Text categorization methods for automatic estimation of verbal intelligence

Fernando Fernández-Martínez , Kseniya Zablotskaya , Wolfgang Minker

A B S T R A C T

In this paper we investigate whether conventional text categorization methods may suffice to infer different verbal intelligence levels. This research goal relies on the hypothesis that the vocabulary that speakers make use of reflects their verbal intelligence levels. Automatic verbal intelligence estimation of users in a spoken language dialog system may be useful when defining an optimal dialog strategy by improving its adaptation capabilities. The work is based on a corpus containing descriptions (i.e. monologs) of a short film by test persons yielding different educational backgrounds and the verbal intelligence scores of the speakers. First, a one-way analysis of variance was performed to compare the monologs with the film transcription and to demonstrate that there are differences in the vocabulary used by the test persons yielding different verbal intelligence levels. Then, for the classification task, the monologs were represented as feature vectors using the classical TF–IDF weighting scheme. The Naive Bayes, *k*-nearest neighbors and Rocchio classifiers were tested. In this paper we describe and compare these classification approaches, define the optimal classification parameters and discuss the classification results obtained.

1. Introduction

Next-generation spoken language dialog systems (SLDSs), developed to provide users with required information and/or to help them to accomplish certain goals, are expected to be able to deal with difficult tasks and react to a wide range of situations and problems. They should help users to feel free and comfortable when interacting with them. Moreover, they should also be user-friendly and easy to use. Including aspects of adaptation to users into SLDS may help to increase the systems' communicative competences and influence on their acceptability (Fig. 1). Next-generation SLDS may change the level of dialog depending on users' experience. For example, a spoken dialog system aimed at providing guidance and support for the installation of some software may try to estimate whether the user is an expert or a novice in this field. Based on this information suitable words and explanations may be generated. These explanations may be very detailed and without specific vocabulary for a non-experienced user; in contrast, for an expert, the system may provide only a sequence of important steps or inform about more difficult operations. From the beginning of the dialog, SLDS may analyse the user's speech, behavior and requests and also the existing difficulties. When deciding on the best response to a user, the dialog manager may change words and sentence structures based on the information about cognitive processes. Its responses may become more helpful and the user-friendliness of the system may be improved. For this purpose it is necessary to identify differences in language use of people yielding different educational background and abilities to analyse situations and to solve problems.

The ability to use language for accomplishing certain goals is called verbal intelligence (VI) (Cianciolo & Sternberg, 2004; Goethals, Sorenson, & Burns, 2004). In other words, verbal intelligence is "the ability to analyse information and to solve problems using language-based reasoning" (Logsdon, 2011). Automatic verbal intelligence estimation may help dialog systems to choose the level of communication and be more simple, useful and effective.

Fig. 2 explains the adaptation process of spoken dialog systems based on verbal intelligence estimation in more detail. When talking to the system, all j spoken utterances of a user are analysed for the verbal intelligence determination. This means that the intelligence level is re-estimated at each turn based on features extracted from the new spoken utterances and from all the phrases which were pronounced at the previous turns. In Fig. 2 the SLDS has three different dialog scenarios corresponding to users yielding a higher, an average and a lower verbal intelligence. At the beginning of the dialog, the systems uses scenarios corresponding to users yielding an average verbal intelligence. At the following turns, the system might switch to alternative dialog scenarios.

The automatic estimation of users' verbal intelligence may help SLDS to more effectively control the flow of the dialogs, engage users in the interaction and be more attentive to human needs

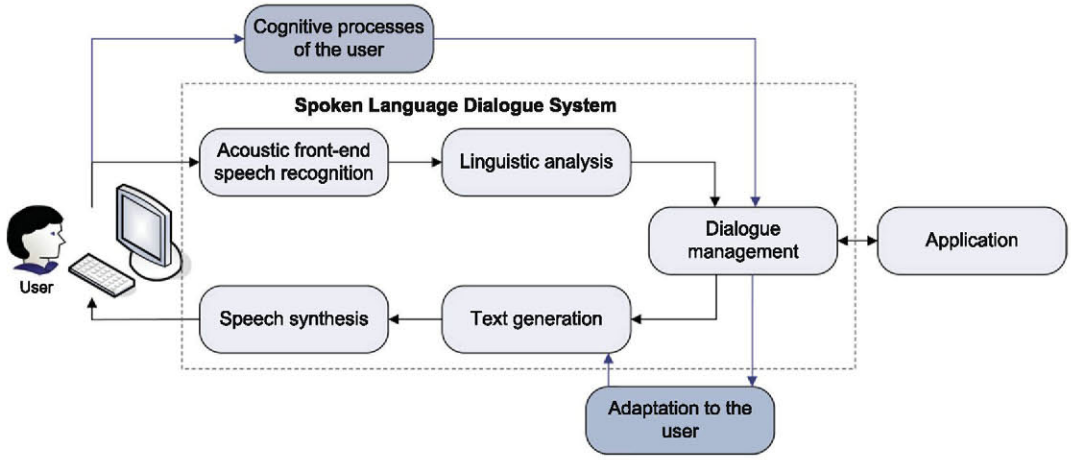

Fig. 1. Spoken language dialog system.

