Automated Classification and Quantification of Verbatims via Machine Learning

Fabrizio Sebastiani
(Joint work with Andrea Esuli)

Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
56124 Pisa, Italy
E-mail: fabrizio.sebastiani@isti.cnr.it

Conference of the ASC
London, UK – September 25, 2012

Outline

Outline

Machine Learning for Automated Verbatim Coding

- In the last 10 years we have championed an approach to automatically coding open-ended answers ("verbatims") based on "machine learning";[1][2][3]
- Based on these principles we have built a software system, called VCS ("Verbatim Coding System"), which we have variously applied to coding surveys in the social sciences, customer relationship management, and market research.

[1] 2003 : D. Giorgetti and F. Sebastiani. Automating survey coding by multiclass text categorization techniques. *Journal of the American Society for Information Science and Technology*, 54(14).

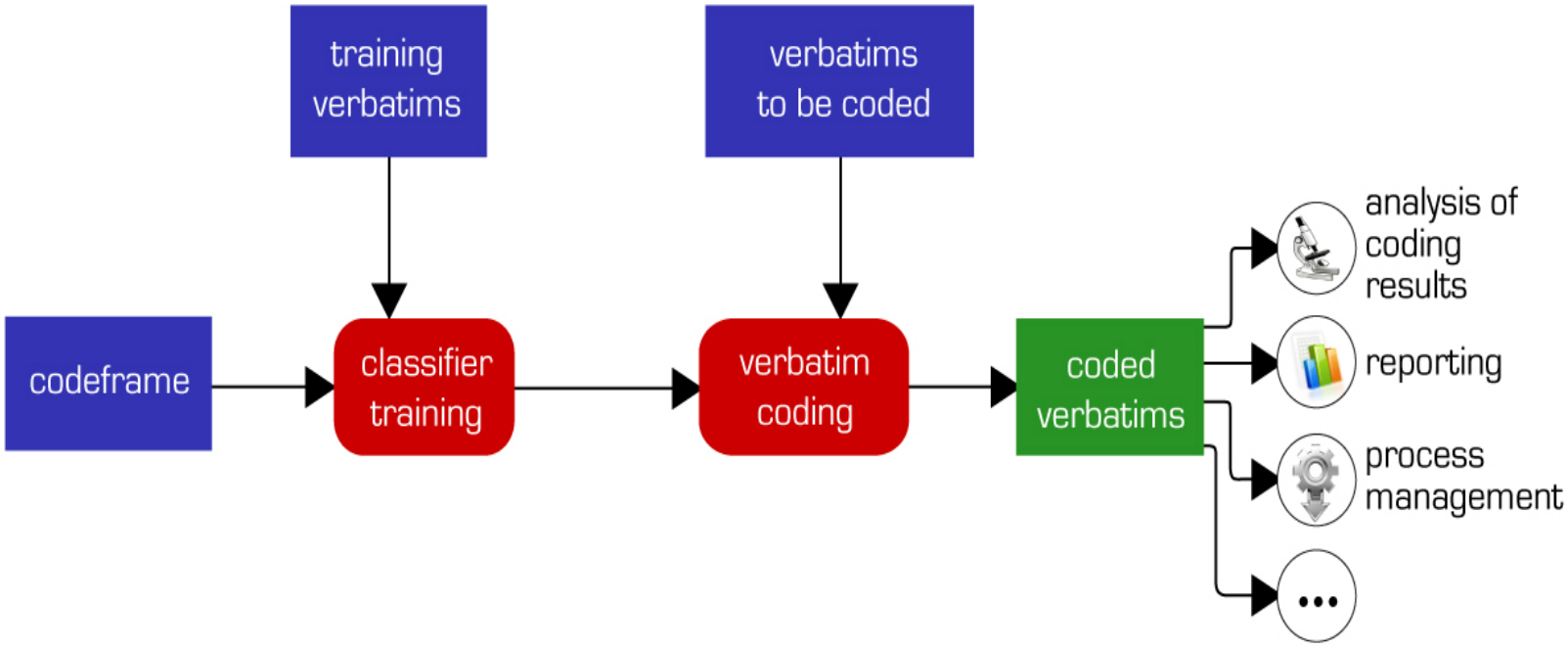
[2] 2007 : T. Macer, M. Pearson, and F. Sebastiani. Cracking the Code: What Customers Say, in Their Own Words. In *Proceedings of the 50th Annual Conference of the Market Research Society*, Brighton, UK. (Best New Thinking Award, Shortlisted for Best Paper Award and for ASC/MRS Tech Effectiveness Award)

[3] 2010 : A. Esuli and F. Sebastiani. Machines that learn how to code open-ended survey data. *International Journal of Market Research*, 52(6). (Shortlisted for best 2010 IJMR paper)

ML for Automated Verbatim Coding (cont'd)

- VCS is based on a "supervised learning" metaphor : the classifier learns (or: is trained), from sample manually classified verbatims, the characteristics a new verbatim should have in order to be attributed a given code.
- The human operator who feeds sample manually classified verbatims to the system plays the role of the "supervisor".

