Automatic Coding of Open-ended Questions Using Text Categorization Techniques

Daniela Giorgetti[1], Irina Prodanof[1], Fabrizio Sebastiani[2]

[1] Istituto di Linguistica Computazionale, Consiglio Nazionale delle Ricerche, Pisa, Italy
[2] Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy
{daniela.giorgetti,irina.prodanof}@ilc.cnr.it, fabrizio@iei.pi.cnr.it

Abstract. Open-ended questions do not limit respondents' answers in terms of linguistic form and semantic content, but bring about severe problems in terms of cost and speed, since their coding requires trained professionals to manually identify and tag meaningful text segments. To overcome these problems, a few automatic approaches have been proposed in the past, some based on matching the answer with textual descriptions of the codes, others based on manually building rules that check the answer for the presence or absence of code-revealing words. While the former approach is scarcely effective, the major drawback of the latter approach is that the rules need to be developed manually, and before the actual observation of text data.
We propose a new approach, inspired by work in information retrieval (IR), that overcomes these drawbacks. In this approach survey coding is viewed as a task of *multiclass text categorization* (MTC), and is tackled through techniques originally developed in the field of *supervised machine learning*. In MTC each text belonging to a given corpus has to be classified into exactly one from a set of predefined categories. In the supervised machine learning approach to MTC, a set of categorization rules is built *automatically* by learning the characteristics that a text should have in order to be classified under a given category. Such characteristics are automatically learnt from a set of training examples, i.e. a set of texts whose category is known.
For survey coding, we equate the set of codes with categories, and all the collected answers to a given question with texts. Giorgetti and Sebastiani [5] have carried out automatic coding experiments with two different supervised learning techniques, one based on a naïve Bayesian method and the other based on multiclass support vector machines. Experiments have been run on a corpus of social surveys carried out by the National Opinion Research Center, University of Chicago (NORC). These experiments show that our methods outperform, in terms of accuracy, previous automated methods tested on the same corpus.

1 Introduction

The survey process typically consists of several steps which have to be performed sequentially, and include the design, construction, administration, analysis and evaluation of the survey itself. Survey questions may be open-ended, i.e. they allow respondents to answer in their own words, or close-ended, i.e. they consist of questions where the answers are limited by a multiple choice list or entail a true/false, agree/disagree reply. In this paper we will only deal with coding, a core step in open-ended surveys, in which text data extracted from free responses to open-ended questions are mapped into codes according to a predefined code scheme. While the

coding of a close-ended survey is straightforward, the coding of an open-ended survey is a subjective task traditionally performed manually by trained professionals, and thus it is costly, time-consuming and affected by the inter-coder agreement problem (which occurs when there is not agreement between two different coders on the code to be assigned to a specific text segment). These are the reasons why social scientists tend to avoid including too many open-ended questions in their surveys, preferring the less problematic multiple choice questions.

A few solutions have been proposed to automate the coding phase of open-ended surveys, but they still require some manual engineering effort. Our approach instead, relies on fully automatic text categorization techniques developed in the information retrieval and machine learning fields.

The structure of this paper is as follows. In Section 2 we focus on the task of automatic survey coding and we discuss how previous approaches have tried to automate (at least partially) the coding task. In order to make the paper self-contained, in Section 3 we give some background on information retrieval and text categorization. In Section 4 we describe how we map the task of coding onto a text categorization task. In Section 5 we present experimental results [5] obtained on three corpora from NORC General Social Survey using two different text categorization classifiers and in Section 6 we summarize our approach and the results achieved and give some hints for possible future developments.

2 Automatic survey coding of open-ended surveys

Coding a survey may be viewed as the task of identifying and labelling common meaningful themes in the answers given by different respondents to a survey questionnaire. Sometimes novel, unforeseen themes are sought in the answers being analyzed, but most of the times surveys are run with a clear purpose, i.e. with a clear set of previously identified concepts whose presence in the text corpus must be assessed or measured in some way. We will concentrate on the most frequent case, in which themes sought are known a priori.

The issue of automating survey coding has been dealt with in software packages developed in the context of text analysis for the social sciences, which includes survey analysis. However, these software packages are not usually tailored for the specific task of survey analysis; in particular, the solutions they provide for the coding task are still unsatisfactory. Many of these applications mostly aim at facilitating the manual coding of data, and their display in several convenient ways. A few of these packages instead do perform automatic coding, mainly by relying on specialized hand-developed *dictionaries* (or *rules*). This means that text fragments are automatically assigned to a specific category if and only if they contain words matching those in the dictionary relevant to the category.

