Automated Verbatim Coding: State of the Art and Future Perspectives

Fabrizio Sebastiani
http://www.isti.cnr.it/People/F.Sebastiani/

Istituto di Scienza e Tecnologie dell’Informazione
Consiglio Nazionale delle Ricerche
Via Giuseppe Moruzzi, 1 – 56124 Pisa, Italy
E-mail: fabrizio.sebastiani@isti.cnr.it

2007 Language Logic Verbatim Management Conference
Cincinnati, OH – October 3-4, 2007
Coding verbatim responses is a bit like doing the dishes after hosting a dinner party: a somewhat tedious and time-consuming experience (...). At least, that was the case before dishwashers became commonplace.

[Tim Macer, Quirk’s Marketing Research Review, 16(7), 2002.]
Outline

1. Introduction
2. VCS: an automated Verbatim Coding System
3. VCS: Effectiveness Tests
   - Effectiveness at the individual level
   - Effectiveness at the aggregate level
4. VCS: Efficiency Tests
5. The future
1 Introduction

2 VCS: an automated Verbatim Coding System

3 VCS: Effectiveness Tests
   - Effectiveness at the individual level
   - Effectiveness at the aggregate level

4 VCS: Efficiency Tests

5 The future
About myself ...

- A senior research scientist at ISTI-CNR, and a former professor of the Department for Pure and Applied Mathematics of the University of Padova, Italy, ...
- ... and the leader of the Automatic Verbatim Coding Project at ISTI-CNR;

- CNR: the Italian National Council of Research, i.e., the largest government-owned research institution in Italy, devoted to basic and applied research in all fields of science;

- ISTI-CNR: the Institute for the Science and Technologies of Information of CNR, the largest (and oldest) Italian institute for research in computer science.
A senior research scientist at ISTI-CNR, and a former professor of the Department for Pure and Applied Mathematics of the University of Padova, Italy, ...

... and the leader of the **Automatic Verbatim Coding Project** at ISTI-CNR;

**CNR**: the Italian National Council of Research, i.e., the largest government-owned research institution in Italy, devoted to basic and applied research in all fields of science;

**ISTI-CNR**: the Institute for the Science and Technologies of Information of CNR, the largest (and oldest) Italian institute for research in computer science.
A senior research scientist at ISTI-CNR, and a former professor of the Department for Pure and Applied Mathematics of the University of Padova, Italy, ...

... and the leader of the Automatic Verbatim Coding Project at ISTI-CNR;

CNR: the Italian National Council of Research, i.e., the largest government-owned research institution in Italy, devoted to basic and applied research in all fields of science;

ISTI-CNR: the Institute for the Science and Technologies of Information of CNR, the largest (and oldest) Italian institute for research in computer science.
About myself ...

- A senior research scientist at ISTI-CNR, and a former professor of the Department for Pure and Applied Mathematics of the University of Padova, Italy, ...

- ... and the leader of the Automatic Verbatim Coding Project at ISTI-CNR;

- **CNR**: the Italian National Council of Research, i.e., the largest government-owned research institution in Italy, devoted to basic and applied research in all fields of science;

- **ISTI-CNR**: the Institute for the Science and Technologies of Information of CNR, the largest (and oldest) Italian institute for research in computer science.
Since 1998: “information retrieval”, “automatic learning”, “computational linguistics”, with an emphasis to applications in automatic text coding (e.g., coding newswire reports, coding medical discharge reports, coding patent applications, etc.)

Since 2003, I have also worked in (automatically) coding verbatim text returned to open-ended questions (e.g., from social surveys, or from customer satisfaction surveys)

Since 2005 I have also worked on opinion mining, i.e., automatically analyzing text with a special eye to the opinions expressed therein.
Since 1998: “information retrieval”, “automatic learning”, “computational linguistics”, with an emphasis to applications in automatic text coding (e.g., coding newswire reports, coding medical discharge reports, coding patent applications, etc.)

Since 2003, I have also worked in (automatically) coding verbatim text returned to open-ended questions (e.g., from social surveys, or from customer satisfaction surveys)

Since 2005 I have also worked on opinion mining, i.e., automatically analyzing text with a special eye to the opinions expressed therein.
... and my research interests

- Since 1998: “information retrieval”, “automatic learning”, “computational linguistics”, with an emphasis to applications in automatic text coding (e.g., coding newswire reports, coding medical discharge reports, coding patent applications, etc.)
- Since 2003, I have also worked in (automatically) coding verbatim text returned to open-ended questions (e.g., from social surveys, or from customer satisfaction surveys)
- Since 2005 I have also worked on opinion mining, i.e., automatically analyzing text with a special eye to the opinions expressed therein.
Introduction

VCS: an automated Verbatim Coding System

VCS: Effectiveness Tests
- Effectiveness at the individual level
- Effectiveness at the aggregate level

VCS: Efficiency Tests

The future
At ISTI-CNR we have recently developed an automated Verbatim Coding System (VCS), described in the paper


which has met with considerable success ...
At ISTI-CNR we have recently developed an automated Verbatim Coding System (VCS), described in the paper


which has met with considerable success ...

- Won the “2006 Amerigo Vespucci Award” (by the Italian Industrialists Association) for Market Research;
- Shortlisted for “Best Paper Award”, MRS’07 Conference;
- Shortlisted for “Best New Thinking”, MRS’07 Conference;
- Shortlisted for the 2007 Technology Effectiveness Award of the Association for Survey Computing.
At ISTI-CNR we have recently developed an automated Verbatim Coding System (VCS), described in the paper


which has met with considerable success ... 

- Won the “2006 Amerigo Vespucci Award” (by the Italian Industrialists Association) for Market Research;
- Shortlisted for “Best Paper Award”, MRS’07 Conference;
- Shortlisted for “Best New Thinking”, MRS’07 Conference;
- Shortlisted for the 2007 Technology Effectiveness Award of the Association for Survey Computing.
At ISTI-CNR we have recently developed an automated Verbatim Coding System (VCS), described in the paper


which has met with considerable success ...

- Won the “2006 Amerigo Vespucci Award” (by the Italian Industrialists Association) for Market Research;
- Shortlisted for “Best Paper Award”, MRS’07 Conference;
- Shortlisted for “Best New Thinking”, MRS’07 Conference;
- Shortlisted for the 2007 Technology Effectiveness Award of the Association for Survey Computing.
Originally commissioned by Egg plc, the largest purely online bank in the world (now part of Citigroup);

Developed in collaboration with Archimede Informatica, a sw company in Pisa, Italy;

Deployed in July 2006, now fully operational and managing all of Egg’s customer satisfaction verbatim data (≈ 20,000 questionnaires per month, plus huge backlogs).
VCS: the underlying philosophy

- VCS is an adaptive system for automatically coding verbatim responses under any user-specified codeframe (aka “codebook”); given such a codeframe, VCS automatically generates an automatic coding system for this codeframe.

- Actually, the basic unit along which VCS works is the code: given a codeframe consisting of several codes, for each such code VCS automatically generates a binary classifier, i.e., a system that decides whether a given verbatim should or should not be attributed the code.
VCS: the underlying philosophy

VCS is an adaptive system for automatically coding verbatim responses under any user-specified codeframe (aka “codebook”); given such a codeframe, VCS automatically generates an automatic coding system for this codeframe.

Actually, the basic unit along which VCS works is the code: given a codeframe consisting of several codes, for each such code VCS automatically generates a binary classifier, i.e., a system that decides whether a given verbatim should or should not be attributed the code.
VCS: the underlying philosophy (cont’d)

- VCS is based on a learning metaphor: the system learns from manually coded data the characteristics a new verbatim should have in order to be attributed the code; the manually coded data need to include positive examples of the code and negative examples of the code;

- Providing manually coded examples of the code to the system is by no means different than providing a child with (positive and negative) examples of, say, what a tiger is, in order to teach him to recognize tigers.
VCS: the underlying philosophy (cont’d)

- VCS is based on a learning metaphor: the system learns from manually coded data the characteristics a new verbatim should have in order to be attributed the code; the manually coded data need to include positive examples of the code and negative examples of the code;

- Providing manually coded examples of the code to the system is by no means different than providing a child with (positive and negative) examples of, say, what a tiger is, in order to teach him to recognize tigers.
Introduction
VCS: an automated Verbatim Coding System
VCS: Effectiveness Tests
VCS: Efficiency Tests
The future

This is a tiger!
This is another tiger!
Introduction

VCS: an automated Verbatim Coding System
VCS: Effectiveness Tests
VCS: Efficiency Tests
The future

This is yet another tiger!
Also a tiger!
This is a NOT a tiger!
NOT a tiger either!
Introduction
VCS: an automated Verbatim Coding System
VCS: Effectiveness Tests
VCS: Efficiency Tests
The future

Is this a tiger?
The VCS information flow

Introduction

VCS: an automated Verbatim Coding System
VCS: Effectiveness Tests
VCS: Efficiency Tests
The future

The VCS information flow diagram:

- Trainer
  - Training verbatims
  - Uncoded verbatims
  - Coding engine
    - Coded verbatims
    - Reports
      - "Card has good features/benefits"
      - interest rate decreased

Validation process:

- Verbatim Coders
  - Validated verbatims
  - Human coder
    - Alerts & Customer feedback
Advantages of learning metaphor

- No need for expert to write coding rules in arcane language; the system only needs user-coded examples for training;
- Easy update to shifted meaning of existing code, revised codeframe, brand new codeframe or brand new survey since the system only needs user-coded examples for training that reflect the new situation;
- Does not use any specialized resource (e.g., thesauri);
- Pretty good effectiveness at the “individual level”, excellent effectiveness at the “aggregate level”, excellent learning and coding speed.
Advantages of learning metaphor

