EC999: Text Normalization

Thiemo Fetzer

University of Chicago & University of Warwick

March 30, 2017

Quantitative Text Analysis as process

Above all, there needs to be a formulatedresearch question or **goal** to be achieved.

```
When I presented the
supplementary budget to
this House last April, I
said we could work our
way through this period
of severe economic
distress. Today, I can
report that
notwithstanding the
difficulties of the past
eight months, we are now
on the road to economic
recovery.
```

```
In this next phase of the
Government's plan we must
stabilise the deficit in
a fair way, safeguard
those worst hit by the
recession, and stimulate
crucial sectors of our
economy to sustain and
create jobs. The worst is
over.
```

```
This Government has the
moral authority and the
well-grounded optimism
rather than the cynicism
of the Opposition. It has
the imagination to create
the new jobs in energy,
agriculture, transport
and construction that
this green budget will
```

docs	words made	because	had	into	get	some	through	next	where	many	irish
t06_kenny_fg	12	11	5	4	8	4	3	4	5	7	10
t05_cowen_ff	9	4	8	5	5	5	14	13	4	9	8
t14_ocaolain_sf	3	3	3	4	7	3	7	2	3	5	6
t01_lenihan_ff	12	1	5	4	2	11	9	16	14	6	9
t11_gormley_green	0	0	0	3	0	2	0	3	1	1	2
t04_morgan_sf	11	8	7	15	8	19	6	5	3	6	6
t12_ryan_green	2	2	3	7	0	3	0	1	6	0	0
t10_quinn_lab	1	4	4	2	8	4	1	0	1	2	0
t07_odonnell_fg	5	4	2	1	5	0	1	1	0	3	0
t09_higgins_lab	2	2	5	4	0	1	0	0	2	0	0
t03_burton_lab	4	8	12	10	5	5	4	5	8	15	8
t13_cuffe_green	1	2	0	0	11	0	16	3	0	3	1
t08_gilmore_lab	4	8	7	4	3	6	4	5	1	2	11
t02_bruton_fg	1	10	6	4	4	3	0	6	16	5	3

Descriptive statistics on words

Scaling documents

Classifying documents

Extraction of topics

Vocabulary analysis

Sentiment analysis

Building A Corpus

- Textual data can be stored in multiple different formats
  - JSON (JavaScript Object Notation) is a lightweight data-interchange format for structured data.
  - XML
  - Flat text files
  - (machine readable?) PDFs
- Parsing or reading text data into R can be achieved by a range of functions.

```r
# con can be any connection, could be a URL or a path to a file
TEXT <- readLines(con = "https://www.dropbox.com/s/eynnvac4kurnjon/speaches-2016-election.json?dl=1",
    encoding = "UTF-8")
# commonly used for tabular data formats
TEXT <- read.table(file = "file.txt")
# may need to iterate over a whole folder of documents
TEST <- lapply(list.files("Path"), function(x) readLines(con = x))
```

An Example: Congressional speeches

- Loading Congressional speaches by important figures important to the 2016 presidential election.
- Data is JSON format
- Each line is a speach given by a member of congress.
- JSON data provides string excerpt as well as meta-information: date, party, speaker, chamber,...


```
{"congress":104,"title":"JOIN THE SENATE AND PASS A CONTINUING RESOLUTION","text":"Mr.
Speaker, 480,000 Federal employees are working without pay, a form of involuntary
servitude; 280,000 Federal employees are not working, and they will be paid. Virtually all
of these workers have mortgages to pay, children to feed, and financial obligations to
meet.\nMr. Speaker, what is happening to these workers is immoral, is wrong, and must be
rectified immediately. Newt Gingrich and the Republican leadership must not continue to
hold the House and the American people hostage while they push their disastrous 7-year
balanced budget plan. The gentleman from Georgia, Mr. Gingrich, and the Republican
leadership must join Senator Dole and the entire Senate and pass a continuing resolution
now, now to reopen Government.\nMr. Speaker, that is what the American people want, that
is what they need, and that is what this body must
do.","chamber":"House","speaker_party":"I","date":"1996-01-04","speaker_name":"Bernie
Sanders"}
{"congress":104,"title":"MEETING THE CHALLENGE","text":"Mr. Speaker, a relationship, to
work and survive, has got to be honest and we have got to deal with each other in good
faith. For a government to govern well, we have to be honest and we have to deal with each
other in good faith.\nThe President has vetoed every measure we have sent to him that
would balance the budget. He has a constitutional right to do that. If he believes that
our budget devastates the elderly, he has a moral obligation to fight us. I will never,
never say bad things about somebody that follows their beliefs because that is what they
should do. There comes a time, though, that one has an obligation to do more than just say
no.\nMr. President, if you do not like our view of a balanced budget, give us your view.
We cannot negotiate against ourselves anymore. You have a legal and a moral obligation to
fight us when you think we are wrong. You have a legal and moral obligation to fulfill
your commitment you made 40 days ago to put a budget on the table that balances. Please
fulfill your
obligation.","chamber":"House","speaker_party":"R","date":"1996-01-04","speaker_name":"
Lindsey Graham"}
```