The closest approach to ours is probably the dictionary-based approach as it is described in Viechnicki's work [16]. In this paper responses to questions from the NORC[3] General Social Survey [3] are classified by means of a set of codes predefined by NORC social scientists. Viechnicki proposes two alternative approaches. In the first one, words that characterize a given category may be combined by means of Boolean operators (AND, OR, NOT), and the answer is classified under the category whose Boolean description it matches. The second method is instead based on computing the similarity between two weighted vectors of words (see Section 3.1 for an explanation of how weighted vectors of words and their similarity are computed) extracted from the answer and from a textual explicatory caption of the code, and choosing the code with the highest similarity score. Similarly, Macchia

[3] `http://www.norc.uchicago.edu/`

and Murgia [10] present a dictionary-based automated approach in which the answer is assigned a unique code if there is an exact match with phrases belonging to a previously defined dictionary associated with the code, or it is assigned a "best code" if the match is partial.

These approaches have the typical drawback of dictionary-based methods, which need a dictionary to be manually developed before the actual coding step takes place, i.e. when data is still totally unknown. Besides, dictionaries need to be manually updated as a result of changes in the structure and/or semantics of the coding scheme. For example, if a new category is added to the scheme, a new dictionary has to be created for it, and the dictionaries for the old categories need to be updated in order to avoid "capturing" the answers that are instead to be filed under the newly introduced category.

Our approach to survey coding has several advantages with respect to the dictionary-based approach.

Firstly, in our learning-based approach the manual effort is directed towards the manual coding of a relatively small training set of answers, and not towards the creation of specialized dictionaries. This is advantageous, as it is easier to manually classify a set of documents than to build and tune a dictionary of words that trigger the attribution of the code, for the simple fact that it is easier to characterize a concept extensionally (i.e. to select instances of it) than intensionally (i.e. to describe the concept in words, or to describe a procedure for recognizing its instances).

Secondly, our approach is solidly grounded in machine learning theory, and it can leverage on a wealth of results and techniques developed within text categorization, a discipline which has been bursting with activity in the last ten years (see e.g. [13]) and has produced systems whose accuracy rivals or exceeds that of a human (i.e. systems capable of generating codes that correlate with those attributed by a coder at least as well as the codes attributed by two human coders correlate with each other).

Of course, our approach is mostly useful for medium- to large-sized surveys, as in the learning phase we need a hand-coded set of answers to train the inductive learner. This means that if a survey is somewhat limited in the number of surveyed people, hand-coding the training set may coincide with hand-coding the entire set. NORC's surveys are examples of relatively large-sized surveys, since 40,933 interviews have been completed in the years from 1972 to 2000.

3 Basics of Text Categorization

This section gives a brief introduction to text categorization and to some basic concepts adopted from the related field of information retrieval; these notions will provide us with the background for solving the survey coding task.

Text categorization (TC, also known as *text classification*) is the task of approximating the unknown *target function* $\Phi : \mathcal{D} \times \mathcal{C} \rightarrow \{T, F\}$ that describes how documents ought to be classified by means of a function $\hat{\Phi} : \mathcal{D} \times \mathcal{C} \rightarrow \{T, F\}$ called the *classifier*, where $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$ is a predefined set of thematic categories, $\mathcal{D}$ is a domain of documents, and T, F represent the boolean values *true* and *false*. If $\Phi(d_j, c_i) = T$, then d_j is called a *positive example* (or a *member*) of c_i, while if $\Phi(d_j, c_i) = F$ it is called a *negative example* of c_i. The categories are just symbolic labels, and no additional knowledge of their meaning is usually available. Classification has to be accomplished only on the basis of knowledge extracted from the documents contents, as no metadata (such as e.g. publication date, document type, publication source) are usually available.

Text categorization is a *subjective* task: when two experts (human or artificial) decide whether to classify document d_j under category c_i, they may disagree, and

this in fact happens quite frequently. A news article on Tony Blair visiting Tuscany could be filed under Politics, or under Gossip, or under both, or even under neither, depending on the subjective judgment of the expert.