- No need for expert to write coding rules in arcane language; the system only needs user-coded examples for training;

- Easy update to
  - shifted meaning of existing code
  - revised codeframe
  - brand new codeframe or brand new survey

since the system only needs user-coded examples for training that reflect the new situation;

- Does not use any specialized resource (e.g., thesauri);

- Pretty good effectiveness at the “individual level”, excellent effectiveness at the “aggregate level”, excellent learning and coding speed.
Advantages of learning metaphor

- No need for expert to write coding rules in arcane language; the system only needs user-coded examples for training;
- Easy update to
  - shifted meaning of existing code
  - revised codeframe
  - brand new codeframe or brand new survey
  since the system only needs user-coded examples for training that reflect the new situation;
- Does not use any specialized resource (e.g., thesauri);
- Pretty good effectiveness at the “individual level”, excellent effectiveness at the “aggregate level”, excellent learning and coding speed.
Advantages of learning metaphor

- No need for expert to write coding rules in arcane language; the system only needs user-coded examples for training;
- Easy update to
  - shifted meaning of existing code
  - revised codeframe
  - brand new codeframe or brand new survey

  since the system only needs user-coded examples for training that reflect the new situation;
- Does not use any specialized resource (e.g., thesauri);
- Pretty good effectiveness at the “individual level”, excellent effectiveness at the “aggregate level”, excellent learning and coding speed.
Advantages of learning metaphor

- No need for expert to write coding rules in arcane language; the system only needs user-coded examples for training;
- Easy update to
  - shifted meaning of existing code
  - revised codeframe
  - brand new codeframe or brand new survey
  since the system only needs user-coded examples for training that reflect the new situation;
- Does not use any specialized resource (e.g., thesauri);
- Pretty good effectiveness at the “individual level”, excellent effectiveness at the “aggregate level”, excellent learning and coding speed.
Advantages of learning metaphor

- No need for expert to write coding rules in arcane language; the system only needs user-coded examples for training;
- Easy update to
  - shifted meaning of existing code
  - revised codeframe
  - brand new codeframe or brand new survey
  since the system only needs user-coded examples for training that reflect the new situation;
- Does not use any specialized resource (e.g., thesauri);
- Pretty good effectiveness at the “individual level”, excellent effectiveness at the “aggregate level”, excellent learning and coding speed.
Outline

1. Introduction

2. VCS: an automated Verbatim Coding System

3. VCS: Effectiveness Tests
   - Effectiveness at the individual level
   - Effectiveness at the aggregate level

4. VCS: Efficiency Tests

5. The future

Fabrizio Sebastiani
Automated Verbatim Coding: State of the Art and Future Perspectives
Testing Effectiveness

- By **accuracy** (or **effectiveness**) of a coding system we refer to the frequency with which the coding decisions of the system are expected to agree with the coding decisions that an expert coder (the “gold standard”) would make.

- We estimate the effectiveness of a coding system by comparing the system’s coding decisions with those of an expert coder on one or more test **datasets** (each consisting of a set of manually coded verbatims plus the corresponding codeframe).
Testing Effectiveness

- By **accuracy** (or **effectiveness**) of a coding system we refer to the frequency with which the coding decisions of the system are expected to agree with the coding decisions that an expert coder (the “gold standard”) would make.

- We estimate the effectiveness of a coding system by comparing the system’s coding decisions with those of an expert coder on one or more test **datasets** (each consisting of a set of manually coded verbatims plus the corresponding codeframe).
Effectiveness: individual or aggregate?

- Effectiveness may be measured at two different levels:
  - At the **individual level**: the perfect system is the one which, for a code $C$, assigns $C$ to the verbatim exactly when the expert coder would have assigned $C$.
  - At the **aggregate level**: the perfect system is the one which, for a code $C$, assigns $x\%$ of the verbatims to $C$ exactly when the expert coder would have assigned $x\%$ of the verbatims to $C$.

- The former is especially interesting for customer satisfaction applications, while the latter is especially interesting for survey analysis and market research.

- Accuracy at the individual level implies accuracy at the aggregate level, but not vice versa!
Effectiveness: individual or aggregate?

- Effectiveness may be measured at two different levels:
  - At the **individual level**: the perfect system is the one which, for a code $C$, assigns $C$ to the verbatim exactly when the expert coder would have assigned $C$.
  - At the **aggregate level**: the perfect system is the one which, for a code $C$, assigns $x\%$ of the verbatims to $C$ exactly when the expert coder would have assigned $x\%$ of the verbatims to $C$.

- The former is especially interesting for customer satisfaction applications, while the latter is especially interesting for survey analysis and market research.

- Accuracy at the individual level implies accuracy at the aggregate level, but not vice versa!
Effectiveness: individual or aggregate?

- Effectiveness may be measured at two different levels:
  - At the **individual level**: the perfect system is the one which, for a code $C$, assigns $C$ to the verbatim exactly when the expert coder would have assigned $C$.
  - At the **aggregate level**: the perfect system is the one which, for a code $C$, assigns $x\%$ of the verbatims to $C$ exactly when the expert coder would have assigned $x\%$ of the verbatims to $C$.

- The former is especially interesting for customer satisfaction applications, while the latter is especially interesting for survey analysis and market research.

- Accuracy at the individual level implies accuracy at the aggregate level, but not vice versa!
Effectiveness: individual or aggregate?

- Effectiveness may be measured at two different levels:
  - At the **individual level**: the perfect system is the one which, for a code $C$, assigns $C$ to the verbatim exactly when the expert coder would have assigned $C$.
  - At the **aggregate level**: the perfect system is the one which, for a code $C$, assigns $x\%$ of the verbatims to $C$ exactly when the expert coder would have assigned $x\%$ of the verbatims to $C$.

- The former is especially interesting for customer satisfaction applications, while the latter is especially interesting for survey analysis and market research.

- Accuracy at the individual level implies accuracy at the aggregate level, but not vice versa!
Outline

1. Introduction

2. VCS: an automated Verbatim Coding System

3. VCS: Effectiveness Tests
   - Effectiveness at the individual level
   - Effectiveness at the aggregate level

4. VCS: Efficiency Tests

5. The future
Effectiveness testing requires an effectiveness measure to be defined and agreed upon. The one we adopt, called $F_1$, relies on the following two notions:

- For a given code $C$, precision (denoted $\pi$) measures the ability of the system to avoid “overcoding”, i.e., attributing $C$ when it should not be attributed; that is, the ability of the system to avoid “false positives” (aka “errors of commission”, or “Type I errors”) for code $C$.

- For a given code $C$, recall (denoted $\rho$) measures the ability of the system to avoid “undercoding”, i.e., failing to attribute $C$ when it should instead be attributed; that is, the ability of the system to avoid “false negatives” (aka “errors of omission”, or “Type II errors”) for code $C$. 
Effectiveness testing requires an **effectiveness measure** to be defined and agreed upon. The one we adopt, called $F_1$, relies on the following two notions:

- **Precision** (denoted $\pi$) measures the ability of the system to avoid “overcoding”, i.e., attributing code $C$ when it should not be attributed; that is, the ability of the system to avoid “false positives” (aka “errors of commission”, or “Type I errors”) for code $C$.

- **Recall** (denoted $\rho$) measures the ability of the system to avoid “undercoding”, i.e., failing to attribute code $C$ when it should instead be attributed; that is, the ability of the system to avoid “false negatives” (aka “errors of omission”, or “Type II errors”) for code $C$. 
Precision and Recall

- Effectiveness testing requires an effectiveness measure to be defined and agreed upon. The one we adopt, called $F_1$, relies on the following two notions:

- For a given code $C$, **precision** (denoted $\pi$) measures the ability of the system to avoid “overcoding”, i.e., attributing $C$ when it should not be attributed; that is, the ability of the system to avoid “false positives” (aka “errors of commission”, or “Type I errors”) for code $C$.

- For a given code $C$, **recall** (denoted $\rho$) measures the ability of the system to avoid “undercoding”, i.e, failing to attribute $C$ when it should instead be attributed; that is, the ability of the system to avoid “false negatives” (aka “errors of omission”, or “Type II errors”) for code $C$. 
The $F_1$ measure

- In a given experiment, precision and recall are computed from a contingency table:

<table>
<thead>
<tr>
<th>Code</th>
<th>coder says</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$</td>
<td>YES</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>system says</th>
<th>YES</th>
<th>$TP$</th>
<th>$FP$</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>$FN$</td>
<td>$TN$</td>
<td></td>
</tr>
</tbody>
</table>

- Precision is defined as $\pi = \frac{TP}{TP + FP}$

- Recall is defined as $\rho = \frac{TP}{TP + FN}$

- The effectiveness measure we adopt is $F_1$, the “harmonic mean” of precision and recall, defined as

$$F_1 = \frac{2 \cdot \pi \cdot \rho}{\pi + \rho} = \frac{2 \cdot TP}{(2 \cdot TP) + FP + FN}$$
The $F_1$ measure

- In a given experiment, precision and recall are computed from a contingency table:

<table>
<thead>
<tr>
<th>Code</th>
<th>coder says</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>NO</td>
</tr>
<tr>
<td>system says</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>$TP$</td>
</tr>
<tr>
<td></td>
<td>$FP$</td>
</tr>
<tr>
<td>system says</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>$FN$</td>
</tr>
<tr>
<td></td>
<td>$TN$</td>
</tr>
</tbody>
</table>

