Text Categorization for Automated Survey Coding

Daniela Giorgetti
Istituto di Linguistica Computazionale
Consiglio Nazionale delle Ricerche
56124 Pisa, Italy
E-mail: daniela.giorgetti@ilc.cnr.it

Fabrizio Sebastiani
Istituto di Elaborazione dell'Informazione
Consiglio Nazionale delle Ricerche
56124 Pisa, Italy
E-mail: fabrizio@iei.pi.cnr.it

Abstract

Survey coding is the task of assigning a symbolic code from a predefined set of such codes to the answer that a person has given in response to an open-ended question of a questionnaire (aka *survey*). This task is usually carried out in order to group respondents according to a predefined scheme based on their answers.

Survey coding is a difficult task, since the code that should be attributed to a respondent based on the answer she has given is a matter of subjective judgment, and thus requires expertise. It is thus unsurprising that this task has traditionally been performed manually, by trained coders. Some feeble attempts have been made at automating this task, most of them based on detecting the similarity between the answer and textual descriptions of the meanings of the candidate codes.

We take a radically new stand, and formulate the problem of automated survey coding as a *text categorization* problem, i.e. as the problem of classifying the answers into categories belonging to a predefined classification scheme. Following common practice, we tackle text categorization by means of supervised machine learning techniques, i.e. we learn a model of the association between answers and categories from a training set of pre-coded answers, and apply the resulting model to the classification of new answers.

In this paper we experiment with two different learning techniques, one based on naïve Bayesian classification and one on support vector machines, and test the resulting framework on a corpus of social surveys. The results we have obtained by far outperform the results obtained by previous automated survey coding approaches.

1 Introduction

Survey coding is the task of assigning a symbolic code from a predefined set of such codes to a textual expression representing the answer that a person has given in response to an open-ended question of a questionnaire, or *survey*. By *open-ended* we mean a question that requires or allows an answer consisting of free text, i.e. text not in a predefined format; open-ended questions are the opposite of *multiple-choice* questions, which instead require "checking" one among a predefined set of answers[1].

[1] In this paper we will only deal with text in *written* form. The implications of coding survey material in *audio* form are discussed in Section 6.

Survey coding is usually carried out in order to classify responses (and respondents) into a predefined scheme, thus superimposing a structure on what would otherwise be a totally unstructured corpus of questions and answers. Survey coding has several applications, especially in the social sciences, ranging from the simple classification of respondents to the extraction of statistics on political opinions, health and lifestyle habits, customer satisfaction, brand fidelity, and patient satisfaction.

As an example, in 1996 interviewers asked the following question (among many) to a carefully chosen sample of 1370 subjects, in the framework of the General Social Survey [7] carried out by the US National Opinion Research Center (NORC)[2]:

```
Within the past month, think about the last time you felt really angry,
irritated or annoyed. Could you describe in a couple of sentences
what made you feel that way - what the situation was?
```

Professional coders were then asked to classify the answers in exactly one among the following categories, each consisting of a code label and a short explicatory caption:

ANGRYWRK:	Situation involved work.
ANGRYFAM:	Situation involved family.
ANGRYGVT:	Situation involved government or government officials.
WRK&FAM:	Situation involved both work and family.
WRK&GVT:	Situation involved both work and government.
FAM&GVT:	Situation involved both family and government.
OTHER:	Situation did not fit the above categories.

Sample answers[3] included

```
trying to teach my son something and he was being stubborn and wouldn't
listen to me i got angry at him
```

or

```
my wife went shopping & spent too much on a dress & it made me feel
angry
```

which coders both classified under the ANGRYFAM header.

Survey coding is a difficult task, since the code that should be attributed to a respondent based on the answer she has given is a matter of subjective judgment, and thus requires expertise. For instance, different coders, especially if little trained, might have different opinions as whether the answer

```
when people in authorities arent treating people right
```

should be classified under ANGRYGVT, or ANGRYWRK, or WRK&GVT, or even under OTHER.

Given the difficulty of the task, it is thus unsurprising that it has traditionally been performed manually, by professional coders. This means that survey coding is also an expensive task, and this is the reason why social scientists (or other professionals in charge of designing and running surveys) tend to avoid including too many open-ended questions in their surveys, and to rely more on the less expensive multiple-choice questions, which by definition do not require a coding phase, but on the other hand strictly limit the respondents' possible answers.

