Machines that Learn how to Code Open-Ended Survey Data

Underlying Principles, Experimental Data, and Methodological Issues

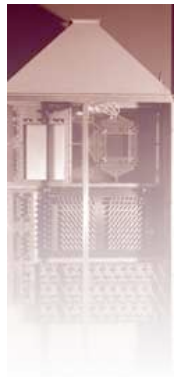
Fabrizio Sebastiani
http://www.isti.cnr.it/People/F.Sebastiani/

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

Conference on Optimal Methods of Coding Open-Ended Survey Data
Ann Arbor, Michigan – December 4-5, 2008

Outline

Outline

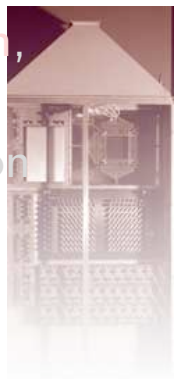


VCS, an Automated Verbatim Coding System

- At ISTI-CNR in the last 4 years we have developed an automated Verbatim Coding System (VCS), described in the paper
  - Tim Macer, Mark Pearson and Fabrizio Sebastiani, **Cracking the Code: What Customers Say, in their Own Words**, Proceedings of the 50th Annual Conference of the Market Research Society (MRS'07), Brighton, UK, March 2007.

which has met with considerable success ...

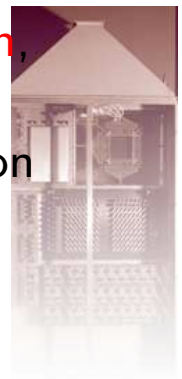
- Winner of the "2007 Best New Thinking Award", Market Research Society, London, UK;
- Winner of the "2006 Amerigo Vespucci Award" for Market Research, Italian Industralists Association;
- Nominated for the 2007 Technology Effectiveness Award, Association for Survey Computing, London, UK;
- Nominated for "Best Paper Award", MRS'07 Conference;



VCS, an Automated Verbatim Coding System

- At ISTI-CNR in the last 4 years we have developed an automated Verbatim Coding System (VCS), described in the paper
  - Tim Macer, Mark Pearson and Fabrizio Sebastiani, **Cracking the Code: What Customers Say, in their Own Words**, Proceedings of the 50th Annual Conference of the Market Research Society (MRS'07), Brighton, UK, March 2007.

  which has met with considerable success ...
  - Winner of the "2007 Best New Thinking Award", Market Research Society, London, UK;
  - Winner of the "2006 Amerigo Vespucci Award" for Market Research, Italian Industralists Association;
  - Nominated for the 2007 Technology Effectiveness Award, Association for Survey Computing, London, UK;
  - Nominated for "Best Paper Award", MRS'07 Conference;



VCS (cont'd)

- A successor of a much more primitive system described in
  - Daniela Giorgetti and Fabrizio Sebastiani, **Automating Survey Coding by Multiclass Text Categorization Techniques**, Journal of the American Society for Information Science and Technology, 54(14): 1269–1277, 2003.
- Originally developed for Egg plc, the largest purely online bank in the world (now part of Citigroup);
  - Deployed in July 2006;
  - Now fully operational and managing all of Egg's customer satisfaction verbatim data ($\approx$ 20,000 questionnaires per month, plus huge backlogs).
- From early 2008, available from within the Ascribe^{TM} platform, by Language Logic LLC, with coverage for five major European languages.

VCS (cont'd)

- A successor of a much more primitive system described in
  - Daniela Giorgetti and Fabrizio Sebastiani, **Automating Survey Coding by Multiclass Text Categorization Techniques**, Journal of the American Society for Information Science and Technology, 54(14): 1269–1277, 2003.
- Originally developed for Egg plc, the largest purely online bank in the world (now part of Citigroup);
  - Deployed in July 2006;
  - Now fully operational and managing all of Egg's customer satisfaction verbatim data ($\approx$ 20,000 questionnaires per month, plus huge backlogs).
- From early 2008, available from within the Ascribe^{TM} platform, by Language Logic LLC, with coverage for five major European languages.

VCS (cont'd)

- A successor of a much more primitive system described in
  - Daniela Giorgetti and Fabrizio Sebastiani, **Automating Survey Coding by Multiclass Text Categorization Techniques**, Journal of the American Society for Information Science and Technology, 54(14): 1269–1277, 2003.
- Originally developed for Egg plc, the largest purely online bank in the world (now part of Citigroup);
  - Deployed in July 2006;
  - Now fully operational and managing all of Egg's customer satisfaction verbatim data ($\approx$ 20,000 questionnaires per month, plus huge backlogs).
- From early 2008, available from within the Ascribe$^{\text{TM}}$ platform, by Language Logic LLC, with coverage for five major European languages.

VCS: the underlying philosophy

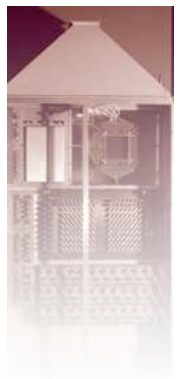
- VCS is an adaptive system for automatically coding verbatim responses under any user-specified codeframe (aka "codebook"); given such a codeframe, VCS automatically generates an automatic coder for this codeframe.
- Applications include coding open-ended responses in customer satisfaction analysis, market research, and social/political studies
- 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 coder, 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 coder for this codeframe.
- Applications include coding open-ended responses in customer satisfaction analysis, market research, and social/political studies
- 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 coder, i.e., a system that decides whether a given verbatim should or should not be attributed the code.

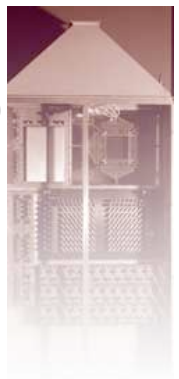
VCS: the underlying philosophy

- 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 coder for this codeframe.
- Applications include coding open-ended responses in customer satisfaction analysis, market research, and social/political studies
- 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 coder, i.e., a system that decides whether a given verbatim should or should not be attributed the code.



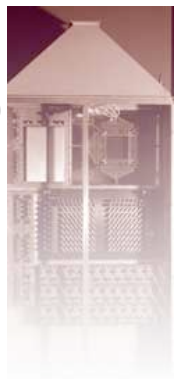
VCS: the underlying philosophy (cont'd)

- 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;
  - Coding is a subjective task, and VCS learns to replicate the subjective behaviour of the human coder who has coded the training examples;
- 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.