Depending on the application, it might be the case that exactly one label $c_i \in \mathcal{C}$ must be assigned to each $d_j \in \mathcal{D}$, or that any number $0 \leq n_j \leq |\mathcal{C}|$ of categories may be assigned to each $d_j \in \mathcal{D}$. The former case is usually dubbed the *binary* case or the *multiclass* case, depending on whether $|\mathcal{C}| = 2$ or $|\mathcal{C}| > 2$, respectively. Since the $|\mathcal{C}| > 2$ case will be the object of interest in this paper, from here on when speaking of TC we will actually mean *multiclass* TC.

We can roughly distinguish three different phases in the life cycle of a TC system: document indexing, classifier learning, and classifier evaluation. The three following paragraphs are devoted to these three phases, respectively; for a more detailed treatment see Sections 5, 6 and 7, respectively, of [13].

3.1 Document indexing

Document indexing denotes the activity of mapping a document d_j into a compact representation of its content that can be directly interpreted (i) by a classifier-building algorithm and (ii) by a classifier, once it has been built. The document indexing methods usually employed in TC are borrowed from IR, where a text d_j is typically represented as a vector of term *weights* $\boldsymbol{d}_j = \langle w_{1j}, \ldots, w_{|\mathcal{T}|j} \rangle$. Here, $\mathcal{T}$ is the *dictionary*, i.e. the set of *terms* (also known as *features*) that occur at least once in at least α documents, and $0 \leq w_{kj} \leq 1$ quantifies the importance of t_k in characterizing the semantics of d_j. Typical values of α are between 1 and 5.

An indexing method is characterized by (i) a definition of what a term is, and (ii) a method to compute term weights. Concerning (i), the most frequent choice is to identify terms either with the *words* occurring in the documents (with the exception of *stop words*, i.e. topic-neutral words such as articles and prepositions, which are eliminated in a pre-processing phase), or with their *stems* (i.e. their morphological roots, obtained by applying a stemming algorithm). When terms coincide with words (either stemmed or not) the approach to document representation is called *bag of words*. Experimental results[**?**,12, **?**] on the use of *phrases* as additional indexing terms have not been uniformly encouraging up to now. Lewis [8] argues that the likely reason for the discouraging results is that, although indexing languages based on phrases have superior semantic qualities, they have inferior statistical qualities with respect to word-only indexing languages.

Concerning (ii), either statistical or probabilistic techniques are used to compute terms weights, the former being the most common option. Binary weights are a special case where 1 denotes the presence and 0 the absence of the term in the document. The choice of binary or non-binary weights depends on the kind of input that the classifier learning algorithm adopted requires. A popular class of statistical term weighting functions yelding non binary weights is $tfidf$ (see e.g. [11]), where two intuitions are at play: (a) the more frequently t_k occurs in d_j, the more important for d_j it is (the *term frequency* intuition); (b) the more documents t_k occurs in, the less discriminating it is, i.e. the smaller its contribution is in characterizing the semantics of a document in which it occurs (the *inverse document frequency* intuition). Weights computed by $tfidf$ are often normalized so as to contrast the tendency of $tfidf$ to emphasize long documents. Note that formulae such as $tfidf$ weigh the importance of a term to a document in terms of occurrence considerations only, disregarding for example the order in which terms appear in the document and their syntactic role.

In TC, unlike in IR, a *dimensionality reduction* phase is often applied so as to reduce the size of the document representations from $\mathcal{T}$ to a much smaller, predefined number. This has both the effect of reducing *overfitting* (i.e. the tendency of the

classifier to better classify the data it has been trained on than new unseen data), and to make the problem more manageable for the learning method, since many such methods are known not to scale well to high problem sizes. Dimensionality reduction often takes the form of *feature selection*: each term is scored by means of a scoring function that captures its degree of (positive, and sometimes also negative) correlation with c_i, and only the highest scoring terms are used for document representation.

