Getting Started with DKPro Agreement

Christian M. Meyer, Margot Mieskes, Christian Stab and Iryna Gurevych: **DKPro Agreement: An Open-Source Java Library for Measuring Inter-Rater Agreement**, in: *Proceedings of the 25th International Conference on Computational Linguistics* (Coling), pp. 105–109, August 2014. Dublin, Ireland.
https://code.google.com/p/dkpro-statistics/

DKPro Agreement in a Nutshell

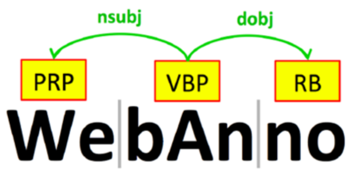
DKPro Agreement is an open-licensed Java library for computing inter-rater agreement using a shared interface and data model.

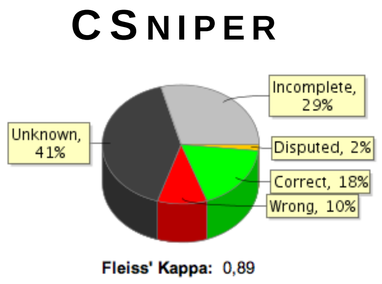
Highlights:

- Support for all commonly used inter-rater agreement measures
- Calculation of multiple coefficients using the same data model
- Both coding <u>and</u> unitizing setups are possible
- Multiple diagnostic devices and visual aids for analyzing disagreement
- Thoroughly tested on a wide range of examples from the literature
- Available as open source software under the Apache License 2.0 (ASL)
- Integrates well with existing Java-based NLP frameworks
- Ready-to-use via Maven Central
- Part of DKPro Statistics collection

Motivation

- Reliability is a necessary precondition of high quality datasets
- Long tradition of assessing inter-rater agreement in psychology, medicine, content analysis
- In NLP/CL often ignored or limited
- Researchers rely on manual calculations, hasty implementation, or insufficiently documented online calculators
- Measures are often not comparable
- Urgent need for software that
  - implements the most important measures
  - allows for diagnosing disagreement
  - integrates with existing projects and annotation workbenches (e.g., WebAnno, CSniper)





License and Availability

DKPro Agreement

http://code.google.com/p/dkpro-statistics/

The latest version of DKPro Agreement is available via Maven Central.

If you use Maven as your build tool, then you can add DKPro Agreement as a dependency in your `pom.xml` file:

```
<dependency>
   <groupId>de.tudarmstadt.ukp.dkpro.statistics</groupId>
   <artifactId>dkpro-statistics-agreement</artifactId>
   <version>2.0.0</version>
</dependency>
```

The software is available open source under the Apache License 2.0 (ASL). The software thus comes “as is” without any warranty (see license text for more details).

Step 0:

Understand the Data Model

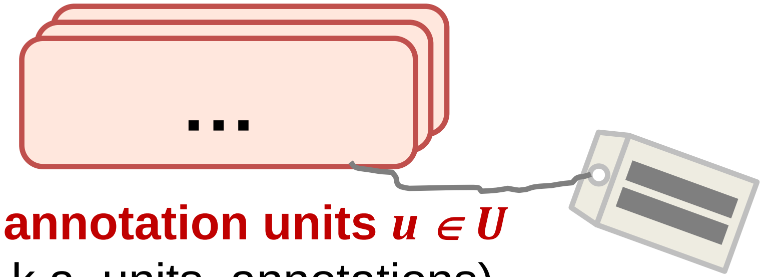
Terminology

Annotation study S:

Basic representation of an annotation experiment

Terminology

Annotation study S:



n_u annotation units $u \in U$
(a.k.a. units, annotations)

n_r raters $r \in R$ (a.k.a. coders, annotators, human observers)

- binary (yes, no)
- nominal (NN, VB, JJ,…)
- ordinal (1st, 2nd, 3rd,…)
- probabilistic (0.03, 0.49,…)
- …

n_c categories $c \in C$ (a.k.a. labels, codes, annotation types)

Annotation Units

Annotation study S:

n_u **annotation units** $u \in U$
(a.k.a. units, annotations)

An annotation unit is a specific part or segment of the input data, which has been coded by a certain rater $r \in R$ with one of the categories $c \in C$.