[2] http://www.norc.uchicago.edu/

[3] Actually, rather than from the real answers, NORC coders work from typewritten versions of the handwritten notes taken by the interviewers. Of course, reporting the answers may originate syntactically ill-formed sentences.

Some attempts have been made in the past at automating the survey coding task. Most of them have exploited simple techniques from the tradition of text retrieval, for detecting the similarity between the answer and textual descriptions of the meanings of the candidate codes [25].

In this paper we take a radically new stand, and formulate the problem of automated survey coding as a *text categorization* problem, i.e. as the problem of classifying the answers into *categories* (or *classes*) belonging to a predefined classification scheme. Following common practice, we tackle this text categorization problem by supervised machine learning techniques, i.e. we learn a model of the association between answers and categories from a "training set" of pre-coded answers, and apply the resulting model to the classification of new answers. In this paper we experiment with two different learning techniques (one based on naïve Bayesian classification [16], and another based on support vector machines [13]) and test the resulting framework on a corpus of social surveys conducted by NORC. The results we obtain by far outperform the results obtained by previous automated survey coding approaches.

This paper is structured as follows. Section 2 introduces survey coding, and reviews related work attempting to automate this task. Section 3 gives a brief introduction to text categorization, its methodology and its main techniques. In Section 4 we frame the survey coding task as a problem of automated text categorization, and describe how the tools outlined in Section 3 can be effectively used to this end. Section 5 illustrates the experiments we have performed in the application of naïve Bayesian classification and support vector machines to the problem of coding a corpus of social surveys collected by NORC. Section 6 concludes, commenting on our results and discussing possible avenues for further research.

2 Survey coding and its automation

From a general point of view, survey analysis shares many aspects with the more general framework of *text analysis for the social sciences*, whereby quantitative methods are applied to the study of text corpora representative of a sample of a given population, in order to infer properties of the population itself. In [1] Alexa describes the various issues involved in the computer-assisted analysis of text for the social sciences, while in [2] Alexa and Züll review relevant commercial and research software packages. In this paper we will not deal with the general problem of text analysis for the social sciences, but will focus just on the coding task of the survey analysis process.

Survey coding may be viewed as the task of identifying common meaningful concepts across different responses to the same set of questions. This task may come in two variants, depending whether the concepts sought are known in advance (in which case the task really consists in checking whether a given concept is present or not in the text under analysis) or not (in which case novel, unforeseen concepts may be "discovered" in the text being analyzed). The former variant is the most common, for the simple fact that surveys are usually 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. As a consequence, it is the former variant we will concentrate on in this paper.

Performing survey coding manually has several drawbacks. First of all, it is slow, since a lot of manual effort by different professional figures is involved. For example, NORC interviewers take handwritten notes of the answers returned during the interview, and typists produce a typewritten text from these notes at a later stage; this text is then analyzed by professional coders, who perform the final coding task. This example shows that the survey coding process is not only slow but also expensive, since it requires a lot of manpower. Yet another drawback is that the process may produce faulty encodings with high probability, as there are several potential sources of error: interviewers may misread or misunderstand the answers or misrepresent them by their notes, typists may misunderstand the handwritten notes or introduce further typing errors, and coders may misinterpret the meaning either of the answers or of the codes.

A further problem in survey coding is the so-called *inter-coder agreement* problem, i.e. the fact that different coders may classify the same data in different ways simply because they have different opinions as to the meaning of the answer and/or the code. Note that this problem does not only affect *manually performed* survey coding, since we may also expect different automated methods to differ in their decisions[4].

Given that text analysis for the social sciences is an important problem, several software packages that address it have been developed. However, they are not usually tailored for the specific task of survey analysis, and the solutions that they provide for the survey coding task are fairly unsatisfactory. Most of these software packages (see [2] for a review) mostly concentrate on helping coders in coding their data *manually*, and in visualizing them in several convenient ways. A few of these packages instead do perform automatic coding, by relying mostly on specialized dictionaries. 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. One of the disadvantages of this approach is that dictionaries have to be created *before* the coding process begins, i.e. when data is still totally unknown; this approach is thus extremely static. The second drawback is that specialized dictionaries need to be developed, one for each category of interest; this requires the intervention of expert personnel, who is then responsible for deciding which words, if present (either alone or in combination) in an answer to an open-ended question, should trigger the attribution of the code to the answer. As this approach is word-based, some explicit mechanism to disambiguate word meanings is also needed, since it is usually word senses, rather than words, that belong (at least conceptually) to the dictionary relevant to the category. There exist several variations on this "dictionary approach"; for instance, Viechnicki [25] describes a method in which words that characterize categories are combined by means of Boolean operators, thus allowing a richer procedural description of categories.

