

Introduction

IN4080

Natural Language Processing

Yves Scherrer & Pierre Lison

Course overview

This course

An introduction to the most common problems and methods in NLP

- Some statistical (non-neural) methods:
 - Naïve Bayes, Logistic regression, sequence labeling, N-gram language models
- High-level introduction to neural methods:
 - Embeddings, multi-layer perceptron, recurrent neural networks, transformers
- Tasks:
 - Text classification, tagging, speech processing, dialog systems, machine translation
- Ethical considerations

Challenges (for us teachers)

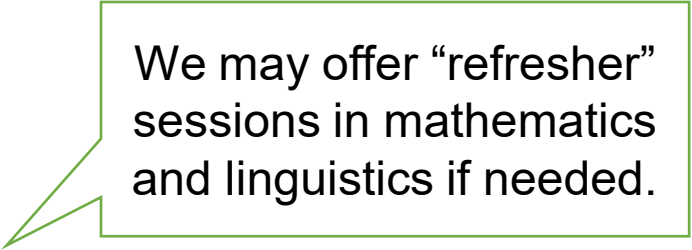
- You have (very) different backgrounds
 - For some, there will be a lot of repetition
 - Others may experience a steep learning curve
- How much weight should we give to non-neural methods?
 - Still useful for getting acquainted with basic ML principles
 - Most tasks are independent of learning algorithm, and can be easier understood using simpler models
 - For the practical parts, we'll stick to what can be done on a CPU
 - There's a dedicated course on neural methods in Spring

Let's do a quick poll...

- In what program are you currently enrolled?
 - Master Language Technology
 - Master Data Science
 - Another IFI Master program
 - Something else
- Have you previously done courses in...
 - Language Technology / NLP?
 - Machine Learning?
 - Corpus Linguistics / Quantitative Linguistics?

Our expectations

- Programming:
 - Familiar with Python and Jupyter notebooks
 - Basic knowledge about machine learning
- Mathematics:
 - Probabilities
 - Logarithms
 - Linear algebra (vectors and matrices)
- Linguistics:
 - Basic knowledge of linguistic structures and corpus linguistic analysis



We may offer “refresher” sessions in mathematics and linguistics if needed.

Organization

- Lectures on Tuesdays (14-16)
 - Teachers: Yves or Pierre
 - Recordings and slides online
- Lab sessions on Fridays (12-14)
 - Teachers: Victor or Huiling
 - No recordings or slides
 - **We already start this week!**
 - This week: set up computing environment, refresher on Python data science stack
- Communication:
 - Discourse forum
 - Group e-mail to all teachers (TBC)

Evaluation

- 3 mandatory assignments
 - Deadlines: 18 September, 16 October, 6 November
 - Practical programming assignments
 - Pass/fail grading, prerequisite for final exam
- Final written exam
 - 7 December
 - Closed book, mostly theoretical questions
 - Determines course grade

Schedule

22. Aug.	Introduction [YS]	
29. Aug.	Text classification I [YS]	
5. Sep.	Text classification II [YS]	Mandatory 1 (text classification)
12. Sep.	Sequence labeling [YS]	
19. Sep.	Language models [YS]	
26. Sep.	Vectors and embeddings [YS]	
3. Oct.	Neural networks I [YS]	Mandatory 2 (sequence labeling)
10. Oct.	Neural networks II [YS]	
17. Oct.	Dialog systems I [PL]	
24. Oct.	Dialog systems II [PL]	Mandatory 3 (dialog systems)
31. Oct.	Dialog systems III [PL]	
7. Nov.	Machine translation [YS]	
14. Nov.	Ethics [PL]	
21. Nov.	Ethics [PL] + Summary [YS]	

Syllabus

- Jurafsky & Martin: Speech and Language Processing, 3rd edition
 - <https://web.stanford.edu/~jurafsky/slp3/>
 - This book is work in progress and thus only available online
 - We use the January 2023 draft
- Bird, Klein & Loper: Natural Language Processing with Python
 - <https://www.nltk.org/book/>
 - Online version of an extensive tutorial of the NLTK library