An Example: Congressional speaches

```r
options(stringsAsFactors = FALSE)
library(data.table)
library(RJSONIO)
library(quanteda)
TEXT <- readLines(con = "https://www.dropbox.com/s/eynnvac4kurnjon/speaches-2016-election.json?dl=1")
TEXT[1]
```

```
## [1] "{\"congress\":104,\"title\":\"JOIN THE SENATE AND PASS A CONTINUING RESOLUTION\",\"text\":\"Mr. Speake
```

```r
SPEECHES <- lapply(TEXT, function(x) data.frame(fromJSON(x)))
SPEECHES <- rbindlist(SPEECHES)
SPEECHES[1]
```

```
##    congress                                            title
## 1:      104 JOIN THE SENATE AND PASS A CONTINUING RESOLUTION
##
## 1: Mr. Speaker, 480,000 Federal employees are working without pay, a form of involuntary servitude; 280,000
##    chamber speaker_party       date   speaker_name
## 1:   House             I 1996-01-04 Bernie Sanders
```

An Example: A Corpus of Congressional speaches

```r
CORPUS <- corpus(SPEECHES$text)
CORPUS[["congress"]] <- SPEECHES$congress
CORPUS[["speaker_name"]] <- SPEECHES$speaker_name
CORPUS[["speaker_party"]] <- SPEECHES$speaker_party
CORPUS[["date"]] <- SPEECHES$date
summary(CORPUS, n = 10)
```

```
## Corpus consisting of 11376 documents, showing 10 documents.
##
##    Text Types Tokens Sentences congress   speaker_name speaker_party       date
##   text1    86    163         6      104 Bernie Sanders             I 1996-01-04
##   text2   111    218        12      104 Lindsey Graham             R 1996-01-04
##   text3   158    337        17      104 Bernie Sanders             I 1996-01-05
##   text4   104    176         6      104 Bernie Sanders             I 1996-01-05
##   text5   589   1852        80      104  Rick Santorum             R 1996-01-22
##   text6    16     18         1      104  Rick Santorum             R 1996-01-22
##   text7   123    197         6      104 Bernie Sanders             I 1996-01-24
##   text8   115    182         4      104 Bernie Sanders             I 1996-01-25
##   text9    18     20         1      104 Bernie Sanders             I 1996-01-25
##  text10    98    171         6      104 Bernie Sanders             I 1996-01-25
##
## Source:  /Users/thiemo/Dropbox/Teaching/Quantitative Text Analysis/Week 2a/* on x86_64 by thiemo
## Created: Wed Nov 16 11:54:00 2016
## Notes:
```

Fundamentals about text data

There are very few "fundamental law's" in computational linguistic. The exception are *Heap's Law* and *Zipf's Law*, which highlights why most text data is *sparse*.

Typically we will define a model of language that is a *stochastic* process.

- Study the single occurence of a word, not its frequency - *Bernoulli process*
- Modeling word frequencies: *Poisson* or *multionomial* distribution.

Heap's Law

Heaps' law (also called Herdan's law) is an empirical relationship which describes the number of *distinct words* in a document (or set of documents) as a function of the *document length* (so called type-token relation). It can be formulated as

$$|V| = kN^{\beta}$$

In log-log form, this power law becomes a straight line

$$\log(|V|) = k + \beta \log(N)$$

where $|V|$ is the size of the vocabulary (the number of types) and N is the number of tokens.

Illustration of Heap's Law in State of Union speeches

```r
library(quanteda)
library(data.table)
data(SOTUCorpus, package = "quantedaData")
DF <- summary(SOTUCorpus)
plot(log(DF$Types), log(DF$Tokens)) + abline(lm(log(DF$Tokens) ~ log(DF$Types)))
```

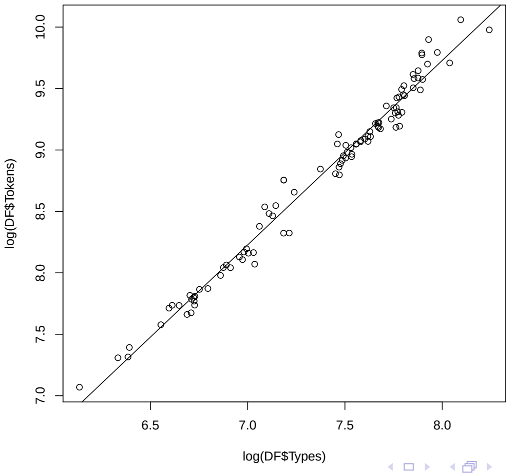


Illustration of Heap's Law in State of Union speeches

```
summary(lm(log(DF$Tokens) ~ log(DF$Types)))

##
## Call:
## lm(formula = log(DF$Tokens) ~ log(DF$Types))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.2245 -0.0664 -0.0120  0.0603  0.2751
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -2.2803     0.1569   -14.5   <2e-16 ***
## log(DF$Types)   1.5011     0.0212    70.7   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1 on 98 degrees of freedom
## Multiple R-squared:  0.981,Adjusted R-squared:  0.981
## F-statistic: 5e+03 on 1 and 98 DF,  p-value: <2e-16
```

For larger corpora, the coefficient is typically smaller. Stemming and further tokenization typically lowers the vocabulary space.