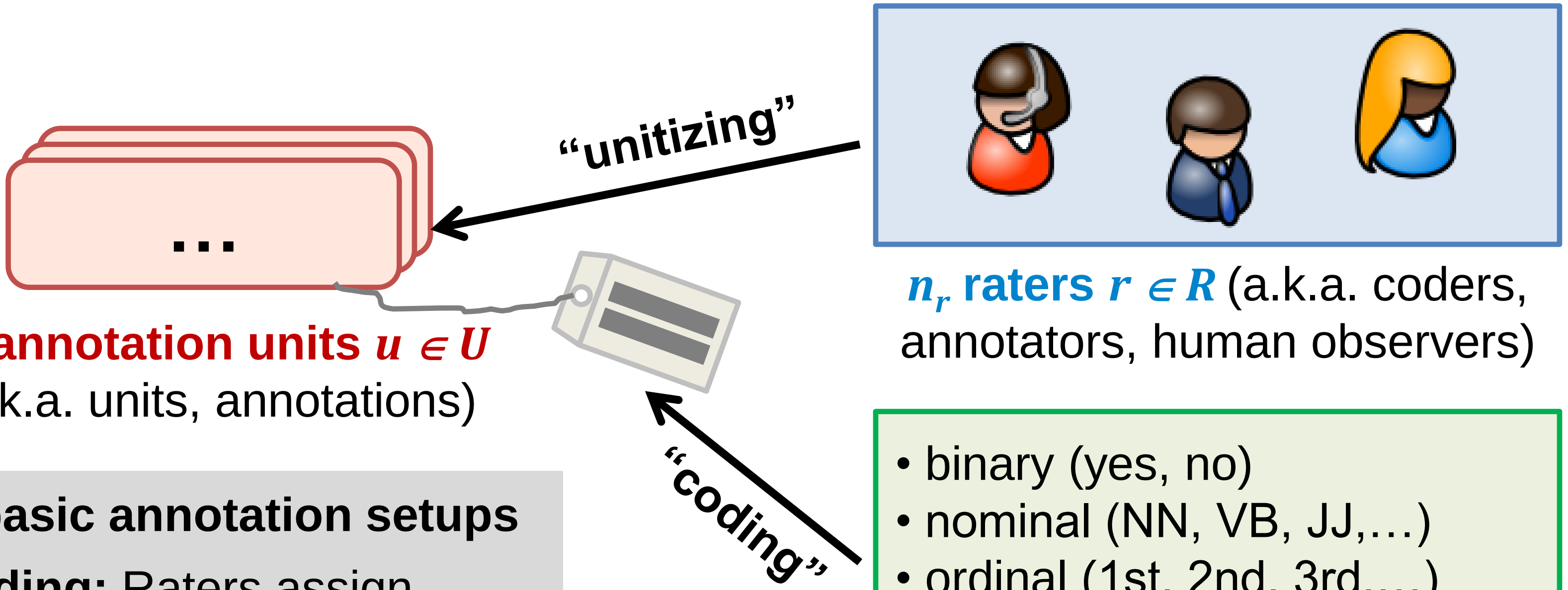
n_r **raters** $r \in R$ (a.k.a. coders, annotators, human observers)

- binary (yes, no)
- nominal (NN, VB, JJ,…)
- ordinal (1st, 2nd, 3rd,…)
- probabilistic (0.03, 0.49,…)
- …

n_c **categories** $c \in C$ (a.k.a. labels, codes, annotation types)

Annotation Setups

Annotation study S:



n_r **raters** $r \in R$ (a.k.a. coders, annotators, human observers)

n_u **annotation units** $u \in U$ (a.k.a. units, annotations)

- binary (yes, no)
- nominal (NN, VB, JJ,…)
- ordinal (1st, 2nd, 3rd,…)
- probabilistic (0.03, 0.49,…)
- …

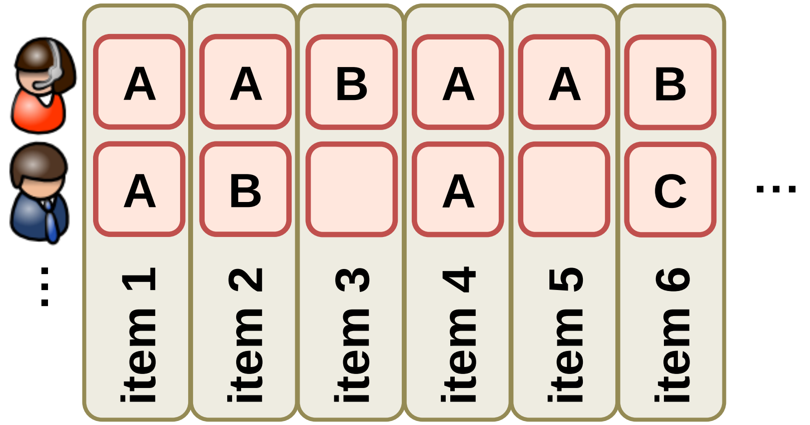
n_c **categories** $c \in C$ (a.k.a. labels, codes, annotation types)

