Cleaning and preprocessing text data, exploring descriptive statistics

Mate Akos

2019 September

MTA TK PTI

For today

1. Importance of research design
2. Some notes on expert coding
3. Validity versus reliability
4. Sampling
5. Unit of analysis
6. Preprocessing and cleaning
7. Feature selection

Consider research design

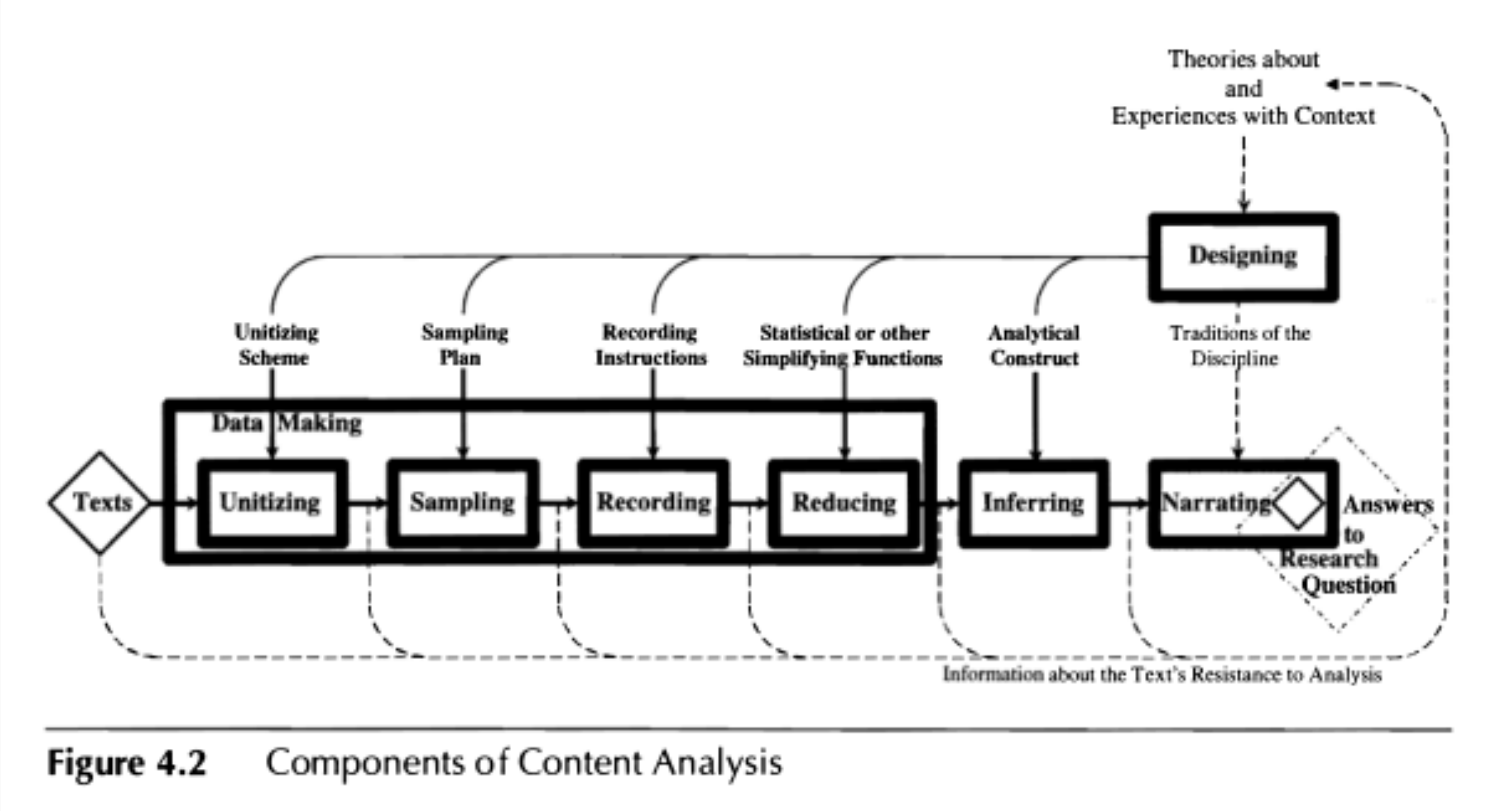

Figure 4.2 Components of Content Analysis

Krippendorff (2004, 86)

Unitizing

Research design choices about the unit of analysis (for expert coding)

