Statistical Analysis of Corpus Data with R

You shall know a word by the company it keeps!

Collocation extraction with statistical association measures

— Part 2 —

Designed by Marco Baroni[1] and Stefan Evert[2]

[1]Center for Mind/Brain Sciences (CIMeC)
University of Trento

[2]Institute of Cognitive Science (IKW)
University of Osnabrück

Outline

Scaling up

- We know how to compute association scores (X^2, Fisher, and $\log\theta$) for individual contingency tables now …

  … but we want to do it automatically for 24,000 bigrams in the Brown data set, or an even larger number word pairs
- Of course, you can write a loop (if you know C/Java):

```
> attach(Brown)
> result <- numeric(nrow(Brown))
> for (i in 1:nrow(Brown)) {
    if ((i %% 100) == 0) cat(i, " bigrams done\n")
    A <- rbind(c(O11[i],O12[i]), c(O21[i],O22[i]))
    result[i] <- chisq.test(A)$statistic
  }
```

☞ `fisher.test()` is even slower …

Vectorising algorithms

- Standard iterative algorithms (loops, function calls) are excruciatingly slow in R
  - R is an interpreted language designed for interactive work and small scripts, not for implementing complex algorithms
- Large amounts of data can be processed efficiently with **vector** and **matrix** operations ➪ vectorisation
  - even computations involving millions of numbers are carried out instantaneously
- How do you store a vector of contingency tables?

☞ as vectors O_{11}, O_{12}, O_{21}, O_{22} in a data frame

Vectorising algorithms

- High-level functions like `chisq.test()` and `fisher.test()` cannot be applied to vectors
  - only accept a single contingency table
  - or vectors of cross-classifying factors from which a contingency table is built automatically
- Need to implement association measures ourselves
  - i.e. calculate a test statistic or effect-size estimate to be used as an association score

➪ have to take a closer look at the statistical theory

Observed and expected frequencies

	w_2	$\neg w_2$	
w_1	O_{11}	O_{12}	$= R_1$
$\neg w_1$	O_{21}	O_{22}	$= R_2$
	$= C_1$	$= C_2$	$= N$

	w_2	$\neg w_2$
w_1	$E_{11} = \frac{R_1 C_1}{N}$	$E_{12} = \frac{R_1 C_2}{N}$
$\neg w_1$	$E_{21} = \frac{R_2 C_1}{N}$	$E_{22} = \frac{R_2 C_2}{N}$

- R_1, R_2 are the **row sums** (R_1 = marginal frequency f_1)
- C_1, C_2 are the **column sums** (C_1 = marginal frequency f_2)
- N is the **sample size**
- E_{ij} are the **expected frequencies** under independence H_0

Adding marginals and expected frequencies in R

```
# first, keep R from performing integer arithmetic
> Brown <- transform(Brown,
  O11=as.numeric(O11), O12=as.numeric(O12),
  O21=as.numeric(O21), O22=as.numeric(O22))

> Brown <- transform(Brown,
  R1=O11+O12, R2=O21+O22,
  C1=O11+O21, C2=O12+O22,
  N=O11+O12+O21+O22)

# we could also have calculated them laboriously one by one:
Brown$R1 <- Brown$O11 + Brown$O12  # etc.

> Brown <- transform(Brown,
  E11=(R1*C1)/N, E12=(R1*C2)/N,
  E21=(R2*C1)/N, E22=(R2*C2)/N)
# now check that E11, ..., E22 always add up to N!
```

Statistical association measures

Measures of significance

- Statistical association measures can be calculated from the observed, expected and marginal frequencies
- E.g. the chi-squared statistic X^2 is given by

$$\textbf{chi-squared} = \sum_{ij} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

  (you can check this in any statistics textbook)
- The `chisq.test()` function uses a different version with Yates' continuity correction applied:

$$\textbf{chi-squared}_{\textbf{corr}} = \frac{N\big(|O_{11}O_{22} - O_{12}O_{21}| - N/2\big)^2}{R_1 R_2 C_1 C_2}$$

Statistical association measures

Measures of significance

- P-values for Fisher's exact test are rather tricky (and computationally expensive)
- Can use likelihood ratio test statistic G^2, which is less sensitive to small and skewed samples than X^2 (Dunning 1993, 1998; Evert 2004)
  - G^2 uses same scale (asymptotic χ^2_1 distribution) as X^2, but you will notice that scores are entirely different

$$\textbf{log-likelihood} = 2 \sum_{ij} O_{ij} \log \frac{O_{ij}}{E_{ij}}$$

Significance measures in R

Watch your numbers!