Two basic annotation setups

- **Coding:** Raters assign categories to fixed items
- **Unitizing:** Raters segment data into codable units

Coding Setup

Annotation study S:



n_u annotation units $u \in U$
n_i annotation items $i \in I$

In a coding setup, the raters receive a set of annotation items $i \in I$ with fixed boundaries, which each of them should code ("annotate") with one of the categories $c \in C$.

$$n_i = n_u \cdot n_r$$

n_r raters $r \in R$ (a.k.a. coders, annotators, human observers)

- binary (yes, no)
- nominal (NN, VB, JJ,…)
- ordinal (1st, 2nd, 3rd,…)
- probabilistic (0.03, 0.49,…)
- …

n_c categories $c \in C$ (a.k.a. labels, codes, annotation types)

Coding Setup: Examples

Example 1: Classify newspaper articles by topic

- raters R = {Alice, Bob}, categories C = {politics, economics, feuilleton}
- items I = {article1, article2, article3}
- units U = {

items	article1	article2	article3
Alice	politics	politics	econ.
Bob	politics	econ.	

}

"missing value"

Example 2: Part-of-speech tagging

- raters R = {Claire, Dave, Estelle}, categories C = {NN, VB, JJ, RB}
- items I = {Colorless, green, ideas, sleep, furiously}
- units U = {

items	Colorless	green	ideas	sleep	furiously
Claire	JJ	JJ	NN	VB	RB
Dave	JJ	JJ	NN	VB	RB
Estelle	RB	JJ	NN	VB	RB

}

Coding Setup: Examples

Example 3: medical diagnosis (Fleiss, 1971)

- raters R = six psychiatrists
- categories C = {depression, personality disorder, schizophrenia, neurosis, other}
- items I = 30 patients, units U = see table 1 →

Example 4: Dialog act tagging

(Artstein&Poesio, 2008)

- raters R = 2 students (rater A and B)
- categories C = {statement, info-request}
- items I = 100 utterances
- units U

units U		rater A		
		Stat	IReq	∑
rater B	Stat	20	20	40
	IReq	10	50	60
	∑	30	70	100