3.2 Classifier learning

A text classifier for categories $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$ is automatically generated by a general inductive process (the *learner*) which, by observing the characteristics of a set of documents preclassified under $\mathcal{C}$, gleans the characteristics that a new unseen document should have in order to belong to a generic category $c_i \in \mathcal{C}$. In order to build classifiers for $\mathcal{C}$, one thus needs a labelled corpus Ω of documents such that the value of $\Phi(d_j, c_i)$ is known for every $\langle d_j, c_i \rangle \in \Omega \times \mathcal{C}$. In experimental TC it is customary to partition Ω into three disjoint sets Tr (the *training set*), Va (the *validation set*), and Te (the *test set*). The training set consists of documents the learner "observes" to build up the classifier. This is called a *supervised* learning activity, since learning is "supervised" by the information on the membership of training documents in categories. The validation set is the set of documents the engineer uses to fine-tune the classifier, e.g. choosing for a parameter p on which the classifier depends, the value that has yielded the best effectiveness when evaluated on Va. The test set is used for the final evaluation of classifier effectiveness. In both the validation and test phase, "evaluating the effectiveness" means running the classifier on a set of preclassified documents (Va or Te) and checking the degree of correspondence between the output of the classifier and the preassigned labels (which are assumed to be correct).

Several methods have been proposed in the text categorization literature for learning a text classifier from training data (see [13] for a review), including probabilistic methods, regression methods, decision tree and decision rule learners, neural networks, batch and incremental learners of linear classifiers, example-based methods, support vector machines, genetic algorithms, hidden Markov models, and classifier committees.

3.3 Classifier evaluation

Training efficiency (i.e. average time required to build a classifier $\hat{\Phi}$ from a given corpus Ω), as well as *classification efficiency* (i.e. average time required to classify a new document by means of $\hat{\Phi}$), and *effectiveness* (i.e. average correctness of $\hat{\Phi}$'s classification behaviour) are different measures of success for a learner. However, effectiveness is usually considered the most important criterion, since in most applications one is willing to trade training time and classification time for correct decisions. Also, it is the most reliable one when it comes to comparing different learners, since efficiency depends on too volatile parameters (e.g. different software/hardware platforms). As a result, we will only measure the success of our approach in terms of effectiveness.

In multiclass TC, effectiveness is usually equated to *accuracy*, which is defined as the percentage of classification decisions that are actually correct.

4 Automatic coding of open-ended surveys by a Text Categorization approach

In this section we describe how survey coding may be handled as a text categorization task, i.e. as the task of automatically generating a classifier that automatically selects, from a set of predefined codes, the correct code to attach to a given answer. The set of all answers to a given question q play the role of the domain $\mathcal{D}$, and the set of all possible codes that may be attributed to an answer to question q play the role of the set of categories $\mathcal{C}$ (coding the answers to different questions thus corresponds to different TC tasks).

The input to the learners (and to the classifiers, once they have been built), consists of a set of answers d_j represented as vectors of term weights $\boldsymbol{d}_j = \langle w_{1j}, \ldots, w_{|\mathcal{T}|j} \rangle$. Note that the fact that many of the answers in the corpus are ill-formed (like in the sentence "`my boyfriend went to court yestreday and if he doesn't have a drivers liceces insurance by the 28th of this month he goes to jail for 6 months`", extracted from the NORC corpus *angry_at* described in Section 5) makes the bag of words approach to representation even more appropriate: given that current linguistic analysis techniques have not proven worthy (i.e. well-performing and robust at the same time) in standard TC, where we usually deal with syntactically well-formed text, it is easy to conjecture that they could hardly prove worthy here.

Giorgetti and Sebastiani [5] have run a series of experiments with two different classifier-learning methods. The first learner used is a probabilistic naïve Bayesian learner, as implemented in the RAINBOW package[4]. Probabilistic text classification methods assume that the data was generated by a parametric model, and use the training set to estimate the parameters of this model. Bayes' theorem then allows to estimate from this model the probability that a given category has generated the document to be classified; classification thus consists in selecting the category with the highest probability.

The second learning method used is a *multiclass support vector machine* (SVM) learner as embodied in the MCSVM software[5]. SVMs attempt to learn a hyperplane in $|\mathcal{T}|$-dimensional space that separates the positive training examples of category c_i from the negative ones with the maximum possible margin, i.e. such that the minimal distance between the hyperplane and a training example is maximum; results in computational learning theory indicate that this tends to minimize the generalization error, i.e. the error of the resulting classifier on yet unseen examples. SVMs were initially conceived for solving binary classification problems, and only recently they have been adapted to multiclass classification.

Crammer and Singer describe in [2] an algorithmic implementation of multiclass SVMs based on a notion of margin generalized to multiclass problems, which allows to train directly a multiclass classifier (while in most of previous work the multiclass problem is decomposed into multiple independent binary classification tasks [6]).

