Text Quantification: Current Research and Future Challenges

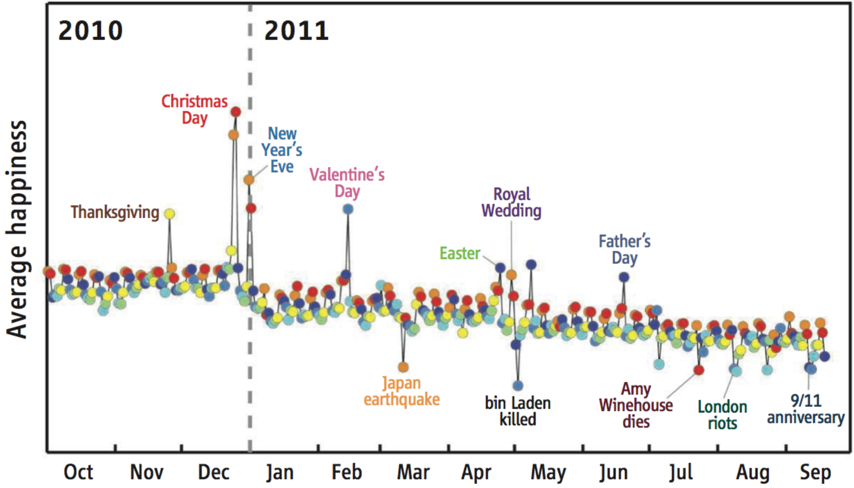
Fabrizio Sebastiani
(Joint work with Shafiq Joty and Wei Gao)

Qatar Computing Research Institute
Qatar Foundation
PO Box 5825 – Doha, Qatar
E-mail: fsebastiani@qf.org.qa
http://www.qcri.com/

FIRE 2016
Kolkata, IN – December 7-10, 2016

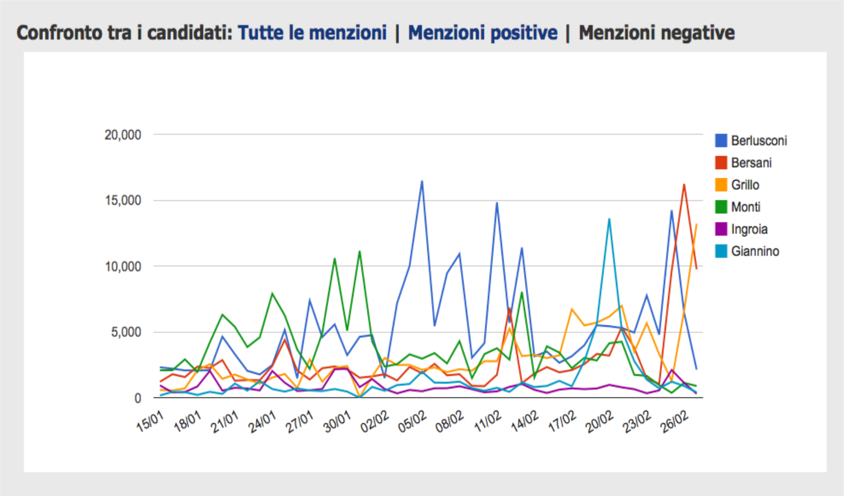
What is quantification?

1



[1] Dodds, Peter et al. Temporal Patterns of Happiness and Information in a Global Social Network: Hedonometrics and Twitter. PLoS ONE, 6(12), 2011.

What is quantification? (cont'd)



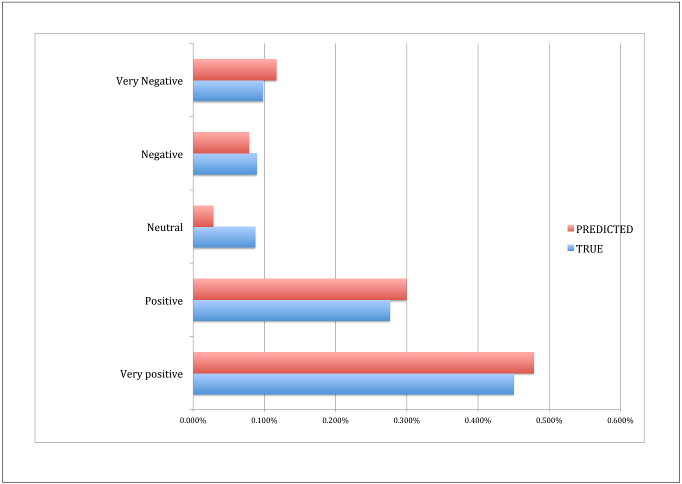
What is quantification? (cont'd)