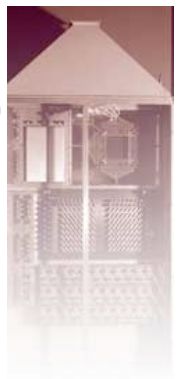
VCS: the underlying philosophy (cont'd)

- 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;
  - Coding is a subjective task, and VCS learns to replicate the subjective behaviour of the human coder who has coded the training examples;
- 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.



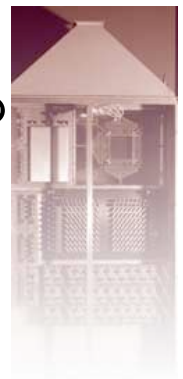
VCS: the underlying philosophy (cont'd)

- 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;
  - Coding is a subjective task, and VCS learns to replicate the subjective behaviour of the human coder who has coded the training examples;
- 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;
  - Coding is a subjective task, and VCS learns to replicate the subjective behaviour of the human coder who has coded the training examples;
- 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.



Training Phase. Teacher: "This is a tiger!"

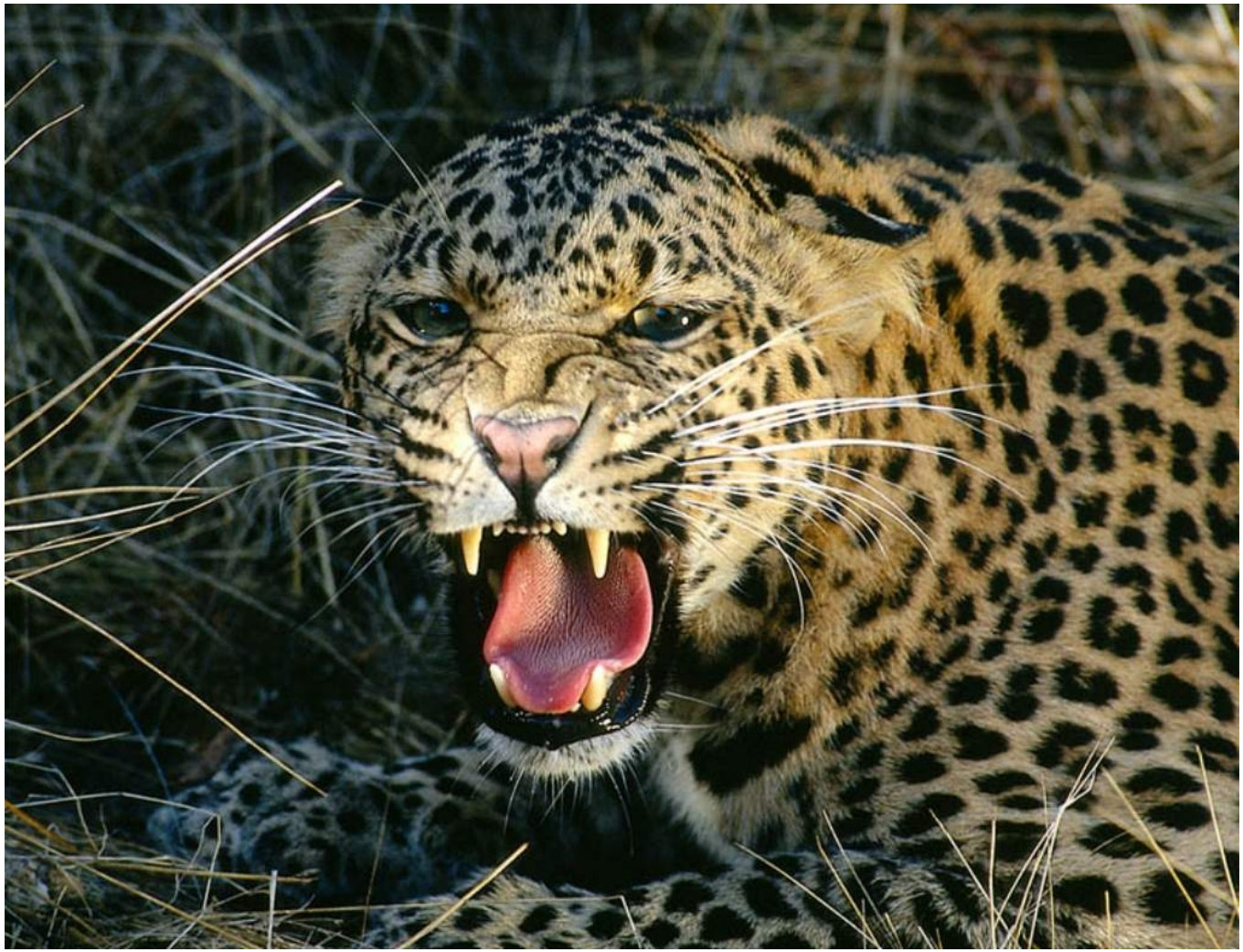
Training Phase. Teacher: "This is another tiger!"

Training Phase. Teacher: "This is yet another tiger!"

Training Phase. Teacher: "Also a tiger!"