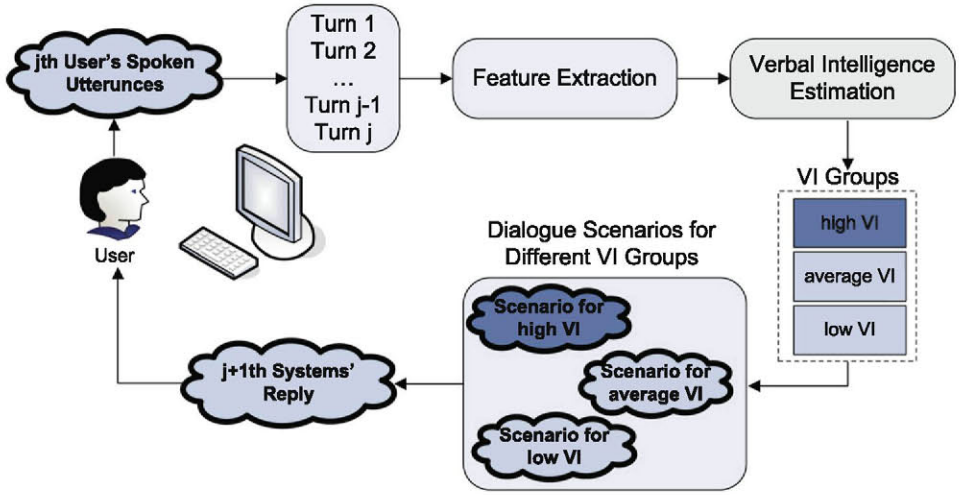

Fig. 2. Adaptation to the user.

and preferences. For training machine learning algorithms, we need to know a maximum number of language features that reflect differences in language use of people yielding different verbal intelligence. In this work we investigate to which extent the vocabulary of test persons reflect their levels of verbal intelligence when they all describe the same event and explain their thoughts and feelings about it. The investigation is based on a corpus containing descriptions of a short film along with the corresponding intelligence scores of the speakers (Zablotskaya, Walter, & Minker, 2010).

The paper is structured as follows. In Section 2 we describe the corpus which was used for the experimental research. Section 3 describes our primary efforts at defining film related features which could be useful for distinguishing test persons yielding different verbal intelligence. Section 4 describes typical TF–IDF approaches and explains the details of the feature selection process for the monologs. In Section 5 we describe and compare the Naive Bayes, *k*-nearest neighbors and Rocchio classifiers. Classification results are presented and discussed in Section 6. Finally, Section 7 presents conclusions and future work.

2. Corpus description

For the data acquisition in Zablotskaya et al. (2010), a short film was shown to German native speakers. It described an experiment on how long people could stay without sleep. The test persons were asked to imagine that they met an old friend and wanted to tell him about this film. Our goal was to record everyday speech when talking to relatives and friends. This corpus, described in Zablotskaya et al. (2010), consists of 56 descriptions (3, 5 h of audio data) of a short film (i.e. monologs).

The test persons were also asked to participate in the verbal part of the Hamburg Wechsler Intelligence Test for Adults (HAWIE) (Wechsler, 1982). According to Wechsler, intelligence is "the global capacity of a person to act purposefully, to think rationally, and to deal effectively with his environment" (Wechsler, 1939). The verbal part consists of the following subtests:

- *Information*: this subtest measures general knowledge and includes questions about history, geography, literatures, etc. For example, *What is the capital of Italy?*
- *Comprehension*: test persons are asked to solve different practical problems and explain some social situations. For example, *What would you do if you lost your way in a forest?*
- *Digit Span*: test persons are asked to repeat increasingly longer strings of numbers forward and then backward; the subtest measures short-term memory.
- *Arithmetic*: test persons are asked to solve some arithmetic problems given in a story-telling way; the subtest measures their concentration and computational ability. For example, *How many rolls you can buy if you have 36 cents and one roll costs 4 cents?*

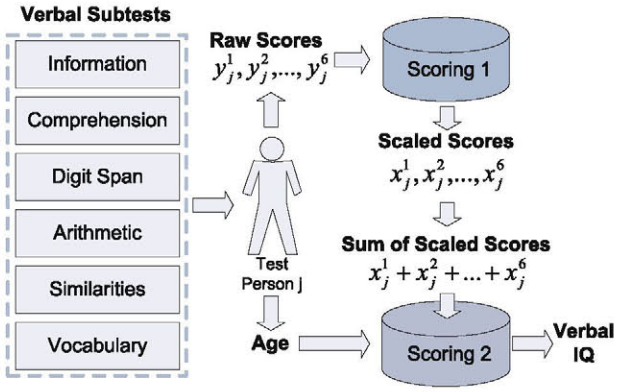

Fig. 3. Verbal part of the hamburg wechsler intelligence test for adults.