Programming setup

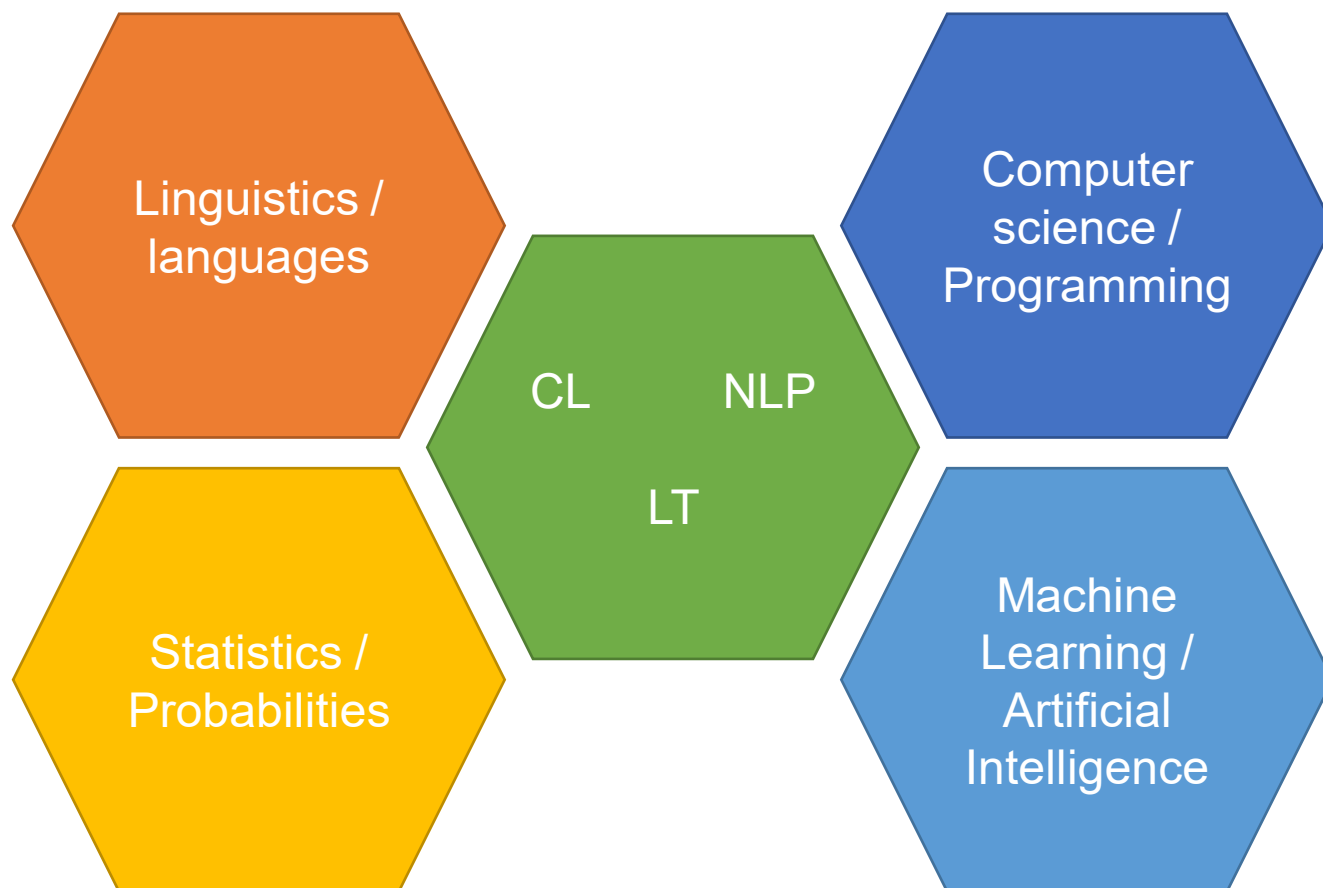
- Python + Jupyter Notebooks
- Data science stack:
 - Pandas, Numpy, Scipy, Matplotlib, ...
- Machine learning libraries:
 - Scikit-learn, PyTorch
- NLP-specific libraries:
 - NLTK, gensim, ...

What is NLP?

Names...