Zipf's Law

Zipf's Law is a law about the frequency distribution of words *within a document*.
Zipf's Law states that s frequency of any word is inversely proportional to its rank in the frequency table.
Formally: Word frequency

$$f = \frac{a}{r^b}$$

where r is the rank in the (empirical) word frequency distribution.
Again, logging

$$\log(f) = \log(a) - b\log(r)$$

Illustration of Zipf's Law

```
OBAMA <- subset(SOTUCorpus, filename == "su2012.txt")
TOK <- tokenize(OBAMA, removePunct = TRUE)
TOK <- data.table(token = tolower(unlist(TOK)))
TOK <- TOK[, .N, by = token][order(N, decreasing = TRUE)]
TOK[1:20]
```

```
##     token   N
##  1:   the 294
##  2:    to 230
##  3:   and 204
##  4:    of 170
##  5:     a 160
##  6:  that 144
##  7:    in 108
##  8:   our  84
##  9:    we  84
## 10:   for  63
## 11:    is  59
## 12:  will  57
## 13:  this  52
## 14:    on  51
## 15:     i  50
## 16:    it  48
## 17:  with  47
## 18:  from  47
## 19:  more  43
## 20:    as  39
```

```
TOK[, `:=`(rank, 1:nrow(TOK))]
```

Illustration of Zipf's Law

```
plot(log(TOK$N), log(TOK$rank)) + abline(lm(log(TOK$N) ~ log(TOK$rank)))
## numeric(0)
```

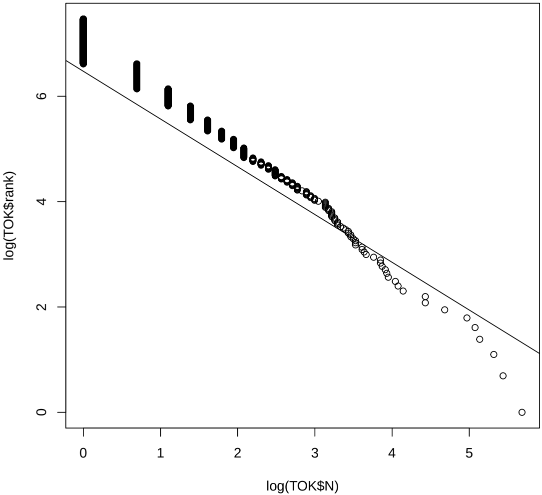


Illustration of Zipf's Law

```
summary(lm(log(TOK$N) ~ log(TOK$rank)))
##
## Call:
## lm(formula = log(TOK$N) ~ log(TOK$rank))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.7907 -0.1110  0.0411  0.1406  0.2938
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     6.47428    0.02937     220   <2e-16 ***
## log(TOK$rank) -0.90642    0.00449    -202   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.186 on 1747 degrees of freedom
## Multiple R-squared:  0.959,Adjusted R-squared:  0.959
## F-statistic: 4.08e+04 on 1 and 1747 DF,  p-value: <2e-16
```

Implications of Heap's and Zipf's Law

- *Heap's Law* and *Zipf's Law* imply that data matrices constructed from text data is very *sparse*.
- Sparsity implies that there would be many zeroes.
- Most data processing steps for text data involve *densifying* the word frequency distribution.
- We next discuss a range of steps commonly used to densify.

Word Tokenization and Normalization

- **Tokenization** - task of segmenting running text into words.
  - Plain vanilla approaches would just `str_split(text," ")` - splitting by white spaces.
  - More sophisticated methods apply *locale* (language) specific algorithms.
- **Normalization**- task of putting words/tokens into a standardized format.
  - For example `we're` to `we are`.
  - Casefolding of tokens (lower-case or upper case)

quanteda tokenization routine

We have already used the functionality in a few illustrations, but lets systematically introduce it here.

```
tokenize(x, what = c("word", "sentence", "character", "fastestword", "fasterword"), removeNumbers = FALSE,
    removePunct = FALSE, removeSymbols = FALSE, removeSeparators = TRUE, removeTwitter = FALSE,
    removeHyphens = FALSE, removeURL = FALSE, ngrams = 1L, skip = 0L, concatenator = "_", simplify = FALSE,
    verbose = FALSE, ...)
```

Tokenization function allows separation of words, sentences and individual characters from a character vector x or a corpus object.

`what` the unit for splitting the text, available alternatives are:

`"word"`
: (recommended default) smartest, but slowest, word tokenization method; see stringi-search-boundaries for details.

`"fasterword"`
: dumber, but faster, word tokenizeation method, uses stri_split_charclass(x, "\\pWHITE_SPACE")

`"fastestword"`
: dumbest, but fastest, word tokenization method, calls `stri_split_fixed(x, " ")`

`"character"`
: tokenization into individual characters

`"sentence"`
: sentence segmenter, smart enough to handle some exceptions in English such as "Prof. Plum killed Mrs. Peacock." (but far from perfect).