Regarding effectiveness, the text categorization literature has shown that naïve Bayesian approaches are, with respect to other learning methods, no more than average performers (see e.g. [4, 7, 9, 17]). On the contrary, support vector machines are currently (together with "boosting"-based classifier committees) the unsurpassed top performers in the TC field [4, 7]. The reason why we experiment with RAINBOW is that we want to show that a text categorization approach to survey coding is much more effective than the dictionary-based approach *regardless of the specific*

[4] RAINBOW was implemented by Andrew McCallum and can be downloaded from `http://www.cs.cmu.edu/~mccallum/rainbow`.

[5] MCSVM was implemented by Koby Crammer and Yoram Singer, and we were kindly provided with a pre-release version of it.

learning method adopted, i.e. that even with an average-performing learning method our text categorization approach to survey coding can outperform the dictionary-based method. Instead, the reason why we experiment with MCSVM is that we want to show what level of effectiveness this approach can achieve, once instantiated with a top-performing learning algorithm[6].

A binary representation as input to RAINBOW, and a non-binary one as input to MCSVM have been adopted. This is due to the fact that the probabilistic models upon which RAINBOW is based require binary inputs, while this is not the case for SVMs. In the binary representation, w_{kj} represents just presence or absence of term t_k in answer d_j. Our non-binary representation is instead the $tfidf$ function in its standard "ltc" variant [11], i.e.

$$tfidf(t_k, d_j) = tf(t_k, d_j) \cdot \log \frac{|Tr|}{\#_{Tr}(t_k)} \tag{1}$$

where $\#_{Tr}(t_k)$ denotes the number of answers in the training set Tr in which t_k occurs at least α times and

$$tf(t_k, d_j) = \begin{cases} 1 + \log \#(t_k, d_j) & \text{if } \#(t_k, d_j) > 0 \\ 0 & \text{otherwise} \end{cases}$$

where $\#(t_k, d_j)$ denotes the number of times t_k occurs in answer d_j. Weights obtained by Equation 1 are normalized by cosine normalization, yielding

$$w_{kj} = \frac{tfidf(t_k, d_j)}{\sqrt{\sum_{s=1}^{|\mathcal{T}|} tfidf(t_s, d_j)^2}} \tag{2}$$

In all the experiments, stop words, punctuation, and numbers, have been removed, and all letters have been converted to lowercase. No feature selection (see e.g. [13, Section 5.1]) has been performed. The reason is that, as shown in extensive experiments by Brank et al. [1], the effectiveness of SVMs is usually worsened by feature selection, irrespectively of the feature selection algorithm used and of the chosen reduction factor (this is also independently confirmed by the results of [15]), and the effectiveness of naïve Bayesian methods does not show systematic patterns of improvement either.

5 Experimental settings and results

As already pointed out, the experiments have been carried out on data from NORC's General Social Survey. This survey, which is ongoing since 1972, aims at investigating how people assess their physical and mental health, the balancing of security and civil liberties, external and internal security threats, intergroup relations and cultural pluralism, religious congregations, etc. We deal with three datasets (see Table 1) from the NORC General Social Survey administered in 1996. Each of these datasets (here nicknamed *angry_at*, *angry_why*, and *brkdhlp*) consists of a set of answers to a given question, plus their associated category codes manually chosen by NORC's professional coders from a predefined set of category codes[7]. The task consists in choosing exactly one code for each answer.

We have chosen these three datasets because they are the same datasets used in [16], which means that we will be able to obtain a direct comparison between the

[6] Note that there are no published results yet concerning the application to TC of *multi-class* SVMs, because multiclass SVMs are a recent development, and because most TC applications are binary. The assumption that multiclass SVMs would be a top performer

Dataset	Category	# of instances
angry_at	ANGRYFAM	275
	ANGRYWRK	345
	ANGRYGVT	74
	WRK&GVT	8
	WRK&FAM	27
	FAM&GVT	16
	OTHER	625
	total	1370
angry_why	SELF	29
	PREVENTED	36
	CRITICAL	88
	DEMANDING	60
	EXPECT	196
	OTHER	51
	total	460
brkdhlp	FAMILY	57
	FRIEND	33
	GROUP	2
	CLERGY	55
	PSYCHIATRIST	56
	AGENCY	16
	OTHER	148
	total	367

Table 1. Distribution of the categories in the three datasets used in the experiments.

effectiveness of a method representative of the dictionary-based approach to survey coding and the effectiveness of our supervised learning approach.
For each dataset, the experiments were carried out according to the following main steps:

1. preprocess the data in order to obtain a data format compatible with the learners (this had to be repeated once for RAINBOW and once for MCSVM, since the two systems require different data representations);
2. partition the set of answers in each dataset in four random disjoint subsets of equal size;
3. run the learner to generate a classifier, using three of the four subsets as the training set (75% of data) and the fourth as the test set (25% of data);
4. run the classifier to categorize the data in each test set of each dataset and evaluate the results in terms of accuracy.