- *Similarities*: test persons are asked to find a similarity between a pair of words. For example, *Please find a similarity between "wood" and "alcohol"?*
- *Vocabulary*: test persons are asked to explain increasingly more difficult words using their vocabulary. For example, *What does "to creep" mean?*

Raw scores of each test person on the verbal test are based on the correct answers (Fig. 3). The raw scores are then converted into "Scaled Scores" using special tables (Wechsler, 1982). The Scaled Scores vary between 0 and 16 and may be used to compare the performance of the participants. The sum of the scaled scores and the age of a test person are used to estimate the corresponding verbal intelligence score.

Overall, 56 test persons yielding different educational levels were tested, therefore 56 monologs about the same topic were collected. All the monologs and the film were transcribed according to the transcription standards by Mergenthaler (1996).

3. Modelling verbal intelligence by using film derived features

To analyse the vocabulary of people yielding different verbal intelligence when describing the same event, at first we decided to compare the monologs with the film transcription. Fig. 4 shows excerpts from the film and from one of the monologs.[1]

For the comparison, the following features were extracted:

- *Number of reused words* – number of words which a test person "reused" from the film. For our example in Fig. 4 the reused words are: *fifty, eight, hours, they, and, they, despite, they, and, the, and, the.*
- *Number of unique reused words.* It includes the number of reused words without repetitions. In Fig. 4, the unique reused words are: *fifty, eight, hours, they, and, despite, the.*
- *Number of all reused lemmas.* This feature is similar to the *Number of all reused words*, but referred to lemmas instead.
- *Number of unique reused lemmas.* This feature is similar to the *Number of unique reused words*, but considering unique lemmas instead.
- *Cosine similarity* between the film and a k_{th} monolog using lemmas. For this feature extraction, we created a matrix consisting of all unique lemmas from the film, including the frequency of these lemmas within the film and within a k_{th} monolog. Table 1 shows this matrix for the texts from our example (Fig. 4).

  The frequencies were normalized by the total amount of words in the corresponding text; the cosine similarity between the two normalized vectors (lemma frequencies within the film and lemma frequencies within a k_{th} monolog) was calculated as:

$$\text{similarity} = \frac{\sum_{i=1}^{n} a_i b_i}{\sum_{i=1}^{n} a_i^2 \sum_{i=1}^{n} b_i^2},$$

  where n is the number of unique lemmas in the film, a_i is the frequency of i_{th} lemma in the film, b_i is the frequency of i_{th} lemma in the monolog.[2]
- *Number of reused n-grams.* For this feature we have calculated the number of n-g ($n = \overline{2,10}$) that were used in the film and then reused by a test-person in the corresponding monolog. In our example, *the number of reused 2-g* equals to 2 (reused 2-g are *fifty-eight* and *eight hour*), *the number of reused 3-g* equals to 1 (*fifty-eight hour*), etc.
- *Cosine similarity using n-grams.* The cosine similarity was calculated from a feature vector composed by the counts of different n-g for each monolog.
- We also determined the number of lemmas that were used by the candidates but were not used in the film. For each monolog the following features have been calculated:

$$\text{Own lemmas}_1 = \sum_{i=1}^{n} \text{frequency}(\text{lemma}_i) * \text{count}(\text{lemma}_i)$$

and

$$\text{Own lemmas}_2 = \sum_{i=1}^{n} \text{frequency}(\text{lemma}_i),$$

where n is the number of unique lemmas that were used by a test person but were not used in the film; count(word$_i$) shows how many times lemma$_i$ was used in the monolog; frequency(lemma$_i$) shows the frequency of lemma$_i$ according to a frequency dictionary of the German language (Kupietz, Belica, Keibe, & Witt, 2010). This dictionary consists of 40,000 German words with frequency from 1 to 17: 1 corresponds to more frequent words, 17 corresponds to less frequent words. If a word from the monologs was not found in the dictionary, its frequency was set to 20.