- Computational linguistics (CL)
 - Traditional name (cf. ACL)
 - Emphasizes applications that support (theoretical) linguistic research
- Natural language processing (NLP)
 - Originated in CS (natural vs. formal languages)
 - Emphasis on engineering, practical applications
 - Most often used name today
- Language technology (LT)
 - Sometimes used to combine CL and NLP (and speech processing)

An interdisciplinary field



Predominant approaches

Rule-based / knowledge-based

- 1950-2000
- Explicitly encode linguistic expert knowledge
- Difficult to scale, expensive

Statistic / data-driven / machine learning

- 1990-today
- Use machine learning methods to generalize from annotated data
- Division of labor between annotators / linguists and ML people
- Annotated datasets are costly to produce, but can easily be reused

Predominant approaches

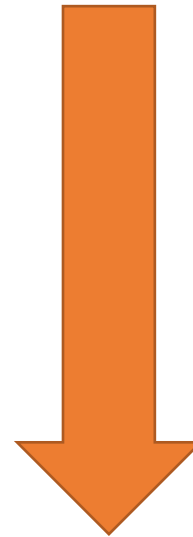
Statistic / data-driven / machine learning

Non-neural models



Neural models

Task-specific data



Pretraining on task-independent data

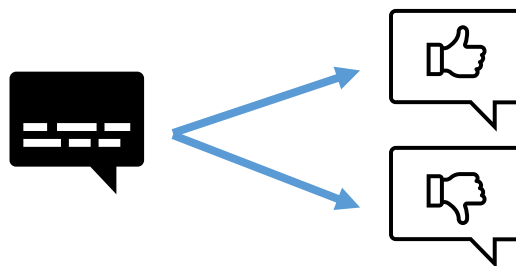
Main NLP tasks

Natural language generation



- Machine translation
- Question answering
- Grammatical error correction
- ...

Annotation (natural language understanding)



- Hate speech detection
- Sentiment analysis
- Language identification
- Syntactic analysis
- ...

The classical NLP pipeline

Split into sentences	Obama says he didn't fear for 'democracy' when running against McCain, Romney.
Tokenize (normalize)	Obama says he did not fear for ' democracy ' when running against McCain , Romney .
Tag	Obama_N says_V he_PN did_V not_ADV fear_V ...
Lemmatize	Says_V → say_V, did_V → do_V, running_V → run_V ...
Parsing (dependency)	<pre> graph TD shot[shot] --- SBJ[I] shot --- OBJ[an elephant] shot --- NMOD[in] shot --- PMOD[my pajamas] an[an] --- DETMOD1[DETMOD] elephant[elephant] --- DETMOD2[DETMOD] in[in] --- NMOD2[NMOD] my[my] --- DETMOD3[DETMOD] pajamas[pajamas] --- DETMOD4[DETMOD] </pre>
Coreference resolution	Obama says he did not
Semantic relation detect.	Fear(Obama, Democracy) Run_against(Obama, McCain),..
Negation detection	... did not fear ... → Not(Fear(Obama, Democracy))

This pipeline is less and less relevant nowadays:

- Most often, the goal is not to produce some kind of logical expression of the sentence meaning.
- Models can directly produce the lower levels without the upper ones.

Working with text data

A dataset, as presented in a typical ML course

	spam	chars	lines breaks	'dollar' occurs. numbers	'winner' occurs?	format	number
1	no	21,705	551	0	no	html	small
2	no	7,011	183	0	no	html	big
3	yes	631	28	0			
4	no	2,454	61	0			
5	no	41,623	1088	9			
...							
50	no	15,829	242	0	no	html	small

Most data doesn't come as nicely formatted tables...
How do we get from raw text to such tables?

Output class
Dependent variable

Input features
Explanatory/independent variables

A dataset, as presented in a typical ML course

Why exactly these features/columns?

	spam	chars	lines breaks	'dollar' occurs. numbers	'winner' occurs?	format	number
1	no	21,705	551	0	no	html	small
2	no	7,011	183	0	no	html	big
3	yes	631	28	0	no	text	none
4	no	2,454	61	0	no	text	small
5	no	41,623	1088	9	no	html	small
...							
50	no	15,829	242	0	no	html	small

How are these values defined?

Another dataset

1	However	ADV
2	,	PUNCT
3	it	PRON
4	is	AUX
5	not	PART
6	enough	ADJ
7	to	PART
8	have	AUX
9	attained	VERB
10	such	ADJ
11	native	ADJ
12	-	PUNCT
13	like	ADJ
14	levels	NOUN
15	.	PUNCT

These are part-of-speech (POS) tags.