- Precision is defined as $\pi = \frac{TP}{TP + FP}$
- Recall is defined as $\rho = \frac{TP}{TP + FN}$
- The effectiveness measure we adopt is $F_1$, the “harmonic mean” of precision and recall, defined as

$$F_1 = \frac{2 \cdot \pi \cdot \rho}{\pi + \rho} = \frac{2 \cdot TP}{(2 \cdot TP) + FP + FN}$$
The $F_1$ measure

- In a given experiment, precision and recall are computed from a contingency table:

<table>
<thead>
<tr>
<th>Code</th>
<th>coder says</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$</td>
<td>YES</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>system says</th>
<th>YES</th>
<th>$TP$</th>
<th>$FP$</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>$FN$</td>
<td>$TN$</td>
<td></td>
</tr>
</tbody>
</table>

- Precision is defined as $\pi = \frac{TP}{TP + FP}$
- Recall is defined as $\rho = \frac{TP}{TP + FN}$
- The effectiveness measure we adopt is $F_1$, the “harmonic mean” of precision and recall, defined as

$$F_1 = \frac{2 \cdot \pi \cdot \rho}{\pi + \rho} = \frac{2 \cdot TP}{(2 \cdot TP) + FP + FN}$$
The $F_1$ measure

- In a given experiment, precision and recall are computed from a contingency table:

<table>
<thead>
<tr>
<th>Code C coder says</th>
<th>system says NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>YES</td>
<td>$TP$</td>
</tr>
<tr>
<td>NO</td>
<td>$FN$</td>
</tr>
<tr>
<td></td>
<td>$TP + FN$</td>
</tr>
</tbody>
</table>

- Precision is defined as $\pi = \frac{TP}{TP + FP}$

- Recall is defined as $\rho = \frac{TP}{TP + FN}$

- The effectiveness measure we adopt is $F_1$, the “harmonic mean” of precision and recall, defined as

$$F_1 = \frac{2 \cdot \pi \cdot \rho}{\pi + \rho} = \frac{2 \cdot TP}{(2 \cdot TP) + FP + FN}$$
## Testing effectiveness on an example dataset

Example: 100 verbatims, codeframe consisting of two codes $C_i$ and $C_j$:

<table>
<thead>
<tr>
<th>Code</th>
<th>coder says</th>
<th>YES</th>
<th>NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_i$</td>
<td>system says</td>
<td>YES</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>system says</td>
<td>NO</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code</th>
<th>coder says</th>
<th>YES</th>
<th>NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_j$</td>
<td>system says</td>
<td>YES</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>system says</td>
<td>NO</td>
<td>5</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\pi &= \frac{15}{15 + 7} = \frac{15}{22} = .682 \\
\rho &= \frac{15}{15 + 8} = \frac{15}{23} = .652 \\
F_1 &= \frac{2 \cdot .682 \cdot .652}{.682 + .652} = .667
\end{align*}
\]

\[
\begin{align*}
\pi_j &= \frac{22}{22 + 13} = \frac{22}{35} = .629 \\
\rho_j &= \frac{22}{22 + 5} = \frac{22}{27} = .815 \\
F_1 &= \frac{2 \cdot .629 \cdot .815}{.629 + .815} = .710
\end{align*}
\]
Testing effectiveness on an example dataset

Example: 100 verbatims, codeframe consisting of two codes $C_i$ and $C_j$:

<table>
<thead>
<tr>
<th>Code $C_i$</th>
<th>coder says</th>
<th>$\pi = \frac{15}{15 + 7} = \frac{15}{22} = .682$</th>
</tr>
</thead>
<tbody>
<tr>
<td>system says</td>
<td>YES</td>
<td>15</td>
</tr>
<tr>
<td>NO</td>
<td>8</td>
<td>70</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code $C_j$</th>
<th>coder says</th>
<th>$\pi_j = \frac{22}{22 + 13} = \frac{22}{35} = .629$</th>
</tr>
</thead>
<tbody>
<tr>
<td>system says</td>
<td>YES</td>
<td>22</td>
</tr>
<tr>
<td>NO</td>
<td>5</td>
<td>60</td>
</tr>
</tbody>
</table>

| $\rho = \frac{15}{15 + 8} = \frac{15}{23} = .652$ |
| $F_1 = \frac{2 \cdot .682 \cdot .652}{.682 + .652} = .667$ |

| $\rho_j = \frac{22}{22 + 5} = \frac{22}{27} = .815$ |
| $F_1 = \frac{2 \cdot .629 \cdot .815}{.629 + .815} = .710$ |
Computing effectiveness wrt an entire codeframe

- Precision, recall and $F_1$ can also be computed relative to an entire codeframe by using a “combined” contingency table

<table>
<thead>
<tr>
<th>Codes $C_i$ and $C_j$</th>
<th>coder says</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>YES</td>
</tr>
<tr>
<td>system says</td>
<td>15 + 22</td>
</tr>
<tr>
<td>says</td>
<td>8 + 5</td>
</tr>
</tbody>
</table>

$$\pi^\mu = \frac{(15 + 22)}{(15 + 22) + (7 + 13)} = \frac{37}{57} = .649$$

$$\rho^\mu = \frac{(15 + 22)}{(15 + 22) + (8 + 5)} = \frac{37}{50} = .740$$

$$F_1^\mu = \frac{2 \cdot .649 \cdot .740}{.649 + .740} = .692$$
Computing effectiveness wrt an entire codeframe

- Precision, recall and $F_1$ can also be computed relative to an entire codeframe by using a “combined” contingency table

<table>
<thead>
<tr>
<th>Codes $C_i$ and $C_j$</th>
<th>coder says</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>YES</td>
</tr>
<tr>
<td>system says NO</td>
<td>8 + 5</td>
</tr>
<tr>
<td>system says YES</td>
<td>15 + 22</td>
</tr>
</tbody>
</table>

$$
\pi^\mu = \frac{(15 + 22)}{(15 + 22) + (7 + 13)} = \frac{37}{57} = .649
$$

$$
\rho^\mu = \frac{(15 + 22)}{(15 + 22) + (8 + 5)} = \frac{37}{50} = .740
$$

$$
F_1^\mu = \frac{2 \cdot .649 \cdot .740}{.649 + .740} = .692
$$
Why is $F_1$ a good measure of effectiveness?

- $F_1 = 0$ for the “pervert system” ($TP = TN = 0$) and $F_1 = 1$ for the “perfect system” ($FN = FP = 0$).
- It partially rewards partial success: i.e., if the true codes of a verbatim are $c_1$, $c_2$, $c_3$, $c_4$, attributing $c_1$, $c_2$, $c_3$ is rewarded more than attributing $c_1$ only.
- It is not easy to game: it has very low values for “trivial” coding systems (e.g. the “trivial rejector” has $F_1 = 0$, the “trivial acceptor” has $F_1 = \frac{TP+FN}{TP+FP+FN+TN}$, which is usually low).
- It rewards systems that balance precision and recall.
- It is symmetric; i.e., the agreement between system and coder is the same as the agreement between coder and system.
- It is (thus) an “industry standard” in the field of text coding.
Why is $F_1$ a good measure of effectiveness?

- $F_1 = 0$ for the “pervert system” ($TP = TN = 0$) and $F_1 = 1$ for the “perfect system” ($FN = FP = 0$).
- It partially rewards partial success: i.e., if the true codes of a verbatim are $c_1, c_2, c_3, c_4$, attributing $c_1, c_2, c_3$ is rewarded more than attributing $c_1$ only.
- It is not easy to game: it has very low values for “trivial” coding systems (e.g. the “trivial rejector” has $F_1 = 0$, the “trivial acceptor” has $F_1 = \frac{TP + FN}{TP + FP + FN + TN}$, which is usually low).
- It rewards systems that balance precision and recall.
- It is symmetric; i.e., the agreement between system and coder is the same as the agreement between coder and system.
- It is (thus) an “industry standard” in the field of text coding.
Why is $F_1$ a good measure of effectiveness?

- $F_1 = 0$ for the “pervert system” ($TP = TN = 0$) and $F_1 = 1$ for the “perfect system” ($FN = FP = 0$).
- It partially rewards partial success: i.e., if the true codes of a verbatim are $c_1, c_2, c_3, c_4$, attributing $c_1, c_2, c_3$ is rewarded more than attributing $c_1$ only.
- It is not easy to game: it has very low values for “trivial” coding systems (e.g. the “trivial rejector” has $F_1 = 0$, the “trivial acceptor” has $F_1 = \frac{TP+FN}{TP+FP+FN+TN}$, which is usually low).
- It rewards systems that balance precision and recall.
- It is symmetric; i.e., the agreement between system and coder is the same as the agreement between coder and system.
- It is (thus) an “industry standard” in the field of text coding.
Why is $F_1$ a good measure of effectiveness?

- $F_1 = 0$ for the “pervert system” ($TP = TN = 0$) and $F_1 = 1$ for the “perfect system” ($FN = FP = 0$).
- It partially rewards partial success: i.e., if the true codes of a verbatim are $c_1, c_2, c_3, c_4$, attributing $c_1, c_2, c_3$ is rewarded more than attributing $c_1$ only.
- It is not easy to game: it has very low values for “trivial” coding systems (e.g. the “trivial rejector” has $F_1 = 0$, the “trivial acceptor” has $F_1 = \frac{TP+FN}{TP+FP+FN+TN}$, which is usually low).
- It rewards systems that balance precision and recall.
- It is symmetric; i.e., the agreement between system and coder is the same as the agreement between coder and system.
- It is (thus) an “industry standard” in the field of text coding.
Why is $F_1$ a good measure of effectiveness?