The scientific literature on automating survey coding is not very rich. Dillon [9] reports doing "automatic classification" of surveys, but what he actually does is grouping *questions* (and not responses, or respondents, as we do) into groups according to similarity; also, he does not start from a predefined set of groups, but generates the groups from scratch, which means that he uses unsupervised learning (i.e. clustering), and not supervised learning as we instead do. A few other researchers (e.g. [4, 26]) have concentrated on the related issue of *grading* responses to open-ended questions, as those answered by students in their school essays. However, this task deals with concerns very different from those addressed in survey coding, since student essays have to be graded according to quality or merit, while surveys have to be coded according to topic-relatedness.

The approach that is closest in spirit to ours is probably the dictionary-based approach as represented by Viechnicki's work [25], in which responses to questions from a NORC survey are classified by means of a set of codes pre-defined by NORC social scientists. The main difference with our approach is that Viechnicki does not use any learning at all, and computes instead the similarity between the textual answer and a textual definition of the code; the code which results in the highest similarity score is chosen.

Our approach has several advantages on the dictionary-based approach. First, the manual effort is directed towards the manual coding of a 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

[4] *Inter-indexer inconsistency* is a similar phenomenon well-known in information retrieval [5]: when two human experts decide whether to index document d_j with index term c_i, they may disagree, and this in fact happens with relatively high frequency.

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). Second, 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. [22]) and has produced systems whose effectiveness rivals or exceeds that of a human (i.e. systems capable of generating codes that correlate with those 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, and since the same survey (e.g. the General Social Survey mentioned in Section 1) is run from year to year, with many questions being asked unmodified year after year.

3 A short introduction to text categorization

Text categorization (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 categories and $\mathcal{D}$ is a domain of documents. 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 a procedural or declarative nature) of their meaning is usually available. It is usually the case that no metadata (such as e.g. publication date, document type, publication source) are available either; therefore, classification must be accomplished only on the basis of knowledge extracted from the documents themselves.

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 with relatively high frequency. A news article on Clinton attending Dizzy Gillespie's funeral could be filed under Politics, or under Jazz, or under both, or even under neither, depending on the subjective judgment of the expert.

Depending on the application, TC may be either *single-label* (i.e. exactly one $c_i \in \mathcal{C}$ must be assigned to each $d_j \in \mathcal{D}$), or *multi-label* (i.e. any number $0 \leq n_j \leq |\mathcal{C}|$ of categories may be assigned to a document $d_j \in \mathcal{D}$). A special case of single-label TC is *binary* TC, in which, given a category c_i, each $d_j \in \mathcal{D}$ must be assigned either to c_i or to its complement $\overline{c}_i$. Multi-label TC under $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$ is usually tackled as $|\mathcal{C}|$ independent binary classification problems under $\{c_i, \overline{c}_i\}$, for $i = 1, \ldots, |\mathcal{C}|$. A *classifier for* c_i is then a function $\hat{\Phi}_i : \mathcal{D} \rightarrow \{T, F\}$ that approximates the unknown target function $\Phi_i : \mathcal{D} \rightarrow \{T, F\}$.

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 [22].

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* $\vec{d_j} = \langle w_{1j}, \ldots, w_{|\mathcal{T}|j} \rangle$. Here, $\mathcal{T}$ is the *dictionary*, i.e. the set of *terms* (also known as *features*) that occur at least once in at least k documents, and $0 \leq w_{kj} \leq 1$ quantifies the importance of t_k in characterizing the semantics of d_j. Typical values of k 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 document (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). A popular choice is to add to the set of words or stems a set of *phrases*, i.e. longer (and semantically more significant) language units extracted from the text by shallow parsing and/or statistical techniques. Concerning (ii), either statistical or probabilistic techniques are used to compute terms weights, the former being the most common option. One popular class of statistical term weighting functions is $tf * idf$ (see e.g. [21]), 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 $tf * idf$ techniques are often normalized so as to contrast the tendency of $tf * idf$ to emphasize long documents.

