Lecture 7 – Text Analysis

Part 1: Text Laws & Preprocessing

THE UNIVERSITY of EDINBURGH
informatics

THE UNIVERSITY of EDINBURGH
School of Social
& Political Science

Prof. Walid Magdy, Informatics School

Scenario

- You are given access to a new dataset
  - 2 corpora, each contains thousands of text files
  - You want to understand and quantify:
    - What is the *content* of these documents? What are they *about*?
    - How does the content of these corpora *differ*?
- How can you analyse?

Lecture Objectives

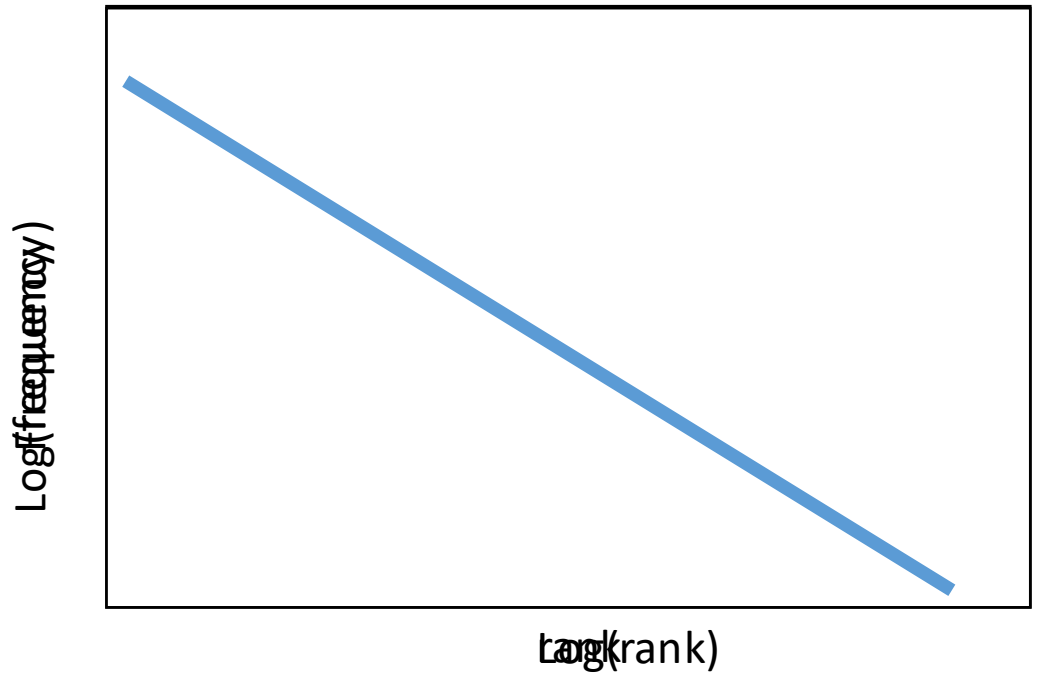
- Learn about some text laws
  - Zipf's law
  - Heap's law
  - Clumping/contagion
- Text Pre-processing
  - Preparation for analysis

Words' nature

- Word → basic unit to represent text
- Certain characteristics are observed for the words we use!
- These characteristics are very consistent, that we can apply laws for them
- These laws apply for:
  - Different languages
  - Different domains of text

Frequency of words

- Some words are very frequent
  e.g. “the”, “of”, “to”
- Many words are less frequent
  e.g. “schizophrenia”, “bazinga”
- ~50% terms appears once
- Frequency of words has
  hard exponential decay



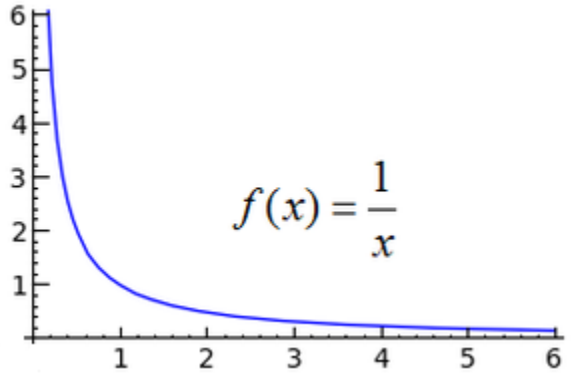
Zipf's Law:

- For a given collection of text, ranking unique terms according to their frequency, then:

$$r \times P_r \cong const$$

- r, rank of term according to frequency
- P_r, probability of appearance of term

