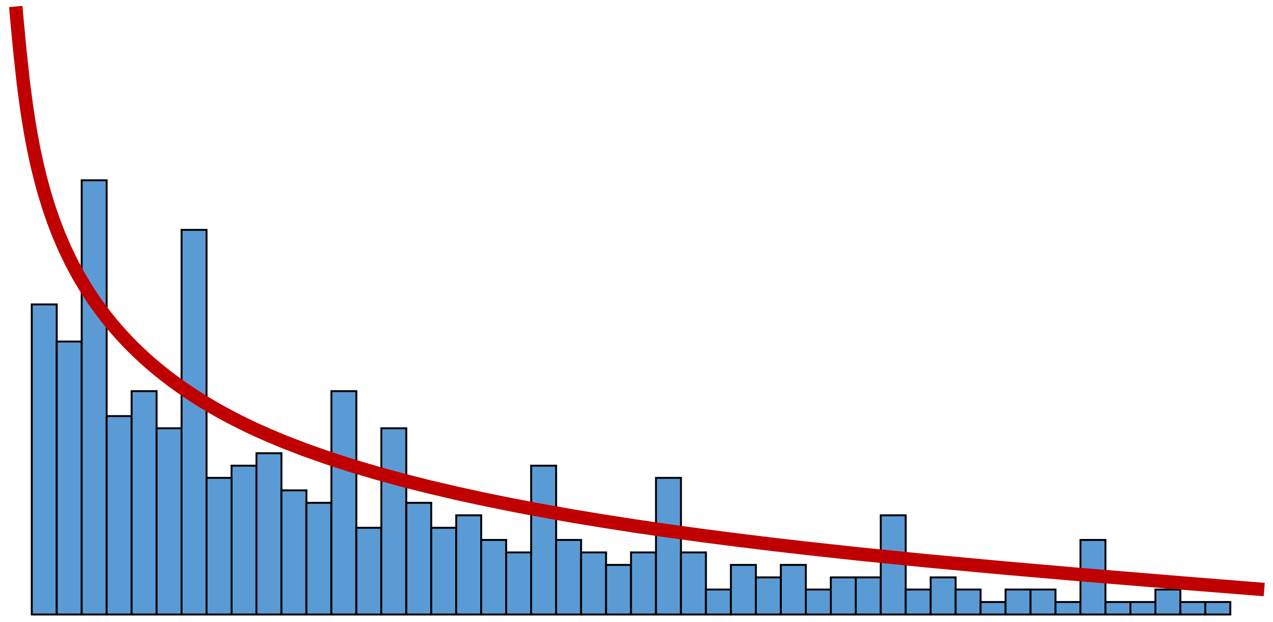
- Where are the labeled documents coming from?!
- If we need to label them, then:
    - Who is gonna label them?
    - How many documents should be labeled?
    - How do we choose documents to be labeled?
    - And – how do we choose the classes?
- All this becomes a nightmare in the case of Big Data

Let’s recall what we started with

- Words are too few ☺
- They are organized in stable phrases
- And – not all of them are important!

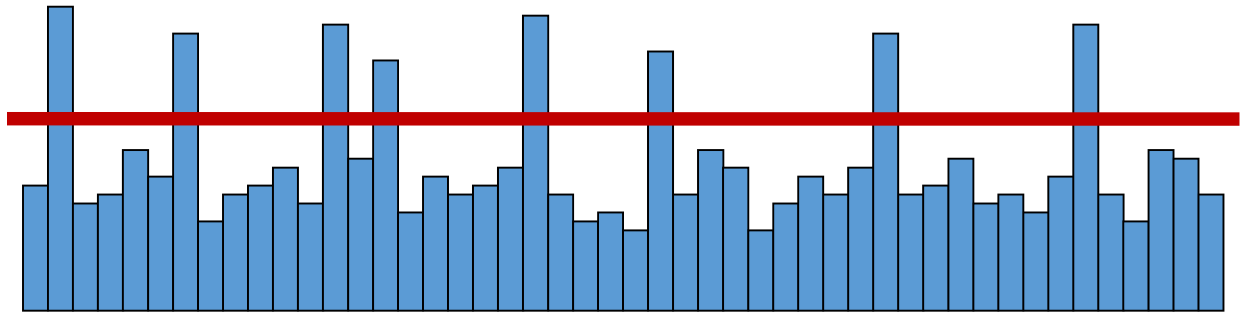
“Surprising” words

- We plot word frequencies in a document
- When words are sorted by their frequency in English
  - Estimated over Web1T or Google Books