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. Alternatively, dimensionality reduction may take the form of *feature extraction*: a set of "artificial" terms is generated from the original term set (by techniques such as supervised or unsupervised term clustering, or latent semantic indexing) in such a way that the newly generated terms are both fewer and stochastically more independent from each other than the original ones used to be.

3.2 Classifier learning

A text classifier for c_i is automatically generated by a general inductive process (the *learner*) which, by observing the characteristics of a set of documents preclassified under c_i or $\overline{c}_i$, gleans the characteristics that a new unseen document should have in order to belong to c_i. In order to build classifiers for $\mathcal{C}$, one thus needs a 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 is the set of documents observing which the learner builds the classifier. The validation set is the set of documents on which the engineer fine-tunes 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 the set on which the effectiveness of the classifier is finally evaluated. 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.

Different learners have been applied in the TC literature, 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. Some of these methods generate binary-valued classifiers of the required form $\hat{\Phi} : \mathcal{D} \times \mathcal{C} \rightarrow \{T, F\}$, but some others generate real-valued functions of the form $CSV : \mathcal{D} \times \mathcal{C} \rightarrow [0, 1]$ (CSV standing for *categorization status value*). For these latter, a set of thresholds τ_i needs to be determined (typically, by experimentation on a validation set) allowing to turn real-valued CSVs into the final binary decisions.

3.3 Classifier evaluation

Training efficiency (i.e. average time required to build a classifier $\hat{\Phi}_i$ from a given corpus Ω), as well as *classification efficiency* (i.e. average time required to classify a document by means of $\hat{\Phi}_i$), and *effectiveness* (i.e. average correctness of $\hat{\Phi}_i$'s classification behaviour) are 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 sw/hw platforms).

In TC, effectiveness wrt category c_i is often measured by a combination of *precision wrt* c_i (π_i), the percentage of documents deemed to belong to c_i that in fact belong to it, and *recall wrt* c_i (ρ_i), the percentage of documents belonging to c_i that are in fact deemed to belong to it. Since it is often the case that a classifier can be arbitrarily tuned to emphasize one at the expense of the other, only combinations of the two are significant, the most popular combination nowadays being their harmonic mean $F_1(i) = \frac{2\pi_i\rho_i}{\pi_i+\rho_i}$. However, in single-label TC, effectiveness wrt category c_i is often measured by *accuracy wrt* c_i (α_i), the percentage of classification decisions that are in fact correct, since the tuning described above is not possible[5]. When effectiveness is computed for several categories, the results for individual categories must be averaged in some way; here, one may opt for *microaveraging* ("categories count proportionally to the number of their positive training examples") or for *macroaveraging* ("all categories count the same"), depending on the application (see Table 1). The former rewards classifiers that behave well on *frequent categories* (i.e. categories with many positive training examples), while classifiers that perform well also on infrequent categories are emphasized by the latter.

4 Automated survey coding by text categorization

In this section we describe our experiments with survey coding handled as a text categorization task, i.e. as the task of learning a classifier that automatically selects, from a set of predefined codes the correct code (or codes) to attach to a given answer.

We have run a series of experiments with two different classifier learning methods. The first learner we use is a probabilistic naïve Bayesian classifier, as implemented in the RAINBOW package developed by McCallum [18]. Probabilistic text classification methods assume that the data was generated by a parametric model, and use the training documents to estimate the parameters of this model. From this generative model, Bayes'

[5] To see this, note that in single-label TC if a document d_j is incorrectly classified under category c_m (a so-called *false positive* of c_m), which results in a decrement of π_m, it is also not classified under the correct category c_n (a so-called *false negative* of c_n), which results in a decrement of ρ_n.