The VCS information flow



Advantages of learning metaphor

- No need for expert to engage in time-consuming writing (in a formal language) of coding rules; the system only needs user-classified examples for training;

  "if you can manually classify texts, you can operate VCS"

- Easy update to
  - revised codeframe
  - brand new codeframe

  since the system only needs manually classified examples for training that reflect the current codeframe;
- No need for specialized resources (e.g., dictionaries);
- Very good accuracy, excellent learning and classification speed.

Advantages of learning metaphor

- No need for expert to engage in time-consuming writing (in a formal language) of coding rules; the system only needs user-classified examples for training;

  "if you can manually classify texts, you can operate VCS"

- Easy update to
  - revised codeframe
  - brand new codeframe

  since the system only needs manually classified examples for training that reflect the current codeframe;
- No need for specialized resources (e.g., dictionaries);
- Very good accuracy, excellent learning and classification speed.

Testing Coding Accuracy

- We have tested the coding accuracy of VCS on several datasets of verbatim data. Given such a dataset
  1. we split the dataset into a training set and a test set;
  2. we train our classifier on the training set;
  3. we test the trained classifier on the test set, i.e., we compare the manually attributed codes with the codes attributed by the classifier.
- The comparison is performed by using a mathematical measure of accuracy. We use the F_1 measure, defined as

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

Mathematically, F_1 is the "harmonic mean" of precision (the ability of the system to avoid overcoding, i.e., generating false positives) and recall (the ability of the system to avoid undercoding, i.e., generating false negatives).

Testing Coding Accuracy

- We have tested the coding accuracy of VCS on several datasets of verbatim data. Given such a dataset
  1. we split the dataset into a training set and a test set;
  2. we train our classifier on the training set;
  3. we test the trained classifier on the test set, i.e., we compare the manually attributed codes with the codes attributed by the classifier.
- The comparison is performed by using a mathematical measure of accuracy. We use the F_1 measure, defined as

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

Mathematically, F_1 is the "harmonic mean" of precision (the ability of the system to avoid overcoding, i.e., generating false positives) and recall (the ability of the system to avoid undercoding, i.e., generating false negatives).

Some tests: the Language Logic data & the Egg data

Dataset	$\lvert Tr \rvert$	$\lvert Te \rvert$	$\lvert C \rvert$	F_1^μ
LL-A	201	65	18	0.92
LL-B	501	10299	34	0.90
LL-C	201	425	20	0.89
LL-D	501	698	27	0.85
LL-E	201	720	39	0.84
LL-F	501	999	57	0.82
LL-G	501	1898	104	0.80
LL-H	501	699	86	0.79
LL-I	501	699	69	0.78
LL-L	501	698	65	0.75
Egg-A	700	300	14	0.63
Egg-B	653	273	20	0.60

Some tests: the Language Logic data & the Egg data

Dataset	$\mid Tr \mid$	$\mid Te \mid$	$\mid C \mid$	F_1^μ
LL-A	201	65	18	0.92
LL-B	501	10299	34	0.90
LL-C	201	425	20	0.89
LL-D	501	698	27	0.85
LL-E	201	720	39	0.84
LL-F	501	999	57	0.82
LL-G	501	1898	104	0.80
LL-H	501	699	86	0.79
LL-I	501	699	69	0.78
LL-L	501	698	65	0.75
Egg-A	700	300	14	0.63
Egg-B	653	273	20	0.60

Some tests: the Language Logic data & the Egg data

Dataset	$\mid Tr \mid$	$\mid Te \mid$	$\mid C \mid$	F_1^μ
LL-A	201	65	18	0.92
LL-B	501	10299	34	0.90
LL-C	201	425	20	0.89
LL-D	501	698	27	0.85
LL-E	201	720	39	0.84
LL-F	501	999	57	0.82
LL-G	501	1898	104	0.80
LL-H	501	699	86	0.79
LL-I	501	699	69	0.78
LL-L	501	698	65	0.75
Egg-A	700	300	14	0.63
Egg-B	653	273	20	0.60