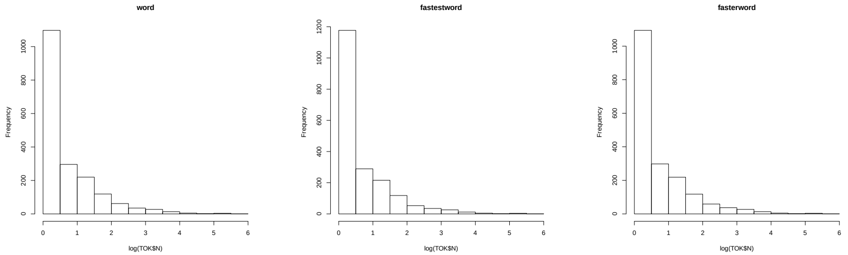
quanteda tokenization routine

- Dumb tokenization approach works reasonably well for languages based on latin alphabet.
- You may end up tokenizing features that you do not really want to separate, like *Named Entities* - `New York` (next week we will work on detecting such n-grams)
- Works poorly for languages that do not use white space character for separation (e.g. Chinese)
- `word` option uses the `BreakIterator` algorithm that implements the *Unicode Text Segmentation* standard
- Words boundaries are identified according to the rules in `http://www.unicode.org/reports/tr29/#Word_Boundaries`, supplemented by a word dictionary for text in Chinese, Japanese, Thai or Khmer. The rules used for locating word breaks take into account the alphabets and conventions used by different languages.

quanteda tokenization routine

Lets look at impact of the three alternative word tokenization methods for Obama's speeches.

```
for (i in c("word", "fastestword", "fasterword")) {
    TOK <- data.table(tok = unlist(tokenize(OBAMA, what = i, removePunct = TRUE)))[, .N, by = tok][order(N,
        decreasing = TRUE)]
    TOK[, `:=`(rid, 1:nrow(TOK))]
    plot(hist(log(TOK$N)), main = i)
}
```



We *may* want to shift more mass to the right (higher counts). Little effect of densification of distribution of token counts using different methods.

quanteda tokenization routine

- Depending on the application or which the data is prepared, it may make sense to normalize text by lowercaseing it, removing punctuation (as we have done already).
- This may introduce *noise* or inaccuracies, but its important to bear in mind what is the goal of the application.
- in *R*, lowercasing is achieved with the function `tolower()`.

Lemmatization and Stemming

Sparsity is a central issue as it blows up the underlying data matrices we work with. There are a range of methods to select features and densify resulting data matrices.

document frequency cutoffs around how many documents does a term appear.

term frequency cutoffs around how often a term appears in a corpus

lemmatization densification based on identified linguistic roots, disregarding the underlying parts of speech (verbs and adjective)

deliberate disregard exclude a range of stop words: words that do not provide independent substantive content

purposive selection use of dictionaries of words or phrases, possible identified from the underlying data (like collocations) or identified as having "predictive content" along dimension of interest.

declared equivalency class work of synonyms and map word (stems) to their underlying synonym

We will discuss these in the next set of slides...

Lemmatization and Stemming

Lemmatization is the task of determining that two words have the same linguisting root.

- `am, are, is` have the same root being be
- Plural's for nouns, in English usually identified by an added `s` share the same root.
- Other gramatic constructs, like *superlatives*...

The most common approach for English is to work with the *Porter stemmer*, which simply chops off affixes. More complex methods use look up tables or augment process with information on the Part of Speech.

Porter Stemmer

- Algorithm dates from 1980
- Still the default "go-to" stemmer as it provides a good trade-off between speed, readability, and accuracy
- Stems using a set of rules, or transformations, applied in a succession of steps
- In total there are about 60 rules in 6 steps that are applied iteratively

The sequence of steps can be summarized as follows:

1. Get rid of plurals and `-ed` or `-ing` suffixes
2. Turns terminal `y` to `i` when there is another vowel in the stem
3. Maps double suffixes to single ones: `-ization`, `-ational`, etc.
4. Deals with suffixes, `-full`, `-ness` etc.
5. Takes off `-ant`, `-ence`, etc.
6. Removes a final `-e`

Porter Stemmer Examples

1. Get rid of plurals and `-ed` or `-ing` suffixes
2. Turns terminal `y` to `i` when there is another vowel in the stem
3. Maps double suffixes to single ones: `-ization`, `-ational`, etc.
4. Deals with suffixes, `-full`, `-ness` etc.
5. Takes off `-ant`, `-ence`, etc.
6. Removes a final `-e`

Semantically → semantically → semanticalli → semantical → semantic → semant → semant.

Destructiveness → destructiveness → destructiveness → destructive → destructive → destruct → destruct

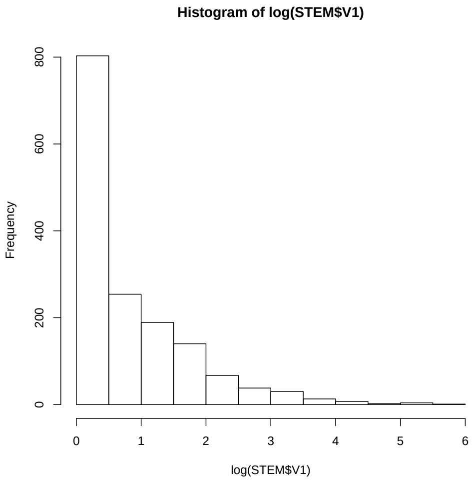
Recognizing → recognize → recognize → recognize → recognize → recognize → recognize
Online illustration: `http://9ol.es/porter_js_demo.html`