- $P_r \cong \frac{const}{r} \rightarrow f(x) \cong \frac{1}{x}$



Practical

Collection	# words	File size
Bible	824,054	4.24 MB
Wiki abstracts	80,460,749	472 MB

Zipf's Law:

Wikipedia abstracts

→ 3.5M En abstracts

$r \times P_r \cong const \rightarrow$

$r \times freq_r \cong const$

Term	Rank	Frequency	r x freq
the	1	5,134,790	5,134,790
of	2	3,102,474	6,204,948
in	3	2,607,875	7,823,625
a	4	2,492,328	9,969,312
is	5	2,181,502	10,907,510
and	6	1,962,326	11,773,956
was	7	1,159,088	8,113,616
to	8	1,088,396	8,707,168
by	9	766,656	6,899,904
an	10	566,970	5,669,700
it	11	557,492	6,132,412
for	13	493,374	5,970,456
as	14	480,277	6,413,862
on	15	471,544	6,723,878
from	16	412,785	7,073,160

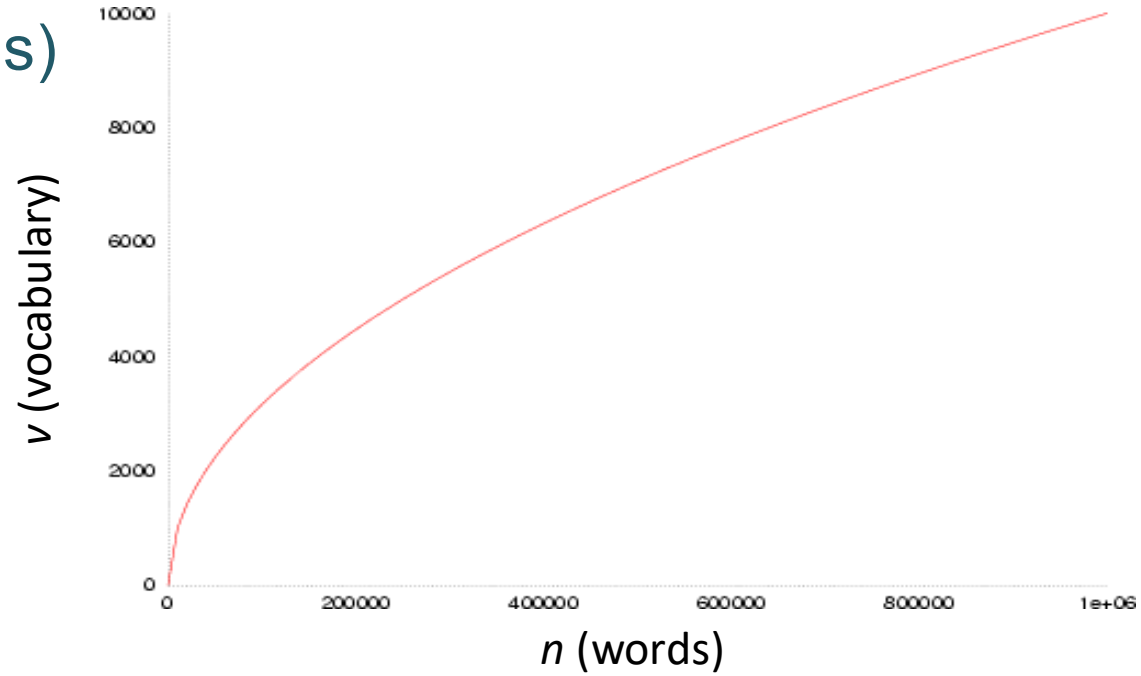
Heap's Law:

- While going through documents, the number of new terms noticed will reduce over time
- For a book/collection, while reading through, record:
  - n: number of words read
  - v: number of news words (unique words)
- Vocabulary growth:

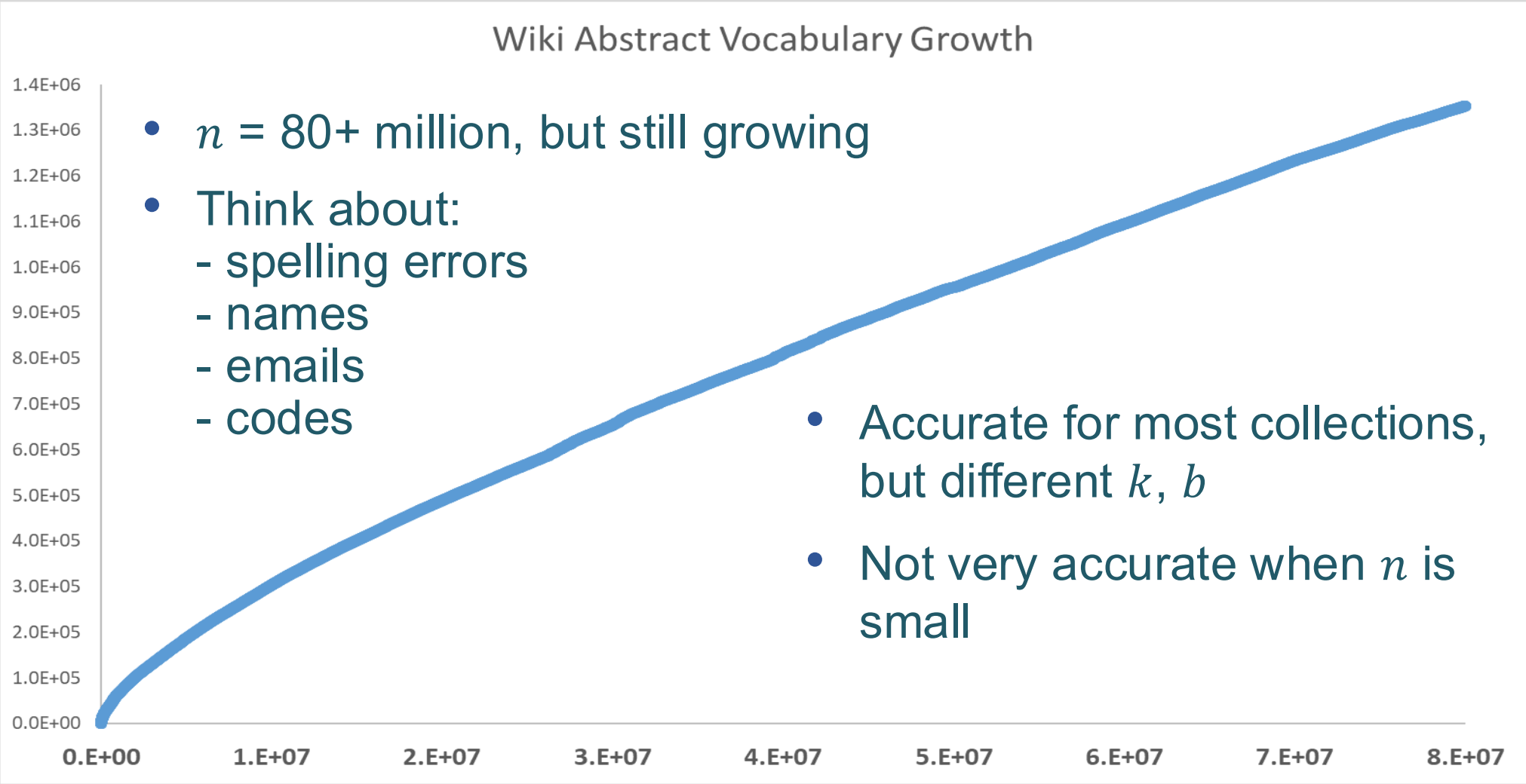
$$v(n) = k \times n^b$$

where, $b < 1$

typically, $0.4 < b < 0.7$



Heap's Law: shouldn't it saturate?



- n = 80+ million, but still growing
- Think about:
  - spelling errors
  - names
  - emails
  - codes

- Accurate for most collections, but different k, b
- Not very accurate when n is small

Clumping/Contagion in text

- From Zipf's law, we notice:
  - Most words do not appear that much!
  - Once you see a word once → expect to see again!
  - Words are like:
    - Rare contagious disease
    - Not, rare independent lightening

- Words are rare events, but they are contagious

How to see documents?