	Microaveraging	Macroaveraging
Precision (π)	$\pi = \frac{\sum_{i=1}^{\lvert\mathcal{C}\rvert} TP_i}{\sum_{i=1}^{\lvert\mathcal{C}\rvert} TP_i + FP_i}$	$\pi = \frac{\sum_{i=1}^{\lvert\mathcal{C}\rvert} \pi_i}{\lvert\mathcal{C}\rvert} = \frac{\sum_{i=1}^{\lvert\mathcal{C}\rvert} \frac{TP_i}{TP_i + FP_i}}{\lvert\mathcal{C}\rvert}$
Recall (ρ)	$\rho \frac{\sum_{i=1}^{\lvert\mathcal{C}\rvert} TP_i}{\sum_{i=1}^{\lvert\mathcal{C}\rvert} TP_i + FN_i}$	$\rho = \frac{\sum_{i=1}^{\lvert\mathcal{C}\rvert} \rho_i}{\lvert\mathcal{C}\rvert} = \frac{\sum_{i=1}^{\lvert\mathcal{C}\rvert} \frac{TP_i}{TP_i + FN_i}}{\lvert\mathcal{C}\rvert}$
Accuracy (α)	$\alpha = \frac{\sum_{i=1}^{\lvert\mathcal{C}\rvert} TP_i + TN_i}{\sum_{i=1}^{\lvert\mathcal{C}\rvert} TP_i + TN_i + FP_i + FN_i}$	$\alpha = \frac{\sum_{i=1}^{\lvert\mathcal{C}\rvert} \alpha_i}{\lvert\mathcal{C}\rvert} = \frac{\sum_{i=1}^{\lvert\mathcal{C}\rvert} \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i}}{\lvert\mathcal{C}\rvert}$

Table 1: Effectiveness measures used in text categorization; TP_i, TN_i, FP_i and FN_i refer to the sets of true positives, true negatives, false positives, and false negatives wrt c_i, respectively.

theorem allows the derivation of a predictive model that once instantiated by the parameter estimates computed in the previous phase, computes the probability that a given category has generated the document to be classified. Classification thus consists in selecting the category with the highest probability. There are two well-known variants of this model, the multi-variate Bernoulli model and the multinomial model. In this paper we chose the latter, since in comparative text classification experiments it has given superior results than the former [19].

The second learning method is a support vector machine (SVM) learner as embodied in the SVMLIGHT package (version 4.0) implemented by Joachims [14]. SVMs attempt to learn a hyperplane in $|\mathcal{T}|$-dimensional space that separates the positive training examples 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. We have simply opted for the default parameter setting of SVMLIGHT; in particular, this means that a linear kernel has been used.

With respect to effectiveness, the text categorization literature has shown that naïve Bayesian approaches of the type implemented in RAINBOW are, relatively to other learning methods, no more than average performers [10, 13, 17, 22, 29]. On the contrary, the same literature has shown that support vector machines of the type implemented in SVMLIGHT are (together with "boosting"-based classifier committees) the unsurpassed top performers [10, 13, 17, 22, 29]. 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 learning method adopted*, i.e. that even with an average-performing learning method our text categorization approach to survey coding can largely outperform the dictionary-based method. Instead, the reason we experiment with SVMLIGHT is that we want to show what level of effectiveness this approach can achieve, once instantiated with a top-performing learning algorithm.

As for the document indexing phase, we have used a standard term-based representation of the type discussed in Section 3.1; in our context the documents coincide with the individual answers that respondents have given to the same question (w_{kj} thus represents the weight of term t_k in answer d_j). Note that the fact that many of the answers in the corpus are syntactically ill-formed (see Section 1) makes the "bag of words" approach to representation (i.e. the approach that just considers term occurrence and frequency of occurrence, disregarding deeper syntactic and semantic aspects) even more appropriate: if syntactic and semantic analysis have not proven worthy in standard text categorization, where we usually deal with syntactically well-formed text, they could hardly prove worthy here!

We have used a binary representation as input to the first learner, and a weighted one as input to the second learner. This is due to the fact that the naïve Bayesian method we have used (as most other probabilistic methods) requires binary inputs, while this is not the case for SVMs. Our weighted representation in input to SVMLIGHT is the *tfidf* function in its standard "ltc" variant [21], 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 once and

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

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

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

In all the experiments discussed in this paper, stop words, punctuation, and numbers, have been removed, and all letters have been converted to lowercase. No dimensionality reduction has been performed. The reason is that, as shown in extensive experiments by Brank et al. [3], 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 [24]), and that of naïve Bayesian methods does not show systematic patterns of improvement either.