- $\log 0$ is undefined, so G^2 cannot be calculated if any of the observed frequencies O_{ij} are zero
  - Why are the expected frequencies E_{ij} unproblematic?
- For these terms, we can substitute $0 = 0 \cdot \log 0$

```
> Brown <- transform(Brown,
  logl = 2 * (
    ifelse(O11>0, O11*log(O11/E11), 0) +
    ifelse(O12>0, O12*log(O12/E12), 0) +
    ifelse(O21>0, O21*log(O21/E21), 0) +
    ifelse(O22>0, O22*log(O22/E22), 0)
  ))
# ifelse() is a vectorised if-conditional
```

Significance measures in R

```
# chi-squared statistic with Yates' correction
> Brown <- transform(Brown,
    chisq = N *
    (abs(O11*O22 - O12*O21) - N/2)^2 /
    (R1 * R2 * C1 * C2)
  )

# Compare this to the output of chisq.test() for some bigrams.
# What happens if you do not apply Yates' correction?

> Brown <- transform(Brown,
  logl = 2 * (
    O11*log(O11/E11) + O12*log(O12/E12) +
    O21*log(O21/E21) + O22*log(O22/E22)
  ))

> summary(Brown$logl)    # do you notice anything strange?
```

Effect-size measures

- Direct implementation allows a wide variety of effect size measures to be calculated
  - but only direct maximum-likelihood estimates, confidence intervals are too complex (and expensive)
- Mutual information and Dice coefficient give two different perspectives on collocativity:

$$\textbf{MI} = \log_2 \frac{O_{11}}{E_{11}} \qquad \textbf{Dice} = \frac{2O_{11}}{R_1 + C_1}$$

- Modified log odds ratio is a reasonably good estimator:

$$\textbf{odds-ratio} = \log \frac{(O_{11} + \frac{1}{2})(O_{22} + \frac{1}{2})}{(O_{12} + \frac{1}{2})(O_{21} + \frac{1}{2})}$$

Further reading

- There are many other association measures
  - Pecina (2005) lists 57 different measures

- Evert, S. (to appear). Corpora and collocations. In A. Lüdeling and M. Kytö (eds.), *Corpus Linguistics. An International Handbook*, article 57. Mouton de Gruyter, Berlin.
  - explains characteristic properties of the measures
  - contingency tables for textual and surface cooccurrences
- Evert, Stefan (2004). *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. Dissertation, Institut für maschinelle Sprachverarbeitung, University of Stuttgart. Published in 2005, URN urn:nbn:de:bsz:93-opus-23714.
  - full sampling models and detailed mathematical analysis
- Online repository: **`www.collocations.de/AM`**
  - with reference implementations in the UCS toolkit software

☞ all these sources use the notation introduced here

Implementiation of the effect-size measures

```
# Can you compute the association scores without peeking ahead?

> Brown <- transform(Brown,
  MI = log2(O11/E11),
  Dice = 2 * O11 / (R1 + C1),
  log.odds = log(
    ((O11 + .5) * (O22 + .5)) /
    ((O12 + .5) * (O21 + .5))
  ))

# check summary(Brown): are there any more NA's?
```

How to use association scores

- Goal: use association scores to identify "true" collocations

- **Strategy 1**: select word pairs with score above threshold
  - no theoretically motivated thresholds for effect size
  - significance thresholds not meaningful for collocations (How many bigrams are significant with $p < .001$?)
  - alternative: take $n = 100, 500, 1000, \ldots$ highest-scoring word pairs ➪ **n-best list** (empirical threshold)