Some tests: the Language Logic data & the Egg data

Dataset	$\lvert Tr \rvert$	$\lvert Te \rvert$	$\lvert C \rvert$	F_1^{μ}
LL-A	201	65	18	0.92
LL-B	501	10299	34	0.90
LL-C	201	425	20	0.89
LL-D	501	698	27	0.85
LL-E	201	720	39	0.84
LL-F	501	999	57	0.82
LL-G	501	1898	104	0.80
LL-H	501	699	86	0.79
LL-I	501	699	69	0.78
LL-L	501	698	65	0.75
Egg-A	700	300	14	0.63
Egg-B	653	273	20	0.60

Some tests: the Language Logic data & the Egg data

Dataset	$\lvert Tr \rvert$	$\lvert Te \rvert$	$\lvert C \rvert$	F_1^{μ}
LL-A	201	65	18	0.92
LL-B	501	10299	34	0.90
LL-C	201	425	20	0.89
LL-D	501	698	27	0.85
LL-E	201	720	39	0.84
LL-F	501	999	57	0.82
LL-G	501	1898	104	0.80
LL-H	501	699	86	0.79
LL-I	501	699	69	0.78
LL-L	501	698	65	0.75
Egg-A	700	300	14	0.63
Egg-B	653	273	20	0.60

Outline

Automated Verbatim Quantification

- Q : is the correct classification of individual verbatims always our final goal? or is it sometimes the case that the final goal is just correctly estimating, for each code, the percentage of respondents that should be attributed the code?

Automated Verbatim Quantification (cont'd)

Example Question 1 (asked by telecom service provider)

"How do you rate the quality of service we are providing to you?"

- In surveys such as these we are interested in accurately determining where every single respondent belongs, since every single respondent may require individualized action depending on the response returned;
- If so, classification accuracy needs to be measured via a mathematical function that rewards the correct classification of each single verbatim; F_1 is such a measure.

Automated Verbatim Quantification (cont'd)

Example Question 2 (asked by fast food company)

"How do you like the fact that we have introduced onions in cheeseburgers?"

Example Question 3 (asked by social science institute)

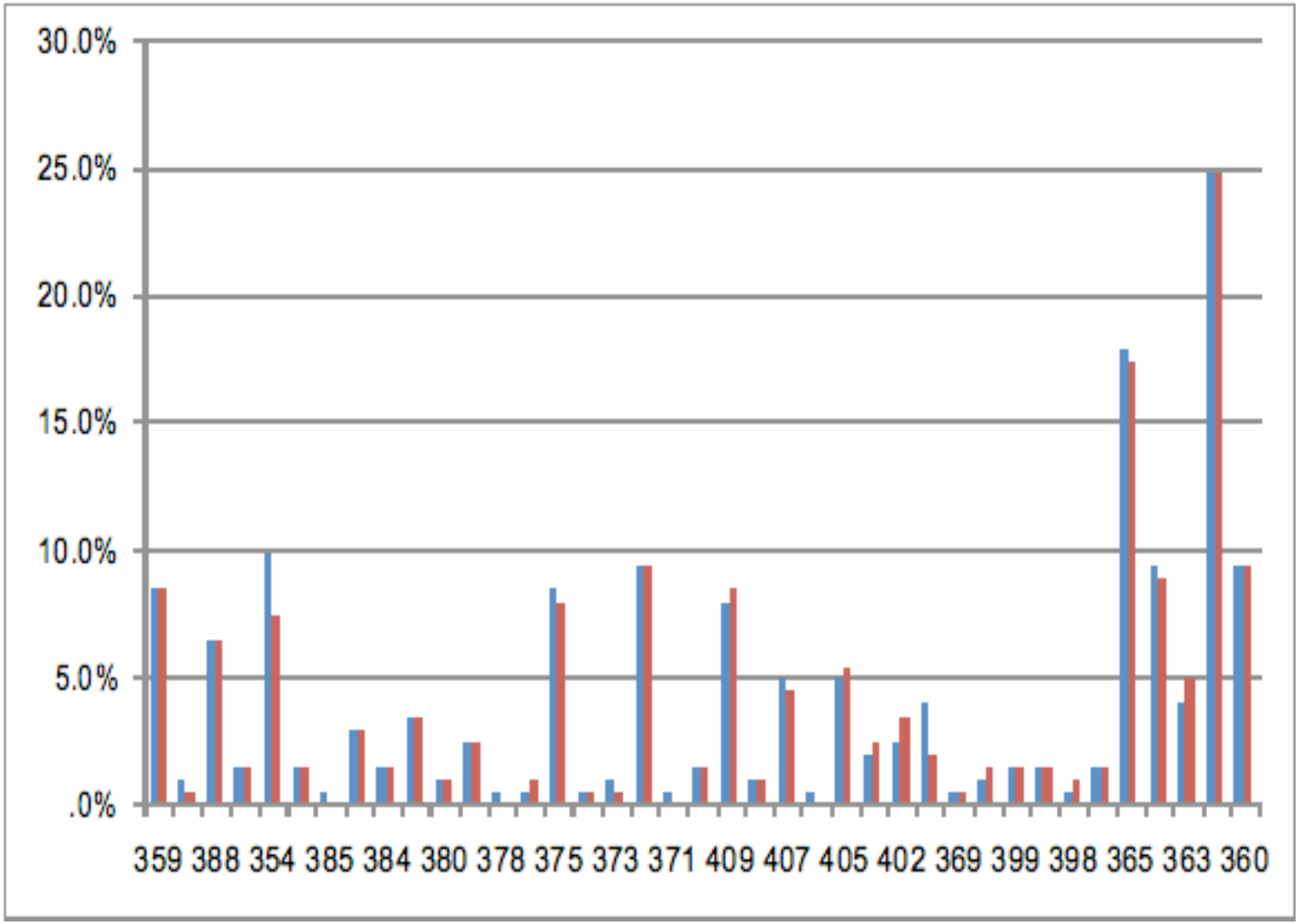
"Do you think that more stringent laws on gun possession should be enforced, and why?"