- $F_1 = 0$ for the “pervert system” ($TP = TN = 0$) and $F_1 = 1$ for the “perfect system” ($FN = FP = 0$).
- It partially rewards partial success: i.e., if the true codes of a verbatim are $c_1, c_2, c_3, c_4$, attributing $c_1, c_2, c_3$ is rewarded more than attributing $c_1$ only.
- It is not easy to game: it has very low values for “trivial” coding systems (e.g. the “trivial rejector” has $F_1 = 0$, the “trivial acceptor” has $F_1 = \frac{TP + FN}{TP + FP + FN + TN}$, which is usually low).
- It rewards systems that balance precision and recall.
- It is symmetric; i.e., the agreement between system and coder is the same as the agreement between coder and system.
- It is (thus) an “industry standard” in the field of text coding.
Why is $F_1$ a good measure of effectiveness?

- $F_1 = 0$ for the “pervert system” ($TP = TN = 0$) and $F_1 = 1$ for the “perfect system” ($FN = FP = 0$).
- It partially rewards partial success: i.e., if the true codes of a verbatim are $c_1, c_2, c_3, c_4$, attributing $c_1, c_2, c_3$ is rewarded more than attributing $c_1$ only.
- It is not easy to game: it has very low values for “trivial” coding systems (e.g. the “trivial rejector” has $F_1 = 0$, the “trivial acceptor” has $F_1 = \frac{TP+FN}{TP+FP+FN+TN}$, which is usually low).
- It rewards systems that balance precision and recall.
- It is symmetric; i.e., the agreement between system and coder is the same as the agreement between coder and system.
- It is (thus) an “industry standard” in the field of text coding.
## Real tests: the Language Logic data & the Egg data

<table>
<thead>
<tr>
<th>DS</th>
<th>Tr</th>
<th>Te</th>
<th>C</th>
<th>ATC</th>
<th>D</th>
<th>AVL</th>
<th>LR</th>
<th>$F_1^\mu$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LL-A</td>
<td>201</td>
<td>65</td>
<td>18</td>
<td>21.00</td>
<td>61</td>
<td>1.35</td>
<td>.344</td>
<td>.92</td>
</tr>
<tr>
<td>LL-B</td>
<td>501</td>
<td>10299</td>
<td>34</td>
<td>26.65</td>
<td>151</td>
<td>1.65</td>
<td>.176</td>
<td>.90</td>
</tr>
<tr>
<td>LL-C</td>
<td>201</td>
<td>425</td>
<td>20</td>
<td>10.05</td>
<td>60</td>
<td>1.61</td>
<td>.168</td>
<td>.89</td>
</tr>
<tr>
<td>LL-D</td>
<td>501</td>
<td>698</td>
<td>27</td>
<td>45.30</td>
<td>471</td>
<td>3.32</td>
<td>.096</td>
<td>.85</td>
</tr>
<tr>
<td>LL-E</td>
<td>201</td>
<td>720</td>
<td>39</td>
<td>8.41</td>
<td>155</td>
<td>2.57</td>
<td>.054</td>
<td>.84</td>
</tr>
<tr>
<td>LL-F</td>
<td>501</td>
<td>999</td>
<td>57</td>
<td>37.58</td>
<td>551</td>
<td>6.99</td>
<td>.068</td>
<td>.82</td>
</tr>
<tr>
<td>LL-G</td>
<td>501</td>
<td>1898</td>
<td>104</td>
<td>21.30</td>
<td>611</td>
<td>6.25</td>
<td>.035</td>
<td>.80</td>
</tr>
<tr>
<td>LL-H</td>
<td>501</td>
<td>699</td>
<td>86</td>
<td>30.08</td>
<td>817</td>
<td>7.87</td>
<td>.037</td>
<td>.79</td>
</tr>
<tr>
<td>LL-I</td>
<td>501</td>
<td>699</td>
<td>69</td>
<td>33.16</td>
<td>764</td>
<td>7.70</td>
<td>.043</td>
<td>.78</td>
</tr>
<tr>
<td>LL-L</td>
<td>501</td>
<td>698</td>
<td>65</td>
<td>29.40</td>
<td>673</td>
<td>5.58</td>
<td>.044</td>
<td>.75</td>
</tr>
<tr>
<td>Egg-A</td>
<td>700</td>
<td>300</td>
<td>14</td>
<td>91.14</td>
<td>2948</td>
<td>28.60</td>
<td>.031</td>
<td>.63</td>
</tr>
<tr>
<td>Egg-B</td>
<td>653</td>
<td>273</td>
<td>20</td>
<td>50.32</td>
<td>3620</td>
<td>27.60</td>
<td>.014</td>
<td>.60</td>
</tr>
</tbody>
</table>
Real tests: the Language Logic data & the Egg data

<table>
<thead>
<tr>
<th>DS</th>
<th>$Tr$</th>
<th>$Te$</th>
<th>$C$</th>
<th>ATC</th>
<th>$D$</th>
<th>AVL</th>
<th>LR</th>
<th>$F_1^\mu$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LL-A</td>
<td>201</td>
<td>65</td>
<td>18</td>
<td>21.00</td>
<td>61</td>
<td>1.35</td>
<td>.344</td>
<td>.92</td>
</tr>
<tr>
<td>LL-B</td>
<td>501</td>
<td>10299</td>
<td>34</td>
<td>26.65</td>
<td>151</td>
<td>1.65</td>
<td>.176</td>
<td>.90</td>
</tr>
<tr>
<td>LL-C</td>
<td>201</td>
<td>425</td>
<td>20</td>
<td>10.05</td>
<td>60</td>
<td>1.61</td>
<td>.168</td>
<td>.89</td>
</tr>
<tr>
<td>LL-D</td>
<td>501</td>
<td>698</td>
<td>27</td>
<td>45.30</td>
<td>471</td>
<td>3.32</td>
<td>.096</td>
<td>.85</td>
</tr>
<tr>
<td>LL-E</td>
<td>201</td>
<td>720</td>
<td>39</td>
<td>8.41</td>
<td>155</td>
<td>2.57</td>
<td>.054</td>
<td>.84</td>
</tr>
<tr>
<td>LL-F</td>
<td>501</td>
<td>999</td>
<td>57</td>
<td>37.58</td>
<td>551</td>
<td>6.99</td>
<td>.068</td>
<td>.82</td>
</tr>
<tr>
<td>LL-G</td>
<td>501</td>
<td>1898</td>
<td>104</td>
<td>21.30</td>
<td>611</td>
<td>6.25</td>
<td>.035</td>
<td>.80</td>
</tr>
<tr>
<td>LL-H</td>
<td>501</td>
<td>699</td>
<td>86</td>
<td>30.08</td>
<td>817</td>
<td>7.87</td>
<td>.037</td>
<td>.79</td>
</tr>
<tr>
<td>LL-I</td>
<td>501</td>
<td>699</td>
<td>69</td>
<td>33.16</td>
<td>764</td>
<td>7.70</td>
<td>.043</td>
<td>.78</td>
</tr>
<tr>
<td>LL-L</td>
<td>501</td>
<td>698</td>
<td>65</td>
<td>29.40</td>
<td>673</td>
<td>5.58</td>
<td>.044</td>
<td>.75</td>
</tr>
<tr>
<td>Egg-A</td>
<td>700</td>
<td>300</td>
<td>14</td>
<td>91.14</td>
<td>2948</td>
<td>28.60</td>
<td>.031</td>
<td>.63</td>
</tr>
<tr>
<td>Egg-B</td>
<td>653</td>
<td>273</td>
<td>20</td>
<td>50.32</td>
<td>3620</td>
<td>27.60</td>
<td>.014</td>
<td>.60</td>
</tr>
</tbody>
</table>
# Real tests: the Language Logic data & the Egg data

<table>
<thead>
<tr>
<th>DS</th>
<th>Tr</th>
<th>Te</th>
<th>C</th>
<th>ATC</th>
<th>D</th>
<th>AVL</th>
<th>LR</th>
<th>$F^\mu_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LL-A</td>
<td>201</td>
<td>65</td>
<td>18</td>
<td>21.00</td>
<td>61</td>
<td>1.35</td>
<td>.344</td>
<td>.92</td>
</tr>
<tr>
<td>LL-B</td>
<td>501</td>
<td>10299</td>
<td>34</td>
<td>26.65</td>
<td>151</td>
<td>1.65</td>
<td>.176</td>
<td>.90</td>
</tr>
<tr>
<td>LL-C</td>
<td>201</td>
<td>425</td>
<td>20</td>
<td>10.05</td>
<td>60</td>
<td>1.61</td>
<td>.168</td>
<td>.89</td>
</tr>
<tr>
<td>LL-D</td>
<td>501</td>
<td>698</td>
<td>27</td>
<td>45.30</td>
<td>471</td>
<td>3.32</td>
<td>.096</td>
<td>.85</td>
</tr>
<tr>
<td>LL-E</td>
<td>201</td>
<td>720</td>
<td>39</td>
<td>8.41</td>
<td>155</td>
<td>2.57</td>
<td>.054</td>
<td>.84</td>
</tr>
<tr>
<td>LL-F</td>
<td>501</td>
<td>999</td>
<td>57</td>
<td>37.58</td>
<td>551</td>
<td>6.99</td>
<td>.068</td>
<td>.82</td>
</tr>
<tr>
<td>LL-G</td>
<td>501</td>
<td>1898</td>
<td>104</td>
<td>21.30</td>
<td>611</td>
<td>6.25</td>
<td>.035</td>
<td>.80</td>
</tr>
<tr>
<td>LL-H</td>
<td>501</td>
<td>699</td>
<td>86</td>
<td>30.08</td>
<td>817</td>
<td>7.87</td>
<td>.037</td>
<td>.79</td>
</tr>
<tr>
<td>LL-I</td>
<td>501</td>
<td>699</td>
<td>69</td>
<td>33.16</td>
<td>764</td>
<td>7.70</td>
<td>.043</td>
<td>.78</td>
</tr>
<tr>
<td>LL-L</td>
<td>501</td>
<td>698</td>
<td>65</td>
<td>29.40</td>
<td>673</td>
<td>5.58</td>
<td>.044</td>
<td>.75</td>
</tr>
<tr>
<td>Egg-A</td>
<td>700</td>
<td>300</td>
<td>14</td>
<td>91.14</td>
<td>2948</td>
<td>28.60</td>
<td>.031</td>
<td>.63</td>
</tr>
<tr>
<td>Egg-B</td>
<td>653</td>
<td>273</td>
<td>20</td>
<td>50.32</td>
<td>3620</td>
<td>27.60</td>
<td>.014</td>
<td>.60</td>
</tr>
</tbody>
</table>
Real tests: the Language Logic data & the Egg data