- In many applications of classification, the real goal is determining the relative frequency (or: prevalence) of each class in the unlabelled data; this is called quantification, or supervised prevalence estimation
- E.g.
  - Among the tweets concerning the next presidential elections, what is the percentage of pro-Democrat ones?
  - Among the posts about the Apple Watch 2 posted on forums, what is the percentage of "very negative" ones?
  - How have these percentages evolved over time recently?
- This task has been studied within IR, ML, DM, and has given rise to learning methods and evaluation measures specific to it
- We will mostly deal with text quantification

Where we are

What is quantification? (cont'd)

- Quantification may be also defined as the task of approximating a true distribution by a predicted distribution



Distribution drift

- The need to perform quantification arises because of distribution drift, i.e., the presence of a discrepancy between the class distribution of *Tr* and that of *Te*.
- Distribution drift may derive when
  - the environment is not stationary across time and/or space and/or other variables, and the testing conditions are irreproducible at training time
  - the process of labelling training data is class-dependent (e.g., "stratified" training sets)
  - the labelling process introduces bias in the training set (e.g., if active learning is used)
- Distribution drift clashes with the IID assumption, on which standard ML algorithms are instead based.

The "paradox of quantification"

- Is "classify and count" the optimal quantification strategy? No!
- A perfect classifier is also a perfect "quantifier" (i.e., estimator of class prevalence), but ...
- ... a good classifier is not necessarily a good quantifier (and vice versa) :

	FP	FN
Classifier A	18	20
Classifier B	20	20

- Paradoxically, we should choose quantifier B rather than quantifier A, since A is biased
- This means that quantification should be studied as a task in its own right

Applications of quantification

A number of fields where classification is used are not interested in individual data, but in data aggregated across spatio-temporal contexts and according to other variables (e.g., gender, age group, religion, job type, ...); e.g.,

- Social sciences : studying indicators concerning society and the relationships among individuals within it

  > *[Others] may be interested in finding the needle in the haystack, but social scientists are more commonly interested in characterizing the haystack.*
  >
  > *(Hopkins and King, 2010)*

- Political science : e.g., predicting election results by estimating the prevalence of blog posts (or tweets) supporting a given candidate or party

Applications of quantification (cont'd)

- Epidemiology : concerned with tracking the incidence and the spread of diseases; e.g.,
  - estimate pathology prevalence from clinical reports where pathologies are diagnosed
  - estimate the prevalence of different causes of death from verbal accounts of symptoms
- Market research : concerned with estimating the incidence of consumers' attitudes about products, product features, or marketing strategies; e.g.,
  - estimate customers' attitudes by quantifying verbal responses to open-ended questions
- Others : e.g.,
  - estimating the proportion of no-shows within a set of bookings
  - estimating the proportions of different types of cells in blood samples

How do we evaluate quantification methods?

- Evaluating quantification means measuring how well a predicted distribution $\hat{p}(c)$ fits a true distribution $p(c)$
- The goodness of fit between two distributions can be computed via divergence functions, which enjoy
  1. $D(p, \hat{p}) = 0$ only if $p = \hat{p}$ (identity of indiscernibles)
  2. $D(p, \hat{p}) \geq 0$ (non-negativity)

  and may enjoy (as exemplified in the binary case)

  3. If $\hat{p}'(c_1) = p(c_1) - a$ and $\hat{p}''(c_1) = p(c_1) + a$, then $D(p, \hat{p}') = D(p, \hat{p}'')$ (impartiality)
  4. If $\hat{p}'(c_1) = p'(c_1) \pm a$ and $\hat{p}''(c_1) = p''(c_1) \pm a$, with $p'(c_1) < p''(c_1) \leq 0.5$, then $D(p, \hat{p}') > D(p, \hat{p}'')$ (relativity)

How do we evaluate quantification methods? (cont'd)

Divergences frequently used for evaluating (multiclass) quantification are

- $\text{MAE}(p, \hat{p}) = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} |\hat{p}(c) - p(c)|$ (Mean Abs Error)
- $\text{MRAE}(p, \hat{p}) = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{|\hat{p}(c) - p(c)|}{p(c)}$ (Mean Relative Abs Error)
- $\text{KLD}(p, \hat{p}) = \sum_{c \in \mathcal{C}} p(c) \log \frac{p(c)}{\hat{p}(c)}$ (Kullback-Leibler Divergence)

	Impartiality	Relativity
Mean Absolute Error	Yes	No
Mean Relative Absolute Error	Yes	Yes
Kullback-Leibler Divergence	No	Yes

Quantification methods: CC

- Classify and Count (CC) consists of
  1. generating a classifier from Tr
  2. classifying the items in Te
  3. estimating $p_{Te}(c_j)$ by counting the items predicted to be in c_j, i.e.,

$$\hat{p}^{CC}_{Te}(c_j) = p_{Te}(\delta_j)$$

- But a good classifier is not necessarily a good quantifier ...
- CC suffers from the problem that "standard" classifiers are usually tuned to minimize $(FP + FN)$ or a proxy of it, but not $|FP - FN|$
  - E.g., in recent experiments of ours, out of 5148 binary test sets averaging 15,000+ items each, standard (linear) SVM brought about an average FP/FN ratio of 0.109.