In order to achieve better statistical significance, in all experiments steps 3 and 4 were repeated four times, for all four possible choices of the test set. Each of the results we report is thus the result of averaging across four different experiments. We have computed the accuracy on the three datasets both with RAINBOW and with MCSVM; the results are reported in Table 2, where they are compared with the accuracy obtained in [16] on the same datasets.

The first observation we can make is that the supervised learning approach to survey coding significantly outperforms the dictionary-based approach: the improvements with respect to the best-performing method reported in [16] are significant, a +18% on average for RAINBOW and a +26% for MCSVM. The improvement is especially noteworthy on the "non-obvious" datasets: for instance, *angry_why* appears to be a hard to characterize dataset, as shown by the poor performance of the two dictionary-based methods, and on this dataset the supervised learning methods improve up to +43% with respect to them. The *angry_at* dataset looks somehow "easier" than *angry_why*, as witnessed by the fact that all four methods listed in Table 2 perform better on *angry_at* than on *angry_why*. This might also be explained by the fact that, as it can be seen from Table 1, it contains more data, since each category in *angry_at* has 195 positive examples on average, while this goes down to 76 for *angry_why*. On the contrary, the *brkdhlp* dataset seems easy to tackle by simple Boolean rules, as shown by the .747 accuracy figure of the Boolean method; in this case RAINBOW underperforms the Boolean method by 13%, while MCSVM virtually delivers the same performance as the Boolean method.

Moreover, the supervised learning approach delivers a more stable performance across the three datasets, since the reductions in standard deviation with respect to the same best-performing method are very significant, a -28% for RAINBOW and a -15% for MCSVM. These improvements are even more significant once we remember that they are obtained by a method that is much cheaper than the dictionary-based method in terms of expert humanpower.

---

once used in a multiclass TC context is based on the top performance that multiclass SVMs have delivered in multiclass application contexts other than TC [2].

[7] The *angry_at* and *angry_why* datasets actually involve the same question, which deals with the description of a situation that caused anger to the respondent; each answer was classified according to *two* different sets of codes, one concerning the object of anger, the other concerning the cause of anger. Actually, *angry_why* contains only a subset of the answers contained in *angry_at*, in the sense that NORC coders classified some of the answers only according to the *angry_at* set of codes. The *brkdhlp* dataset (called *breakdown* in [16]) consists of answers to the question as to what source of help was used to deal with a nervous breakdown.

	Dictionary-Based [16]		Supervised Learning	
	Vector	Boolean	RAINBOW	MCSVM
angry_at	0.451	0.465	0.714 (+54%)	**0.756 (+63%)**
angry_why	0.211	0.272	**0.389 (+43%)**	0.376 (+38%)
brkdhlp	0.646	**0.747**	0.653 (-13%)	0.746 (-0.13%)
Average	0.436	0.495	0.585 (+18%)	**0.626 (+26%)**
Std. Dev.	0.218	0.239	**0.173 (-28%)**	0.216 (-10%)

Table 2. Comparative accuracy results obtained on the *angry_at*, *angry_why* and *brkdhlp* datasets using a Boolean and a vector-based method and using a naïve Bayesian and a multiclass SVM TC methods. The percentile improvements in accuracy and average accuracy, and the percentile reductions in standard deviation, are reported with respect to the Boolean method, the best dictionary-based method in [16]. **Boldface** indicates the best performance on the dataset.

The fact that improvements of this order of magnitude are obtained even with a method, such as the naïve Bayesian technique implemented in RAINBOW, which is known as an average performer in the text categorization literature, bears witness to the superiority of the supervised learning approach to survey coding.

The fact that multiclass SVMs, known top-performers in the machine learning literature (see e.g. [2]), outperform RAINBOW, only by a very small margin, is more surprising. A possible explanation to this might be that the vocabulary of the NORC corpora exhibits low internal stochastic dependence, hence approximating the conditions under which Bayesian approaches are theoretically optimal.