Does it make sense to distinguish auxiliary verbs from other verbs, and is this really an instance of an auxiliary?

Why is *native-like* split into three pieces?

Punctuation signs are generally separated from the preceding words. This is more difficult than it may seem...

Data preparation

- Preprocessing
 - Remove unwanted elements (empty lines, HTML tags...)
 - Split text into sentences
 - Tokenize (split sentences into “words” / tokens)
 - Case folding
- Feature extraction
 - Define instance scope (word, sentence)
 - Define features and their types
 - Extract feature values from data

Sentence splitting

- How many sentences are in the following text?
There are a number of ways this could happen, the churchmen pointed out, and here is an example. Last month in Ghana an American missionary discovered when he came to pay his hotel bill that the usual rate had been doubled. When he protested, the hotel owner said: “Why do you worry?”
- How would you detect sentence boundaries?
 - Split after .?!:
 - Maybe take into account capitalization
- Difficulties / corner cases?

Sentence splitting

- Corner cases:
 - Is the colon (:) a sentence boundary?
 - The full stop does not always mark sentence boundaries (e.g. abbreviations)
 - But what about abbreviations at the end of a sentence?
 - Titles typically do not end with a punctuation sign, but should be considered as sentences
 - What about incomplete utterances (e.g. on social media)?

Sentence splitting

Example with NLTK:

```
import nltk.tokenize
text = "God is Great! I won a lottery."
sentences = nltk.tokenize.sent_tokenize(text)
print(sentences)
>>> ['God is Great!', 'I won a lottery.']
```

- NLTK `sent_tokenize` provides language-specific models
- Sentence splitters are mostly rule-based (regular expressions)
- SpaCy Sentencizer can use custom rules:
 - <https://spacy.io/api/sentencizer>

Tokenization

- How many words/tokens does the following sentence contain?

For example, this isn't a well-formed example.

- Split at whitespace: 7
- Separate punctuation (,.): 9
- What should be done with hyphenated words?
 - well-formed or well | -formed or well | - | formed ?
- What about contractions?
 - isn't or is | n't or isn | ' | t or is | not ?
- Tokenization is mostly a matter of conventions. There is generally not a single correct solution.

Tokenization

Example with NLTK:

```
import nltk.tokenize
text = "For example, this isn't a well-formed example."
tokens = nltk.tokenize.word_tokenize(text)
print(tokens)
>>> ['For', 'example', ',', 'this', 'isn', "'", 't', 'a', 'well-
formed', 'example', '.']
```

- Uses the Punkt tokenization model for English
 - Simpler tokenizers (based on regular expressions) are available

Case folding

- Should **The** be a different token than **the**?
- Should **Apple** be a different token than **apple**?

- Simplest solution:
 - Convert everything to lowercase
- Truecasing:
 - Collect a list of unique (lowercased) words
 - Count how often each word appears lowercased and capitalized
 - Replace all occurrences at the beginning of a sentence by the most frequent case

Working with text data



Look at the data

- Before and after preprocessing



A lot of things can go wrong here

- But they are usually easy to fix!



Ensure that:

- The character encoding is correct
- There is no unwanted markup
- The correct language is set for language-specific tools
- The same preprocessing is applied to all datasets

Statistical properties of text data

Statistical properties of text data

- What can we do with a large corpus of text?
 - Count words, characters, word sequences, ...
- With these counts, we can
 - Compare different corpora with each other
 - Discover statistical regularities (frequency laws)
 - Build statistical models that can generate new texts and classify existing texts

Count words

- The result is a **frequency distribution**
- Results depend on preprocessing
- What words do you expect to be most frequent?

Word Tally

the	
been	
message	
persevere	
nation	

<https://www.nltk.org/book/ch01.html>

Count words

- These are the most common words in Tom Sawyer:

Chris Manning & Hinrich Schütze (1999):
Foundations of Statistical Natural Language Processing, MIT Press.

- What would the least common words be?
- Words that only occur once in a text are called **hapax legomena** (or hapaxes).