- In surveys such as these we are just interested in accurately estimating, for each code, the percentage of respondents that should be attributed the code ("code prevalence"), since respondents do not require individualized action that depends on the response returned.

Automated Verbatim Quantification (cont'd)

- This task is variously called
  - quantification (via classification) ; or
  - code prevalence estimation; or
  - distribution estimation.
- Quantification is also interesting in fields other than survey computing; e.g., quantifying `VeryPositive`, `Positive`, `Negative`, and `VeryNegative` comments about product X on the Web, and tracking how these code prevalences evolve in time.

An Early Example: the LL-E dataset



Measuring Quantification Accuracy

- Quantification accuracy needs to be measured via a mathematical function that rewards the correct estimation of code prevalence.
- The previous histogram visually uses absolute bias (i.e., discrepancy between true and predicted class prevalence) as the measure
  - This is considered a naive measure, since mis-predicting 10% in place of 1% is a more serious mistake than mis-predicting 50% in place of 41%.
- We might revert to relative bias (i.e., absolute bias divided by the true class prevalence) ...
  - ... but this is again a naive measure, since it (a) it is undefined when the true prevalence is 0%, and (b) it returns excessively large values (tending to infinity) when true prevalence is very small.

Measuring Quantification Accuracy

- Quantification accuracy needs to be measured via a mathematical function that rewards the correct estimation of code prevalence.
- The previous histogram visually uses absolute bias (i.e., discrepancy between true and predicted class prevalence) as the measure
  - This is considered a naive measure, since mis-predicting 10% in place of 1% is a more serious mistake than mis-predicting 50% in place of 41%.
- We might revert to relative bias (i.e., absolute bias divided by the true class prevalence) ...
  - ... but this is again a naive measure, since it (a) it is undefined when the true prevalence is 0%, and (b) it returns excessively large values (tending to infinity) when true prevalence is very small.

Measuring Quantification Accuracy (cont'd)

- The "industry standard" for measuring quantification accuracy is

  (normalized) Kullback-Leibler divergence

  (*NKLD* – aka normalized relative entropy), which is defined as "a measure of the inefficiency of assuming that a probability distribution is $\hat{p}$ when the true distribution is p"
- It is mathematically defined as

$$NKLD(p||\hat{p}) = 1 - \frac{2}{\pi} \arctan\left(\sum_{x \in X} p(x) \log \frac{p(x)}{\hat{p}(x)}\right)$$

- $NKLD = 1$ indicates no discrepancy between true and predicted distribution, $NKLD = 0$ indicates total discrepancy.

Should we just "classify and count"?