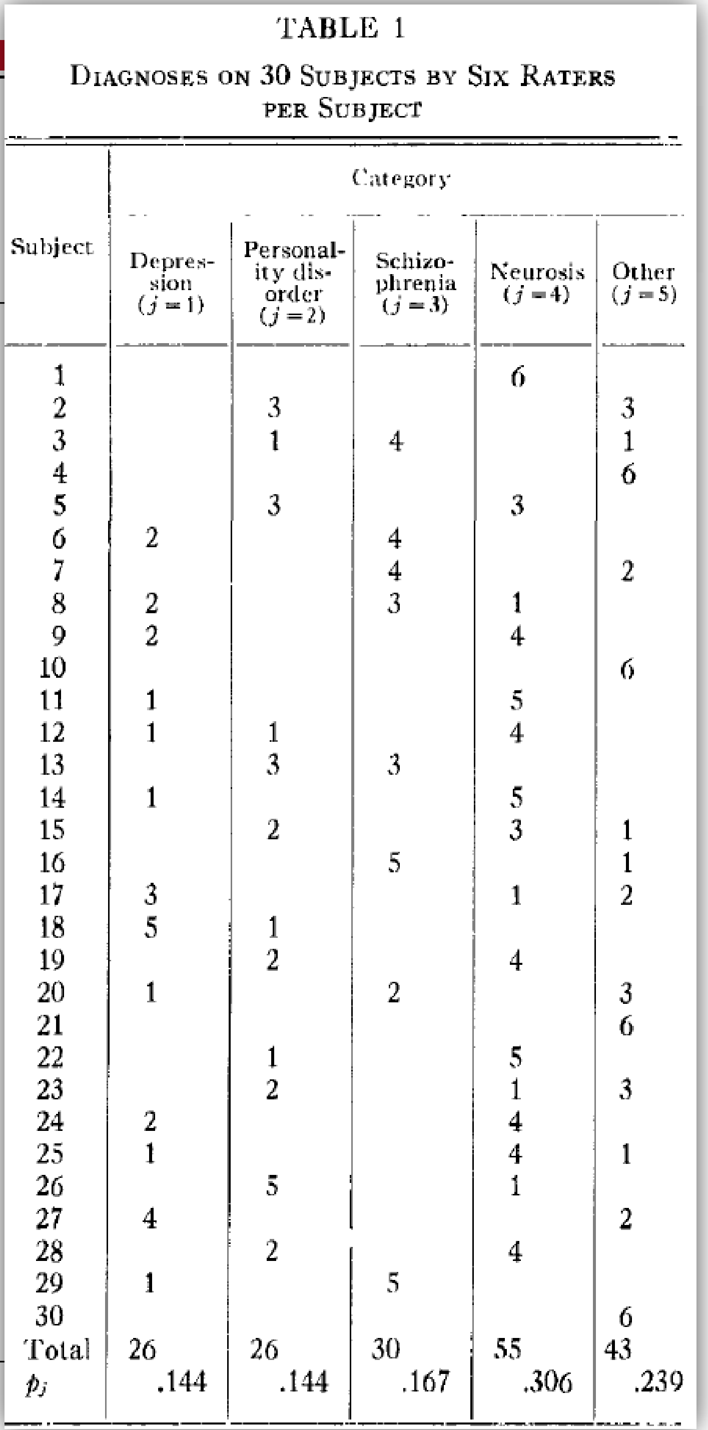


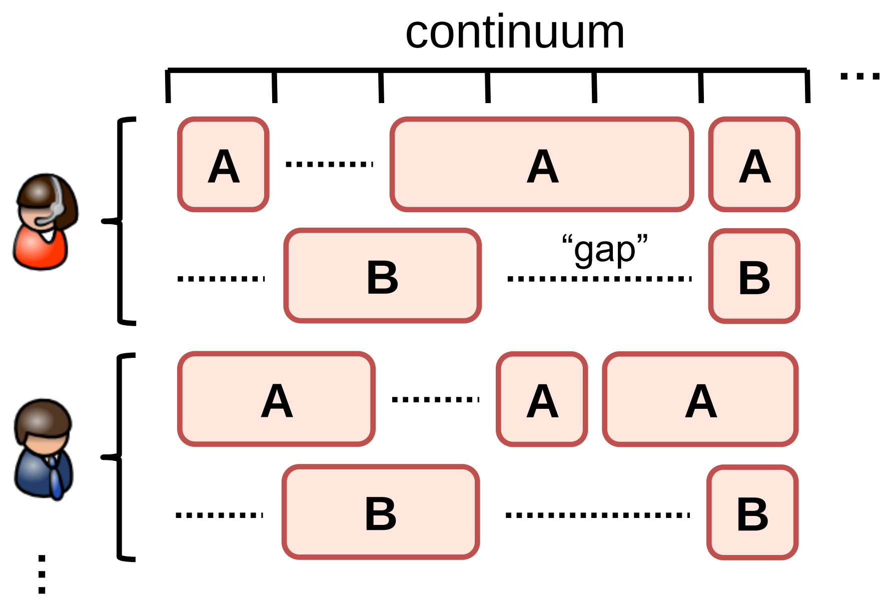
TABLE 1

DIAGNOSES ON 30 SUBJECTS BY SIX RATERS PER SUBJECT

Subject	Category				
	Depression ($j=1$)	Personality disorder ($j=2$)	Schizophrenia ($j=3$)	Neurosis ($j=4$)	Other ($j=5$)
1				6	
2		3			3
3		1	4		1
4					6
5		3		3	
6	2		4		
7			4		2
8	2		3	1	
9	2			4	
10					6
11	1			5	
12	1	1		4	
13		3	3		
14	1			5	
15		2		3	1
16			5		1
17	3			1	2
18	5	1			
19		2		4	
20	1		2		3
21					6
22		1		5	
23		2		1	3
24	2			4	
25	1			4	1
26		5		1	
27	4				2
28		2		4	
29	1		5		
30					6
Total	26	26	30	55	43
p_j	.144	.144	.167	.306	.239