<table>
<thead>
<tr>
<th>DS</th>
<th>Tr</th>
<th>Te</th>
<th>C</th>
<th>ATC</th>
<th>D</th>
<th>AVL</th>
<th>LR</th>
<th>$F_1^\mu$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LL-A</td>
<td>201</td>
<td>65</td>
<td>18</td>
<td>21.00</td>
<td>61</td>
<td>1.35</td>
<td>.344</td>
<td>.92</td>
</tr>
<tr>
<td>LL-B</td>
<td>501</td>
<td>10299</td>
<td>34</td>
<td>26.65</td>
<td>151</td>
<td>1.65</td>
<td>.176</td>
<td>.90</td>
</tr>
<tr>
<td>LL-C</td>
<td>201</td>
<td>425</td>
<td>20</td>
<td>10.05</td>
<td>60</td>
<td>1.61</td>
<td>.168</td>
<td>.89</td>
</tr>
<tr>
<td>LL-D</td>
<td>501</td>
<td>698</td>
<td>27</td>
<td>45.30</td>
<td>471</td>
<td>3.32</td>
<td>.096</td>
<td>.85</td>
</tr>
<tr>
<td>LL-E</td>
<td>201</td>
<td>720</td>
<td>39</td>
<td>8.41</td>
<td>155</td>
<td>2.57</td>
<td>.054</td>
<td>.84</td>
</tr>
<tr>
<td>LL-F</td>
<td>501</td>
<td>999</td>
<td>57</td>
<td>37.58</td>
<td>551</td>
<td>6.99</td>
<td>.068</td>
<td>.82</td>
</tr>
<tr>
<td>LL-G</td>
<td>501</td>
<td>1898</td>
<td>104</td>
<td>21.30</td>
<td>611</td>
<td>6.25</td>
<td>.035</td>
<td>.80</td>
</tr>
<tr>
<td>LL-H</td>
<td>501</td>
<td>699</td>
<td>86</td>
<td>30.08</td>
<td>817</td>
<td>7.87</td>
<td>.037</td>
<td>.79</td>
</tr>
<tr>
<td>LL-I</td>
<td>501</td>
<td>699</td>
<td>69</td>
<td>33.16</td>
<td>764</td>
<td>7.70</td>
<td>.043</td>
<td>.78</td>
</tr>
<tr>
<td>LL-L</td>
<td>501</td>
<td>698</td>
<td>65</td>
<td>29.40</td>
<td>673</td>
<td>5.58</td>
<td>.044</td>
<td>.75</td>
</tr>
<tr>
<td>Egg-A</td>
<td>700</td>
<td>300</td>
<td>14</td>
<td>91.14</td>
<td>2948</td>
<td>28.60</td>
<td>.031</td>
<td>.63</td>
</tr>
<tr>
<td>Egg-B</td>
<td>653</td>
<td>273</td>
<td>20</td>
<td>50.32</td>
<td>3620</td>
<td>27.60</td>
<td>.014</td>
<td>.60</td>
</tr>
</tbody>
</table>
Real tests: the Language Logic data & the Egg data

<table>
<thead>
<tr>
<th>DS</th>
<th>Tr</th>
<th>Te</th>
<th>C</th>
<th>ATC</th>
<th>D</th>
<th>AVL</th>
<th>LR</th>
<th>$F_1^{μ}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LL-A</td>
<td>201</td>
<td>65</td>
<td>18</td>
<td>21.00</td>
<td>61</td>
<td>1.35</td>
<td>.344</td>
<td>.92</td>
</tr>
<tr>
<td>LL-B</td>
<td>501</td>
<td>10299</td>
<td>34</td>
<td>26.65</td>
<td>151</td>
<td>1.65</td>
<td>.176</td>
<td>.90</td>
</tr>
<tr>
<td>LL-C</td>
<td>201</td>
<td>425</td>
<td>20</td>
<td>10.05</td>
<td>60</td>
<td>1.61</td>
<td>.168</td>
<td>.89</td>
</tr>
<tr>
<td>LL-D</td>
<td>501</td>
<td>698</td>
<td>27</td>
<td>45.30</td>
<td>471</td>
<td>3.32</td>
<td>.096</td>
<td>.85</td>
</tr>
<tr>
<td>LL-E</td>
<td>201</td>
<td>720</td>
<td>39</td>
<td>8.41</td>
<td>155</td>
<td>2.57</td>
<td>.054</td>
<td>.84</td>
</tr>
<tr>
<td>LL-F</td>
<td>501</td>
<td>999</td>
<td>57</td>
<td>37.58</td>
<td>551</td>
<td>6.99</td>
<td>.068</td>
<td>.82</td>
</tr>
<tr>
<td>LL-G</td>
<td>501</td>
<td>1898</td>
<td>104</td>
<td>21.30</td>
<td>611</td>
<td>6.25</td>
<td>.035</td>
<td>.80</td>
</tr>
<tr>
<td>LL-H</td>
<td>501</td>
<td>699</td>
<td>86</td>
<td>30.08</td>
<td>817</td>
<td>7.87</td>
<td>.037</td>
<td>.79</td>
</tr>
<tr>
<td>LL-I</td>
<td>501</td>
<td>699</td>
<td>69</td>
<td>33.16</td>
<td>764</td>
<td>7.70</td>
<td>.043</td>
<td>.78</td>
</tr>
<tr>
<td>LL-L</td>
<td>501</td>
<td>698</td>
<td>65</td>
<td>29.40</td>
<td>673</td>
<td>5.58</td>
<td>.044</td>
<td>.75</td>
</tr>
<tr>
<td>Egg-A</td>
<td>700</td>
<td>300</td>
<td>14</td>
<td>91.14</td>
<td>2948</td>
<td>28.60</td>
<td>.031</td>
<td>.63</td>
</tr>
<tr>
<td>Egg-B</td>
<td>653</td>
<td>273</td>
<td>20</td>
<td>50.32</td>
<td>3620</td>
<td>27.60</td>
<td>.014</td>
<td>.60</td>
</tr>
</tbody>
</table>
## Real tests: the Language Logic data & the Egg data

<table>
<thead>
<tr>
<th>DS</th>
<th>Tr</th>
<th>Te</th>
<th>C</th>
<th>ATC</th>
<th>D</th>
<th>AVL</th>
<th>LR</th>
<th>( F_1^{\mu} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>LL-A</td>
<td>201</td>
<td>65</td>
<td>18</td>
<td>21.00</td>
<td>61</td>
<td>1.35</td>
<td>.344</td>
<td>.92</td>
</tr>
<tr>
<td>LL-B</td>
<td>501</td>
<td>10299</td>
<td>34</td>
<td>26.65</td>
<td>151</td>
<td>1.65</td>
<td>.176</td>
<td>.90</td>
</tr>
<tr>
<td>LL-C</td>
<td>201</td>
<td>425</td>
<td>20</td>
<td>10.05</td>
<td>60</td>
<td>1.61</td>
<td>.168</td>
<td>.89</td>
</tr>
<tr>
<td>LL-D</td>
<td>501</td>
<td>698</td>
<td>27</td>
<td>45.30</td>
<td>471</td>
<td>3.32</td>
<td>.096</td>
<td>.85</td>
</tr>
<tr>
<td>LL-E</td>
<td>201</td>
<td>720</td>
<td>39</td>
<td>8.41</td>
<td>155</td>
<td>2.57</td>
<td>.054</td>
<td>.84</td>
</tr>
<tr>
<td>LL-F</td>
<td>501</td>
<td>999</td>
<td>57</td>
<td>37.58</td>
<td>551</td>
<td>6.99</td>
<td>.068</td>
<td>.82</td>
</tr>
<tr>
<td>LL-G</td>
<td>501</td>
<td>1898</td>
<td>104</td>
<td>21.30</td>
<td>611</td>
<td>6.25</td>
<td>.035</td>
<td>.80</td>
</tr>
<tr>
<td>LL-H</td>
<td>501</td>
<td>699</td>
<td>86</td>
<td>30.08</td>
<td>817</td>
<td>7.87</td>
<td>.037</td>
<td>.79</td>
</tr>
<tr>
<td>LL-I</td>
<td>501</td>
<td>699</td>
<td>69</td>
<td>33.16</td>
<td>764</td>
<td>7.70</td>
<td>.043</td>
<td>.78</td>
</tr>
<tr>
<td>LL-L</td>
<td>501</td>
<td>698</td>
<td>65</td>
<td>29.40</td>
<td>673</td>
<td>5.58</td>
<td>.044</td>
<td>.75</td>
</tr>
<tr>
<td>Egg-A</td>
<td>700</td>
<td>300</td>
<td>14</td>
<td>91.14</td>
<td>2948</td>
<td>28.60</td>
<td>.031</td>
<td>.63</td>
</tr>
<tr>
<td>Egg-B</td>
<td>653</td>
<td>273</td>
<td>20</td>
<td>50.32</td>
<td>3620</td>
<td>27.60</td>
<td>.014</td>
<td>.60</td>
</tr>
</tbody>
</table>
### Real tests: the Language Logic data & the Egg data