R implementation

Most implementaiton of Porter stemmer used in R are actually coded in C, as $C++$ is much faster in processing.

```
library("SnowballC")

wordStem("Amazing")

## [1] "Amaze"

# multiple languages are supported
getStemLanguages()

##  [1] "danish"     "dutch"      "english"    "finnish"    "french"     "german"
##  [7] "hungarian"  "italian"    "norwegian"  "porter"     "portuguese" "romanian"
## [13] "russian"    "spanish"    "swedish"    "turkish"

wordStem("Liebschaften", language = "de")

## [1] "Liebschaft"

wordStem("amaren", language = "es")

## [1] "amar"

# densification?
TOK[, `:=`(stemmed, wordStem(tok))]
nrow(TOK)

## [1] 1877

summary(TOK$N)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     1.0     1.0     3.7     3.0   278.0

STEM <- TOK[, sum(N), by = stemmed]
plot(hist(log(STEM$V1)))
```

Stemming reduces dimensionality



Mass is shifted to the right, away from words occuring just once.

Stemming Issues

Stemming is an approximation to the Lemmatization task which generally provides for a good trade-off between accuracy and speed. Most are stimple rule based algorithms.

- Stemmers are rudimentary approach to morphological analysis
- No word sense disambiguation ("Police" vs "policing")
- No Part of Speech disambiguation ("Policing" could be noun or verb, but "hitting" could only be verb)
- However other approaches to lemmatization in practice does not do much better.

We just briefly introduce an alternative *R* package that implements a morphological approach.

hunspell package

- `hunspell` is actually the spell checker used in Google Chrome, which is also used by other proprietary software packages.
- Has a significant capacities to identify lemmas of words using a dictionary lookup approach.

```
words <- c("severing", "several", "ironic", "iron", "animal", "animated")
wordStem(words)
```

```
## [1] "sever" "sever" "iron"  "iron"  "anim"  "anim"
```

```
library(hunspell)
# hunspell_stem(words)
hunspell_analyze(words)
```

```
## [[1]]
## [1] " st:severing"   " st:sever fl:G"
##
## [[2]]
## [1] " st:several"
##
## [[3]]
## [1] " st:ironic"
##
## [[4]]
## [1] " st:iron"
##
## [[5]]
## [1] " st:animal"
##
## [[6]]
## [1] " st:animated"     " st:animate fl:D"
```

stopwords

Stopwords are words that typically contain no informational content, they may be articles, prepositions, ...

```
stopwords("english")[1:20]

##  [1] "i"          "me"         "my"         "myself"     "we"         "our"
##  [7] "ours"       "ourselves"  "you"        "your"       "yours"      "yourself"
## [13] "yourselves" "he"         "him"        "his"        "himself"    "she"
## [19] "her"        "hers"

stopwords("spanish")[1:20]

##  [1] "de"   "la"   "que"  "el"   "en"   "y"    "a"    "los"  "del"  "se"   "las"  "por"
## [13] "un"   "para" "con"  "no"   "una"  "su"   "al"   "lo"

stopwords("german")[1:20]

##  [1] "aber"    "alle"    "allem"   "allen"   "aller"   "alles"   "als"     "also"
##  [9] "am"      "an"      "ander"   "andere"  "anderem" "anderen" "anderer" "anderes"
## [17] "anderm"  "andern"  "anderr"  "anders"
```

Identifying words that can be removed as they are stopwords may use statistical methods, such as corpus dissimilarity, which we will introduce in the collocation detection lecture this week.
In quanteda you can remove features from a `tokenize`-object by applying the `removeFeatures(x, features)`, where `features` could be `stopwords("english")`.

Wordnet based densification

WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations.

- Need to separately install wordnet, in Mac can be done quickly using homebrew. `brew install wordnet`
- R-package called `wordnet`
- On loading, need to set path to wordnet dictionary installation.
- Available to browse on `http://wordnetweb.princeton.edu/perl/webwn?s=car`

Wordnet based densification

```
library(wordnet)
# set path to dictionary
setDict("/usr/local/Cellar/wordnet/3.1/dict")
synonyms("company", "NOUN")
```

```
## [1] "caller"         "companionship"  "company"        "fellowship"     "party"
## [6] "ship's company" "society"        "troupe"
```

Could list word list (running part of speech tagging first) and then replace synonyms of most frequently appearing words to reduce the vocabulary.

Minimum Edit Distances

A lot of NLP work consists of identifying which texts are similar to others. We will illustrate this later, when we turn to a *bag of words* language model that allows simple *vector based* comparisons of text.

We introduce the idea of computing string similarity introducing the idea of Edit Distance.

Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other. I will show an application from my research.

This is extremely useful when working with (messy) data - such as OCRed documents, where you need to get standardize and get rid of non-systematic typos.

Levenshtein Distance

- Levenshtein distance assumes a cost of deletion/ insertion of a character to be 1.
- Assumes a cost of substition of character of 1 (sometimes 2).
- So the Levenshtein distance between `car` and `can` is equal to 1.
- Unit cost allows express adjustments needed relative to string length.
- Levenshtein computation uses **dynamic programming** and is thus very fast.