Unitizing Setup

Annotation study S:



n_u **annotation units** $u \in U$

In unitizing studies, the raters are asked to identify the annotation units $u \in U$ themselves by marking their boundaries.

n_r **raters** $r \in R$ (a.k.a. coders, annotators, human observers)

- binary (yes, no)
- nominal (NN, VB, JJ,…)
- ordinal (1st, 2nd, 3rd,…)
- probabilistic (0.03, 0.49,…)
- …

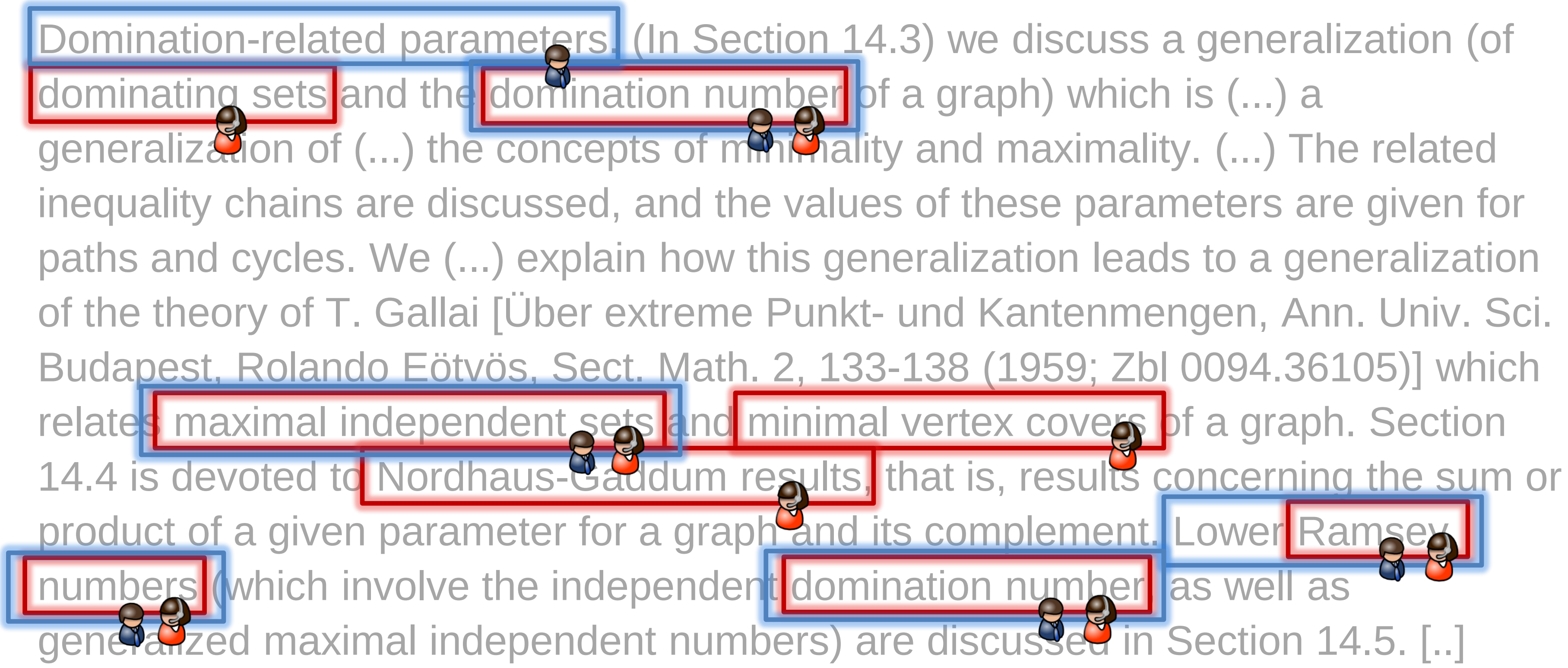
n_c **categories** $c \in C$ (a.k.a. labels, codes, annotation types)

Unitizing Setup: Examples

Example 1: Keyphrase identification

raters $\boldsymbol{R}$ = { 👩, 👨 }, categories $\boldsymbol{C}$ = {keyphrase}

units $\boldsymbol{U}$:



Domination-related parameters. (In Section 14.3) we discuss a generalization (of dominating sets and the domination number of a graph) which is (...) a generalization of (...) the concepts of minimality and maximality. (...) The related inequality chains are discussed, and the values of these parameters are given for paths and cycles. We (...) explain how this generalization leads to a generalization of the theory of T. Gallai [Über extreme Punkt- und Kantenmengen, Ann. Univ. Sci. Budapest, Rolando Eötvös, Sect. Math. 2, 133-138 (1959; Zbl 0094.36105)] which relates maximal independent sets and minimal vertex covers of a graph. Section 14.4 is devoted to Nordhaus-Gaddum results, that is, results concerning the sum or product of a given parameter for a graph and its complement. Lower Ramsey numbers (which involve the independent domination number as well as generalized maximal independent numbers) are discussed in Section 14.5. [..]