- unit of analysis (exogenous vs endogenous debate)
- quasi sentence vs natural language units (sentences)
  - stick to natural units, results are by and large the same (Daubler et al 2012)
- validity-reliability tradeoff

Possible units of analysis (for CATA)

- words
- n-grams
- sentences
- paragraphs
- pages

It all comes down to research design

Reliability vs Validity

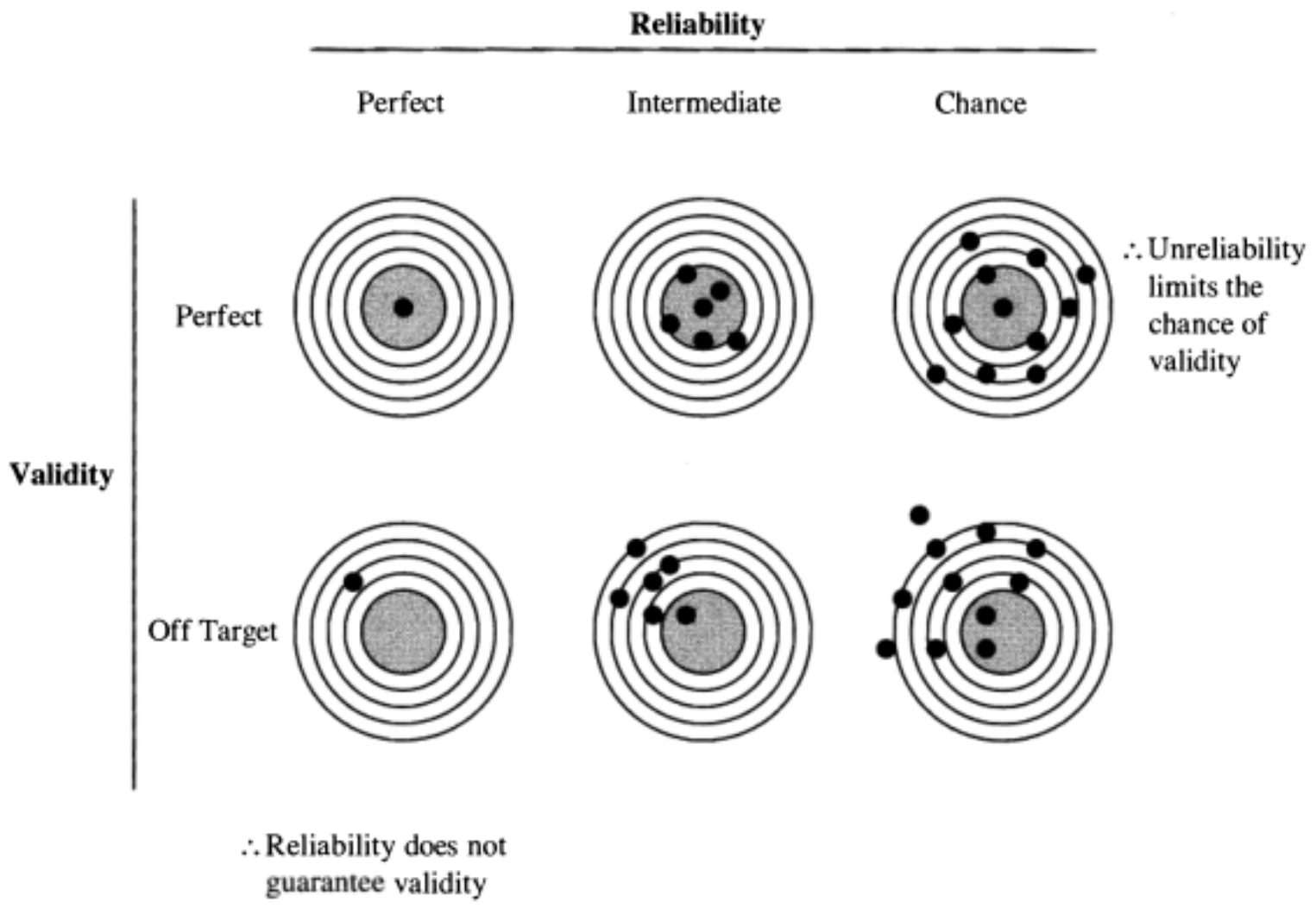

Figure 11.1 The Relationship Between Reliability and Validity

Some measurement theory

- Reliability: *"a research procedure is reliable when it responds to the same phenomena in the same way regardless of the circumstances of its implementation."* (Krippendorff, 2004: 211)
- Validity: How unbiased our measurement is, is it objective? *"provides assurances that the claims emerging from the research are borne out in fact"*
  - **Content validity:** the extent to which our measure reflects the content of the concept
  - **Criterion validity:** the extent to which our measure correlates with another measure which reflects the same concept
  - **Construct validity:** the extent to which our measure behaves as expected within the theoretical context
  - (Zeller & Carmines (1980))