5 Experiments

As already pointed out, our 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 two datasets (see Table 2) from the NORC General Social Survey administered in 1996. Each of these datasets (here nicknamed *angry_at* and *angry_why*) consists of a set of answers to a given question, and their associated category code manually chosen by NORC's professional coders from a pre-defined set of category codes; the task consists in choosing exactly one category code for each answer[6].

Therefore, note that the task we are dealing with here is a *single-label* categorization task, as defined at the beginning of Section 3. As a result, we have measured effectiveness in terms of accuracy (defined as in Section 3.3), which has also the advantage that we can directly compare our results with those of [25], which are also presented in

[6] 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).

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

Table 2: Characteristics of the two datasets used in this experiment.

terms of accuracy. Both macroaveraging and microaveraging were computed (in [25] only macroaveraging results are reported).

While the RAINBOW package is explicitly geared towards single-label categorization, this is not true of the SVMLIGHT package, which is designed for binary (hence also for multi-label) categorization. This means that, when using SVMLIGHT in order to choose among $|\mathcal{C}|$ codes, we learn $|\mathcal{C}|$ binary classifiers that behave independently of each other. This means that SVMLIGHT might classify an answer under no category at all, or might classify an answer under more than one category; in both cases accuracy will be penalized, since in the former case there will be one false negative for the true category c_m, while in the latter case there will be one or more false positives, each for a category c_n different from the true category[7].

We have chosen these two datasets because these are the same datasets used in Viechnicki's work [25], which means that we will be able to obtain a direct comparison between the effectiveness of his method (which is representative of the dictionary-based approach to survey coding) and the effectiveness of ours [8].

Note that both datasets include a class OTHER. This consideration alone indicates that these datasets are not "easy", since

- the classifier cannot simply "take a guess" by picking "the least inappropriate" category, since if all choices are sufficiently inappropriate, the category OTHER applies[9];

[7] An alternative policy might have been to pick the category c_i for which SVMLIGHT returns the highest categorization status value. Unfortunately, also this is notoriously not an optimal policy for binary SVMs; a possible way out would be to use SVM-based methods which are explicitly geared to single-label categorization, such as those proposed in [6, 20], or to use some general means of using multi-label learners in a single-label context, such as *error-correcting output codes* [8, 11]. Our use of SVMLIGHT in this context should thus be considered as a first attack to this problem.

[8] Viechnicki [25] actually used a third dataset, nicknamed *breakdo*. We have not experimented on it because neither Viechnicki nor NORC could locate a copy of it to give us.

[9] This is confirmed by everybody's experience with multiple-choice tests. No student likes to take a multiple-choice test in which the last choice is always "None of the preceding answers apply"!

	Vector [25]	**Boolean [25]**	RAINBOW	SVMLIGHT
angry_at	0.451	0.464	0.842 (+81%)	**0.862 (+85%)**
angry_why	0.210	0.271	0.793 (+292%)	**0.828 (+305%)**
average	0.330	0.367	0.817 (+222%)	**0.845 (+230%)**

Table 3: Comparative macroaveraged accuracy results obtained on the *angry_at* and *angry_why* datasets using a Boolean and a vector-based method (results reported from [25]), and using a naïve Bayesian and a support vector machine text categorization methods. The percentile improvements are reported with respect to the Boolean method. **Boldface** indicates the best performance on the dataset.

- the category OTHER will typically be a very hard category to work with, since it will not be characterized by a specific terminology, as is instead the case with categories that are strongly characterized in a topical sense. The presence of a category OTHER in the category set always tends to deteriorate the global performance of any text classifier.

For each dataset, the main steps we went through to run our experiments are the following:

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 SVMLIGHT);
2. randomly split each dataset into a training set (consisting of 75% of the answers) and a test set (the remaining 25%);
3. run the learner to learn a classifier from the training set. Note that in SVMLIGHT this has meant learning $|C|$ binary classifiers, one for each category; in this process, all the positive training examples of categories $c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{|C|}$ have been used as negative training examples of c_i.
4. run the classifier to classify the data in the test set. Similarly to the previous step, all the positive test examples of categories $c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{|C|}$ have been used as negative test examples of c_i.
5. evaluate the results.