Unitizing Setup: Examples

Example 2: Krippendorff (2004)

raters ***R*** = { ***i, j*** }, categories ***C*** = { ***c, k*** }

units ***U*** :

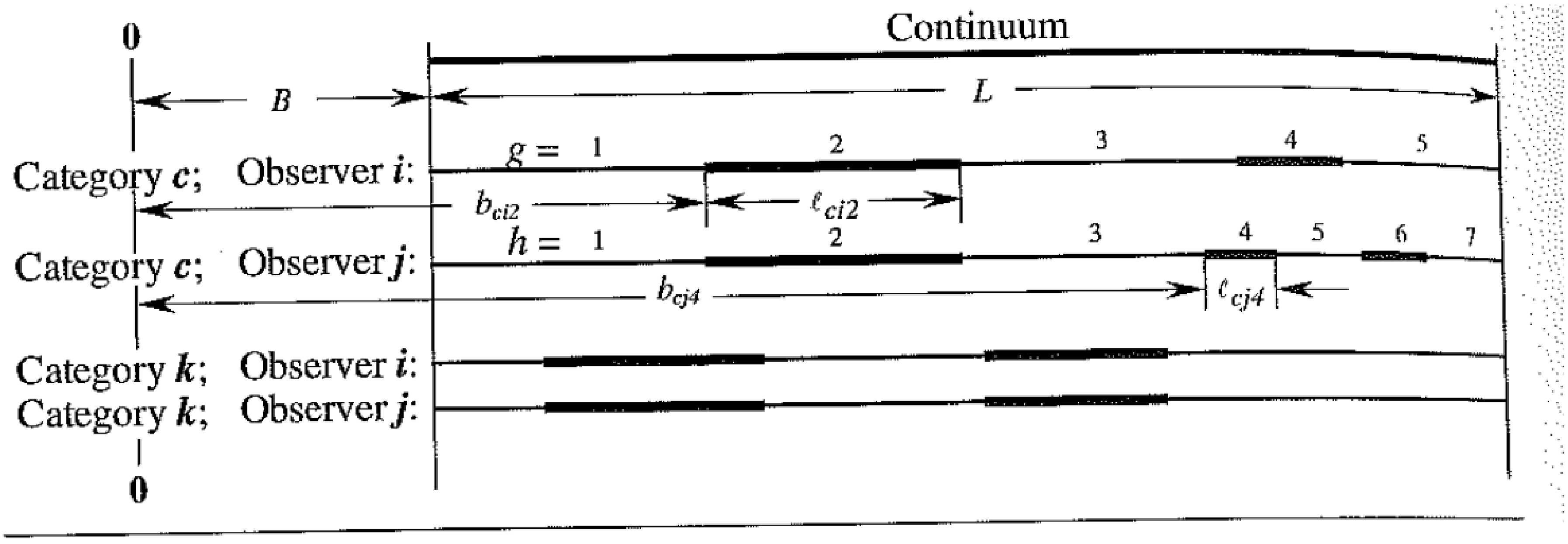


Figure 11.5 Unitizing Terms

Step 1:

Represent the Annotated Data

Create the Annotation Study

Depending on your annotation setup, instanciate the corresponding annotation study

For coding setups:

```
CodingAnnotationStudy study =
   new CodingAnnotationStudy(<rater-count>);
```

For unitizing setups:

```
UnitizingAnnotationStudy study =
   new UnitizingAnnotationStudy(<rater-count>,
   <continuum-offset>, <continuum-length>);
```

Define the Annotations

(1) Manually define your data in the source code.
Particularly suitable for small studies or tests.

```
study.addItem(Object… <annotations>)
```

Code Example:

```
  study.addItem("A", "A", "B", "A");
  study.addItem("B", "B", "B", "B");
  study.addItem("B", "C", null, "B");
```

```
study.addUnit(<offset>, <length>, <rater>, <category>)
```

Code Example:

```
  study.addUnit(10, 4, 2, "A");
  study.addUnit(20, 1, 1, "B");
  study.addUnit(20, 3, 2, "B");
```

Define the Annotations

(2) Load the annotation data from flat-files or from a database.