Quantification methods: PCC

- Probabilistic Classify and Count (PCC) estimates p_{Te} by simply counting the expected fraction of items predicted to be in the class, i.e.,

$$\hat{p}_{Te}^{PCC}(c_j) = E_{Te}[c_j] = \frac{1}{|Te|} \sum_{\mathbf{x} \in Te} p(c_j|\mathbf{x})$$

- The rationale is that posterior probabilities contain richer information than binary decisions, which are obtained from posterior probabilities by thresholding.

Quantification methods: ACC

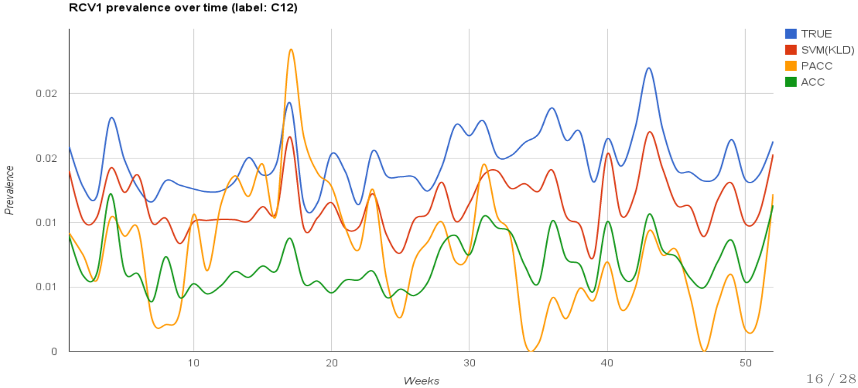
- Adjusted Classify and Count (ACC) is based on the observation that, after we have classified the test documents Te,

$$p_{Te}(\delta_j) = \sum_{c_i \in \mathcal{C}} p_{Te}(\delta_j | c_i) \cdot p_{Te}(c_i)$$

- The $p_{Te}(\delta_j)$'s are observed
- The $p_{Te}(\delta_j | c_i)$'s can be estimated on Tr via k-fold cross-validation (these latter represent the system's bias).
- This results in a system of $|\mathcal{C}|$ linear equations (one for each c_j) with $|\mathcal{C}|$ unknowns (the $p_{Te}(c_i)$'s).
- ACC consists in solving this system, and consists in correcting the class prevalence estimates obtained by CC according to the estimated system's bias.

Quantification methods: SVM(KLD)

- SVM(KLD) consists in performing CC with an SVM in which the minimized loss function is KLD
- KLD (and all other measures for evaluating quantification) is non-linear and multivariate, so optimizing it requires “SVMs for structured output”, which can label entire structures (in our case: sets) in one shot



Where do we go from here?

Where do we go from here?

- Quantification research has assumed quantification to require predictions at an individual level as an intermediate step; e.g.,
  - **PCC** : Use expected counts (from posterior probabilities) instead of actual counts
  - **ACC** : Perform CC and then correct for the classifier's estimated bias
  - **SVM(KLD)** : Perform CC via classifiers optimized for quantification loss functions
- Radical change in direction :

  Can quantification be performed without predictions at an individual level?

Vapnik's Principle

- Key observation: classification is a more general problem than quantification
- Vapnik's principle:

  > *"If you possess a restricted amount of information for solving some problem, try to solve the problem directly and never solve a more general problem as an intermediate step. It is possible that the available information is sufficient for a direct solution but is insufficient for solving a more general intermediate problem."*

- This suggests solving quantification directly, without solving classification as an intermediate step

(Binary) quantification as a regression problem

- Formally, quantification does not require classification!
  - (Binary) Classification : learn function $h_c : \mathcal{X} \rightarrow \{-1, +1\}$
  - (Binary) Quantification : learn function $q_c : 2^{\mathcal{X}} \rightarrow [0, 1]$
  - (Univariate) Regression : learn function $r_c : \mathcal{X} \rightarrow \mathbb{R}$
- Quantification is an instance of regression!, provided we
  - constrain the output to be in [0,1]
  - make the subsets in $2^{\mathcal{X}}$ the objects of prediction
- In some applications, viewing quantification as an instance of regression is more natural ; e.g.
  - Topic-based sentiment quantification in tweets
  - Cell type quantification in blood samples
  - Estimating the proportion of no-shows within a set of bookings