3.1. Feature analysis

The k-means algorithm, which is frequently used for data clustering in machine learning, was applied on the scaled scores of the test persons (Fig. 5). For the feature analysis two experiments were performed. In the first experiment the observations were partitioned into two clusters: cluster P_1 consisted of test persons yielding a lower verbal intelligence, P_2 contained candidates yielding a higher verbal intelligence. In the second experiment the test persons were partitioned into three clusters: P_1 is the lower verbal intelligence, P_2 is the average verbal intelligence, and P_3 is the higher verbal intelligence.

The averaged values of all the features from the two clusters were compared using a one-way analysis of variance (ANOVA).

In Experiment I with two clusters, features with small p-values were:

- *Number of reused 3-g* (averaged value for the first class $AV_{low} = 0.021$, averaged value for the second class $AV_{high} = 0.031$, $p = 0.012$, $F = 6.63$);
- *Cosine similarity using lemmas* ($AV_{\text{low}} = 0.79$, $AV_{\text{high}} = 0.83$, $p = 0.03$, $F = 4.64$);

[1] As the conversation language is German, the example was directly translated into English.

[2] Cosine similarity will be further used in the Rocchio classification approach that will be explained in detail in Section 5.2.

Excerpt from the film

Max and Funda have been without sleep for fifty eight hours. They have laid down on the sofa. Is it a mistake? Actually they would like to move. But now they cannot any more. The blood pressure is down, the energy reserves are over. They both are freezing despite the fire-place and the jacket. The question is who closes the eyes first. It is Max. Funda wins. She stays awake a few minutes longer.

Excerpt from a corresponding monologue

After fifty eight hours, they were really tired. And, they had frozen. Despite they had very warm clothes. And then the man fell asleep and then the woman.

Fig. 4. Excerpts from the film and one of the recorded monologs.

Table 1
Matrix for lemma frequency.

Lemmas from film	Frequency (film)	Frequency (monolog)
Max	2	0
and	1	3
Funda	1	0
have	2	2
be	6	1
without	1	0
sleep	1	0
for	1	0
fifty	1	1
eight	1	1
hour	1	1

- *Cosine similarity using repeated n-g* ($AV_{low} = 0.13$, $AV_{high} = 0.15$, $p = 0.01$, $F = 7.07$).

In Experiment II with three clusters, a feature with a small p-value was:

- *Cosine similarity using repeated n-g* ($AV_{low} = 0.13$, $AV_{aver} = 0.14$, $AV_{high} = 0.16$, $p = 0.01$, $F = 7.07$).

As we can see, participants with a higher verbal intelligence used more words from the film and the similarity between their descriptions and the film was higher than the similarity of participants with an average and a lower verbal intelligence. This may be explained in the following way.

Test persons yielding a higher verbal intelligence (class *HIGH*) may have a better ability to listen to and recall spoken information from the film. Memory is indeed one of the verbal sub-tests of HAWIE so that a high memory score relates to a high verbal intelligence score of a test person. Therefore, people with good memory (i.e. higher verbal intelligence) were easier able to remember many details of the film and to use words which they heard when watching the program. They may also better understand the relationships between language concepts, make more sophisticated language analogies or comparisons and perform a more complex language-based analysis.

Hence, we may conclude that the vocabulary of test persons yielding different verbal intelligence was different when they talked about the same event, even despite they were asked to talk about this film just after they had watched it.

4. Text categorization solutions

Film derived features presented in the previous section showed to be good predictors of verbal intelligence. Particularly, some of them suggested that test persons belonging to different verbal intelligence classes may be distinguished by word or lemma patterns, even regardless of the order of these words and lemmas in the monologs.

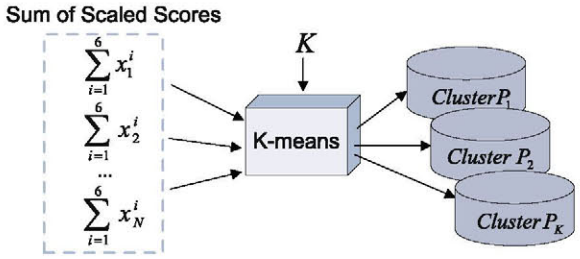

Fig. 5. The k-means algorithm.

This result led us to the main hypothesis that we investigate in this work: is it possible to solve the problem of inferring the corresponding level of verbal intelligence by simply applying conventional text categorization (TC) techniques?

To validate this hypothesis, typical TF–IDF features (introduced in the next section) have been extracted from the transcripts of the monologs (henceforth we do not make any use of the film transcription).

Three of the most popular TC methods have been applied for the automatic classification of monologs into three groups: test persons yielding a lower, an average and a higher verbal intelligence.

4.1. TF–IDF based approaches

TF–IDF (term frequency–inverse document frequency) based approaches are often used in information retrieval and text mining. As an example of a typical text mining task we may refer to the text categorization. The goal of TC is the classification of documents into a fixed number of predefined categories.