“Surprising” words (continued)

- Why don’t we plot the ratio of word frequencies in the document and in English



$$\frac{p(w)}{q(w)}$$

Bekkerman & Crammer, EMNLP-2008

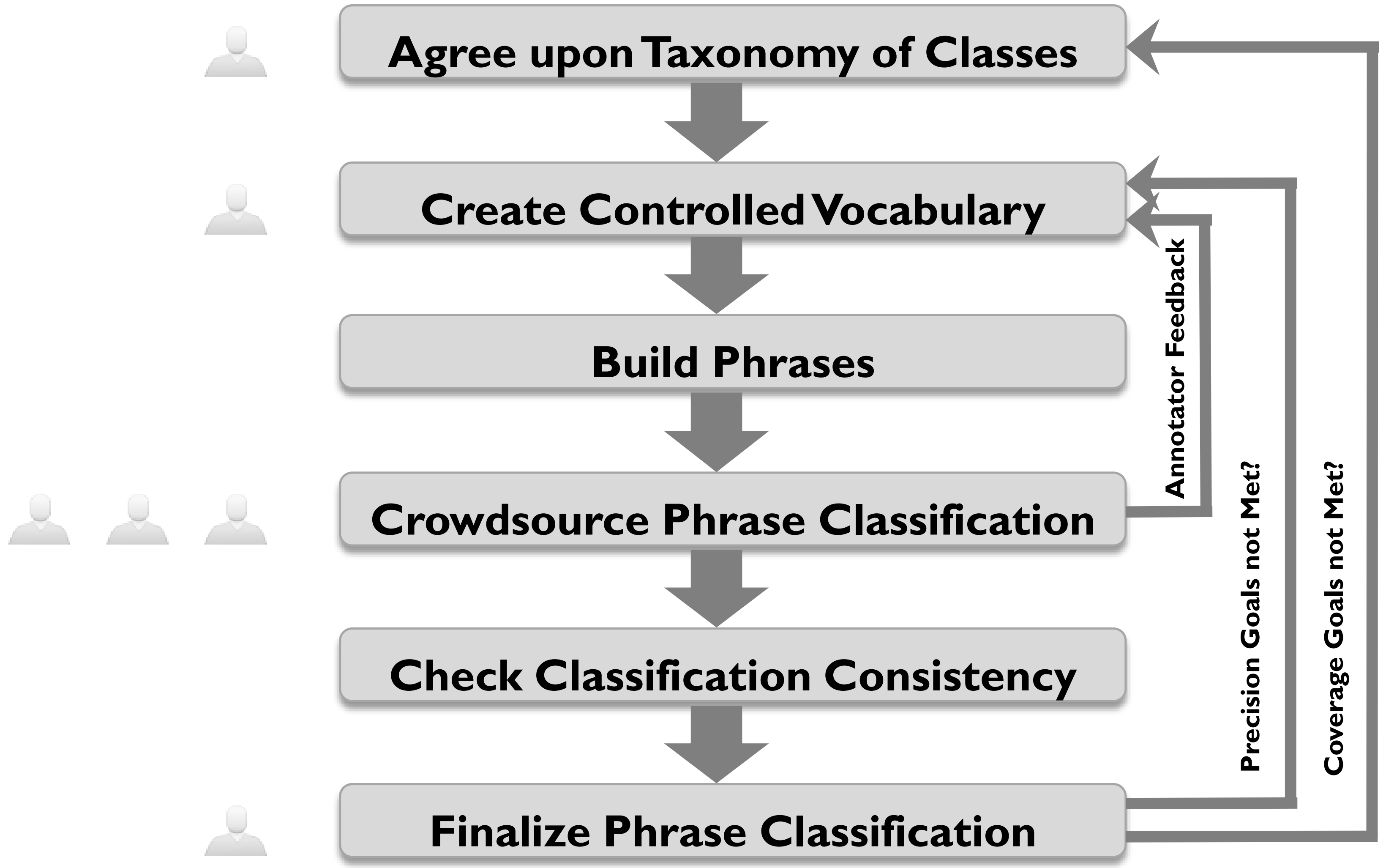
Example of a document

Top words	Bottom words
oath	as
ideals	there
prosperity	by
courage	which
nation	in
generations	from
journey	were
our	an
seek	was
crisis	i