  > *dynamic programming (also known as dynamic optimization) is a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems*

Minimum Edit Distances

Suppose you have two strings s and t of length n and m.
Below provides the algorithm

Step	Description
1	Set n to be the length of a.
	Set m to be the length of b.
	If $n = 0$, return m and exit.
	If $m = 0$, return n and exit.
	Construct a matrix containing $0..m$ rows and $0..n$ columns.
2	Initialize the first row to $0..n$.
	Initialize the first column to 0..m.
3	Examine each character of a (i from 1 to n).
4	Examine each character of b (j from 1 to m).
5	If $a[i]$ equals $b[j]$, the cost is 0.
	If $a[i]$ doesn't equal $b[j]$, the cost is 1.
6	Set cell d[i,j] of the matrix equal to the minimum of:
	a. The cell immediately above plus 1: $d[i-1, j] + 1$.
	b. The cell immediately to the left plus 1: $d[i, j-1] + 1$.
	c. The cell diagonally above and to the left plus the cost: $d[i-1, j-1] + cost$.
7	After the iteration steps (3, 4, 5, 6) are complete, the distance is found in cell $d[n, m]$.

Levenshtein Distance

Formally, Levenshtein Distance is computed as

$$\text{lev}_{a,b}(i,j) = min \begin{cases} \text{lev}_{a,b}(i-1,j)+1 \\ \text{lev}_{a,b}(i,j-1)+1 \\ \text{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases}$$

An Illustration

$$\text{lev}_{a,b}(i,j) = min \begin{cases} \text{lev}_{a,b}(i-1,j)+1 \\ \text{lev}_{a,b}(i,j-1)+1 \\ \text{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases}$$

Initialization Step 1 and 2

		e	x	e	c	u	t	i	o	n
	0	1	2	3	4	5	6	7	8	9
i	1									
n	2									
t	3									
e	4									
n	5									
t	6									
i	7									
o	8									
n	9									

An Illustration

Step 3 for each row , for each column ...

		e	x	e	c	u	t	i	o	n
	0	1	2	3	4	5	6	7	8	9
i	1	**1**	2	3	4	5	6	6	7	8
n	2									
t	3									
e	4									
n	5									
t	6									
i	7									
o	8									
n	9									

Converting "e" to "i": min of

- Converting "empty string" to "i", plus deletion (left cell)
- Converting "e" to "empty string", plus insertion (upper cell)
- Converting "empty" to "empty", plus substitution of "e" for "i"

An Illustration

Step 3 for each row , for each column ...

		e	x	e	c	u	t	i	o	n
	0	1	2	3	4	5	6	7	8	9
i	1	1	**2**	3	4	5	6	6	7	8
n	2									
t	3									
e	4									
n	5									
t	6									
i	7									
o	8									
n	9									

Converting "ex" to "i": min of

- Converting "e" to "i", plus deletion (left cell)
- converting "ex" to "empty string", plus insertion (upper cell)
- Converting "e" to "empty string", plus substitution of "x" for "i"

An Illustration

Step 3 for each row , for each column ...

		e	x	e	c	u	t	i	o	n
	0	1	2	3	4	5	6	7	8	9
i	1	**1**	2	3	4	5	6	6	7	8
n	2	2	**2**	3	4	5	6	7	7	7
t	3	3	3	**3**	4	5	5	6	7	8
e	4	3	4	3	**4**	5	6	6	7	8
n	5	4	4	4	4	**5**	6	7	7	7
t	6	5	5	5	5	5	**5**	6	7	8
i	7	6	6	6	6	6	6	**5**	6	7
o	8	7	7	7	7	7	7	6	**5**	6
n	9	8	8	8	8	8	8	7	6	**5**

`http://www.let.rug.nl/kleiweg/lev/`

With Substition Cost of 2

		e	x	e	c	u	t	i	o	n
	0	1	2	3	4	5	6	7	8	9
i	**1**	2	3	4	5	6	7	6	7	8
n	2	**3**	4	5	6	7	8	7	8	7
t	3	4	**5**	6	7	8	7	8	9	8
e	4	3	4	**5**	**6**	7	8	9	10	9
n	5	4	5	6	7	**8**	9	10	11	10
t	6	5	6	7	8	9	**8**	9	10	11
i	7	6	7	8	9	10	9	**8**	9	10
o	8	7	8	9	10	11	10	9	**8**	9
n	9	8	9	10	11	12	11	10	9	**8**

`http://www.let.rug.nl/kleiweg/lev/`

Finding near matches for messy data...

- Edit distance is a powerful tool to remove typos due to erroneous or bad quality scanned text data.
- A lot of social program data records are (still) paper based and need to be scanned in.
- Scanning errors are usually not linguistic in nature, but rather consist of character omissions.