- Q : Is the "classify and count" strategy optimal for quantification?
- A : No. The reason is that not necessarily the most accurate automated classifier we can generate balances false positives and false negatives.
- Q : Which of the following classifiers should we prefer?

	Classifier 1	Classifier 2
True positives	55	75
False positives	20	20
False negatives	20	0
True negatives	905	905
F_1	0.800	0.925
Predicted percentage	7.5%	9.5%
True percentage	7.5%	7.5%

- A: Classifier 2 should be preferred if classification is the final goal; Classifier 1 should be preferred if quantification is the final goal.

Should we just "classify and count"?

- Q : Is the "classify and count" strategy optimal for quantification?
- A : No. The reason is that not necessarily the most accurate automated classifier we can generate balances false positives and false negatives.
- Q : Which of the following classifiers should we prefer?

	Classifier 1	Classifier 2
True positives	55	75
False positives	20	20
False negatives	20	0
True negatives	905	905
F_1	0.800	0.925
Predicted percentage	7.5%	9.5%
True percentage	7.5%	7.5%

- A: Classifier 2 should be preferred if classification is the final goal; Classifier 1 should be preferred if quantification is the final goal.

Should we just "classify and count"?

- Q : Is the "classify and count" strategy optimal for quantification?
- A : No. The reason is that not necessarily the most accurate automated classifier we can generate balances false positives and false negatives.
- Q : Which of the following classifiers should we prefer?

	Classifier 1	Classifier 2
True positives	55	75
False positives	20	20
False negatives	20	0
True negatives	905	905
F_1	0.800	0.925
Predicted percentage	7.5%	9.5%
True percentage	7.5%	7.5%

- A: Classifier 2 should be preferred if classification is the final goal; Classifier 1 should be preferred if quantification is the final goal.

Going for *NKLD*

- In our recent research we have proposed a machine learning method called SVM(NKLD) which "goes for *NKLD*", i.e., is "conscious" that the final goal is obtaining the highest possible *NKLD* (and not F_1), and trains classifiers accordingly.[4]
- SVM(NKLD) trades (a little bit of) classification accuracy for (a larger bit of) quantification accuracy.
- We have tested it by using a set of Reuters data, consisting of 804,414 manually coded texts coded according to a 99-code codeframe, and generated across 1 year (a "tracker"); we train the classifier on the data from the 1st week, and we test quantification accuracy on 52 test sets, one for each week's worth data.

[4] Andrea Esuli and Fabrizio Sebastiani. Optimizing Text Quantifiers for Multivariate Loss Functions. Unpublished manuscript, 2012.