How to cope with big textual data

- Represent each document as a bag of *surprising* words
  - You'll filter out lots of noise
- **Problem:** some words are too common and ambiguous
- **Solution:** use phrases!
- **Problem:** some phrases don't carry much meaning
- **Solution:** build / apply a vocabulary of terminology!

Phrase-based text classification


Agree upon Taxonomy of Classes
Create Controlled Vocabulary
Build Phrases
Crowdsource Phrase Classification
Check Classification Consistency
Finalize Phrase Classification
Annotator Feedback
Precision Goals not Met?
Coverage Goals not Met?

Phrase-based text classification (contd)

- Should not be too many phrases to categorize ☺
- Once phrases are categorized, locate them in documents
- Categorize documents into a mixture of classes based on phrase classification

Bekkerman & Gavish, KDD-2011

Wikipedia as a terminology vocabulary

- Long history of using Wikipedia in text mining

Gabrilovich & Markovitch IJCAI-2007

- Currently, Wikipedia has 5M content pages and 8M redirect pages
    - Each page title is a term, validated by Wikipedia editors
    - Most content pages are pre-categorized!

How to categorize big textual data

- Represent each document as a bag of Wikipedia terms
- Categorize documents into a mixture of classes based on Wikipedia term classification
- **No labeling is required!!!**
- **Problem:** extracting ngrams from documents, efficiently
- **Solution:** use the *Trie* data structure (prefix tree)

We can do better than classification

- Build *semantic space* of a document collection
  - A graph where nodes are documents and edges are topical connections between the documents
- A topical connection between two documents is an overlap of their Bags of Terms

Semantic space: two problems

- **Problem #1:** the semantic space is quadratic in the number of documents
- **Solution:** use MapReduce

- **Problem #2:** how to deal with synonyms?
- **Solution:** use word2vec

Example: Patentosphere

- Build the semantic space of 8M US patents
  - Filed in the last 40 years
- Drill down to any region of the semantic space

Khoury & Bekkerman, RIPL 2016

How to construct the semantic space

- Input: term $\rightarrow$ $(d_1, w_1), (d_2, w_2), \ldots (d_n, w_n)$
- Mapper: for each t, output all pairs $(d_i, d_j), \min(w_i, w_j)$
- Reducer: sum weights together $(d_i, d_j), \sum\min(w_i, w_j)$

Elsayed, Lin & Oard ACL-2008

How to deal with synonyms: word2vec

- Each word (or phrase, term) is represented as a distribution over its surrounding words
- Semantically similar words have similar distributions

Mikolov, Chen, Corrado & Dean arXiv 2013

- Numerous extensions since then

How to construct word2vec

- Input: documents as sequences of terms $(t_1, t_2, \ldots, t_n)$
- Mapper: for each t_i, output key-value pairs (t_i, t_{i-2}), (t_i, t_{i-1}), (t_i, t_{i+1}), (t_i, t_{i+2})
- Reducer: count how many times term t appeared with surrounding terms $t \rightarrow (t_1, w_1), (t_2, w_2), \ldots (t_n, w_n)$
- Transpose the resulting matrix, and apply the semantic space construction MapReduce (2 slides above)

Conclusion

- Big Data Text Mining suffers from overcomplication
- Very simple methods work surprisingly well, because:
  - Language is very much finite
  - Lots of high-quality manual work has already been done
- The gold is in algorithmic efficiency

From Big Data to All Data

- The time will come when all text ever written will be indexed
- All text in all languages will be translated to English
- All speech ever said will be transcripted
- All common phrases will be categorized

- We will then know the future ☺

Questions? ronb@univ.haifa.ac.il