- **Strategy 2**: rank word pairs by association score
  - reorder data frame by decreasing association scores
  - word pairs at the top are "more collocational"
  - corresponds to n-best lists of arbitrary sizes

Rankings in R

```
> sum(Brown$chisq > qchisq(.999,df=1))  # p < .001
> sum(Brown$logl > qchisq(.999,df=1))

> Brown <- transform(Brown,
  r.logl = rank(-logl),   # rank by decreasing score
  r.MI   = rank(-MI, ties="min"),  # see ?rank
  r.Dice = rank(-Dice, ties="min"))

> subset(Brown, r.logl <= 20,  # 20-best list for log-likelihood
  c(word1,word2,O11,logl,r.logl,r.MI,r.Dice))

# Now do the same for MI and Dice.  What are your observations?

# How many anti-collocations are there among the 100 most
# collocational bigrams according to log-likelihood?
```

Sorting data frames in R

```
> x <- 10 * sample(10)  # 10, 20, ..., 100 in random order

> sort(x)    # sorting a vector is easy (default: ascending)
> sort(x, decreasing=TRUE)

# But for sorting a data frame, we need an index vector that tell us
# in what order to rearrange the rows of the table.

> sort.idx <- order(x)    # also has decreasing option
> sort.idx
> x[sort.idx]
```

Sorting data frames in R: practice time

```
# try to sort bigram data set by log-likelihood measure

> sort.idx <- order(Brown$logl, decreasing=TRUE)
> Brown.logl <- Brown[sort.idx, ]

> Brown.logl[1:20, 1:6]

# Now construct a simple character vector with the first 100 bigrams,
# or show only relevant columns of the data frame for the first 100 rows.

# Show the first 100 noun-noun bigrams (pos code N) and
# the first 100 adjective-noun bigrams (codes J and N).

# If you know some programming, can you write a function that
# displays the first n bigrams for a selected association measure?
```

Sorting data frames in R: practice time

Example solutions for practice questions

```
> paste(Brown.logl$word1, Brown.logl$word2)[1:100]
> paste(Brown$word1, Brown$word2)[sort.idx[1:100]]

# advanced code ahead: make your life easy with some R knowledge
> show.nbest <- function(myData,
  AM=c("chisq","logl","MI","Dice","O11"), n=20) {
    AM <- match.arg(AM)  # allows unique abbreviations
    idx <- order(myData[[AM]], decreasing=TRUE)
    myData[idx[1:n], c("word1","word2","O11",AM)]
  }

> show.nbest(Brown, "chi")

# Can you construct a table that compares the measures side-by-side?
```

Evaluation of association measures

- One way to achieve a better understanding of different association measures is to evaluate and compare their performance in **multiword extraction** tasks
  - published studies include Daille (1994), Krenn (2000), Evert & Krenn (2001, 2005), Pearce (2002) and Pecina (2005)
- “Standard” multiword extraction approach
  - extract (syntactic) collocations from suitable text corpus
  - rank according to score of selected association measure
  - take *n*-best list as **multiword candidates**
  - additional filtering, e.g. by frequency threshold
  - candidates have to be validated manually by expert
- Evaluation based on manual validation
  - expert marks candidates as true (TP) or false (FP) positive
  - calculate **precision** of n-best list $= \#TP/n$
  - if all word pairs are annotated, also calculate **recall**

The PP-verb data set of Krenn (2000)

- Krenn (2000) used a data set of German PP-verb pairs to evaluate the performance of association measures
  - goal: identification of lexicalised German PP-verb combinations such as *zum Opfer fallen* (fall victim to), *ums Leben kommen* (lose one's life), *im Mittelpunkt stehen* (be the centre of attention), etc.
  - manual annotation distinguishes between support-verb constructions and figurative expressions (both are MWE)
  - candidate data for original study extracted from 8 million word fragment of German *Frankfurter Rundschau* corpus
- PP-verb data set used in this session
  - candidates extracted from full *Frankfurter Rundschau* corpus (40 million words, July 1992 – March 1993)
  - more sophisticated syntactic analysis used
  - frequency threshold $f \geq 30$ leaves 5102 candidates