<table>
<thead>
<tr>
<th>DS</th>
<th>Tr</th>
<th>Te</th>
<th>C</th>
<th>ATC</th>
<th>D</th>
<th>AVL</th>
<th>LR</th>
<th>$F_1^\mu$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LL-A</td>
<td>201</td>
<td>65</td>
<td>18</td>
<td>21.00</td>
<td>61</td>
<td>1.35</td>
<td>.344</td>
<td>.92</td>
</tr>
<tr>
<td>LL-B</td>
<td>501</td>
<td>10299</td>
<td>34</td>
<td>26.65</td>
<td>151</td>
<td>1.65</td>
<td>.176</td>
<td>.90</td>
</tr>
<tr>
<td>LL-C</td>
<td>201</td>
<td>425</td>
<td>20</td>
<td>10.05</td>
<td>60</td>
<td>1.61</td>
<td>.168</td>
<td>.89</td>
</tr>
<tr>
<td>LL-D</td>
<td>501</td>
<td>698</td>
<td>27</td>
<td>45.30</td>
<td>471</td>
<td>3.32</td>
<td>.096</td>
<td>.85</td>
</tr>
<tr>
<td>LL-E</td>
<td>201</td>
<td>720</td>
<td>39</td>
<td>8.41</td>
<td>155</td>
<td>2.57</td>
<td>.054</td>
<td>.84</td>
</tr>
<tr>
<td>LL-F</td>
<td>501</td>
<td>999</td>
<td>57</td>
<td>37.58</td>
<td>551</td>
<td>6.99</td>
<td>.068</td>
<td>.82</td>
</tr>
<tr>
<td>LL-G</td>
<td>501</td>
<td>1898</td>
<td>104</td>
<td>21.30</td>
<td>611</td>
<td>6.25</td>
<td>.035</td>
<td>.80</td>
</tr>
<tr>
<td>LL-H</td>
<td>501</td>
<td>699</td>
<td>86</td>
<td>30.08</td>
<td>817</td>
<td>7.87</td>
<td>.037</td>
<td>.79</td>
</tr>
<tr>
<td>LL-I</td>
<td>501</td>
<td>699</td>
<td>69</td>
<td>33.16</td>
<td>764</td>
<td>7.70</td>
<td>.043</td>
<td>.78</td>
</tr>
<tr>
<td>LL-L</td>
<td>501</td>
<td>698</td>
<td>65</td>
<td>29.40</td>
<td>673</td>
<td>5.58</td>
<td>.044</td>
<td>.75</td>
</tr>
<tr>
<td>Egg-A</td>
<td>700</td>
<td>300</td>
<td>14</td>
<td>91.14</td>
<td>2948</td>
<td>28.60</td>
<td>.031</td>
<td>.63</td>
</tr>
<tr>
<td>Egg-B</td>
<td>653</td>
<td>273</td>
<td>20</td>
<td>50.32</td>
<td>3620</td>
<td>27.60</td>
<td>.014</td>
<td>.60</td>
</tr>
</tbody>
</table>
### Real tests: the Language Logic data & the Egg data

<table>
<thead>
<tr>
<th>DS</th>
<th>Tr</th>
<th>Te</th>
<th>C</th>
<th>ATC</th>
<th>D</th>
<th>AVL</th>
<th>LR</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LL-A</td>
<td>201</td>
<td>65</td>
<td>18</td>
<td>21.00</td>
<td>61</td>
<td>1.35</td>
<td>.344</td>
<td>.92</td>
</tr>
<tr>
<td>LL-B</td>
<td>501</td>
<td>10299</td>
<td>34</td>
<td>26.65</td>
<td>151</td>
<td>1.65</td>
<td>.176</td>
<td>.90</td>
</tr>
<tr>
<td>LL-C</td>
<td>201</td>
<td>425</td>
<td>20</td>
<td>10.05</td>
<td>60</td>
<td>1.61</td>
<td>.168</td>
<td>.89</td>
</tr>
<tr>
<td>LL-D</td>
<td>501</td>
<td>698</td>
<td>27</td>
<td>45.30</td>
<td>471</td>
<td>3.32</td>
<td>.096</td>
<td>.85</td>
</tr>
<tr>
<td>LL-E</td>
<td>201</td>
<td>720</td>
<td>39</td>
<td>8.41</td>
<td>155</td>
<td>2.57</td>
<td>.054</td>
<td>.84</td>
</tr>
<tr>
<td>LL-F</td>
<td>501</td>
<td>999</td>
<td>57</td>
<td>37.58</td>
<td>551</td>
<td>6.99</td>
<td>.068</td>
<td>.82</td>
</tr>
<tr>
<td>LL-G</td>
<td>501</td>
<td>1898</td>
<td>104</td>
<td>21.30</td>
<td>611</td>
<td>6.25</td>
<td>.035</td>
<td>.80</td>
</tr>
<tr>
<td>LL-H</td>
<td>501</td>
<td>699</td>
<td>86</td>
<td>30.08</td>
<td>817</td>
<td>7.87</td>
<td>.037</td>
<td>.79</td>
</tr>
<tr>
<td>LL-I</td>
<td>501</td>
<td>699</td>
<td>69</td>
<td>33.16</td>
<td>764</td>
<td>7.70</td>
<td>.043</td>
<td>.78</td>
</tr>
<tr>
<td>LL-L</td>
<td>501</td>
<td>698</td>
<td>65</td>
<td>29.40</td>
<td>673</td>
<td>5.58</td>
<td>.044</td>
<td>.75</td>
</tr>
<tr>
<td>Egg-A</td>
<td>700</td>
<td>300</td>
<td>14</td>
<td>91.14</td>
<td>2948</td>
<td>28.60</td>
<td>.031</td>
<td>.63</td>
</tr>
<tr>
<td>Egg-B</td>
<td>653</td>
<td>273</td>
<td>20</td>
<td>50.32</td>
<td>3620</td>
<td>27.60</td>
<td>.014</td>
<td>.60</td>
</tr>
</tbody>
</table>
How good are these results?

- How good are $F_1 = .75$ and $F_1 = .92$?
- Is $F_1 = .92$ exactly 8% worse than I would get from my coders? No, since your coders won’t get you $F_1 = 1$.
- How good a given $F_1$ value on the part of VCS is can only be measured in an intercoder agreement study, i.e., wrt the value of $F_1$ that two human coders would achieve wrt each other on the same dataset. For codes
  1. “Coke” for question “What is your favourite soft drink?”
  2. “Customer is ready to defect” for question “Are you happy with the quality of our service?”

different levels of $F_1$ may be expected, both by an automatic coding system and by a human coder. Code 2 is inherently more controversial than Code 1.
How good are these results?

- How good are $F_1 = .75$ and $F_1 = .92$?
- Is $F_1 = .92$ exactly 8% worse than I would get from my coders? **No**, since your coders won’t get you $F_1 = 1$.
- How good a given $F_1$ value on the part of VCS is can only be measured in an intercoder agreement study, i.e., wrt the value of $F_1$ that two human coders would achieve wrt each other on the same dataset. For codes
  1. “Coke” for question “What is your favourite soft drink?”
  2. “Customer is ready to defect” for question “Are you happy with the quality of our service?”

different levels of $F_1$ may be expected, both by an automatic coding system and by a human coder. Code 2 is inherently more controversial than Code 1.
How good are these results?

- How good are $F_1 = .75$ and $F_1 = .92$?
- Is $F_1 = .92$ exactly 8% worse than I would get from my coders? **No,** since your coders won’t get you $F_1 = 1$.

How good a given $F_1$ value on the part of VCS is can only be measured in an intercoder agreement study, i.e., wrt the value of $F_1$ that two human coders would achieve wrt each other on the same dataset. For codes

1. “Coke” for question “What is your favourite soft drink?”
2. “Customer is ready to defect” for question “Are you happy with the quality of our service?”

different levels of $F_1$ may be expected, both by an automatic coding system and by a human coder. Code 2 is inherently more controversial than Code 1.
How good are these results?

- How good are $F_1 = .75$ and $F_1 = .92$?
- Is $F_1 = .92$ exactly 8% worse than I would get from my coders? **No,** since your coders won’t get you $F_1 = 1$.
- How good a given $F_1$ value on the part of VCS is can only be measured in an intercoder agreement study, i.e., wrt the value of $F_1$ that two human coders would achieve wrt each other on the same dataset. For codes

  1. “Coke” for question “What is your favourite soft drink?”
  2. “Customer is ready to defect” for question “Are you happy with the quality of our service?”

different levels of $F_1$ may be expected, both by an automatic coding system and by a human coder. Code 2 is inherently more controversial than Code 1.
How good are these results?