The applicability of TC techniques has significantly grown in recent years. Organizing news by subject topics (e.g. to disambiguate information and to provide readers with greater search experiences) or papers by research domains (e.g. for large databases of information that need indexing for retrieval) are just some of the most popular examples. Moreover, Security (e.g. analysis of plain text sources such as internet news), Biomedical (e.g. indexing of patient reports in health care organizations according to disease categories) or Software (e.g. for tracking and monitoring terrorist activities) domains also have benefit from these techniques.

New domains, like Marketing (e.g. analytical customer relationship management) or Sentiment analysis (e.g. analysis of movie reviews), start using text mining solutions. In this work we have

applied these techniques to a new domain: the estimation of speakers' verbal intelligence.

For TC, every document has to be transformed into a representation which could be suitable for learning algorithms and classification tasks. As reviewed in Miao and Kamel (2011), most TC algorithms are based on the vector space model (VSM). TC state-of-the-art systems widely apply the VSM approach (Baeza-Yates & Ribeiro-Neto, 1999; Miao & Kamel, 2011; Vinciarelli, 2005).

Information retrieval (IR) research suggests that words work well as representation units. In VSM, each document in a corpus is represented by a list of words (i.e. bag of words). Each word is considered as a feature; the value of the feature is a weight transformation of the number of times the word occurs in the document (i.e. word's frequency). Thus, a document is represented as a feature vector and its relevance to a query submitted by a user is measured through appropriate matching functions. These matching functions are typically based on statistical measures, like TF–IDF, that basically weight the importance of each word. The importance of a word increases proportionally to its frequency within a document but is offset by its frequency within a corpus.

Variations of this TF–IDF weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query.

4.1.1. Mathematical details

TF–IDF is a common feature transforming or weighting function. The term count, $n_{i,j}$, denotes the frequency of a given term t_i in a given document d_j. This count is usually normalized to prevent a bias towards longer documents. Thus, the term frequency $tf_{i,j}$ measures the importance of a term t_i within a document d_j and is defined as follows:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{1}$$

where the denominator is the number of words in a document d_j, that is, the size of the document $|d_j|$.

The inverse document frequency idf_i is a measure of the general importance of a term:

$$idf_i = \log \frac{|D|}{\{j : t_i \in d_j\}} \tag{2}$$

where $|D|$ is the total number of documents in the corpus, $\{j{:}t_i \in d_j\}$ is the number of documents where the term t_i appears (i.e. documents for $n_{i,j} \neq 0$).

The feature weighting function is then computed by using the following formula:

$$tfidf_{i,j} = tf_{i,j} \cdot idf_i \tag{3}$$

These weights show the importance of the words in each document. As can be seen, more frequent terms in a document are more representative and, if the number of documents in which this term occurs increases, this term becomes less discriminative.

At this point, we may view each document as a vector that contains terms and their corresponding weights. For those terms from the vocabulary that do not occur in a document this weight equals to zero. In the following sections we will show the advantage of such a document representation.

4.2. Feature selection

Typical TC approaches make use of different feature selection techniques to further reduce the dimensionality of the data space by removing irrelevant features that have no contribution to category discrimination.

Different feature selection techniques through information theory were well studied in Yang and Pedersen (1997). As a result of this study, information gain (IG) and v2-test (CHI) were reported to be the top performing methods out of five methods under test in terms of feature removal aggressiveness and classification accuracy improvement. However, the document frequency thresholding approach, the simplest method with the lowest cost in computation, was reported to perform similarly.

The Document Frequency (DF) is the number of documents in which a term occurs. As described in Yang and Pedersen (1997), it is possible to compute the document frequency for each unique term in the training corpus and to remove from the feature space those terms whose document frequency is less than a certain predefined threshold. By doing so we are adopting a basic assumption: rare terms are either non-informative for the category prediction (i.e. intelligence estimation in our case) or not influential in global performance. In either case, removal of rare terms contributes to the reduction of dimensionality of the feature space and improves the classification accuracy (i.e. if rare terms happen to be noise terms).

If we try to summarize both pros and cons of using the document frequency thresholding approach, we may say that positive aspects are:

- It is the simplest technique for vocabulary reduction (easily scalable to a very large corpora).
- Computational complexity is approximately linear with the number of documents.

while on the other hand, negative aspects are:

- The technique is usually considered as an ad hoc approach to improve the efficiency instead of a principled criterion for a predictive features selection.
- The technique is typically considered, from an IR point of view, as a non-appropriate approach for aggressive term removal (low-DF terms are assumed to be relatively informative and therefore should not be removed aggressively).