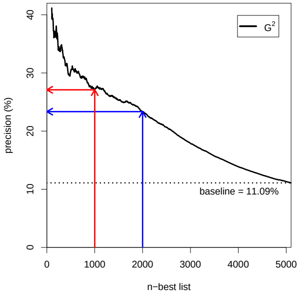
Table of n-best precision values

- Evaluation computes precision (and optionally) recall for various association measures and n-best lists

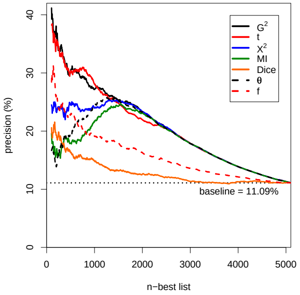
n-best	logl	chisq	t-score	MI	Dice	odds	freq
100	42.0	24.0	38.0	19.0	21.0	17.0	27.0
200	37.5	23.5	35.0	16.5	19.5	14.0	26.5
500	30.4	24.6	30.2	18.0	16.4	19.6	23.0
1,000	27.1	23.9	28.1	21.6	14.9	24.4	19.2
1,500	25.3	25.0	24.8	24.3	13.2	25.3	18.0
2,000	23.4	23.4	21.9	23.1	12.6	23.3	16.3

- More intuitive presentation for arbitrary n-best lists in the form of **precision graphs** (or precision-recall graphs)

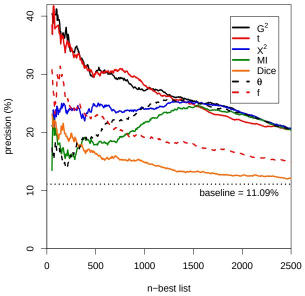
Precision graphs



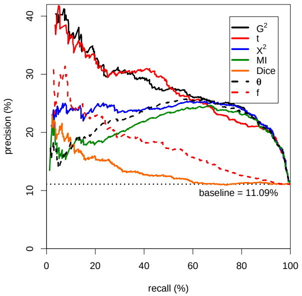
Precision graphs



Precision graphs: zooming in



Precision-by-recall graphs



The PP-verb data set

- `krenn_pp_verb.tbl` available from course homepage
- Data frame with 5102 rows and 14 columns:
  - **PP** = prepositional phrase (lemmatised)
  - **verb** = lexical verb (lemmatised)
  - **is.colloc** = Boolean variable indicating TPs (= MWE)
  - **is.SVC**, **is.figur** distinguish subtypes of MWE
  - **freq**, **MI**, **Dice**, **z.score**, **t.score**, **chisq**, **chisq.corr**, **log.like**, **Fisher** = precomputed association scores
    (Do you recognise all association measures?)
- Our goal is to reproduce the table and plots shown on the previous slides (perhaps not all the bells and whistles)

Precision tables: your turn!

```
> PPV <- read.delim("krenn_pp_verb.tbl")
> colnames(PPV)

> attach(PPV)

# You should now be able to sort the data set and calculate
# precision for some association measures and n-best lists.
# (hint: sum() counts TRUE entries in Boolean vector)
```

Precision tables

```
> idx.logl <- order(log.like, decreasing=TRUE)
> sum(is.colloc[idx.logl[1:500]]) / 500    # n = 500
> sum(is.colloc[idx.logl[1:1000]]) / 1000 # n = 1000

# use cumsum() to calculate precision for all n-best lists
> prec <- cumsum(is.colloc[idx.logl]) /
  (1:nrow(PPV))
> prec[c(100,200,500,1000,1500,2000)]
```

Precision tables: an elegant solution

```
> show.prec <- function(myData, AM, n) {
  stopifnot(AM %in% colnames(myData)) # safety first!
  sort.idx <- order(myData[[AM]], decreasing=TRUE)
  prec <- cumsum(myData$is.colloc[sort.idx]) /
          (1:nrow(myData))
  result <- data.frame(100 * prec[n]) # percentages
  rownames(result) <- n   # add nice row/column labels
  colnames(result) <- AM
  result   # return single-column data frame with precision values
  }

> show.prec(PPV, "chisq", c(100,200,500,1000))
```