Word	Freq.
the	3332
and	2972
a	1775
to	1725
of	1440
was	1161
it	1027
in	906
that	877
he	877
I	783
his	772
you	686
Tom	679
with	642

Lexical diversity

- **Tokens:** number of words in running text (with repetition)
- **Types:** number of unique words in running text
 - “one cat caught five mice and three cats caught one mouse” → 11 tokens, 9 types
- **Type-token ratio (TTR):** types / tokens, expressed in %
 - 81.8% for the example sentence
 - TTR is a measure of **lexical diversity**
 - Which type of text has higher TTR, a newspaper article or a transcript of an informal conversation?

Collocations

- A **collocation** is a sequence of words that occur together unusually often.
- Examples of two-word collocations for two texts:

```
>>> text4.collocations()  
United States; fellow citizens; four years; years ago; Federal  
Government; General Government; American people; Vice President; Old  
World; Almighty God; Fellow citizens; Chief Magistrate; Chief Justice;  
God bless; every citizen; Indian tribes; public debt; one another;  
foreign nations; political parties  
>>> text8.collocations()  
would like; medium build; social drinker; quiet nights; non smoker;  
long term; age open; Would like; easy going; financially secure; fun  
times; similar interests; Age open; weekends away; poss rship; well  
presented; never married; single mum; permanent relationship; slim  
build
```

<https://www.nltk.org/book/ch01.html>

Zipf's law

The product of the frequency of a word f and its position in the frequency list (rank) r is constant:

$$f \cdot r = k$$

- The first word is twice as frequent as the second.
- The first word is three times as frequent as the third.
- ...

George K. Zipf (1932): *Selected Studies on the Principle of Relative Frequency in Language*. Harvard, MA: Harvard University Press.

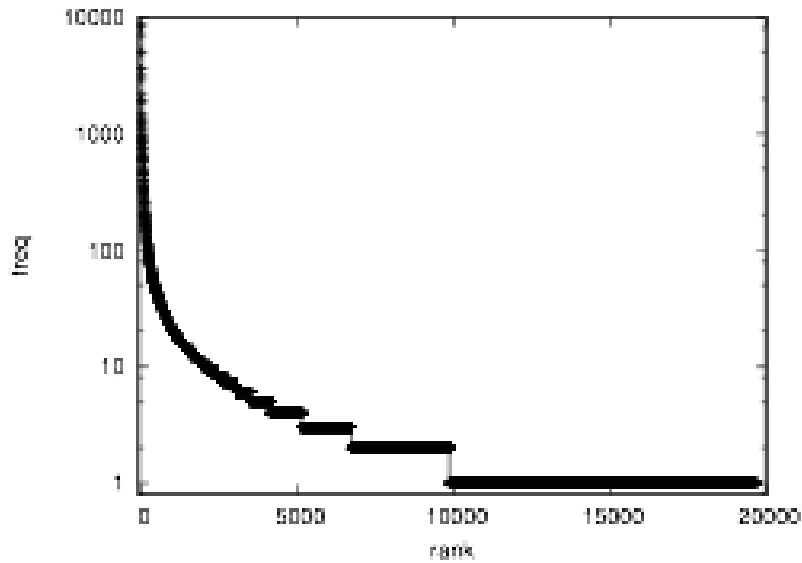
Zipf's law

Word	Freq. (f)	Rank (r)	$f \cdot r$	Word	Freq. (f)	Rank (r)	$f \cdot r$
the	3332	1	3332	turned	51	200	10200
and	2972	2	5944	you'll	30	300	9000
a	1775	3	5235	name	21	400	8400
he	877	10	8770	comes	16	500	8000
but	410	20	8400	group	13	600	7800
be	294	30	8820	lead	11	700	7700
there	222	40	8880	friends	10	800	8000
one	172	50	8600	begin	9	900	8100
about	158	60	9480	family	8	1000	8000
more	138	70	9660	brushed	4	2000	8000
never	124	80	9920	sins	2	3000	6000
Oh	116	90	10440	Could	2	4000	8000
two	104	100	10400	Applausive	1	8000	8000