6 Conclusion and future work

We have shown that automatic coding of responses to open-ended survey questions may be posed as a multiclass text categorization problem, and that text categorization techniques based on supervised learning may significantly outperform dictionary-based techniques, such as those used in [16], that have been up to now the dominant approach to automated survey coding. Another advantage of the supervised learning approach with respect to the dictionary-based approach, which requires that the text classifiers be handcrafted (by a knowledge engineer and a social scientist working together), is that the classifiers can be generated automatically from the training data, with substantive savings in terms of expert humanpower.

The effectiveness levels that text categorization techniques have achieved in our experiments are far from being perfect, and also from being completely satisfactory. Although the results obtained in our research are promising, we think that more research is needed for the automatic approach to survey coding to clearly supersede the manual approach.

Acknowledgements

The open-ended text used in this work was collected in the General Social Surveys of the National Opinion Research Center (NORC), University of Chicago, and supplied by NORC to the authors. We are grateful to Tom Smith and Jennifer Berktold for providing these texts and for assisting us in their interpretation. We are also grateful

to Koby Crammer, Yoram Singer and Andrew McCallum for making the MCSVM and RAINBOW packages available, to Peter Viechnicki for clarifying several points of his experiments, to Henri Avancini for helping with several preprocessing issues.

References

1. Janez Brank, Marko Grobelnik, Natasa Milić-Frayling, and Dunja Mladenić. Interaction of feature selection methods and linear classification models. In *Proceedings of the ICML-02 Workshop on Text Learning*, Sydney, AU, 2002.
2. Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.
3. James A. Davis and Tom Smith. *General Social Surveys, 1972-1996: Cumulative Codebook*. National Opinion Research Center, Chicago, US, 1996.
4. Susan T. Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In Georges Gardarin, James C. French, Niki Pissinou, Kia Makki, and Luc Bouganim, editors, *Proceedings of CIKM-98, 7th ACM International Conference on Information and Knowledge Management*, pages 148–155, Bethesda, US, 1998. ACM Press, New York, US.
5. Daniela Giorgetti and Fabrizio Sebastiani. Automating survey coding by multiclass text categorization techniques. *JASIST*, 2004.
6. Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.
7. Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE. Published in the "Lecture Notes in Computer Science" series, number 1398.
8. David D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In Nicholas J. Belkin, Peter Ingwersen, and Annelise Mark Pejtersen, editors, *Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval*, pages 37–50, Kobenhavn, DK, 1992. ACM Press, New York, US.
9. Hang Li and Kenji Yamanishi. Text classification using ESC-based stochastic decision lists. *Information Processing and Management*, 38(3):343–361, 2002.
10. Stefania Macchia and Manuela Murgia. Coding of textual responses: Various issues on automated coding and computer assisted coding. In *Proceedings of JADT-02, 6th International Conference on the Statistical Analysis of Textual Data*, pages 471–482, St-Malo, FR, 2002.
11. Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988. Also reprinted in [14], pp. 323–328.
12. Hinrich Schütze, David A. Hull, and Jan O. Pedersen. A comparison of classifiers and document representations for the routing problem. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval*, pages 229–237, Seattle, US, 1995. ACM Press, New York, US.
13. Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.
14. Karen Sparck Jones and Peter Willett, editors. *Readings in information retrieval*. Morgan Kaufmann, San Mateo, US, 1997.
15. Hirotoshi Taira and Masahiko Haruno. Feature selection in SVM text categorization. In *Proceedings of AAAI-99, 16th Conference of the American Association for Artificial Intelligence*, pages 480–486, Orlando, US, 1999. AAAI Press, Menlo Park, US.

16. Peter Viechnicki. A performance evaluation of automatic survey classifiers. In Vasant Honavar and Giora Slutzki, editors, *Proceedings of ICGI-98, 4th International Colloquium on Grammatical Inference*, pages 244–256, Ames, US, 1998. Springer Verlag, Heidelberg, DE. Published in the "Lecture Notes in Computer Science" series, number 1433.
17. Yiming Yang and Xin Liu. A re-examination of text categorization methods. In Marti A. Hearst, Fredric Gey, and Richard Tong, editors, *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, pages 42–49, Berkeley, US, 1999. ACM Press, New York, US.