Precision tables: an elegant solution

```
> n.list <- c(100,200,500,1000,1500,2000)

# data frames of same height can be combined in this way
> prec.table <- cbind(
    show.prec(PPV, "log.like", n.list),
    show.prec(PPV, "Fisher", n.list),
    show.prec(PPV, "chisq", n.list),
    show.prec(PPV, "chisq.corr", n.list),
    show.prec(PPV, "z.score", n.list),
    show.prec(PPV, "t.score", n.list),
    show.prec(PPV, "MI", n.list),
    show.prec(PPV, "Dice", n.list),
    show.prec(PPV, "freq", n.list)
  )

> round(prec.table, 1)  # rounded values are more readable
```

Precision graphs

```
# first, generate sort index for each association measure
> idx.ll <- order(log.like, decreasing=TRUE)
> idx.chisq <- order(chisq, decreasing=TRUE)
> idx.t <- order(t.score, decreasing=TRUE)
> idx.MI <- order(MI, decreasing=TRUE)
> idx.Dice <- order(Dice, decreasing=TRUE)
> idx.f <- order(freq, decreasing=TRUE)
```

Precision graphs

```
# second, calculate precision for all n-best lists
> n.vals <- 1:nrow(PPV)

> prec.ll <- cumsum(is.colloc[idx.ll]) *
  100 /  n.vals
> prec.chisq <- cumsum(is.colloc[idx.chisq]) *
  100 / n.vals
> prec.t <- cumsum(is.colloc[idx.t]) *
  100 / n.vals
> prec.MI <- cumsum(is.colloc[idx.MI]) *
  100 / n.vals
> prec.Dice <- cumsum(is.colloc[idx.Dice]) *
  100 / n.vals
> prec.f <- cumsum(is.colloc[idx.f]) *
  100 / n.vals
```

Precision graphs

```
# increase font size, set plot margins (measured in lines of text)
> par(cex=1.2, mar=c(4,4,1,1)+.1)

# third: plot as line, then add lines for further measures
> plot(n.vals, prec.ll, type="l",
  ylim=c(0,42), xaxs="i", # fit x-axis range tightly
  lwd=2, col="black",     # line width and colour
  xlab="n-best list", ylab="precision (%)")
> lines(n.vals, prec.chisq, lwd=2, col="blue")
> lines(n.vals, prec.t, lwd=2, col="red")
> lines(n.vals, prec.MI, lwd=2, col="black",
  lty="dashed") # line type: solid, dashed, dotted, ...
> lines(n.vals, prec.Dice, lwd=2,
  col="blue", lty="dashed")
> lines(n.vals, prec.f, lwd=2,
  col="red", lty="dashed")
```

Precision graphs

```
# add horizontal line for baseline precision
> abline(h = 100 * sum(is.colloc) / nrow(PPV))

# and legend with labels for the precision lines
> legend("topright", inset=.05, # easy positioning of box
  bg="white", # fill legend box so it may cover other graphics
  lwd=2,      # short vectors are recycled as necessary
  col=c("black", "blue", "red"),
  lty=c("solid","solid","solid", # no default values here!
        "dashed","dashed","dashed"),
  # either string vector, or "expression" for mathematical typesetting
  legend=expression(G^2, X^2, t, "MI", "Dice", f))
```

Precision graphs: playtime

- Add further decorations to plot (baseline text, arrows, ...)
- Write functions to simplify plot procedure
  - you may want to explore `type="n"` plots
- Precision values highly erratic for $n < 50$ ➪ don't show
- Graphs look smoother with thinning
  - increment n in steps of 5 or 10 (rather than 1)
- Calculate recall and create precision-by-recall graphs

☞ all those bells, whistles and frills are implemented in the UCS toolkit (`www.collocations.de/software.html`)