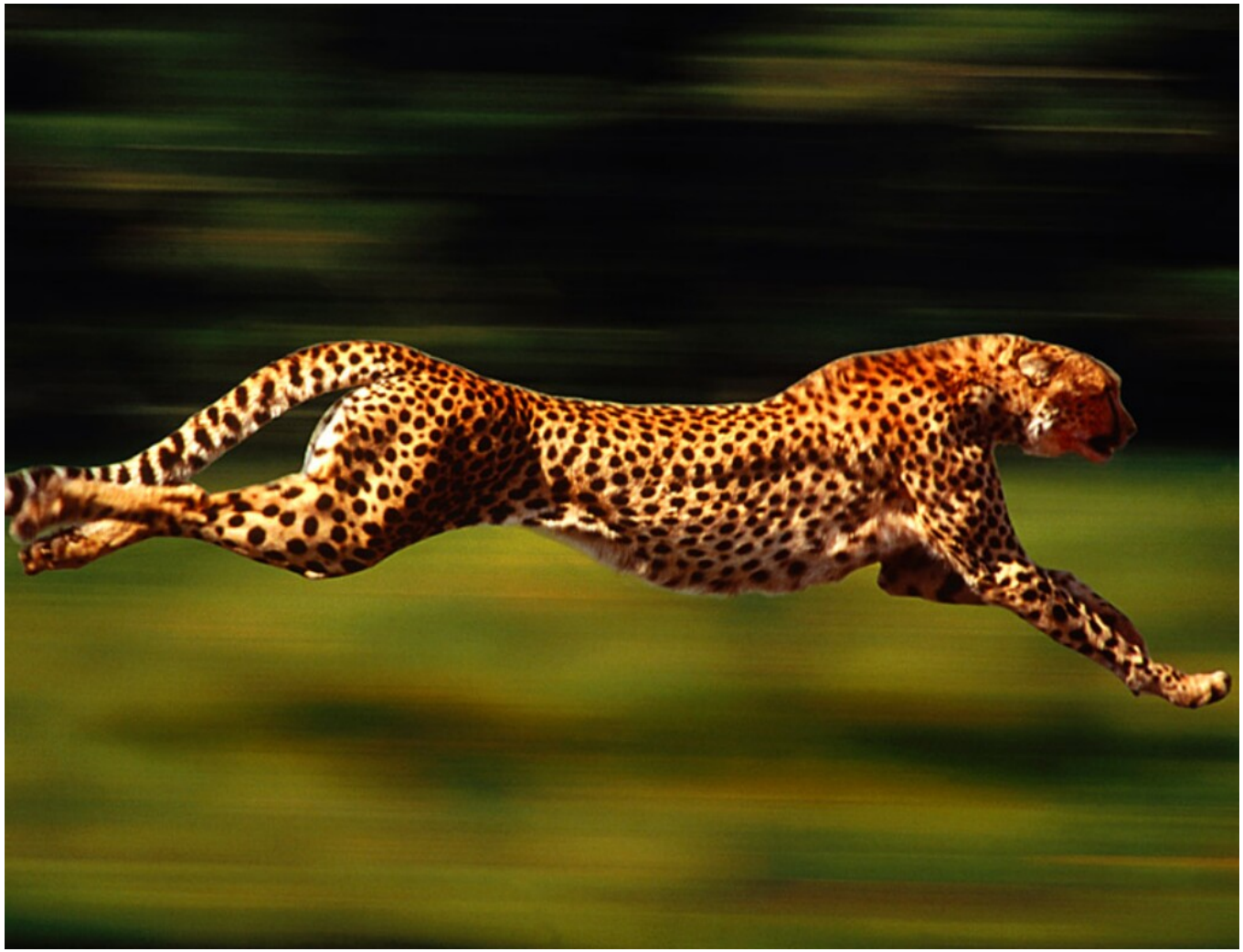
Training Phase. Teacher: "This is a NOT a tiger!"

Training Phase. Teacher: "NOT a tiger either!"



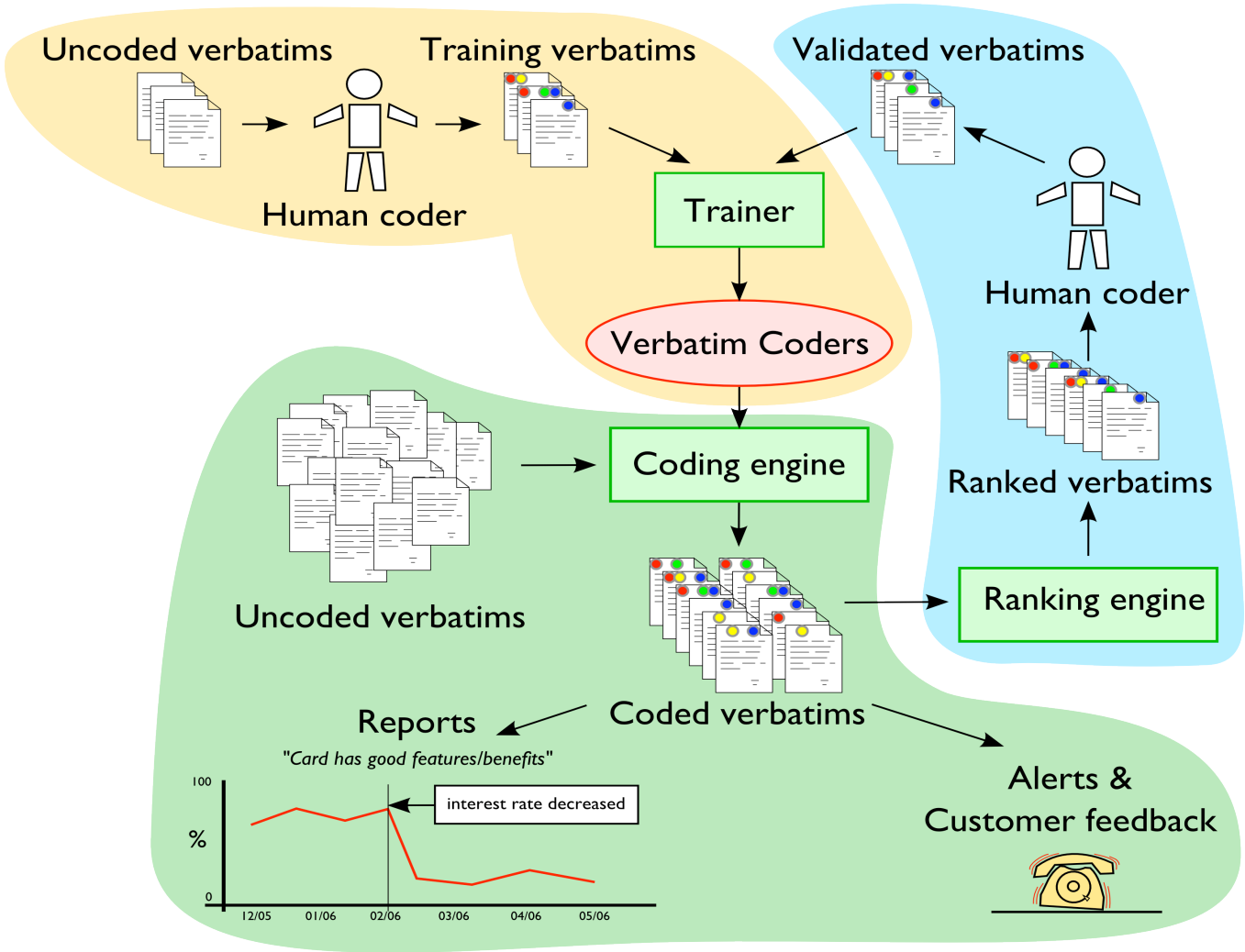
Training Phase. Teacher: "Absolutely NOT a tiger!"

Testing ("Coding") Phase. Teacher: "Is this a tiger?"



Proactive Training phase. Learner: "Is this a tiger?"

