Unit 4: Collocation Extraction with Statistical Association Measures (Pt. 2)
Statistics for Linguists with R – A SIGIL Course

Designed by Marco Baroni and Stefan Evert

1 Center for Mind/Brain Sciences (CIMeC)
   University of Trento, Italy
2 Institute of Cognitive Science (IKW)
   University of Osnabrück, Germany

Copyright © 2007–2010 Baroni & Evert

Scaling up: working with large data sets

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

Examples:

```
Scaling up: working with large data sets

▶ 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 of 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

- 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

<table>
<thead>
<tr>
<th>$w_1$</th>
<th>$O_{11}$</th>
<th>$O_{12}$</th>
<th>$R_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O_{21}$</td>
<td>$O_{22}$</td>
<td>$R_2$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$w_1$</th>
<th>$E_{11} = \frac{R_1 C_1}{N}$</th>
<th>$E_{12} = \frac{R_1 C_2}{N}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_2$</td>
<td>$E_{21} = \frac{R_2 C_1}{N}$</td>
<td>$E_{22} = \frac{R_2 C_2}{N}$</td>
</tr>
</tbody>
</table>

- $R_1, R_2$ 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 $H_0$

Outline

Adding marginals and expected frequencies in R

```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 \( \chi^2 \) is given by

\[
\text{chi-squared} = \sum_{ij} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}
\]

(you can check this in any statistics textbook)

- The chisq.test() function uses a different version with Yates’ continuity correction applied:

\[
\text{chi-squared}_{\text{corr}} = \frac{N(|O_{11}O_{22} - O_{12}O_{21}| - N/2)^2}{R_1R_2C_1C_2}
\]

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 \( \chi^2 \) (Dunning 1993, 1998; Evert 2004)

\( G^2 \) uses same scale (asymptotic \( \chi^2 \) distribution) as \( \chi^2 \), but you will notice that scores are entirely different

\[
\text{log-likelihood} = 2 \sum_{ij} O_{ij} \log \frac{O_{ij}}{E_{ij}}
\]

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 \( 0 \cdot \log 0 \)

\[
\begin{align*}
\text{logl} & = 2 \cdot ( \text{ifelse}(O_{11} > 0, O_{11} \log(O_{11}/E_{11}), 0) + \\
& \quad \text{ifelse}(O_{12} > 0, O_{12} \log(O_{12}/E_{12}), 0) + \\
& \quad \text{ifelse}(O_{21} > 0, O_{21} \log(O_{21}/E_{21}), 0) + \\
& \quad \text{ifelse}(O_{22} > 0, O_{22} \log(O_{22}/E_{22}), 0) )
\end{align*}
\]

# ifelse() is a vectorised if-conditional
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:
  \[ \text{MI} = \log_2 \frac{O_{11}}{E_{11}} \quad \text{Dice} = \frac{2O_{11}}{R_1 + C_1} \]
- Modified log odds ratio is a reasonably good estimator:
  \[ \text{odds-ratio} = \log \left( \frac{(O_{11} + \frac{1}{2})(O_{22} + \frac{1}{2})}{(O_{12} + \frac{1}{2})(O_{21} + \frac{1}{2})} \right) \]

Further reading

- There are many other association measures
  - Pecina (2005) lists 57 different measures
  - explains characteristic properties of the measures
  - full sampling models and detailed mathematical analysis
- Online repository: [www.collocations.de/AM](http://www.collocations.de/AM)
  - with reference implementations in the UCS toolkit software

Implementation of the effect-size measures

```r
# 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?
```

Outline

- Statistical association measures
  - Scaling up: working with large data sets
  - Sorting and ranking data frames
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
  - 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

Sorting data frames in R

```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 tells us
# in what order to rearrange the rows of the table.
> sort.idx <- order(x)  # also has decreasing option
> sort.idx
> x[sort.idx]
```

Rankings in R

```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"))

> Brown <- transform(Brown,  # 20-best list for log-likelihood
  logl = rank(-logl, ties="min")

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

# How 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: practice time

```r
> x <- 10 * sample(10)  # 10, 20, …, 100 in random order
> sort.idx <- order(Brown$logl, decreasing=TRUE)
> Brown.logl <- Brown[sort.idx, ]

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

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

```r
> 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.

```r
> c(word1,word2,MI,logl,r.logl,r.MI,r.Dice)
```

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

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

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

# If you know some programming, can you write a function that
displays the first $n$ bigrams for a selected association measure?
Scaling up: working with large data sets

Sorting and ranking data frames

Sorting data frames in R: practice time

Example solutions for practice questions

```r
> 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

```r
> 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?

The evaluation of association measures

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


▶ “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

The PP-verb data set of Krenn (2000)
Table of n-best precision values

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

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

- More intuitive presentation for arbitrary n-best lists in the form of precision graphs (or precision-recall graphs)
The evaluation of association measures

Outline

Scaling up: working with large data sets
Statistical association measures
Sorting and ranking data frames

The evaluation of association measures
Precision/recall tables and graphs
MWE evaluation in R

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!

```r
> 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

```r
> 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

```r
> 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))
```

```r
> 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)
)
```

# remember the lapply / do.call trick from Unit 2?
```
> prec.list <- lapply(
  c("log.like", "Fisher", "chisq", "chisq.corr",
   "z.score", "t.score", "MI", "Dice", "freq"),
  function (AM) show.prec(PPV, AM, n.list)
)

> prec.table <- do.call(cbind, prec.list)

> round(prec.table, 1)  # rounded values are more readable
```
The evaluation of association measures

# 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)

# 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

# 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",
     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")
> lines(n.vals, prec.Dice, lwd=2, col="blue", lty="dashed")
> lines(n.vals, prec.f, lwd=2, col="red", lty="dashed")

# 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,
     bg="white",
     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"),
     legend=expression(G^2, X^2, t, "MI", "Dice", f))
### References I