- How good are $F_1 = .75$ and $F_1 = .92$?
- Is $F_1 = .92$ exactly 8% worse than I would get from my coders? No, since your coders won’t get you $F_1 = 1$.
- How good a given $F_1$ value on the part of VCS is can only be measured in an intercoder agreement study, i.e., wrt the value of $F_1$ that two human coders would achieve wrt each other on the same dataset. For codes
  1. “Coke” for question “What is your favourite soft drink?”
  2. “Customer is ready to defect” for question “Are you happy with the quality of our service?”

different levels of $F_1$ may be expected, both by an automatic coding system and by a human coder. Code 2 is inherently more controversial than Code 1.
How good can be VCS expected to be on a new dataset?

- We have experimentally observed that the $F_1$ of VCS tends to increase with
  - the average number of training verbatims per code (ATC) provided to the system
  - the degree of “linguistic regularity” (LR) in the training verbatims;
  - how uncontroversial the code is, which can be measured by intercoder agreement. On the Egg datasets VCS was roughly 85% as good as expert human coders.

<table>
<thead>
<tr>
<th></th>
<th>Easier</th>
<th>Harder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average # of Training Verbatim per Code (ATC)</td>
<td>High</td>
<td>Small</td>
</tr>
<tr>
<td>Average Verbatim Length (AVL)</td>
<td>Small</td>
<td>High</td>
</tr>
<tr>
<td>Human Coder Agreement</td>
<td>High</td>
<td>Small</td>
</tr>
</tbody>
</table>
How good can be VCS expected to be on a new dataset?

- We have experimentally observed that the $F_1$ of VCS tends to increase with
  - the average number of training verbatims per code (ATC) provided to the system
  - the degree of “linguistic regularity” (LR) in the training verbatims;
  - how uncontroversial the code is, which can be measured by intercoder agreement. On the Egg datasets VCS was roughly 85% as good as expert human coders.

<table>
<thead>
<tr>
<th></th>
<th>Easier</th>
<th>Harder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average # of Training Verbatim per Code (ATC)</td>
<td>High</td>
<td>Small</td>
</tr>
<tr>
<td>Average Verbatim Length (AVL)</td>
<td>Small</td>
<td>High</td>
</tr>
<tr>
<td>Human Coder Agreement</td>
<td>High</td>
<td>Small</td>
</tr>
</tbody>
</table>
How good can be VCS expected to be on a new dataset?

- We have experimentally observed that the $F_1$ of VCS tends to increase with:
  - the average number of training verbatims per code (ATC) provided to the system
  - the degree of “linguistic regularity” (LR) in the training verbatims;
  - how uncontroversial the code is, which can be measured by intercoder agreement. On the Egg datasets VCS was roughly 85% as good as expert human coders.

<table>
<thead>
<tr>
<th></th>
<th>Easier</th>
<th>Harder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average # of Training Verbatims per Code (ATC)</td>
<td>High</td>
<td>Small</td>
</tr>
<tr>
<td>Average Verbatim Length (AVL)</td>
<td>Small</td>
<td>High</td>
</tr>
<tr>
<td>Human Coder Agreement</td>
<td>High</td>
<td>Small</td>
</tr>
</tbody>
</table>
How good can be VCS expected to be on a new dataset?

We have experimentally observed that the $F_1$ of VCS tends to increase with

- the average number of training verbatims per code (ATC) provided to the system
- the degree of “linguistic regularity” (LR) in the training verbatims;
- how uncontroversial the code is, which can be measured by intercoder agreement. On the Egg datasets VCS was roughly 85% as good as expert human coders.

<table>
<thead>
<tr>
<th></th>
<th>Easier</th>
<th>Harder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average # of Training Verbatims per Code (ATC)</td>
<td>High</td>
<td>Small</td>
</tr>
<tr>
<td>Average Verbatim Length (AVL)</td>
<td>Small</td>
<td>High</td>
</tr>
<tr>
<td>Human Coder Agreement</td>
<td>High</td>
<td>Small</td>
</tr>
</tbody>
</table>
Outline

1. Introduction

2. VCS: an automated Verbatim Coding System

3. VCS: Effectiveness Tests
   - Effectiveness at the individual level
   - Effectiveness at the aggregate level

4. VCS: Efficiency Tests

5. The future
The PD measure

- We measure effectiveness at the aggregate level by PD, the discrepancy between the true percentage and the predicted percentage of respondents belonging to code C; the perfect system has $PD = 0$.

- For each experiment, we compute both the maximum value and the average value of PD across the codes in the same codeframe.

- How good is a given value of PD, again, should be assessed wrt an intercoder agreement study.
The PD measure

- We measure effectiveness at the aggregate level by $PD$, the discrepancy between the true percentage and the predicted percentage of respondents belonging to code $C$; the perfect system has $PD = 0$.

- For each experiment, we compute both the maximum value and the average value of PD across the codes in the same codeframe.

- How good is a given value of PD, again, should be assessed wrt an intercoder agreement study.
The PD measure

- We measure effectiveness at the aggregate level by PD, the discrepancy between the true percentage and the predicted percentage of respondents belonging to code $C$; the perfect system has $PD = 0$.
- For each experiment, we compute both the maximum value and the average value of PD across the codes in the same codeframe.
- How good is a given value of PD, again, should be assessed wrt an intercoder agreement study.
### Effectiveness at the aggregate level

| DS   | \(|C|\) | \(F_1^\mu\) | AvgPD | MaxPD |
|------|--------|-------------|-------|-------|
| LL-A | 18     | .92         | .008  | .040  |
| LL-B | 34     | .90         | .006  | .048  |
| LL-C | 20     | .89         | .007  | .074  |
| LL-D | 27     | .85         | .008  | .056  |
| LL-E | 39     | .84         | .004  | .025  |
| LL-F | 57     | .82         | .007  | .048  |
| LL-G | 104    | .80         | .005  | .052  |
| LL-H | 86     | .79         | .007  | .057  |
| LL-I | 69     | .78         | .008  | .052  |
| LL-L | 65     | .75         | .010  | .096  |
| Egg-A| 14     | .63         |       |       |
| Egg-B| 20     | .60         |       |       |

*DS* refers to different datasets, \(|C|\) denotes the number of codes, \(F_1^\mu\) is the F1 score, AvgPD and MaxPD represent the average and maximum precision drops, respectively.
Example: the LL-E dataset
Why is VCS so good at the aggregate level?

- VCS excels at the aggregate level because it explicitly tries to maximize $F_1$ ...
  - ... and to maximize $F_1$ you need to balance precision and recall ...
  - ... and to balance precision and recall you must balance false positives and false negatives ...
  - ... and if $FP = FN$, then $PD = 0$!

- Contrary to VCS, human coders often have high PD wrt each other, since it is typically the case than one coder may be consistently more liberal (or conservative) than the other.

- On the Egg tests, at the aggregate level VCS proved to be superior to expert human coders!
Why is VCS so good at the aggregate level?

- VCS excels at the aggregate level because it explicitly tries to maximize $F_1$ ...
  - ... and to maximize $F_1$ you need to balance precision and recall ...
  - ... and to balance precision and recall you must balance false positives and false negatives ...
  - ... and if $FP = FN$, then $PD = 0$!

- Contrary to VCS, human coders often have high PD wrt each other, since it is typically the case than one coder may be consistently more liberal (or conservative) than the other.

- On the Egg tests, at the aggregate level VCS proved to be superior to expert human coders!
VCS excels at the aggregate level because it explicitly tries to maximize $F_1$ ...

- ... and to maximize $F_1$ you need to balance precision and recall ...
- ... and to balance precision and recall you must balance false positives and false negatives ...
- ... and if $FP = FN$, then $PD = 0$!

Contrary to VCS, human coders often have high PD wrt each other, since it is typically the case than one coder may be consistently more liberal (or conservative) than the other.

On the Egg tests, at the aggregate level VCS proved to be superior to expert human coders!
Why is VCS so good at the aggregate level?

- VCS excels at the aggregate level because it explicitly tries to maximize $F_1$ ...
  - ... and to maximize $F_1$ you need to balance precision and recall ...
  - ... and to balance precision and recall you must balance false positives and false negatives ...
  - ... and if $FP = FN$, then $PD = 0$!

- Contrary to VCS, human coders often have high PD wrt each other, since it is typically the case than one coder may be consistently more liberal (or conservative) than the other.

- On the Egg tests, at the aggregate level VCS proved to be superior to expert human coders!
Why is VCS so good at the aggregate level?

- VCS excels at the aggregate level because it explicitly tries to maximize $F_1$ ...
  - ... and to maximize $F_1$ you need to balance precision and recall ...
  - ... and to balance precision and recall you must balance false positives and false negatives ...
  - ... and if $FP = FN$, then $PD = 0$!

- Contrary to VCS, human coders often have high PD wrt each other, since it is typically the case than one coder may be consistently more liberal (or conservative) than the other.
  - On the Egg tests, at the aggregate level VCS proved to be superior to expert human coders!
Why is VCS so good at the aggregate level?

- VCS excels at the aggregate level because it explicitly tries to maximize $F_1$ ...
  - ... and to maximize $F_1$ you need to balance precision and recall ...
  - ... and to balance precision and recall you must balance false positives and false negatives ...
  - ... and if $FP = FN$, then $PD = 0$!

- Contrary to VCS, human coders often have high PD wrt each other, since it is typically the case than one coder may be consistently more liberal (or conservative) than the other.