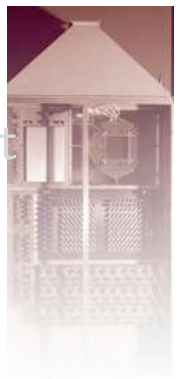
The VCS information flow



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
  - 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 domain-dependent resource (e.g, thesauri), it's a "plug and play" system;
- Pretty good accuracy at the "individual level", excellent accuracy at the "aggregate level", excellent learning and coding speed.



Advantages of learning metaphor

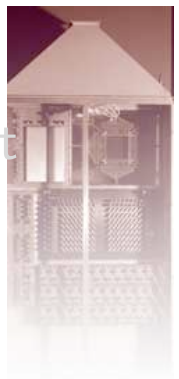
- No need for expert to write coding rules in arcane language; the system only needs user-coded examples for training;
- Easy update to
  - revised codeframe
  - brand new codeframe or brand new survey

  since the system only needs user-coded examples for training that reflect the new situation;
- Does not use any domain-dependent resource (e.g, thesauri), it's a "plug and play" system;
- Pretty good accuracy at the "individual level", excellent accuracy 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
  - revised codeframe
  - brand new codeframe or brand new survey

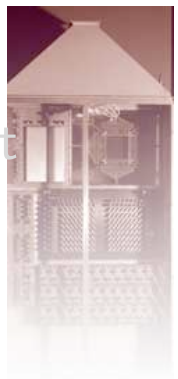
  since the system only needs user-coded examples for training that reflect the new situation;
- Does not use any domain-dependent resource (e.g, thesauri), it's a "plug and play" system;
- Pretty good accuracy at the "individual level", excellent accuracy at the "aggregate level", excellent learning and 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
  - revised codeframe
  - brand new codeframe or brand new survey

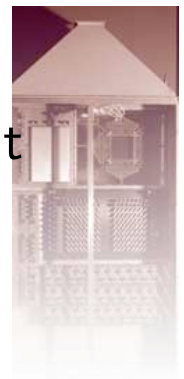
  since the system only needs user-coded examples for training that reflect the new situation;
- Does not use any domain-dependent resource (e.g, thesauri), it's a "plug and play" system;
- Pretty good accuracy at the "individual level", excellent accuracy at the "aggregate level", excellent learning and coding speed.



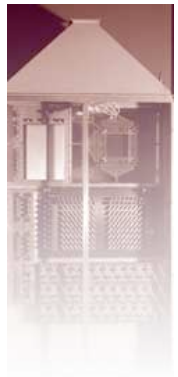
Advantages of learning metaphor

- No need for expert to write coding rules in arcane language; the system only needs user-coded examples for training;
- Easy update to
  - 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 domain-dependent resource (e.g, thesauri), it's a "plug and play" system;
- Pretty good accuracy at the "individual level", excellent accuracy at the "aggregate level", excellent learning and coding speed.



Outline



Testing accuracy

- 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 the coder who has coded the training verbatims would make.
- We estimate the accuracy of a coding system by comparing the system's coding decisions with those of the human coder on one or more test datasets (each consisting of a set of manually coded verbatims plus the corresponding codeframe).

Testing accuracy

- 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 the coder who has coded the training verbatims would make.
- We estimate the accuracy of a coding system by comparing the system's coding decisions with those of the human coder on one or more test datasets (each consisting of a set of manually coded verbatims plus the corresponding codeframe).

Accuracy: individual or aggregate?

- Accuracy may be measured at two different levels:
  - At the individual level (more strict): the perfect system is the one which, for a code C, assigns C to the verbatim exactly when the human coder would have assigned C.
  - At the aggregate level (more lenient): the perfect system is the one which, for a code C, assigns $x\%$ of the verbatims to C exactly when the human 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!

Accuracy: individual or aggregate?

- Accuracy may be measured at two different levels:
  - At the individual level (more strict): the perfect system is the one which, for a code C, assigns C to the verbatim exactly when the human coder would have assigned C.
  - At the aggregate level (more lenient): the perfect system is the one which, for a code C, assigns $x\%$ of the verbatims to C exactly when the human 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!

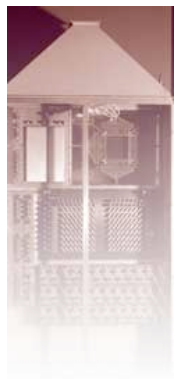
Accuracy: individual or aggregate?

- Accuracy may be measured at two different levels:
  - At the individual level (more strict): the perfect system is the one which, for a code C, assigns C to the verbatim exactly when the human coder would have assigned C.
  - At the aggregate level (more lenient): the perfect system is the one which, for a code C, assigns $x\%$ of the verbatims to C exactly when the human 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!

Accuracy: individual or aggregate?

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



Precision and Recall

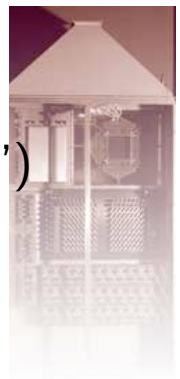
- Accuracy testing requires an accuracy measure to be defined and agreed upon. The one we adopt, called F_1, relies on the following two notions:
  - For a given code C, precision (denoted π) measures the ability of the system to avoid "overcoding", i.e., attributing C when it should not be attributed; 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 ρ) 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.

Precision and Recall

- Accuracy testing requires an accuracy 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 π) 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 ρ) 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.

Precision and Recall

- Accuracy testing requires an accuracy 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 π) 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 ρ) 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:

Code C		coder says **YES**	coder says **NO**
system says	**YES**	TP	FP
	NO	FN	TN

- Precision is defined as $\pi = \frac{TP}{TP + FP}$
- Recall is defined as $\rho = \frac{TP}{TP + FN}$
- The accuracy 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:

Code C		coder says **YES**	coder says **NO**
system says	**YES**	TP	FP
	NO	FN	TN

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

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

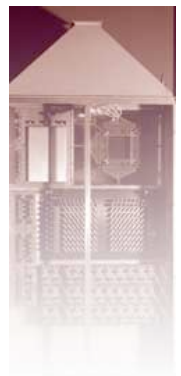
The F_1 measure

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

Code C		coder says **YES**	coder says **NO**
system says	**YES**	TP	FP
	NO	FN	TN

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

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



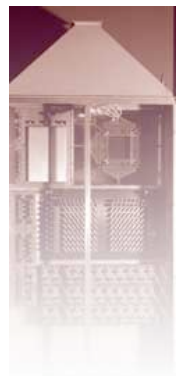
The F_1 measure

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

Code C		coder says **YES**	coder says **NO**
system says	**YES**	TP	FP
	NO	FN	TN

- Precision is defined as $\pi = \dfrac{TP}{TP + FP}$
- Recall is defined as $\rho = \dfrac{TP}{TP + FN}$
- The accuracy 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 accuracy on an example dataset