In this work a slightly modified version of this DF thresholding approach was applied to the data: TF–IDF measures instead of DF measures were used. As another remarkable difference, we did not remove the lowest TF–IDF terms but just selected the highest TF–IDF terms. In particular, instead of defining a threshold for TF–IDF measures, we defined a fixed number of terms to be selected (i.e. N). Therefore, we first sorted all the terms according to their TF–IDF measures. Then, we selected the top N most representative or indicative terms according to their TF–IDF weights. The remaining terms were removed as stop or common words that did not add any meaningful content. By observing the evolution of the classification accuracy with an increasing N value, we determined the minimum size of the vocabulary (i.e. dimensionality) required to achieve the optimum performance.

4.2.1. Class-based vs corpus-based

As stated above, in our framework each word is considered as a feature and each document is represented as a feature vector. In Özgür, Özgür, and Güngör (2005) two alternative ways for implementing the selection of these keywords or features are presented.

In the first one, the so-called corpus-based keyword selection, a common keyword or feature set that reflects the most important words for all classes (i.e. highest TF–IDF terms) in all documents is selected.

In the alternative approach, named as class-based keyword selection, the keyword selection process is performed separately

for each class. In this way, the most important and specific words for each class are determined.

4.2.2. Word lemmatisation

Word lemmatisation is often applied in the area of IR, where the goal is to enhance the system performance and to reduce the number of unique words (Solka, 2008). Particularly, word lemmatisation is part of the data pre-processing required to convert a natural language document to the feature space. Formally, it is the process for reducing inflected (or sometimes derived) words to their lemmas. For example, as a result of lemmatisation, different words like "play", "plays", "playing" and "played" are related to the same feature identification (i.e. lemma) "play".

Word lemmatisation was applied to our monologs to assess its impact on performance (i.e. classification accuracy). Like removing stop words, lemmatisation also contributed to the reduction of the size of the lexicon, thus saving on computational resources.

5. Vector space classification

As stated above, the vector space model represents each document as a vector with one real-valued component (i.e. TF–IDF weight) for each term. Therefore, we need text classification methods that can operate on real-valued vectors. In this section we introduce those ones that have been tested so far.

A number of classifiers has been used to classify text documents, including regression models, Bayesian probabilistic approaches, Nearest Neighbors approaches, Rocchio algorithm, decision trees, inductive rule learning, neural networks, on-line learning, Support Vector Machines (SVMs), and combining classifiers (Hui, Wang, Bell, Bi, & Greer, 2003; Lewis, Yang, Rose, & Li, 2004; Sebastiani & March, 2002; Yang & Liu, 1999). In this work we used three well-known vector space classification methods: Naive Bayes (NB), Rocchio and Nearest Neighbor classification (kNN).

NB is often used as a baseline in text classification research as it combines efficiency (training and classification can be accomplished with one pass over the data) and good accuracy (particularly if there are many equally important features that jointly contribute to the classification decision). The Rocchio algorithm is a very simple and efficient text categorization method for applications such as web searching, and on-line query because of its simplicity in both training and testing (Vinciarelli, 2005). kNN requires no explicit training and can use the unprocessed training set directly in classification. However, it is less efficient than the other classification methods (i.e. with kNN all the work is done at run-time so that it can have poor run-time performance if the training set is large).

Rocchio and Naive Bayes are linear classifiers whereas kNN is an example of a non-linear one. Generally speaking, if a problem is non-linear and its class boundaries cannot be approximated well with linear hyperplanes, non-linear classifiers are often more accurate than linear classifiers (particularly, if the training set is large, then kNN can handle complex classes better than Rocchio and NB). On the other hand, if a problem is linear, then it is better to use a simpler linear classifier. However, this needs to be taken with a little bit of salt since the previous assertion is always conditioned by the well-known bias-variance trade-off (i.e. with limited training data, a more constrained model tends to perform better). These approaches are described in more detail in the following sections.

Among the enumerated alternatives, SVMs are widely used mainly because they have much current theoretical and empirical appeal and perform at the state-of-the-art level. According to Sebastiani and March (2002), SVMs, example based methods, regression methods and boosting based combining classifiers deliver top-notch performance. Lewis et al. (2004) also found that SVMs perform better on Reuters-RCV1 corpus than kNN and Rocchio.

Nonetheless, recent revisions of the selected algorithms have proposed enhanced versions of these methods that achieve relatively close performance to the top-notch TC classifier: SVMs. For instance, Miao and Kamel (2011) have re-examined the applicable assumptions and parameter optimization method of the traditional Rocchio algorithm and proposed an enhanced version of this method that clearly outperforms the former one by using a pairwise optimized strategy. Salles et al. (2010) also presents a methodology to determine the impact that may have temporal effects on TC and to minimize it. By extending the three algorithms (namely kNN, Rocchio and NB) to incorporate a Temporal Weighting Function (TWF), experiments showed that these temporally-aware classifiers achieved significant gains, outperforming (or at least matching) state-of-the-art algorithms.