Code Example:

```
CodingAnnotationStudy study = new CodingAnnotationStudy(3);
BufferedReader reader = new BufferedReader(
   new FileReader("flatfile.tsv"));
String line;
while ((line = reader.readLine()) != null) {
  study.addItemAsArray(line.split("\t"));
}
reader.close();
```

Define the Annotations

(3) Use UIMA annotations (or a similar data format from your framework).

Code Example:

```
UnitizingAnnotationStudy study =
    new UnitizingAnnotationStudy(2,
    jcas.getDocumentText().length());
for (Annotation a : JCasUtil.select(jcas, Annotation.class))
{
  study.addUnit(a.getBegin(), a.getEnd() - a.getBegin(),
      a.getRaterIdx(), true);
}
```

(4) Reuse your own data model by implementing available interfaces.

Choosing Category Types

Categories can be of arbitrary types:

- Basic types
  - Integer
  - Double
  - String
  - Enum
  - …
- Complex types
  - Sets of annotations
  - User-defined types
- Missing values and gaps are represented by `null`

Step 2:

Measure the Inter-Rater Agreement

Available Coefficients

Measure	Type	Raters	Chance-corr.	Weighted
Percentage agreement p	coding	≥ 2	–	–
Bennett et al.'s S (1954)	coding	2	uniform	–
Scott's π (1955)	coding	2	study-specific	–
Cohen's κ (1960)	coding	2	rater-specific	–
Randolph's κ (2005) [multi-S]	coding	≥ 2	uniform	–
Fleiss's κ (1971) [multi-π]	coding	≥ 2	study-specific	–
Hubert's κ (1977) [multi-κ]	coding	≥ 2	rater-specific	–
Krippendorff's α (1980)	coding	≥ 2	study-specific	✓
Cohen's weighted κ_w (1968)	coding	≥ 2	rater-specific	✓
Krippendorff's α_U (1995)	unitizing	≥ 2	study-specific	–

Artstein&Poesio (2008) and Krippendorff (2004) explain these measures.

Compute the Inter-rater Agreement

```
PercentageAgreement pa = new PercentageAgreement(study);
System.out.println(pa.calculateAgreement());

FleissKappaAgreement kappa = new FleissKappaAgreement(study);
System.out.println(kappa.calculateAgreement());

KrippendorffAlphaAgreement alpha =
   new KrippendorffAlphaAgreement(study,
   new NominalDistanceFunction());
System.out.println(alpha.calculateObservedDisagreement());
System.out.println(alpha.calculateExpectedDisagreement());
System.out.println(alpha.calculateAgreement());
```

Step 3:

Analyze the Disagreement

Analyze the Disagreement

Raw agreement scores are of limited help for diagnosing the main sources of disagreement. DKPro Agreement provides multiple diagnostic devices.

Agreement insights:

- Observed agreement
- Expected agreement
- Rater-specific agreement
- Category-specific agreement
- Item-specific agreement

Formatted output and visual aids:

- Coincidence matrix
- Contingency matrix
- Reliability matrix
- Continuum of a unitizing study
- Planned: Hinton diagrams

Analyze the Disagreement

Example: Reliability matrix and category-specific agreement

		items						
		1	**2**	**3**	**4**	**5**	**6**	∑
raters	👩	A	A	B	A	A	B	
	👨	A	B		A		C	
categories	A	2	1		2	1		**6**
	B		1	1			1	**3**
	C						1	**1**

$p = 0.50$
$\kappa = 0.08$
$\alpha = 0.18$

$\alpha(A) = 0.39$
$\alpha(B) = -0.22$
$\alpha(C) = 0.00$

Join the Community!

DKPro Statistics

DKPro Agreement

http://code.google.com/p/dkpro-statistics/

Announcements and discussion:

http://groups.google.com/group/dkpro-statistics-users

Download and issue tracker:

https://code.google.com/p/dkpro-statistics/

Project background:

https://www.ukp.tu-darmstadt.de/software/dkpro-statistics/