Bag-of-words trick

- Can you guess what this is about:
  - per is salary hour €25,000 Ronaldo's
  - obesity French is of full cause and fat fries
- Re-ordering doesn't destroy the topic
  - individual words – "building blocks"
  - "bag" of words: a "composition" of "meanings"

Bag-of-words trick

- Most search engines use BOW
  - treat documents, queries as bags of words
- A “bag” is a set with repetitions
  - match = “degree of overlap” between D,Q
- Retrieval models
  - statistical models (function) that use words as features
  - decide which documents most likely to be relevant
- What should be the top results for Q?
  - BOW makes these models tractable

Bag-of-words: Criticism

- word meaning lost without context
  - True, but BOW doesn't really discard context
    -
- what about negations, etc.?
  - {no, climate change is real} vs. {climate change is no real}
- does not work for all languages
  - No natural "word" unit for Chinese, images, music
  - Solve by "segmentation" or "feature induction"

Preprocessing

Preparing Text – Simple Steps

- There are some very basic steps to process text before comparing it!
- Steps such as:
  - Tokenisation
  - Stopping
  - Normalisation
    - Stemming
- The objective of these steps are:
  - Focus on important terms
  - Normalise similar terms with different surface form

Word or Term?

- What is a word?
- We refer to word-elements as "terms"
  - word "*preprocessing*"
  - part of a word "*pre*"
  - number / code "*INFR08034*"
- Pre-processing steps before indexing:
  - Tokenisation
  - Stopping
  - Stemming
- **Objective** → identify the basic form of the term to make it efficient when analysing a corpora

Tokenisation

- Input: "***Friends, Romans; and Countrymen!***"
- Output: Tokens
  - *Friends*
  - *Romans*
  - *and*
  - *Countrymen*
- Sentence → tokenization (splitting) → tokens
- A **token** is an **instance** of a sequence of characters
- **Typical technique:** split at non-letter characters
- Each such token is now a candidate for an index entry (**term**), after further processing

Issues in Tokenisation

- "*Finland's*" capital → *Finland*? *Finlands*? *Finland's*?
- Hewlett-Packard → one token or two?
  - **state-of-the-art**: break up hyphenated sequence.
  - *co-education*
  - *lowercase, lower-case, lower case ?*
  - It can be effective to get the user to put in possible hyphens
- **Numbers?**
  - 3/20/91 vs. Mar. 20, 1991 vs. 20/3/91
  - This course code is INFR11145
  - (800) 234-2333

Issues in Tokenisation

- **URLs:**
  - http://www.bbc.co.uk
  - http://www.bbc.co.uk/news/world-europe-41376577
- **Social Media**
  - Black lives matter
  - #Black_lives_matter
  - #BlackLivesMatter
  - #blacklivesmatter
  - @blacklivesmatter
- **San Francisco**: one token or two?
  - How do you decide it is one token?

Tokenisation for different languages

- French → ***L'ensemble*** → one token or two?
  - ***L*** ? ***L'*** ? ***Le*** ?
  - Want ***l'ensemble*** to match with ***un ensemble***
  - Until at least 2003, it didn't on Google
- German → compounds
  - ***Lebensversicherungsgesellschaftsangestellter***
    'life insurance company employee'
  - German retrieval systems benefit greatly from a **compound splitter** module → Can give a 15% performance boost for German
- Chinese and Japanese → no spaces between words:
  - 莎拉波娃现在居住在美国东南部的佛罗里达
  - Tokenisation → Segmentation

Tokenisation: common practice

- Just split at non-letter characters
- Add special cases if required
- Some applications have special setup
  - Social media: hashtags/mentions handled differently
  - URLs: no split, split at domain only, remove entirely!
  - Medical: protein & diseases names

Stopping (stop words removal)

- ~~This is a very~~ exciting lecture ~~on the~~ use ~~of the~~ best computational methods ~~for~~ social science
- **Stop words**: the most common words in collection
  → the, a, is, he, she, I, him, for, on, to, very, …
- There are a lot of them ≈ 30-40% of text
- New stop words appear in specific domains
  - Tweets: RT → *"RT @realDonalTrump Iran will …"*
  - Patents: said, claim → *"a said method that extracts …."*
- Stop words
  - influence on sentence structure
  - less influence on topic (aboutness)

Stopping: always apply?

- Sometimes very important:
  - Phrase quotes: “Let it be”, “To be or not to be”
  - Relational relationship:
    - flights to London from Edinburgh
    - flights from London to Edinburgh
- In some analysis, you can keep them.
- You decide based on application and analysis objective

Stopping: stop words

- Common practice in many applications
  → remove stop words
- There are common stop words list for each language
  - NLTK (python)
  - http://members.unine.ch/jacques.savoy/clef/index.html
- There are special stop words list for some applications
- How to create your list:
  - Sort all terms in a collection by frequency
  - Manually select the possible stop words from top *N* terms

Normalisation

- **Objective** → make words with different surface forms look the same
- Document: "This is my car. I told you it is my CAR!!"