Going for *NKLD*

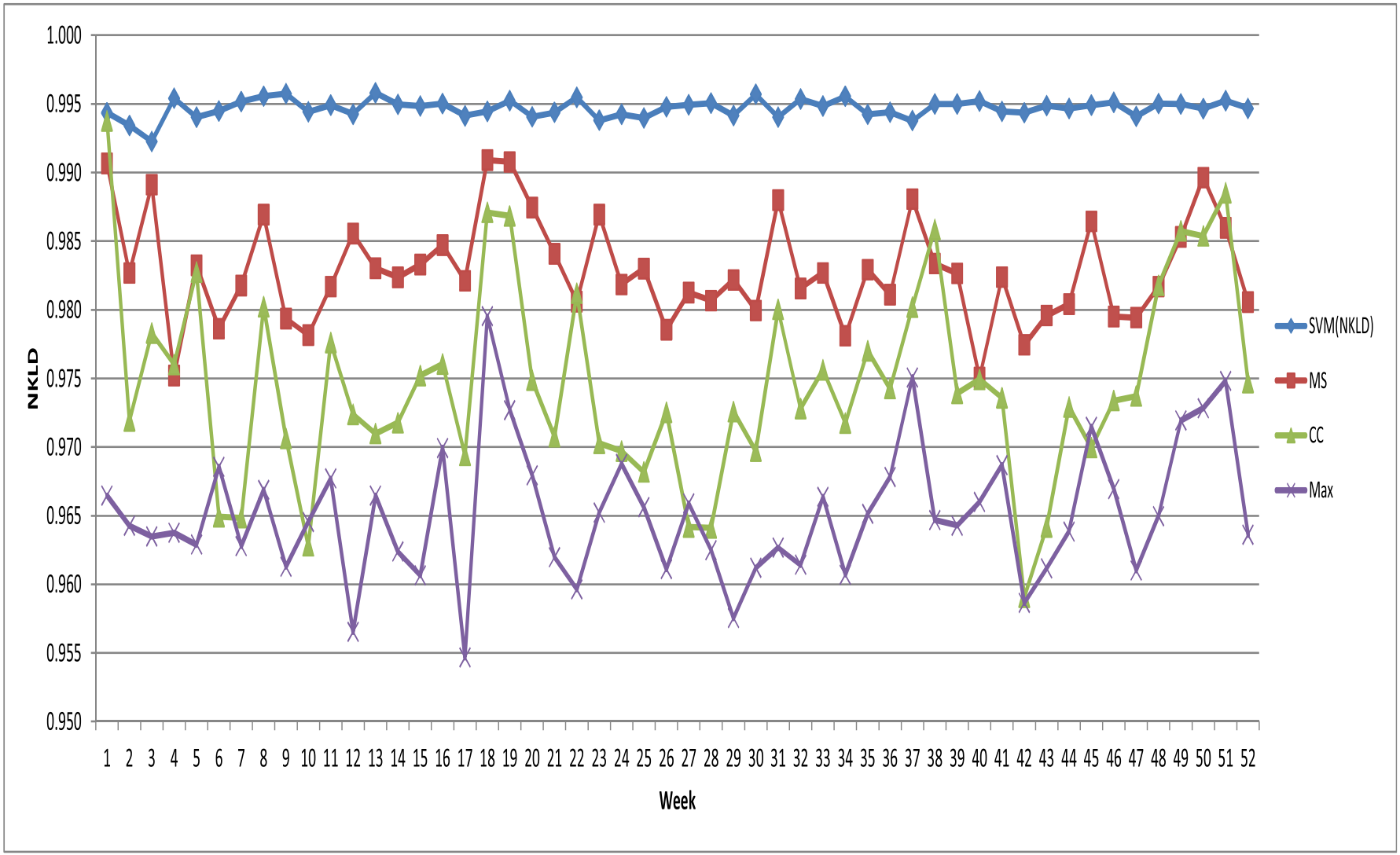
- In our recent research we have proposed a machine learning method called SVM(NKLD) which "goes for *NKLD*", i.e., is "conscious" that the final goal is obtaining the highest possible *NKLD* (and not F_1), and trains classifiers accordingly.[4]
- SVM(NKLD) trades (a little bit of) classification accuracy for (a larger bit of) quantification accuracy.
- We have tested it by using a set of Reuters data, consisting of 804,414 manually coded texts coded according to a 99-code codeframe, and generated across 1 year (a "tracker"); we train the classifier on the data from the 1st week, and we test quantification accuracy on 52 test sets, one for each week's worth data.

[4] Andrea Esuli and Fabrizio Sebastiani. Optimizing Text Quantifiers for Multivariate Loss Functions. Unpublished manuscript, 2012.

The Reuters experiments

		RCV1-v2
ALL	Total # of docs	804,414
	# of codes (i.e., binary tasks)	99
	Time unit used for split	week
TRAINING	# of docs	12,807
	Min # of positive docs per code	2
	Max # of positive docs per code	5,581
	Avg # of positive docs per code	397
	Min code prevalence	0.01%
	Max code prevalence	43.75%
	Avg code prevalence	3.15%
TEST	# of docs	791,607
	# of test sets per code	52
	Avg # of test docs per set	15,212
	Min # of positive docs per code	0
	Max # of positive docs per code	9775
	Avg # of positive docs per code	494
	Min code prevalence	0.00%
	Max code prevalence	53.44%
	Avg code prevalence	3.23%

The Reuters experiments (cont'd)



The Takeaway Message!

1. Supervised machine learning technology for automated verbatim coding works!, and it does so
   1. in a way which is robust to ill-formed input
   2. across different application contexts (CRM, market research, social sciences)
   3. across different languages
2. When applying this technology, it is worth determining in advance whether our final goal is
   1. classification (individualized action required)
   2. quantification-via-classification (no individualized action required)

   and to use quantification-specific technology if the latter is the case.

Thanks!

The Takeaway Message!

1. Supervised machine learning technology for automated verbatim coding works!, and it does so
   1. in a way which is robust to ill-formed input
   2. across different application contexts (CRM, market research, social sciences)
   3. across different languages
2. When applying this technology, it is worth determining in advance whether our final goal is
   1. classification (individualized action required)
   2. quantification-via-classification (no individualized action required)

   and to use quantification-specific technology if the latter is the case.

Thanks!