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

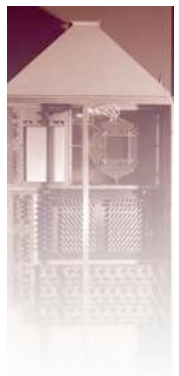
Code C_i		coder says	
		YES	**NO**
system says	**YES**	15	7
	NO	8	70

Code C_j		coder says	
		YES	**NO**
system says	**YES**	22	13
	NO	5	60

$$\pi = \frac{15}{15+7} = \frac{15}{22} = .682 \qquad \pi_j = \frac{22}{22+13} = \frac{22}{35} = .629$$

$$\rho = \frac{15}{15+8} = \frac{15}{23} = .652 \qquad \rho_j = \frac{22}{22+5} = \frac{22}{27} = .815$$

$$F_1 = \frac{2 \cdot .682 \cdot .652}{.682 + .652} = .667 \qquad F_1 = \frac{2 \cdot .629 \cdot .815}{.629 + .815} = .710$$



Testing accuracy on an example dataset

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

Code C_i		coder says	
		YES	**NO**
system says	**YES**	15	7
	NO	8	70

Code C_j		coder says	
		YES	**NO**
system says	**YES**	22	13
	NO	5	60

$$\pi = \frac{15}{15+7} = \frac{15}{22} = .682 \qquad \pi_j = \frac{22}{22+13} = \frac{22}{35} = .629$$

$$\rho = \frac{15}{15+8} = \frac{15}{23} = .652 \qquad \rho_j = \frac{22}{22+5} = \frac{22}{27} = .815$$

$$F_1 = \frac{2 \cdot .682 \cdot .652}{.682 + .652} = .667 \qquad F_1 = \frac{2 \cdot .629 \cdot .815}{.629 + .815} = .710$$

Computing accuracy wrt an entire codeframe

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

Codes C_i and C_j		coder says YES	coder says NO
system says	YES	$15 + 22$	$7 + 13$
	NO	$8 + 5$	$70 + 60$

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

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

$$F_1^{\mu} = \frac{2 \cdot .649 \cdot .740}{.649 + .740} = .692$$

Computing accuracy wrt an entire codeframe

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

Codes C_i and C_j		coder says **YES**	coder says **NO**
system says	**YES**	$15 + 22$	$7 + 13$
	NO	$8 + 5$	$70 + 60$

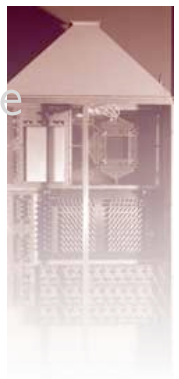
$$\pi^{\mu} = \frac{(15+22)}{(15+22)+(7+13)} = \frac{37}{57} = .649$$

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

$$F_1^{\mu} = \frac{2 \cdot .649 \cdot .740}{.649 + .740} = .692$$

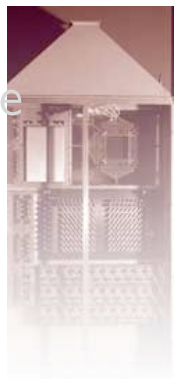
Why is F_1 a good measure of accuracy?

- $F_1 = 0$ 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.



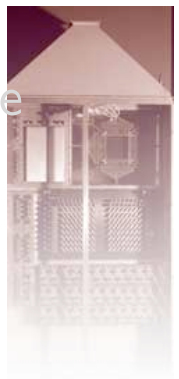
Why is F_1 a good measure of accuracy?

- $F_1 = 0$ 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.



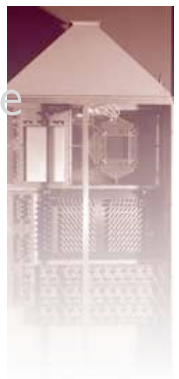
Why is F_1 a good measure of accuracy?

- $F_1 = 0$ 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 accuracy?

- $F_1 = 0$ 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 accuracy?

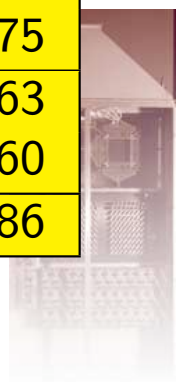
- $F_1 = 0$ for the "pervert system" ($TP = TN = 0$) and $F_1 = 1$ for the "perfect system" ($FN = FP = 0$).
- It partially rewards partial success: i.e., if the true codes of a verbatim are c_1, c_2, c_3, c_4, attributing c_1, c_2, c_3 is rewarded more than attributing c_1 only.
- It is not easy to 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 accuracy?

- $F_1 = 0$ 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

DS	$\lvert Tr \rvert$	$\lvert Te \rvert$	$\lvert C \rvert$	ATC	$\lvert D \rvert$	AVL	RAI	F_1^μ
LL-A	201	65	18	21.00	61	1.35	.344	.92
LL-B	501	10299	34	26.65	151	1.65	.176	.90
LL-C	201	425	20	10.05	60	1.61	.168	.89
LL-D	501	698	27	45.30	471	3.32	.096	.85
LL-E	201	720	39	8.41	155	2.57	.054	.84
LL-F	501	999	57	37.58	551	6.99	.068	.82
LL-G	501	1898	104	21.30	611	6.25	.035	.80
LL-H	501	699	86	30.08	817	7.87	.037	.79
LL-I	501	699	69	33.16	764	7.70	.043	.78
LL-L	501	698	65	29.40	673	5.58	.044	.75
Egg-A	700	300	14	91.14	2948	28.60	.031	.63
Egg-B	653	273	20	50.32	3620	27.60	.014	.60
ANES L/D	2664	2665	1	1396.00	4558	30.83	.306	.86

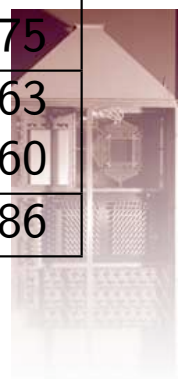


Real tests: the Language Logic data & the Egg data

DS	$\lvert Tr \rvert$	$\lvert Te \rvert$	$\lvert C \rvert$	ATC	$\lvert D \rvert$	AVL	RAI	F_1^μ
LL-A	201	65	18	21.00	61	1.35	.344	.92
LL-B	501	10299	34	26.65	151	1.65	.176	.90
LL-C	201	425	20	10.05	60	1.61	.168	.89
LL-D	501	698	27	45.30	471	3.32	.096	.85
LL-E	201	720	39	8.41	155	2.57	.054	.84
LL-F	501	999	57	37.58	551	6.99	.068	.82
LL-G	501	1898	104	21.30	611	6.25	.035	.80
LL-H	501	699	86	30.08	817	7.87	.037	.79
LL-I	501	699	69	33.16	764	7.70	.043	.78
LL-L	501	698	65	29.40	673	5.58	.044	.75
Egg-A	700	300	14	91.14	2948	28.60	.031	.63
Egg-B	653	273	20	50.32	3620	27.60	.014	.60
ANES L/D	2664	2665	1	1396.00	4558	30.83	.306	.86