In order to achieve better statistical significance, we repeated Steps 2 to 5 four times, i.e. we partitioned the dataset in four random disjoint subsets and used each of them in turn as the testing set while using the other three as the training set. The results we report are thus averaged across four different experiments.

We have computed micro- and macroaveraged accuracy on the two datasets both with RAINBOW and SVMLIGHT; the results are reported in Tables 3 and 4.

The first observation we can do is that the text categorization approach to survey coding vastly outperforms the dictionary-based approach: the improvements with respect to the best-perfoming method reported in [25] are outstanding, a +222% on average for RAINBOW and a +230% for SVMLIGHT. As anticipated in Section 4, the fact that improvements of this order of magnitude are obtained even with a method, such as the naïve Bayesian technique implemented in RAINBOW, that is known as an average performer in the text categorization literature, bears witness to the superiority of the text categorization approach to survey coding.

The fact that SVMLIGHT, a known top-performer in the text categorization literature, outperforms RAINBOW only by a very small margin (3%) is probably due to the fact that, as observed earlier, SVMLIGHT is a multi-label system, and using it in a single-label

	Rainbow	SVMlight	Rainbow	SVMlight
	Micro	Micro	Macro	Macro
angry_at	0.894	0.914	0.842	**0.862**
angry_why	0.808	0.828	0.793	**0.828**
average	0.851	0.871	0.817	**0.845**

Table 4: Micro- and macroaveraged accuracy results obtained on the *angry_at* and *angry_why* datasets using a naïve Bayesian and a support vector machine text categorization methods.

context such as ours can be done only with some straight-jacketing. The application of single-label versions of SVMs to our task is thus a promising avenue of further research.

The *angry_at* dataset looks somehow "easier" than *angry_why*, as witnessed by the fact that all four methods listed in Table 3 perform better on *angry_at* than on *angry_why*. This might also be explained by the fact that, as can be seen from Table 2, 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*. This is confirmed by the results we have obtained on a third dataset, nicknamed *mental*, consisting of 3000 answers classified under 7 categories (an average of 428 positive examples each): the macroaveraged accuracy of SVMlight jumps to 0.935 (8% better than on *angry_at* and 12% better than on *angry_why*), and the one of Rainbow jumps to 0.918 (9% better than on *angry_at* and 15% better than on *angry_why*)[10].

From Table 4 we may further observe that both Rainbow and SVMlight achieve a higher performance on microaveraging than on macroaveraging. This confirms results from the text categorization literature [22, 28], according to which macroaveraged effectiveness measures are always a tougher benchmark for text classifiers than the corresponding microaveraged ones. The reason for this is that, quite naturally, microaveraged effectiveness tends to be heavily determined by the performance of the classifiers on the most populated categories, and these categories are obviously the easiest for any supervised learning method.

6 Conclusion

We have shown that automatic coding of responses to open-ended survey questions may be posed as a text categorization problem, and that text categorization techniques largely outperform the dictionary-based techniques based on similarity computation that have been up to now the dominant approach to automated survey coding.

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.

There are several avenues for further research. One of these is to experiment with single-label versions of SVMs (as discussed in Footnote 7), in order to improve upon the results of SVMlight. Another, which we are currently working at, is simply to run experiments on more survey data, in order to obtain results which are more statistically reliable.

[10] We do not report in detail the results obtained on *mental*, since it was not used in [25] and is thus not useful for our comparisons with the dictionary-based approach. The *mental* dataset deals with questions and answers concerning mental illnesses.

In the future, we plan to combine automated survey coding by text categorization with speech recognition, in order to allow the survey coding task to proceed directly from the audio recording of the interview, since we believe that survey coding may be performed with much better effectiveness only by using better quality input, i.e. more faithful representations of the answers. Proceeding directly from the audio recording can eliminate the sources of noise mentioned in Section 2 (i.e. the noise possibly introduced by interviewer and typist), and also makes for greater savings in term of manpower, which means that the researchers who design the survey could afford having more open-ended questions and less multiple-choice ones. However, such a process would entail the need to apply text categorization techniques to noisy text, since speech recognition software performs imperfectly, especially in dealing with natural speech in possibly noisy environments. To this respect, we think that there are reasons for optimism, since research in text categorization of noisy text [12, 15] has already shown that, by employing noisy texts also in the training phase (i.e. texts affected by the same source of noise that is also at work in the test documents), effectiveness levels comparable to those obtainable in the case of standard text can be achieved. Although the source of noise dealt with in [12, 15] was different (i.e. noise resulting from optical character recognition) there are reasons to believe that similar effectiveness patterns might result also in the case of noise introduced by speech recognition.