  How many times the word "car" appeared?

- Sentence → tokenisation → tokens
  → (stopping) → topical-text
  → normalisation → text to be analysed

Case folding and equivalents

- “A” & “a” are different strings for computers
- Case folding: convert all letters to lower case
  - CAR, Car, caR → car
  - Windows → windows, should we do that?
  - Usually yes, you don’t know how they are written in the documents, especially social media!
- Diacritics/Accents removal
  - French: Château → chateau
  - German: Tüebingen → tuebingen
  - Arabic: كُتُبٍ → كتب

Stemming

- Search for: “play”
  should it match: “played”, “playing”, “player”?
- Many morphological variations of words
  - *inflectional* (plurals, tenses)
  - *derivational* (making verbs nouns etc.)
- In most cases, aboutness does not change
- Stemmers attempt to reduce morphological variations of words to a common stem
  - usually involves removing suffixes (in English)

Stemming

- Usually, improves analysis by 5-10% improvement for English.
- Can be critical for highly inflected languages, e.g.:
  - 30% improvement in Finnish
  - 50% improvement in Arabic

They are Peter's **children**	هؤلاء **أبناء** بيتر
The **children** behaved well	**الأبناء** تصرفوا جيدا
Her **children** are cute	**أبناءها** لطاف
My **children** are funny	**أبنائي** ظرفاء
We have to save our **children**	علينا أن نحمي **أبناءنا**
Patents and **children** are happy	الآباء **والأبناء** سعداء
He loves his **children**	هو يحب **أبناءه**
His **children** loves him	**أبناؤه** يحبونه

Stemming

- Two basic types
  - Dictionary-based: uses lists of related words
  - Algorithmic: uses program to determine related words
- Algorithmic stemmers
  - suffix-s: remove ‘s’ endings assuming plural
  - e.g., cats → cat, lakes → lake, windows → window
  - Many false negatives: supplies → supplie
  - Some false positives: James → Jame

Porter Stemmer

- Most common algorithm for stemming English
- Conventions + 5 phases of reductions
  - phases applied sequentially
  - each phase consists of a set of commands
  - sample convention:
    of the rules in a compound command, select the one that applies to the longest suffix.
- Example rules in Porter stemmer
  - *sses* → *ss* (processes → process)
  - *y* → *i* (reply → repli)
  - *ies* → *i* (replies → repli)
  - *ement* → null (replacement → replac)

Stemmed words are misspelled!!

- repli, replac, suppli, inform retriev, anim
- These are not words anymore, these are terms
- These terms are used for the analysis not to be read in the document
- These represent the optimal form for a better match between different surface forms of a term
  - e.g. replac → replace, replaces, replaced, replacing, replacer, replacers, replacement, replacements.

Pre-processing: Common practice

- Tokenisation: split at non-letter characters
  - Basic regular expression
    → process \w and neglect anything else
  - For tweets, you might want to keep “#” and “@”
- Remove stop words
  - find a common list, and filter these words out
- Apply case folding
- Apply Porter stemmer
  - Other stemmers are available, but Porter is the most famous with many implementations available in different programming languages

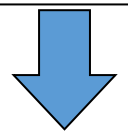
Limitations

- Irregular verbs:
  - saw → see
  - went → go
- Different spellings
  - colour vs. color
  - tokeni**s**ation vs. tokeni**z**ation
  - Television vs. TV
- Synonyms
  - car vs. vehicle
  - UK vs. Britain
- More advanced methods …

Summary

- Text follows well-known phenomena
- Text pre-processing before IR:
  - Tokenisation → Stopping → Stemming

This is an example sentence of how the pre-processing is applied to text in information retrieval. It includes: Tokenization, Stop Words Removal, and Stemming



exampl sentenc pre process appli text inform retriev includ token stop word remov stem

Practical

Collection	Original		After Pre-processing	
	# words	File size	# words	File size
Bible	824,054	4.24 MB	358,112	2.05 MB
Wiki abstracts	78,137,597	472 MB	47,741,065	309 MB

Recourses

- Videos:
  - Zipf's law, Vsouce: https://www.youtube.com/watch?v=fCn8zs912OE

- Common stop words list in 40+ languages
  - http://members.unine.ch/jacques.savoy/clef/index.html