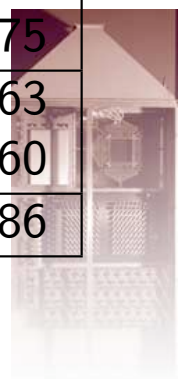
Real tests: the Language Logic data & the Egg data

DS	$\lvert Tr \rvert$	$\lvert Te \rvert$	$\lvert C \rvert$	ATC	$\lvert D \rvert$	AVL	RAI	F_1^μ
LL-A	201	65	18	21.00	61	1.35	.344	.92
LL-B	501	10299	34	26.65	151	1.65	.176	.90
LL-C	201	425	20	10.05	60	1.61	.168	.89
LL-D	501	698	27	45.30	471	3.32	.096	.85
LL-E	201	720	39	8.41	155	2.57	.054	.84
LL-F	501	999	57	37.58	551	6.99	.068	.82
LL-G	501	1898	104	21.30	611	6.25	.035	.80
LL-H	501	699	86	30.08	817	7.87	.037	.79
LL-I	501	699	69	33.16	764	7.70	.043	.78
LL-L	501	698	65	29.40	673	5.58	.044	.75
Egg-A	700	300	14	91.14	2948	28.60	.031	.63
Egg-B	653	273	20	50.32	3620	27.60	.014	.60
ANES L/D	2664	2665	1	1396.00	4558	30.83	.306	.86



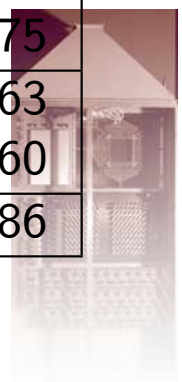
Real tests: the Language Logic data & the Egg data

DS	$\lvert Tr \rvert$	$\lvert Te \rvert$	$\lvert C \rvert$	ATC	$\lvert D \rvert$	AVL	RAI	F_1^μ
LL-A	201	65	18	21.00	61	1.35	.344	.92
LL-B	501	10299	34	26.65	151	1.65	.176	.90
LL-C	201	425	20	10.05	60	1.61	.168	.89
LL-D	501	698	27	45.30	471	3.32	.096	.85
LL-E	201	720	39	8.41	155	2.57	.054	.84
LL-F	501	999	57	37.58	551	6.99	.068	.82
LL-G	501	1898	104	21.30	611	6.25	.035	.80
LL-H	501	699	86	30.08	817	7.87	.037	.79
LL-I	501	699	69	33.16	764	7.70	.043	.78
LL-L	501	698	65	29.40	673	5.58	.044	.75
Egg-A	700	300	14	91.14	2948	28.60	.031	.63
Egg-B	653	273	20	50.32	3620	27.60	.014	.60
ANES L/D	2664	2665	1	1396.00	4558	30.83	.306	.86



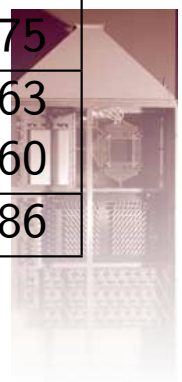
Real tests: the Language Logic data & the Egg data

DS	$\lvert Tr \rvert$	$\lvert Te \rvert$	$\lvert C \rvert$	ATC	$\lvert D \rvert$	AVL	RAI	F_1^μ
LL-A	201	65	18	21.00	61	1.35	.344	.92
LL-B	501	10299	34	26.65	151	1.65	.176	.90
LL-C	201	425	20	10.05	60	1.61	.168	.89
LL-D	501	698	27	45.30	471	3.32	.096	.85
LL-E	201	720	39	8.41	155	2.57	.054	.84
LL-F	501	999	57	37.58	551	6.99	.068	.82
LL-G	501	1898	104	21.30	611	6.25	.035	.80
LL-H	501	699	86	30.08	817	7.87	.037	.79
LL-I	501	699	69	33.16	764	7.70	.043	.78
LL-L	501	698	65	29.40	673	5.58	.044	.75
Egg-A	700	300	14	91.14	2948	28.60	.031	.63
Egg-B	653	273	20	50.32	3620	27.60	.014	.60
ANES L/D	2664	2665	1	1396.00	4558	30.83	.306	.86



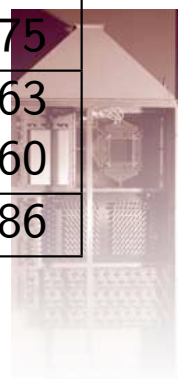
Real tests: the Language Logic data & the Egg data

DS	$\lvert Tr \rvert$	$\lvert Te \rvert$	$\lvert C \rvert$	ATC	$\lvert D \rvert$	AVL	RAI	F_1^μ
LL-A	201	65	18	21.00	61	1.35	.344	.92
LL-B	501	10299	34	26.65	151	1.65	.176	.90
LL-C	201	425	20	10.05	60	1.61	.168	.89
LL-D	501	698	27	45.30	471	3.32	.096	.85
LL-E	201	720	39	8.41	155	2.57	.054	.84
LL-F	501	999	57	37.58	551	6.99	.068	.82
LL-G	501	1898	104	21.30	611	6.25	.035	.80
LL-H	501	699	86	30.08	817	7.87	.037	.79
LL-I	501	699	69	33.16	764	7.70	.043	.78
LL-L	501	698	65	29.40	673	5.58	.044	.75
Egg-A	700	300	14	91.14	2948	28.60	.031	.63
Egg-B	653	273	20	50.32	3620	27.60	.014	.60
ANES L/D	2664	2665	1	1396.00	4558	30.83	.306	.86



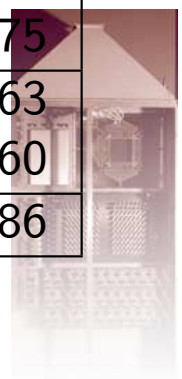
Real tests: the Language Logic data & the Egg data