Measuring Political Turnover: Raw CIA data

AFGHANISTAN

CHIEF OF STATE	ZAHIR SHAH MOHAMMED KING
PRIME MINISTER	DAUD KHAN MOHAMMED
FIRST DEPUTY PRIME MINISTER	MOHAMMED KHAN ALI
SECOND DEPUTY PRIME MINISTER	NAIM KHAN MOHAMMED
MIN OF AGRICULTURE	ADALAT GHOLAM HAIDER
MIN OF COMMERCE	SHERZAD GHOLAM MOHAMMED
MIN OF COMMUNICATIONS	MURID MOHAMMED
MIN OF COURT	ALI SULEIMAN AHMED
MIN OF DEFENSE	DAUD KHAN MOHAMMED
MIN OF EDUCATION	PUPAL ALI AHMED
MIN OF FINANCE	MALIKYAR ABD ALLAH
MIN OF FOREIGN AFFAIRS	NAIM KHAN MOHAMMED
MIN OF INTERIOR	ABDULLAH SEYYID ABD AL HALIM
MIN OF JUSTICE	ABDULLAH SEYYID ABD AL HALIM
MIN OF MINES & INDUSTRIES	YUSEF MIR MOHAMMED
MIN OF PLANNING	DAUD KHAN MOHAMMED
MIN OF PUBLIC HEALTH	DAUD KHAN MOHAMMED
MIN OF PUBLIC WELFARE	USMAN GHULAM FAROOQ KHAN
MIN OF PUBLIC WORKS	

1 13 JULY 1962

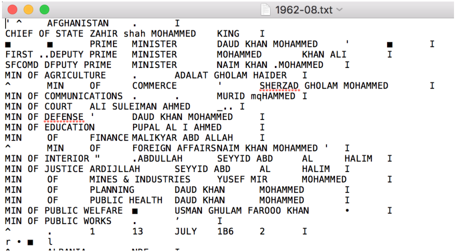
AFGHANISTAN

CHIEF OF STATE	ZAHIR SHAH, MUHAMMAD KING
PRIME MINISTER	DAUD, MUHAMMAD SARDAR
FIRST DEPUTY PRIME MINISTER	MUHAMMAD, ALI SARDAR
SECOND DEPUTY PRIME MINISTER	NAIM, MUHAMMAD SARDAR
MIN OF AGRICULTURE	ADALAT, GHOLAM HAIDER
MIN OF COMMERCE	SHERZAD, GHOLAM MUHAMMAD
MIN OF COMMUNICATIONS	MORID, MUHAMMAD
MIN OF COURT	ALI, SULEIMAN AHMAD SARDAR
MIN OF DEFENSE	DAUD, MUHAMMAD SARDAR
MIN OF EDUCATION	POPAL, ALI AHMAD
MIN OF FINANCE	MALIKYAR, ABDULLAH
MIN OF FOREIGN AFFAIRS	NAIM, MUHAMMAD SARDAR
MIN OF INTERIOR	ABDULLAH, SAYYED
MIN OF JUSTICE	ABDULLAH, SAYYED
MIN OF MINES & INDUSTRIES	YUSEF, MIR MUHAMMAD DR
MIN OF PLANNING	DAUD, MUHAMMAD SARDAR
MIN OF PUBLIC HEALTH	
MIN OF PUBLIC WORKS	
DEPT OF PRESS	SOHEIL, MUHAMMAD ASEF
DEPT OF TRIBAL AFFAIRS	MAJRUH, SAYYED SHAMSUDDIN

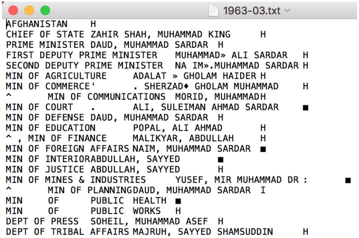
1963-03

Figure: CIA Reports Tracking Political Transitions

Measuring Political Turnover: Raw CIA data



```
' ^     AFGHANISTAN     .      I
CHIEF OF STATE ZAHIR shah MOHAMMED    KING    I
■         ■    PRIME   MINISTER       DAUD KHAN MOHAMMED     '      ■       I
FIRST ..DEPUTY PRIME   MINISTER       MOHAMMED         KHAN ALI     I
SFCOMD DFPUTY PRIME    MINISTER       NAIM KHAN .MOHAMMED     I
MIN OF AGRICULTURE     .     ADALAT GHOLAM HAIDER   I
^       MIN     OF     COMMERCE       '    SHERZAD GHOLAM MOHAMMED       I
MIN OF COMMUNICATIONS .        .      MURID mqHAMMED I
MIN OF COURT    ALI SULEIMAN AHMED      _.. I
MIN OF DEFENSE '        DAUD KHAN MOHAMMED     I
MIN OF EDUCATION        PUPAL AL I AHMED       I
MIN     OF     FINANCE MALIKYAR ABD ALLAH      I
^       MIN    OF      FOREIGN AFFAIRSNAIM KHAN MOHAMMED '   I
MIN OF INTERIOR "       .ABDULLAH      SEYYID ABD     AL      HALIM   I
MIN OF JUSTICE ARDIJLLAH       SEYYID ABD     AL      HALIM   I
MIN     OF     MINES & INDUSTRIES     YUSEF MIR       MOHAMMED       I
MIN     OF     PLANNING        DAUD KHAN       MOHAMMED       I
MIN     OF     PUBLIC HEALTH   DAUD KHAN       MOHAMMED       I
MIN OF PUBLIC WELFARE ■        USMAN GHULAM FAROOO KHAN       •      I
MIN OF PUBLIC WORKS    .       '      I
^       .      1       13      JULY    1B6     2       I
r • ■   l
```



```
AFGHANISTAN    H
CHIEF OF STATE ZAHIR SHAH, MUHAMMAD KING      H
PRIME MINISTER DAUD, MUHAMMAD SARDAR  H
FIRST DEPUTY PRIME MINISTER    MUHAMMAD» ALI SARDAR    H
SECOND DEPUTY PRIME MINISTER   NA IM».MUHAMMAD SARDAR H
MIN OF AGRICULTURE     ADALAT » GHOLAM HAIDER H
MIN OF COMMERCE'       . SHERZAD♦ GHOLAM MUHAMMAD     H
^       MIN OF COMMUNICATIONS  MORID, MUHAMMADH
MIN OF COURT    .      ALI, SULEIMAN AHMAD SARDAR     ■
MIN OF DEFENSE DAUD, MUHAMMAD SARDAR  H
MIN OF EDUCATION       POPAL, ALI AHMAD       H
^ , MIN OF FINANCE     MALIKYAR, ABDULLAH     H
MIN OF FOREIGN AFFAIRS NAIM, MUHAMMAD SARDAR ■
MIN OF INTERIORABDULLAH, SAYYED       ■
MIN OF JUSTICE ABDULLAH, SAYYED       H
MIN OF MINES & INDUSTRIES      YUSEF, MIR MUHAMMAD DR :       ■
^       MIN OF PLANNINGDAUD, MUHAMMAD SARDAR  I
MIN     OF     PUBLIC  HEALTH ■
MIN     OF     PUBLIC  WORKS   H
DEPT OF PRESS  SOHEIL, MUHAMMAD ASEF  H
DEPT OF TRIBAL AFFAIRS MAJRUH, SAYYED SHAMSUDDIN      H
```