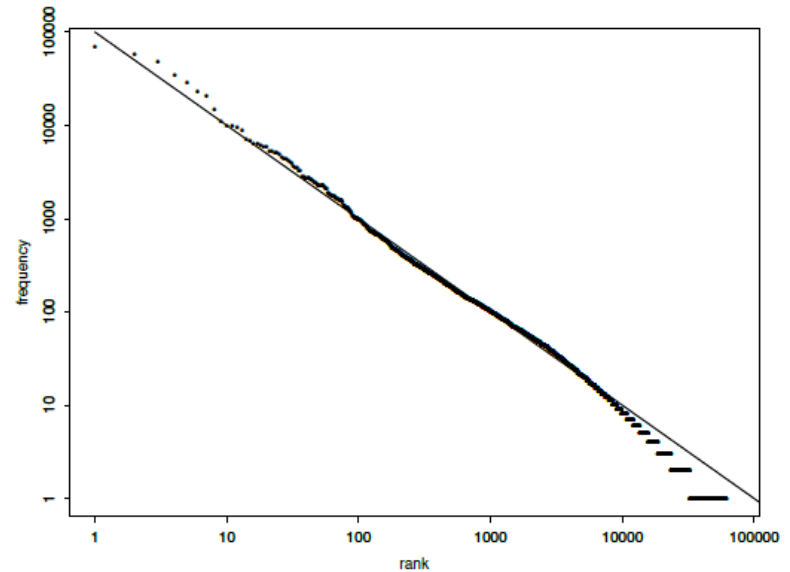
C. Manning & H. Schütze (1999): *Foundations of Statistical Natural Language Processing*, MIT Press, ch. 1.

Zipf's law

Visualization
(log scale on Y-axis
only):



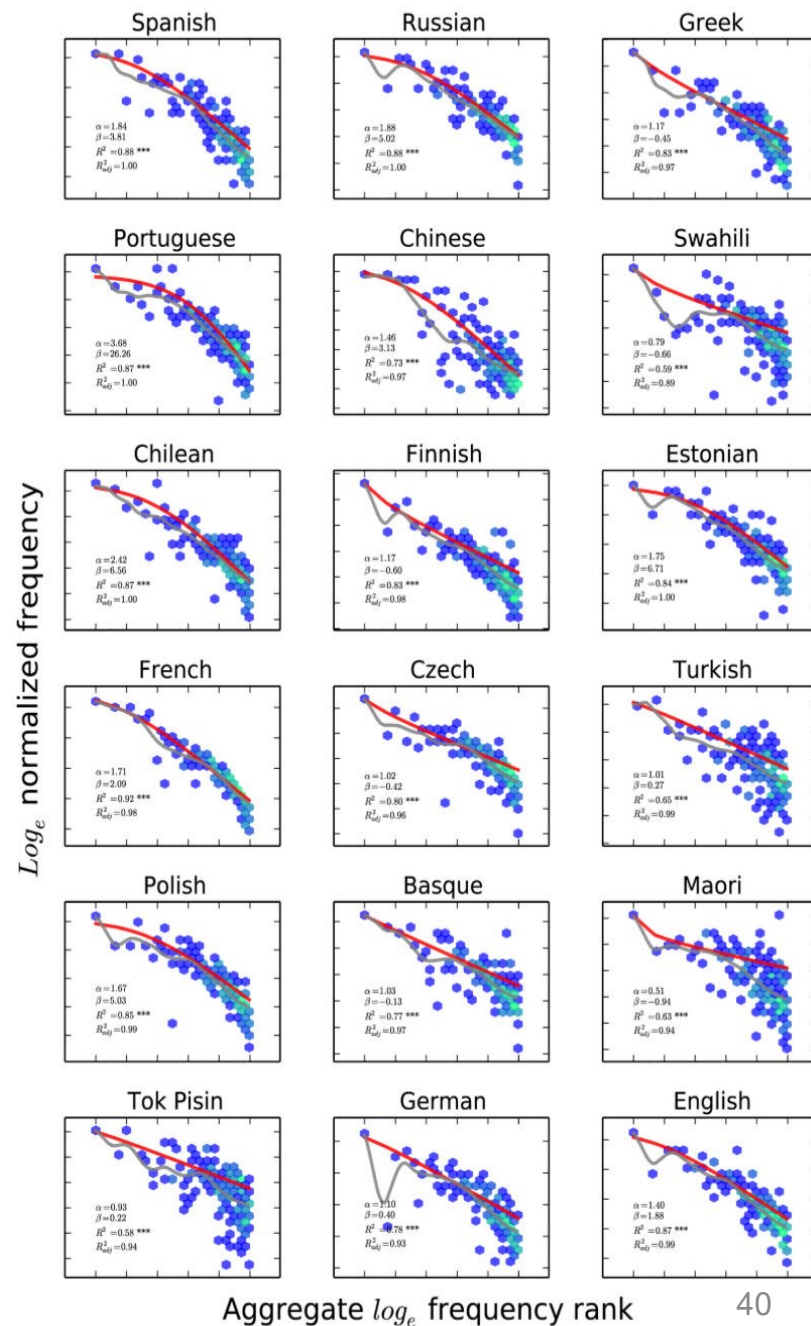
Visualization
(log scale on X- and
Y-axis):