DS	$\lvert Tr \rvert$	$\lvert Te \rvert$	$\lvert C \rvert$	ATC	$\lvert D \rvert$	AVL	RAI	F_1^μ
LL-A	201	65	18	21.00	61	1.35	.344	.92
LL-B	501	10299	34	26.65	151	1.65	.176	.90
LL-C	201	425	20	10.05	60	1.61	.168	.89
LL-D	501	698	27	45.30	471	3.32	.096	.85
LL-E	201	720	39	8.41	155	2.57	.054	.84
LL-F	501	999	57	37.58	551	6.99	.068	.82
LL-G	501	1898	104	21.30	611	6.25	.035	.80
LL-H	501	699	86	30.08	817	7.87	.037	.79
LL-I	501	699	69	33.16	764	7.70	.043	.78
LL-L	501	698	65	29.40	673	5.58	.044	.75
Egg-A	700	300	14	91.14	2948	28.60	.031	.63
Egg-B	653	273	20	50.32	3620	27.60	.014	.60
ANES L/D	2664	2665	1	1396.00	4558	30.83	.306	.86



Real tests: the Language Logic data & the Egg data

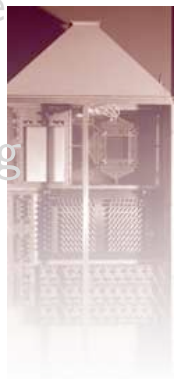
DS	$\lvert Tr \rvert$	$\lvert Te \rvert$	$\lvert C \rvert$	ATC	$\lvert D \rvert$	AVL	RAI	F_1^{μ}
LL-A	201	65	18	21.00	61	1.35	.344	.92
LL-B	501	10299	34	26.65	151	1.65	.176	.90
LL-C	201	425	20	10.05	60	1.61	.168	.89
LL-D	501	698	27	45.30	471	3.32	.096	.85
LL-E	201	720	39	8.41	155	2.57	.054	.84
LL-F	501	999	57	37.58	551	6.99	.068	.82
LL-G	501	1898	104	21.30	611	6.25	.035	.80
LL-H	501	699	86	30.08	817	7.87	.037	.79
LL-I	501	699	69	33.16	764	7.70	.043	.78
LL-L	501	698	65	29.40	673	5.58	.044	.75
Egg-A	700	300	14	91.14	2948	28.60	.031	.63
Egg-B	653	273	20	50.32	3620	27.60	.014	.60
ANES L/D	2664	2665	1	1396.00	4558	30.83	.306	.86



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?"

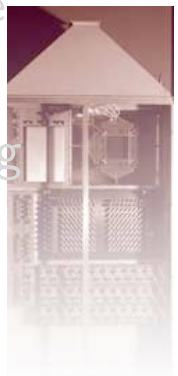
  different levels of F_1 may be expected, both by an automatic coding system and by a human coder. Code 2 is inherently more controversial than Code 1.



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?"

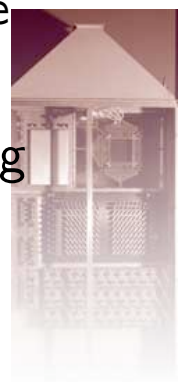
  different levels of F_1 may be expected, both by an automatic coding system and by a human coder. Code 2 is inherently more controversial than Code 1.



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 inverse of the average length of the training verbatims (1/AVL);
  - 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.

	Easier	Harder
Average # of Training Verbatims per Code (ATC)	High	Small
Average Verbatim Length (AVL)	Small	High
Human Coder Agreement	High	Small

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 inverse of the average length of the training verbatims (1/AVL);
  - 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.

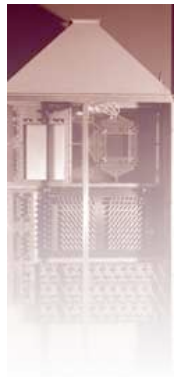
	Easier	Harder
Average # of Training Verbatims per Code (ATC)	High	Small
Average Verbatim Length (AVL)	Small	High
Human Coder Agreement	High	Small

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 inverse of the average length of the training verbatims (1/AVL);
  - 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.

	Easier	Harder
Average # of Training Verbatims per Code (ATC)	High	Small
Average Verbatim Length (AVL)	Small	High
Human Coder Agreement	High	Small

Outline



The *PD* measure

- We measure accuracy at the aggregate level by *PD*, the discrepancy between the true percentage and the predicted percentage of verbatims with 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

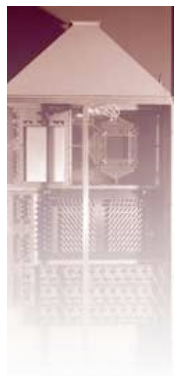
- We measure accuracy at the aggregate level by *PD*, the discrepancy between the true percentage and the predicted percentage of verbatims with code C; the perfect system has $PD = 0$.
- For each experiment, we compute both the maximum value and the average value of *PD* across the codes in the same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