Acknowledgements

The open-ended text used in this work was collected on 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 Thorsten Joachims and Andrew McCallum for making the SVMlight and Rainbow packages available, to Peter Viechnicki for clarifying several points of his experiments, and to David Lewis for suggesting pointers to the automated survey coding literature.

References

[1] Melina Alexa. Computer-assisted text analysis methodology in the social sciences. Technical Report 97/07, Zentrum für Umfragen, Methoden und Analysen, Mannheim, DE, 1997.

[2] Melina Alexa and Cornelia Züll. Text analysis software: Commonalities, differences and limitations. The results of a review. *Quality and Quantity*, 34:299–321, 2000.

[3] 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. Forthcoming.

[4] Jill Burstein, Karen Kukich, Susanne Wolff, Chi Lu, Martin Chodorow, Lisa Braden-Harder, and Mary Dee Harris. Automated scoring using a hybrid feature identification technique. In *Proceedings of the ACL-98, 36th Annual Meeting of the Association for Computational Linguistics*, pages 206–210, Montreal, CA, 1998.

[5] Cyril Cleverdon. Optimizing convenient online access to bibliographic databases. *Information Services and Use*, 4(1):37–47, 1984. Also reprinted in [27], pp. 32–41.

[6] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

[7] James A. Davis and Tom Smith. *General Social Surveys, 1972-1996: Cumulative Codebook.* Chicago, US, 1996.

[8] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[9] Martin Dillon. Automatic classification of Harris survey questions: An experiment in the organization of information. *Journal of the American Society for Information Science*, pages 294–301, 1982.

[10] 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.

[11] Rayid Ghani. Using error-correcting codes for text classification. In Pat Langley, editor, *Proceedings of ICML-00, 17th International Conference on Machine Learning*, pages 303–310, Stanford, US, 2000. Morgan Kaufmann Publishers, San Francisco, US.

[12] David J. Ittner, David D. Lewis, and David D. Ahn. Text categorization of low quality images. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 301–315, Las Vegas, US, 1995.

[13] 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.

[14] Thorsten Joachims. Making large-scale SVM learning practical. In Bernhard Schölkopf, Christopher J. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, chapter 11, pages 169–184. The MIT Press, Cambridge, US, 1999.

[15] Markus Junker and Rainer Hoch. An experimental evaluation of OCR text representations for learning document classifiers. *International Journal on Document Analysis and Recognition*, 1(2):116–122, 1998.

[16] David D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 4–15, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE. Published in the "Lecture Notes in Computer Science" series, number 1398.

[17] Hang Li and Kenji Yamanishi. Text classification using ESC-based stochastic decision lists. *Information Processing and Management*, 38(3):343–361, 2002.

[18] Andrew K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. Available from `http://www.cs.cmu.edu/ mccallum/bow`, 1996.

[19] Andrew K. McCallum and Kamal Nigam. A comparison of event models for naive Bayes text classification. In *Proceedings of the 1st AAAI Workshop on Learning for Text Categorization*, pages 41–48, Madison, US, 1998.

[20] Jason D. Rennie and Ryan Rifkin. Improving multiclass text classification with the support vector machine. Technical Report AIM-2001-026, Massachusetts Institute of Technology, Cambridge, US, 2001.

[21] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988. Also reprinted in [23], pp. 323–328.

[22] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[23] Karen Sparck Jones and Peter Willett, editors. *Readings in information retrieval.* Morgan Kaufmann, San Mateo, US, 1997.

[24] 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.

[25] 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.

[26] Dave Whittington and Helen Hunt. Approaches to the computerized assessment of free text responses. In *Proceedings of the 3rd Annual Computer-Assisted Assessment Conference*, pages 207–219, Loughborough, UK, 1999.

[27] Peter Willett, editor. *Document retrieval systems.* Taylor Graham, London, UK, 1988.

[28] Yiming Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1/2):69–90, 1999.

[29] 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.