(Binary) quantification as a regression problem

- Our process may thus consist in
  1. training, for each class $c \in \{c_1, c_2\}$, a regressor $r_c : 2^{\mathcal{X}} \rightarrow \mathbb{R}$;
  2. generate, for unlabeled set s and for each class $c \in \{c_1, c_2\}$, a prediction $r_c(s)$;
  3. generate, for each class $c \in \{c_1, c_2\}$, prevalence estimates $p_s(c)$ by rescaling the predictions $r_c(s)$, i.e., by computing

$$\hat{p}_s(c) = \frac{r_c(s) - \min_{c \in \{c_1, c_2\}} r_c(s)}{\max_{c \in \{c_1, c_2\}} r_c(s) - \min_{c \in \{c_1, c_2\}} r_c(s)} \tag{1}$$

- Any supervised learned for regression can be used (e.g., ϵ-SVR, Random Forests, etc.)

Generating vectorial representations

- If we switch to regression we need the notions of
  - microexamples : $\mathbf{x}$, $\mathbf{x}_1$, $\mathbf{x}_2$, ... (e.g., documents)
  - macroexamples : $\mathbf{X}$, $\mathbf{X}_1$, $\mathbf{X}_2$, ... (e.g., sets of documents)
- Our learning algorithm is given as input not a set of training microexamples $\{\mathbf{x}_1, ..., \mathbf{x}_m\}$ but an entire set of training macroexamples $\{\mathbf{X}_1, ..., \mathbf{X}_n\}$
- Our regressor r_c is given as input not a single microexample $\mathbf{x}$ but an entire macroexample $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_{|\mathbf{X}|}\}$
- We thus face the task of coming up with (a) a choice of features, and (b) a weighting function
  1. where vectors represent each a macroexample (unusual in IR!)
  2. that capture the nature of our problem, i.e., conveys useful information for predicting class prevalence

Generating vectorial representations (cont'd)

- A potential solution:
  - As features we use all terms that appear in at least one training micro-example
  - As the weight of feature t_k for macroexample $\mathbf{X}_i$ we use macroexample frequency, i.e., the fraction of items $\mathbf{x}_{ij}$ (microexamples) in $\mathbf{X}_i$ in which t_k occurs

$$w_{ki} = \frac{|\{\mathbf{x}_{ij} \in \mathbf{X}_i | t_k \in \mathbf{x}_{ij}\}|}{|\{\mathbf{x}_{ij} \in \mathbf{X}_i\}|}$$

Generating vectorial representations (cont'd)

- Function

$$w_{ki} = \frac{|\{\mathbf{x}_{ij} \in \mathbf{X}_i | t_k \in \mathbf{x}_{ij}\}|}{|\{\mathbf{x}_{ij} \in \mathbf{X}_i\}|}$$

  captures the nature of quantification because it makes reference to microitems, which is what quantification is about (e.g.,

$$w_{ki} = \frac{\sum_{\mathbf{x}_{ij} \in \mathbf{X}_i} \#(t_k, \mathbf{x}_{ij})}{\sum_{t_s \in T} \sum_{\mathbf{x}_{ij} \in \mathbf{X}_i} \#(t_s, \mathbf{x}_{ij})} \qquad (*)$$

  does not make reference to them)
- Other features may be added that describe the macroexample as a whole; e.g., type of topic (for topic-based tweet sentiment quantification), age of patient (for blood cell quantification), etc.

Identifying training items

- While in some applications (e.g., topic-based tweet sentiment quantification) we may have several training macroexamples, in some others we may have only one (e.g., quantifying the distribution of topics in news)
- In the latter case, how do we obtain the many training macroexamples that a regressor needs?
- A possible solution: from the only available set of microexamples, extract many different subsets
- Out of n microexamples, we can generate 2^n training macroexamples; we thus need a selection policy that emphasizes diversity
- Random selection likely to be a reasonable policy, trading off between computational cost (inexpensive) and ability to generate diversity (high, in the long run)

Conclusion

- "Quantification as Regression" :
  - new paradigm, more in line with Vapnik's principle
  - entails challenging problems, esp. concerning how to generate vectorial representations
- This "solves" the paradox of quantification
- Quantification: a relatively (yet) unexplored new task, with many research problems still open
- Growing awareness that quantification is going to be more and more important; given the advent of "big data", application contexts will spring up in which we will simply be happy with analysing data at the aggregate (rather than at the individual) level

Questions?

Thank you!

For any question, email me at
fsebastiani@qf.org.qa