The *PD* measure

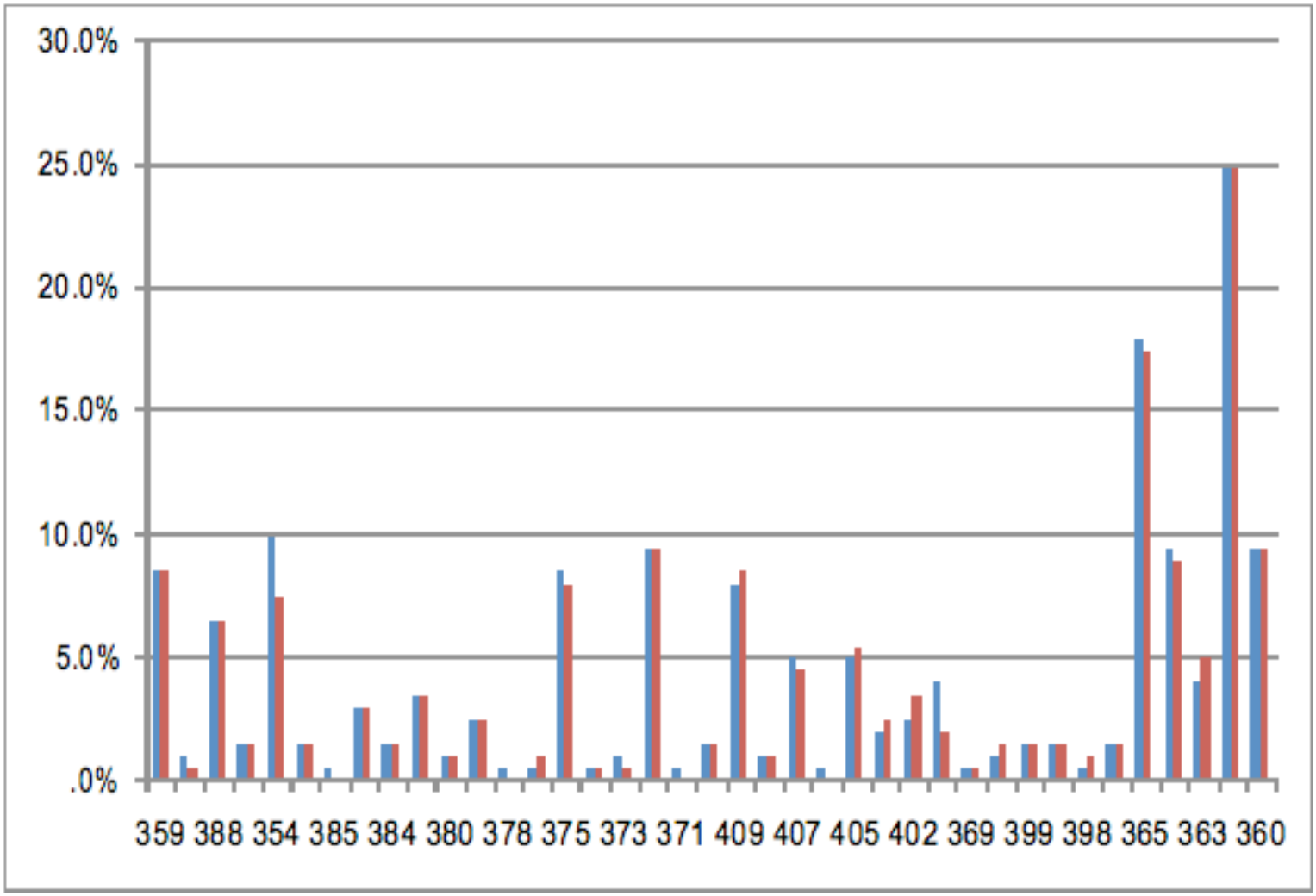
- We measure accuracy at the aggregate level by *PD*, the discrepancy between the true percentage and the predicted percentage of verbatims with code C; the perfect system has $PD = 0$.
- For each experiment, we compute both the maximum value and the average value of *PD* across the codes in the same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

Accuracy at the aggregate level

DS	$\|C\|$	F_1^μ	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	.019	.062
Egg-B	20	.60	.023	.057
ANES L/D	1	.86	.046	.046



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!

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

- There are two sides to efficiency in VCS:
  - Training-time efficiency: how fast can the automated coders for a given codeframe be generated from a given set of training verbatims?
  - Coding-time efficiency: how fast can the coders generated for a given codeframe code new, yet uncoded data?
- Our tests on Egg data indicate that, for a 20-code codeframe
  - The coders 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 lower AVL).
- 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.

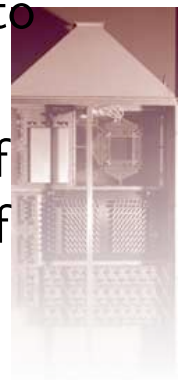
- There are two sides to efficiency in VCS:
  - Training-time efficiency: how fast can the automated coders for a given codeframe be generated from a given set of training verbatims?
  - Coding-time efficiency: how fast can the coders generated for a given codeframe code new, yet uncoded data?
- Our tests on Egg data indicate that, for a 20-code codeframe
  - The coders 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 lower AVL).
- 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 coders for a given codeframe be generated from a given set of training verbatims?
  - Coding-time efficiency: how fast can the coders generated for a given codeframe code new, yet uncoded data?
- Our tests on Egg data indicate that, for a 20-code codeframe
  - The coders 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 lower AVL).
- 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 coders for a given codeframe be generated from a given set of training verbatims?
  - Coding-time efficiency: how fast can the coders generated for a given codeframe code new, yet uncoded data?
- Our tests on Egg data indicate that, for a 20-code codeframe
  - The coders 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 lower AVL).
- 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 coders for a given codeframe be generated from a given set of training verbatims?
  - Coding-time efficiency: how fast can the coders generated for a given codeframe code new, yet uncoded data?
- Our tests on Egg data indicate that, for a 20-code codeframe
  - The coders 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 lower AVL).
- 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 coders for a given codeframe be generated from a given set of training verbatims?
  - Coding-time efficiency: how fast can the coders generated for a given codeframe code new, yet uncoded data?
- Our tests on Egg data indicate that, for a 20-code codeframe
  - The coders 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 lower AVL).
- 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

- Features I have not discussed:
  - Robustness to lexical and syntactic ill-formedness
  - 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 the last 20% of the training examples?
  - "Umbrella coding"
- Features planned for next releases:
  - Support for automatic codeframe generation.
  - Support for (training) "data cleaning".

- Features I have not discussed:
  - Robustness to lexical and syntactic ill-formedness
  - 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 the last 20% of the training examples?
  - "Umbrella coding"
- Features planned for next releases:
  - Support for automatic codeframe generation.
  - Support for (training) "data cleaning".

- Features I have not discussed:
  - Robustness to lexical and syntactic ill-formedness
  - 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 the last 20% of the training examples?
  - "Umbrella coding"
- Features planned for next releases:
  - Support for automatic codeframe generation.
  - Support for (training) "data cleaning".

- Features I have not discussed:
  - Robustness to lexical and syntactic ill-formedness
  - 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 the last 20% of the training examples?
  - "Umbrella coding"
- Features planned for next releases:
  - Support for automatic codeframe generation.
  - Support for (training) "data cleaning".

- Features I have not discussed:
  - Robustness to lexical and syntactic ill-formedness
  - 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 the last 20% of the training examples?
  - "Umbrella coding"
- Features planned for next releases:
  - Support for automatic codeframe generation.
  - Support for (training) "data cleaning".

- Features I have not discussed:
  - Robustness to lexical and syntactic ill-formedness
  - 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 the last 20% of the training examples?
  - "Umbrella coding"
- Features planned for next releases:
  - Support for automatic codeframe generation.
  - Support for (training) "data cleaning".

- Features I have not discussed:
  - Robustness to lexical and syntactic ill-formedness
  - 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 the last 20% of the training examples?
  - "Umbrella coding"
- Features planned for next releases:
  - Support for automatic codeframe generation.
  - Support for (training) "data cleaning".

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.]*

Thanks for your attention!