In any case, and as discussed in Manning, Raghavan, and Schtze (2008), despite believes of many researchers that SVM is better than kNN in terms of effectiveness, kNN is better than Rocchio and Rocchio is better than NB, the ranking of classifiers ultimately depends on the classes, the document collection and the experimental setup.

5.1. Naive Bayes classification

The first supervised learning method we introduce is the multinomial Naive Bayes or multinomial NB model, a probabilistic learning method (Manning et al., 2008). According to this method, the probability of a document d being in a class c can be computed as:

$$P(c,d) \propto P(c) \cdot \prod_{1 \leqslant k \leqslant n_d} P(t_k|c) \tag{4}$$

where $P(t_k|c)$ is the conditional probability of a term t_k occurring in a document of a class c. It may also be interpreted as a measure of how much evidence t_k contributes that c is the correct class. $P(c)$ is the prior probability of a document occurring in a class c. Terms $\langle t_1, t_2, \ldots, t_{nd} \rangle$ are part of the vocabulary that is used for the classification; n_d is the number of terms.

In NB classification, the best class for a document d is determined as:

$$c_{map} = \arg\max_{c \in C} \widehat{P}(c|d) = \arg\max_{c \in C} \widehat{P}(c) \cdot \prod_{1 \leqslant k \leqslant n_d} \widehat{P}(t_k|c) \tag{5}$$

where $\widehat{P}$ refers to the parameters to be estimated from the training data by applying the Maximum Likelihood Estimation (MLE). The interpretation of this equation is rather simple. Each conditional parameter $P(t_k|c)$ is a weight that indicates the quality of an indicator t_k for a class c. Similarly, the prior $P(c)$ is a weight that indicates the relative frequency of c. More frequent classes are more likely to be determined as the correct class.

To reduce the number of parameters, we adopted the Naive Bayes conditional independence assumption where attribute values are independent of each other given the class so that for our multinomial model:

$$P(d|c) = P(\langle t_1, t_2, \ldots, t_{nd} \rangle|c) = \prod_{1 \leqslant k \leqslant n_d} P(X_k = t_k|c) \tag{6}$$

where X_k is a random variable for a position k in the document and the values of X_k are terms from the vocabulary. $P(X_k = t_k|c)$ is the probability that in a document of a class c a term t will occur in a position k.

To further reduce the complexity of our multinomial model (assuming a different probability distribution for each position k

in the document still results in too many parameters), we made a second independence assumption: conditional probabilities for a term are the same regardless of its position in a document:

$$P(X_{k_1} = t|c) = P(X_{k_2} = t|c) \tag{7}$$

where X is a single distribution of terms which is exactly the same for any position $k_1, k_2, \ldots, k_i$. Eq. (7) applies for all terms t and classes c. This positional independence assumption is equivalent to adopting the bag of words model, which we introduced in Section 4.1. This bag-of-words model discards all information that is communicated by the order of words in natural language sentences.

5.1.1. A variant of the multinomial model

A critical step in solving a text classification problem is to choose the document representation. An alternative formalization of the multinomial model represents each document d as a M-dimensional vector of counts: $tf{-}idf_{t_1,d}, tf{-}idf_{t_2,d}, \ldots, tf{-}idf_{t_M,d}$ where $tf{-}idf_{t_i,d}$ is the TF–IDF measure for a term t_i in a document d. $P(d|c)$ is then computed as follows:

$$P(d|c) = P(\langle tf{-}idf_{t_1,d}, tf{-}idf_{t_2,d}, \ldots, tf{-}idf_{t_M,d}\rangle|c) \tag{8}$$

All the model parameters (i.e. class priors and feature probability distributions) may be estimated from the training set by using MLE. For every class' prior we calculated an estimate for the class probability from the training set (i.e. (prior for a given class) = (number of samples in the class)/ (total number of samples)). To estimate the parameters for our feature distribution, we adopted the typical assumption that the continuous values associated with each class are distributed according to a Gaussian distribution. Particularly, assuming that the training data contains continuous attributes, i.e. TF–IDF measures for each term and document, we first segmented the data by the class and then computed the mean and variance of every term specific TF–IDF measure in each class.

5.1.2. About the independence assumptions

Typically, TC tasks rather look at the words themselves and not at their corresponding positions in the documents (i.e. bag of words). This relies on the hypothesis that each topic or class to be distinguished is fairly represented by only some specific words from our bag. The NB models often perform well for TC tasks despite the conditional independence and the positional independence assumptions. In fact, both assumptions are very important to avoid problems in estimation owing to data sparseness.