Figure: CIA Reports Tracking Political Transitions

⇒ 50 years of monthly data, essentially covering all countries of the world. ⇒ 3 million rows of raw data, initially 342,540 unique rows. ⇒ Levenshtein based dimensionality reduction reduces this down to 199,028.

Measuring Political Turnover: Raw CIA data

It is evident that many strings are very very similar, and since typos are idiosyncratic to an individual document, we can take a frequentist approach.

```
library(RecordLinkage)
levenshteinDist("MIN OF EDUCATION\tPUPAL AL I AHMED\tI", "MIN OF EDUCATION\tPOPAL, ALI AHMAD H")

## [1] 6

levenshteinSim("MIN OF EDUCATION\tPUPAL AL I AHMED\tI", "MIN OF EDUCATION\tPOPAL, ALI AHMAD H")

## [1] 0.829

## run on whole vector
VEC <- c("CHIEF OF STATE\tZAHIR SHAH, MUHAMMAD KING\tH", "PRIME MINISTER\tDAUD, MUHAMMAD SARDAR\tH",
    "FIRST DEPUTY PRIME MINISTER\tMUHAMMAD ALI SARDAR\tH", "SECOND DEPUTY PRIME MINISTER\tNA IM.MUHAMMAD SARD
    "MIN OF AGRICULTURE\tADALAT  GHOLAM HAIDER\tH", "MIN OF COMMERCE\t'\t. SHERZAD GHOLAM MUHAMMAD\tH",
    "^\tMIN OF COMMUNICATIONS\tMORID, MUHAMMAD\tH", "MIN OF COURT\t.\tALI, SULEIMAN AHMAD SARDAR\t",
    "MIN OF DEFENSE\tDAUD, MUHAMMAD SARDAR\tH", "MIN OF EDUCATION\tPOPAL, ALI AHMAD\tH", "^ , MIN OF FINANCE
    "MIN OF FOREIGN AFFAIRS\tNAIM, MUHAMMAD SARDAR\t", "MIN OF INTERIOR\tABDULLAH, SAYYED\t",
    "MIN OF JUSTICE\tABDULLAH, SAYYED\tH", "MIN OF MINES & INDUSTRIES\tYUSEF, MIR MUHAMMAD DR\t:\t",
    "^\tMIN OF PLANNING\tDAUD, MUHAMMAD SARDAR\tI", "MIN\tOF\tPUBLIC\tHEALTH\t", "MIN\tOF\tPUBLIC\tWORKS\tH
    "DEPT OF PRESS\tSOHEIL, MUHAMMAD ASEF\tH", "DEPT OF TRIBAL AFFAIRS\tMAJRUH, SAYYED SHAMSUDDIN\tH")
SIM <- levenshteinSim("MIN OF EDUCATION\tPUPAL AL I AHMED\tI", VEC)

SIM

##  [1] 0.262 0.211 0.260 0.189 0.372 0.304 0.439 0.326 0.342 0.857 0.256 0.304 0.400 0.486
## [15] 0.327 0.341 0.314 0.286 0.270 0.280

VEC[which.max(SIM)]

## [1] "MIN OF EDUCATION\tPOPAL, ALI AHMAD\tH"
```

Clustering Based on Edit Distance

Clustering is a very useful machine learning application that typically requires distance objects. Working with text data often requires a disambiguation of alternative spelling varations and clustering can be a very useful tool.

OpenRefine

Developer(s)	Google, open source community
Initial release	November 10, 2010; 6 years ago
Stable release	2.5 / December 11, 2011; 5 years ago [1]
Repository	github.com /OpenRefine /OpenRefine

OpenRefine Clustering