Zipf's law

Is this only true for English?

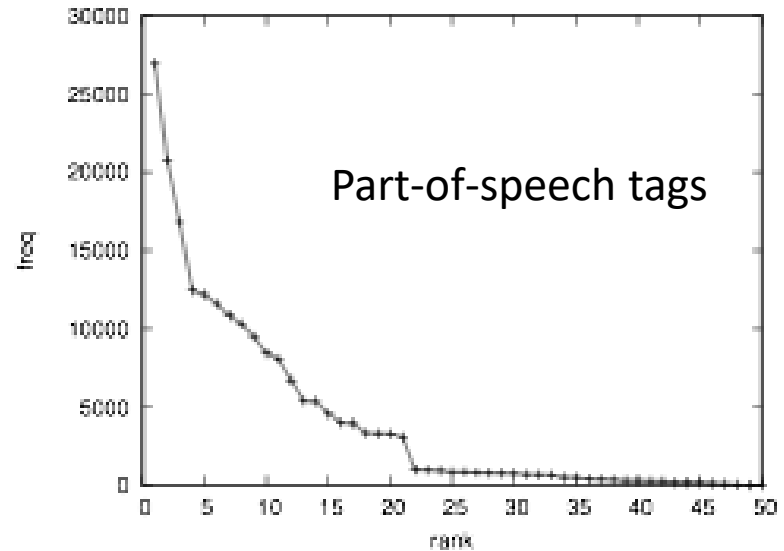
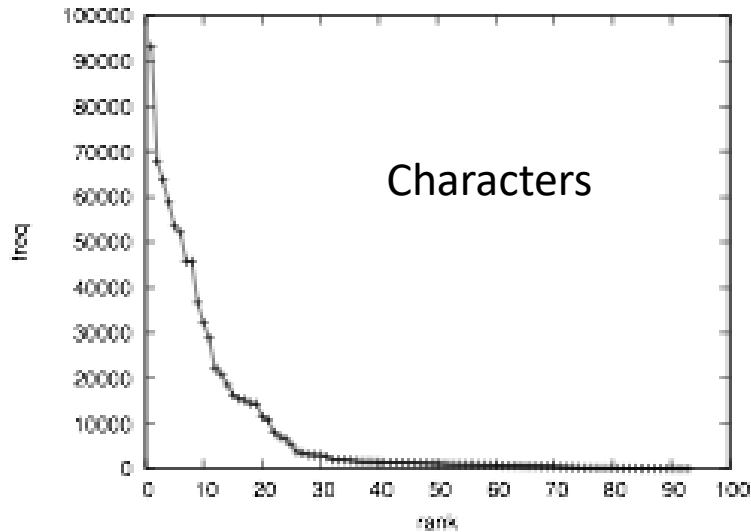
Steven Piantadosi (2014): “Zipf's word frequency law in natural language: A critical review and future directions”. *Psychon Bull Rev.* 21 (5): 1112–1130.