- On the Egg tests, at the aggregate level VCS proved to be superior to expert human coders!
Outline

1. Introduction

2. VCS: an automated Verbatim Coding System

3. VCS: Effectiveness Tests
   - Effectiveness at the individual level
   - Effectiveness at the aggregate level

4. VCS: Efficiency Tests

5. The future
There are two sides to efficiency in VCS:

- **Training-time efficiency**: how fast can the automated classifiers for a given codeframe be generated from a given set of training verbatims?
- **Coding-time efficiency**: how fast can the classifiers generated for a given codeframe code new, yet uncoded data?

Our tests on Egg data indicate that, for a 20-code codeframe:

- The classifiers can be generated from 1000 training examples in approximately 2 minutes altogether;
- 100,000 verbatims can be coded automatically in approximately 8 minutes.

In our tests on Language Logic data both training and coding were, on average, approximately 7.6 times faster than on Egg data (due to higher “linguistic regularity”).

Training time (resp., coding time) increases linearly with number of training verbatims (resp., number of verbatims to code), number of codes in the codeframe, and decreases linearly with degree of linguistic regularity.
There are two sides to efficiency in VCS:

- **Training-time efficiency**: how fast can the automated classifiers for a given codeframe be generated from a given set of training verbatims?
- **Coding-time efficiency**: how fast can the classifiers generated for a given codeframe code new, yet uncoded data?

Our tests on Egg data indicate that, for a 20-code codeframe

- The classifiers can be generated from 1000 training examples in approximately 2 minutes altogether;
- 100,000 verbatims can be coded automatically in approximately 8 minutes.

In our tests on Language Logic data both training and coding were, on average, approximately 7.6 times faster than on Egg data (due to higher “linguistic regularity”).

Training time (resp., coding time) increases linearly with number of training verbatims (resp., number of verbatims to code), number of codes in the codeframe, and decreases linearly with degree of linguistic regularity.
There are two sides to efficiency in VCS:

- **Training-time efficiency**: how fast can the automated classifiers for a given codeframe be generated from a given set of training verbatims?
- **Coding-time efficiency**: how fast can the classifiers generated for a given codeframe code new, yet uncoded data?

Our tests on Egg data indicate that, for a 20-code codeframe

- The classifiers can be generated from 1000 training examples in approximately 2 minutes altogether;
- 100,000 verbatims can be coded automatically in approximately 8 minutes.

In our tests on Language Logic data both training and coding were, on average, approximately 7.6 times faster than on Egg data (due to higher “linguistic regularity”).

Training time (resp., coding time) increases linearly with number of training verbatims (resp., number of verbatims to code), number of codes in the codeframe, and decreases linearly with degree of linguistic regularity.
There are two sides to efficiency in VCS:

- **Training-time efficiency**: how fast can the automated classifiers for a given codeframe be generated from a given set of training verbatims?
- **Coding-time efficiency**: how fast can the classifiers generated for a given codeframe code new, yet uncoded data?

Our tests on Egg data indicate that, for a 20-code codeframe

- The classifiers can be generated from 1000 training examples in approximately 2 minutes altogether;
- 100,000 verbatims can be coded automatically in approximately 8 minutes.

In our tests on Language Logic data both training and coding were, on average, approximately 7.6 times faster than on Egg data (due to higher “linguistic regularity”).

Training time (resp., coding time) increases linearly with number of training verbatims (resp., number of verbatims to code), number of codes in the codeframe, and decreases linearly with degree of linguistic regularity.
There are two sides to efficiency in VCS:

- **Training-time efficiency**: how fast can the automated classifiers for a given codeframe be generated from a given set of training verbatims?
- **Coding-time efficiency**: how fast can the classifiers generated for a given codeframe code new, yet uncoded data?

Our tests on Egg data indicate that, for a 20-code codeframe:

- The classifiers can be generated from 1000 training examples in approximately 2 minutes altogether;
- 100,000 verbatims can be coded automatically in approximately 8 minutes.

In our tests on Language Logic data both training and coding were, on average, approximately 7.6 times faster than on Egg data (due to higher “linguistic regularity”).

Training time (resp., coding time) increases linearly with number of training verbatims (resp., number of verbatims to code), number of codes in the codeframe, and decreases linearly with degree of linguistic regularity.
There are two sides to efficiency in VCS:

- **Training-time efficiency**: how fast can the automated classifiers for a given codeframe be generated from a given set of training verbatims?
- **Coding-time efficiency**: how fast can the classifiers generated for a given codeframe code new, yet uncoded data?

Our tests on Egg data indicate that, for a 20-code codeframe:

- The classifiers can be generated from 1000 training examples in approximately 2 minutes altogether;
- 100,000 verbatims can be coded automatically in approximately 8 minutes.

In our tests on Language Logic data both training and coding were, on average, approximately 7.6 times faster than on Egg data (due to higher “linguistic regularity”).

Training time (resp., coding time) increases linearly with number of training verbatims (resp., number of verbatims to code), number of codes in the codeframe, and decreases linearly with degree of linguistic regularity.
Outline

1. Introduction
2. VCS: an automated Verbatim Coding System
3. VCS: Effectiveness Tests
   - Effectiveness at the individual level
   - Effectiveness at the aggregate level
4. VCS: Efficiency Tests
5. The future
VCS soon to be integrated into Ascribe\textsuperscript{TM} V6, likely by early Spring 2008;

New features we are going to introduce, possibly in Release 1.0:

- Higher robustness to typos
- Sophisticated control panel, for answering the questions
  - What $F_1$ / PD can I can expect on this codeframe, given the amount of training data I have provided?
  - How has the estimated $F_1$ / PD on this codeframe improved as a result of my recently added 50 training verbatims?

New features we are going to introduce, possibly in Release 2.0:

- Support for languages other than English. Priorities will be French, Spanish, Italian, German, Japanese, Chinese, Russian, Portuguese.
- (Support for semi-automatic codeframe generation).
VCS soon to be integrated into Ascribe™ V6, likely by early Spring 2008;

New features we are going to introduce, possibly in Release 1.0:

- Higher robustness to typos
- Sophisticated control panel, for answering the questions
  - What $F_1$ / PD can I expect on this codeframe, given the amount of training data I have provided?
  - How has the estimated $F_1$ / PD on this codeframe improved as a result of my recently added 50 training verbatims?

New features we are going to introduce, possibly in Release 2.0:

- Support for languages other than English. Priorities will be French, Spanish, Italian, German, Japanese, Chinese, Russian, Portuguese.
- (Support for semi-automatic codeframe generation).
- VCS soon to be integrated into Ascribe™ V6, likely by early Spring 2008;
- New features we are going to introduce, possibly in Release 1.0:
  - Higher robustness to typos
  - Sophisticated control panel, for answering the questions
    - What $F_1$ / PD can I can expect on this codeframe, given the amount of training data I have provided?
    - How has the estimated $F_1$ / PD on this codeframe improved as a result of my recently added 50 training verbatims?
- New features we are going to introduce, possibly in Release 2.0:
  - Support for languages other than English. Priorities will be French, Spanish, Italian, German, Japanese, Chinese, Russian, Portuguese.
  - (Support for semi-automatic codeframe generation).
VCS soon to be integrated into Ascribe\textsuperscript{TM} V6, likely by early Spring 2008;

New features we are going to introduce, possibly in Release 1.0:

- Higher robustness to typos
- Sophisticated control panel, for answering the questions
  - What $F_1$ / PD can I can expect on this codeframe, given the amount of training data I have provided?
  - How has the estimated $F_1$ / PD on this codeframe improved as a result of my recently added 50 training verbatims?

New features we are going to introduce, possibly in Release 2.0:

- Support for languages other than English. Priorities will be French, Spanish, Italian, German, Japanese, Chinese, Russian, Portuguese.
- (Support for semi-automatic codeframe generation).
- VCS soon to be integrated into Ascribe\textsuperscript{TM} V6, likely by early Spring 2008;
- New features we are going to introduce, possibly in Release 1.0:
  - Higher robustness to typos
  - Sophisticated control panel, for answering the questions
    - What $F_1$ / PD can I can expect on this codeframe, given the amount of training data I have provided?
    - How has the estimated $F_1$ / PD on this codeframe improved as a result of my recently added 50 training verbatims?
- New features we are going to introduce, possibly in Release 2.0:
  - Support for languages other than English. Priorities will be French, Spanish, Italian, German, Japanese, Chinese, Russian, Portuguese.
  - (Support for semi-automatic codeframe generation).
VCS soon to be integrated into Ascribe<sup>TM</sup> V6, likely by early Spring 2008;

New features we are going to introduce, possibly in Release 1.0:

- Higher robustness to typos
- Sophisticated control panel, for answering the questions
  - What $F_1$ / PD can I can expect on this codeframe, given the amount of training data I have provided?
  - How has the estimated $F_1$ / PD on this codeframe improved as a result of my recently added 50 training verbatims?

New features we are going to introduce, possibly in Release 2.0:

- Support for languages other than English. Priorities will be French, Spanish, Italian, German, Japanese, Chinese, Russian, Portuguese.
- (Support for semi-automatic codeframe generation).
Automated Verbatim Coding: State of the Art and Future Perspectives

Fabrizio Sebastiani
http://www.isti.cnr.it/People/F.Sebastiani/

Istituto di Scienza e Tecnologie dell’Informazione
Consiglio Nazionale delle Ricerche
Via Giuseppe Moruzzi, 1 – 56124 Pisa, Italy
E-mail: fabrizio.sebastiani@isti.cnr.it

2007 Language Logic Verbatim Management Conference
Cincinnati, OH – October 3-4, 2007