plain text and data cleaning

DTL SU @ AU

kristoffer l nielbo

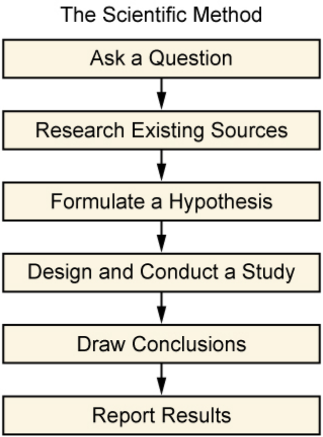
kln@cas.au.dk

github.com/kln-courses/tmgu17

tmgu17.slack.com

DAI|IMC|AARHUS UNIVERSITY

generic research



The Scientific Method

Ask a Question

Research Existing Sources

Formulate a Hypothesis

Design and Conduct a Study

Draw Conclusions

Report Results

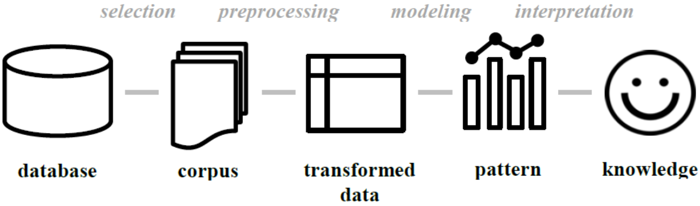
design types:

- exploratory
- descriptive
- causal
- prototyping

Independent of type, the actual process is always iterative

⇒ ‘Mixed model’ of research design

TM workflow



General multistep process of knowledge discovery

Constructed out of a need for handling ‘extraction of useful information (knowledge) from rapidly growing volumes of digital data’

For each project you develop a pipeline within this workflow

Accessing data

Select specific documents (target data/corpus) relevant to your research question from the database.

Online databases and research libraries are excellent resources

- Proprietary issues
- Data Protection Acts
- Avaialbility (e.g., historical sources)

we will focus on documents stored locally in a *plain text* without markup

```
The First Book of Moses, called Genesis

   {1:1} In the beginning God created the heaven and the earth. {1:2}
And the earth was without form, and void; and darkness was upon the
face of the deep. And the Spirit of God moved upon the face of the
waters.

   {1:3} And God said, Let there be light: and there was light. {1:4}
And God saw the light, that it was good: and God divided the light
```

Python is very accommodating though

```
import urllib2
from HTMLParser import HTMLParser

class html_parser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print "start tag:", tag
    def handle_endtag(self, tag):
        print "end tag :", tag
    def handle_data(self, data):
        print "data  :", data

url = ’http://www.au.dk/en/’
response = urllib2.urlopen(url)
webpage = response.read()

parser = html_parser()
parser.feed(webpage)
```

> “You gotta know when to be lazy. Done correctly, it’s an art form that benefits everyone.” (Nicholas Sparks, *The Choice*)

`Library` - collection of resources (code and data) and associated documentation

```
$ conda list
#from /github.com/kln-courses/tmgu17/blob/master/code_sample/week1/list_libs.py
$ python list_libs.py
```

search, install and update packages through Conda

```
$ conda search gensim
$ conda install gensim

$ conda update gensim
$ conda update conda

$ conda env list
$ conda list -n wintermute
$ conda install --name wintermute gensim
```

multiple environments

```
$ conda info --envs
$ conda create --name mrWhite python=2
$ conda info --envs
$ conda create -n mypy3 python=3.6 anaconda
```

use different environments

```
$ source activate mrWhite # Win: activate mrWhite
$ source deactivate # Win: deactivate
```

delete environments

```
conda remove --name mrWhite --all
```

preprocessing

to prepare a document we need to parse, slice and split it at the relevant level(s).

unstructured data are very noisy, so to increase the signal, we therefore remove irrelevant data through `preprocessing`

range of text normalization techniques to `preprocess` the data:

- casefolding
- removal of non-alphanumeric characters (punctuation, blanks)
- removal of numeral and stopwords
- vocablary pruning
- identification of parts of speech
- reduction of inflectional forms through stemming and lemmatization
- disambiguation
- synonym substitution
- ...

remember that **one man's rubbish may be another's treasure**

example #1

`Sentence Boundary Disambiguation` (sentence breaking)
- identyfying where sentences begin and end

- punctuation marks are often ambiguous, "." abbreviation, decimal point, an ellipsis, or an email address – not the end of a sentence

- question marks "?" and exclamation "!" marks may appear in embedded quotations, emoticons, computer code, and slang

- Chinese (& Japanese) have unambiguous sentence-ending markers

naive approach

1. if ".", sentence is ended
2. search dictionary of abbrevations
3. if preceeding character is CAPS, then sentence ended

example #2

`part-of-speech` tagging (word-category disambiguation)
- marking up a word in a text as corresponding to a particular part of speech

- utilize word definition and context

- identification of words as nouns, verbs, adjectives, adverbs &c

- algorithmic approach associate discrete terms in accordance with a set of tags
- taggers: rule-based and stochastic

example #3

`stemming`
- reducing inflected words to their stem, base or root form

- the stem need *not* be identical to the morphological root
- sufficient that related words map to the same stem (stem $\neq$ valid root)

- search engines treat words with the same stem as synonyms (`conflation`)

`Porter stemming algorithm - step 1a`

```
SSES -> SS     caresses -> caress
IES  -> I      ponies   -> poni
               ties     -> ti
SS   ->  SS    caress   -> caress
S    ->        cats     -> cat
```

AARHUS UNIVERSITET

IMC
INTERACTING MINDS CENTRE

example #4

`lemmatization`

- grouping inflected forms of a word so they can be analyzed as a single item - algorithmic process of determining the lemma for a given word based in context, part of speech

- words tend to appear in several inflected forms (”to walk” - ”walk”, ”walked”, ”walks”, ”walking”). The base form (”walk”) is the dictionary form called the lemma of the word - lexeme: combination of the base form and part of speech - while

stemmers operate on a single word with no cotext, lemmatizers use context and can discriminate word meaning depending on part of speech.

- “lemmatization is stemming done right” (anonymous CST researcher), but stemmers are faster (at the cost of accuracy)