Aggregate log_e frequency rank

Zipf's law

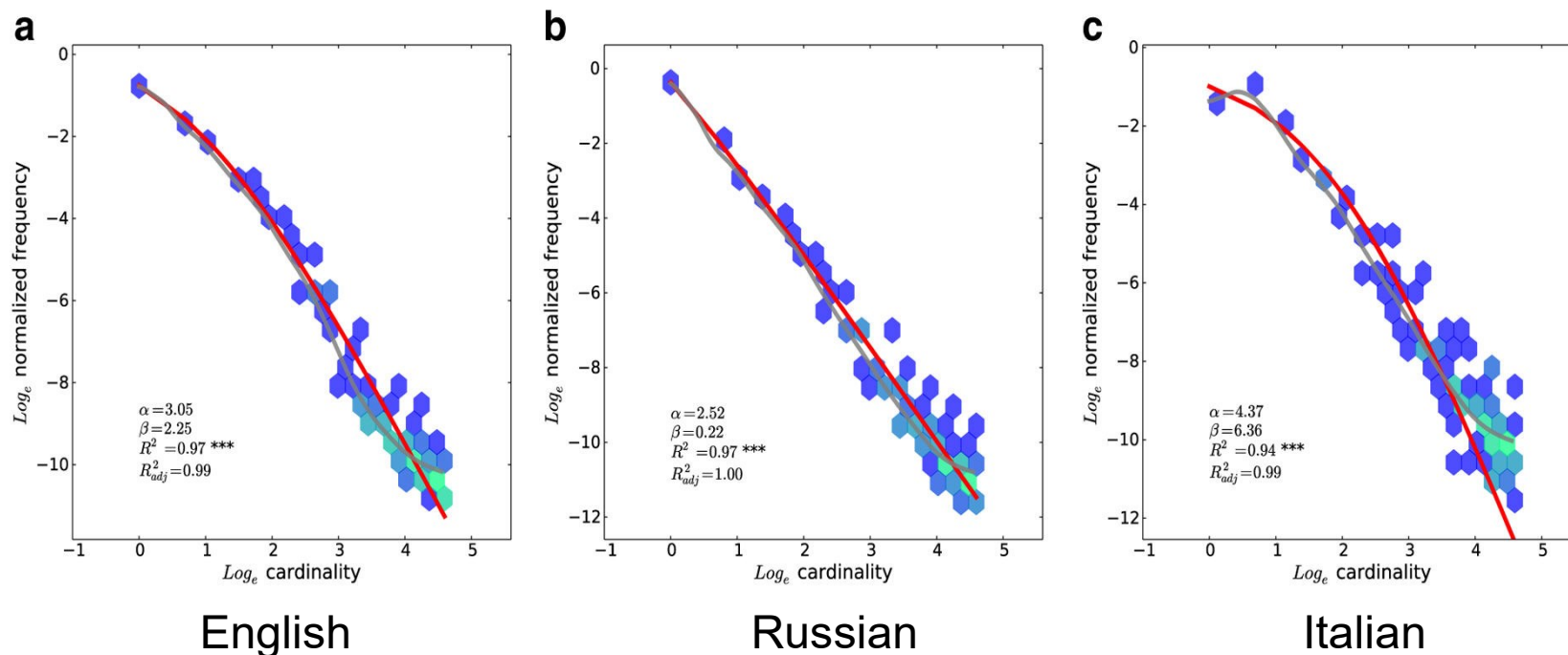
Is this only true for words?



(English)

Zipf's law

The same effect can be seen when looking only at numerals (“one”, “two”, ...) [Piantadosi 2014]:



Zipf's law – Conclusions

Other laws by Zipf:

- The number of **meanings** of a word is correlated with its frequency
- The **length** of a word is inversely correlated with its frequency
- Words tend to clump together in a text

Zipf's law – Conclusions

Zipf's law holds **approximately**:

- Unstable behavior for most frequent words
- Large variation for least frequent words
- The constant (slope of the regression line) differs according to language and data type

Zipf's law – Conclusions

“Psychological” interpretation:

- Zipf: Minimize effort of communication by using a small set of (easily accessible) common words and a large set of rare words (to prevent ambiguity)
- Piantadosi: *Word meaning is a substantial determinant of frequency, and it is perhaps intuitively the best causal force in shaping frequency. “Happy” is more frequent than “disillusioned” because the meaning of the former occurs more commonly in topics people like to discuss.*

Zipf's law – Conclusions

Practical implications:

- Almost all words are rare
- Whatever the size of the corpus, many words occur only once, and many words will not appear at all

Readings

- NLTK book, chapter 1
 - <https://www.nltk.org/book/ch01.html>
 - Simple text statistics
 - Includes a gentle introduction to Python (most of you can probably skip these parts)
- NLTK book, chapter 3
 - <https://www.nltk.org/book/ch03.html>
 - Tokenization, segmentation, text normalization
 - Includes Python and RegEx tutorials (most of you can probably skip these parts)