Sampling

- Sample vs. population (do we really need a sample?)
- random vs nonrandom sampling
    - stratified sampling
    - cluster sampling
    - snowball sampling
    - relevance sampling
    - make sure that what is being analyzed is a valid representation of the whole phenomenon

preprocessing / cleaning

1. Punctuation
2. Numbers, symbols, ligatures (fi, fl, ff, etc.)
3. Lowercasing (case sensitivity of dictionaries and other algorithms)
4. Stemming
5. Stopwords (removing noise)
6. n-gram inclusion (including context but increasing the dfm dimension)
7. Trimming the document feature matrix (by docfreq. and/or termfreq.)

Feature selection

Possible features

- characters
- words
- word stems
- coded text segments

Feature selection (making the computational work easier)

- **term frequency**: the frequency of the given feature in the corpus
- **document frequency**: number of documents the feature appears in
- adding **more stopwords** (domain and corpus specific knowledge)
- **purposive selection**: using a dictionary

Feature weighting

- **Term frequency** ($tf_{t,d}$): frequency of term t ($n_{i,j}$) in document d_j (k is the number of terms in the document)
  - $tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$
- **document frequency** (df_t): number of documents containing term t
- **inverse document frequency** (idf_t): idf of a rare term is high; frequent term is low (exact log base is not important as it does not affect ranking)
  - $\mathrm{idf_t} = \log \frac{N}{df_i}$

Feature weighting

- **term frequency - inverse document frequency** ($tf - idf_{t,d}$):
  - **high** when t occurs many times in small number of docs
  - **lower** when t occurs fewer times, or occurs in many more docs
  - **lowest** when the term is frequent accross docs
  - composite weight: $\mathrm{tf} - \mathrm{idf}_{\mathrm{t,d}} = \mathrm{tf}_{\mathrm{t,d}} \times idf_t$

From words to numbers

Our corpus is made up from two documents:

1. ”think think think”
2. ”you never can tell with bees”

After tokenization (using unigrams as tokens), we create a **document-feature matrix** (or document-term matrix). It looks like this:

```
Document-feature matrix of: 2 documents, 7 features
2 x 7 sparse Matrix of class "dfm"
       features
docs    think you never can tell with bees
  text1     3   0     0   0    0    0    0
  text2     0   1     1   1    1    1    1
```

DFM weighting examples

inverse document frequency:

```
  think     you   never     can    tell    with
0.30103 0.30103 0.30103 0.30103 0.30103 0.30103 0.3
```

term frequency:

```
Document-feature matrix of: 2 documents, 7 features
2 x 7 sparse Matrix of class "dfm"
       features
docs    think       you     never       can       te
  text1     1 0         0         0         0
  text2     0 0.1666667 0.1666667 0.1666667 0.16666
```

tf – idf:

```
Document-feature matrix of: 2 documents, 7 features
2 x 7 sparse Matrix of class "dfm"
       features
docs       think     you   never     can    tell    
  text1 0.90309 0       0       0       0       0   
  text2 0       0.30103 0.30103 0.30103 0.30103 0.3
```

Excercise

We have a corpus with **500 documents**, with **2500 wordcount each**. Our term in question is *"rebellion"*. It appears in **100 documents**. In the first document, **it appears 50 times**.

- term frequency is $50/2500 = 0.02$

Excercise

We have a corpus with **500 documents**, with **2500 wordcount each**. Our term in question is *"rebellion"*. It appears in **100 documents**. In the first document, **it appears 50 times**.

- term frequency is $50/2500 = 0.02$
- document frequency is $500/50 = 10$, thus idf is $log(10) = 2.3025$

Excercise

We have a corpus with **500 documents**, with **2500 wordcount each**. Our term in question is *"rebellion"*. It appears in **100 documents**. In the first document, **it appears 50 times**.

- term frequency is $50/2500 = 0.02$
- document frequency is $500/50 = 10$, thus idf is $log(10) = 2.3025$
- the tf-idf then: $0.02 * 2.3025 = 0.046$

Zipf’s law

The freqiency of a given word is inversely proportional to its rank in the frequency table.

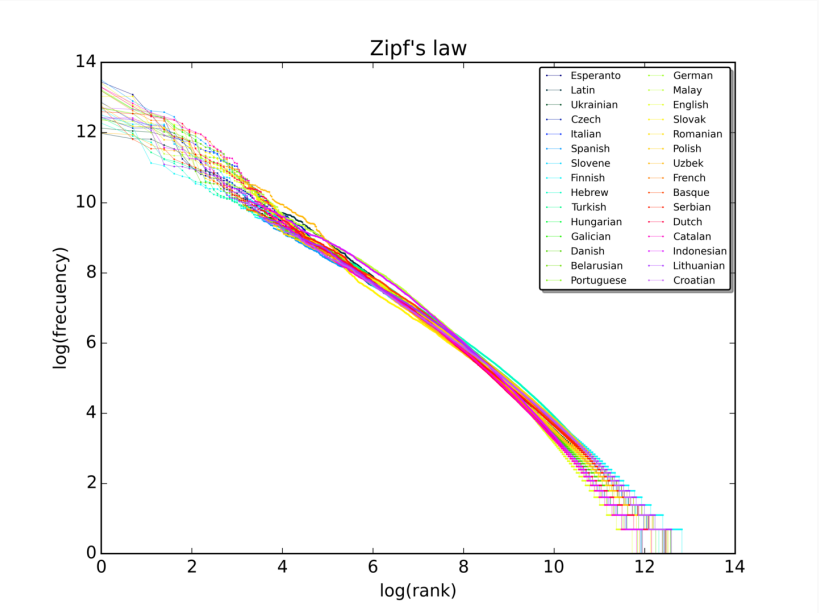

Figure 2: A plot of the rank versus frequency for the first 10 million words in 30 Wikipedias (log-log scale)

(figure source: Wikipedia: Zipf’s law)

QTA in R

An overview of text analysis operations, with the R packages used in this teacher's corner

Operation	R packages	
	example	alternatives
Data preparation		
importing text	readtext	jsonlite, XML, antiword, readxl, pdftools
string operations	stringi	stringr
preprocessing	quanteda	stringi, tokenizers, snowballC, tm, etc.
document-term matrix (DTM)	quanteda	tm, tidytext, Matrix
filtering and weighting	quanteda	tm, tidytext, Matrix
Analysis		
dictionary	quanteda	tm, tidytext, koRpus, corpustools
supervised machine learning	quanteda	RTextTools, kerasR, austin
unsupervised machine learning	topicmodels	quanteda, stm, austin, text2vec
text statistics	quanteda	koRpus, corpustools, textreuse
Advanced topics		
advanced NLP	spacyr	coreNLP, cleanNLP, koRpus
word positions and syntax	corpustools	quanteda, tidytext, koRpus

Figure 1. Order of text analysis operations for data preparation and analysis.

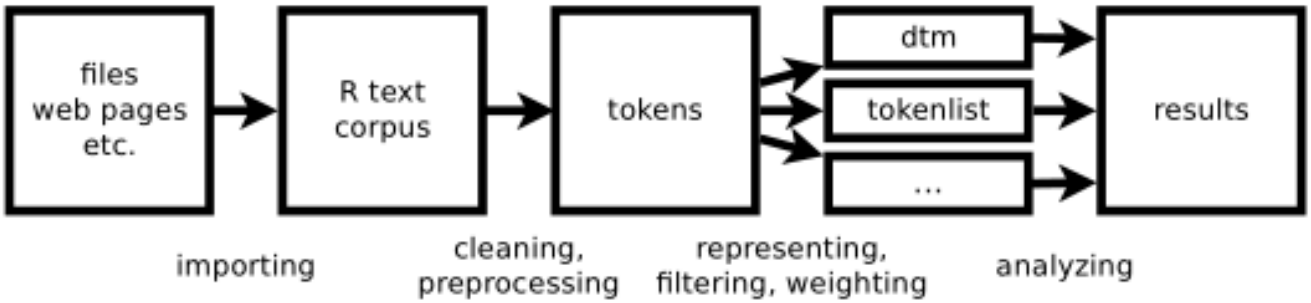

Figure 3

And now, R!