By adopting both independence assumptions, we are committed to a specific way of processing the evidence. Particularly, in the NB classification we look at each term separately so that we do not make a difference between word A followed by word B and word B followed by word A (although there is a difference between them). However, the conditional independence assumption does not really hold for text data as terms are conditionally dependent on each other. Additionally, the position of a term in a document by itself may carry more information about the class than expected.

5.2. The Rocchio approach

The Rocchio classification (Rocchio, 1971) divides the vector space into different regions centered on prototypes. These prototypes or centroids, one for each class, define the class boundaries (i.e. hyperplanes). For a given training dataset, the centroid of a class c can be computed as the vector average or center of mass of its members (i.e. all documents in the class) (Ittner, Lewis, & Ahn, 1995; Manning et al., 2008).

$$\vec{\mu}(c) = \frac{1}{|D_c|}\sum_{d\in D_c}\vec{V}(d) \tag{9}$$

where D_c is the set of documents from class $c : D_c = \{d : \langle d, c\rangle \in D\}$; $\vec{V}(d) = V_1(d)\cdots V_M(d)$ is a vector that contains tf–idf weights for each term of a document d.

As many vector space classifiers (e.g. computing the nearest neighbors in kNN classification), the Rocchio approach relies on distance-based decisions (from a TC point of view, the relatedness of two documents can be typically expressed in terms of similarity or distance). Particularly, the Rocchio classification rule is to classify a point in accordance with the region it falls into. To do this, basically we determine the centroid $\vec{\mu}(c)$ that the point is closest to and then assign it to c.

In our experiments, we used the cosine similarity measure as the underlying distance. Cosine similarity is the cosine of the angle between two vectors and determines whether they are pointing in roughly the same direction. Since the components of our vectors (i.e. tf–idf weights) could not be negative, the angle between two tf–idf vectors could not be greater than 90°.

The vector representation $\vec{V}(d_1)$ and $\vec{V}(d_2)$ of the cosine similarity between two documents d_1 and d_2 is:

$$sim(d_1, d_2) = \frac{\vec{V}(d_1)\cdot\vec{V}(d_2)}{|\vec{V}(d_1)||\vec{V}(d_2)|} \tag{10}$$

where $|\vec{V}(d_1)|$ and $|\vec{V}(d_2)|$ it the Euclidean length of the vectors.

By using this measure, we are also applying a normalization process which makes each vector of the same length (Salton, 1989). If we have a look at the magnitude of the vector difference between two vectors corresponding to documents with very similar content, it may happen that this difference is significantly simple because one is much longer than the other. Cosine similarity measure compensates this effect of document length so that the similarity between document vectors is reduced to only measuring the cosine of the angle between them.

We can rewrite Eq. (10) as follows:

$$sim(d_1, d_2) = \vec{v}(d_1)\cdot\vec{v}(d_2) \tag{11}$$

where $\vec{v}(d_1) = \vec{V}(d_1)/|\vec{V}(d_1)|$ and $\vec{v}(d_2) = \vec{V}(d_2)/|\vec{V}(d_2)|$. The assignment criterion for a document d and its vector representation $\vec{V}(d)$ can be defined as:

$$c_{rocchio} = \arg\max_{c\in C} sim(\vec{\mu}(c), \vec{V}(d)) \tag{12}$$

In our implementation of the Rocchio approach (Dumais, Platt, Heckerman, & Sahami, 1998; Joachims, 1998) only positive training samples are considered for obtaining the prototype for each class (i.e. training samples that belong to the corresponding class). However, recent variations of Rocchio (Bi, Bell, Wang, Guo, & Guan, 2007; Moschitti, 2003; Sebastiani & March, 2002) consider the effects of negative samples (i.e. training documents that belong to all other classes) when computing the prototypes for the defined classes. Different parameters may be used to control the relative importance of positive and negative samples. These Rocchio classifiers reward not only the closeness of a test document to the centroid of the positive training instances, but also its distance from the centroid of the negative training instances.

5.3. k-Nearest neighbors

In pattern recognition, the k-nearest neighbor algorithm (kNN) is a method for classifying objects based on the closest training examples in the feature space. In TC, kNN takes an arbitrary input document and ranks the k nearest neighbors among the training documents through the use of a similarity score (i.e. cosine similarity distance). It then assigns to the input the category or the class of the most similar document or documents. A constant k, defined by a user, denotes the number of neighbors included in the evaluation.

The kNN algorithm is a valid non-parametric method. Despite being amongst the simplest of all machine learning algorithms, it is one of the best methods when the text is described by using VSM (Yang & Liu, 1999). However, traditional kNN has two main drawbacks: the intensive computational effort, especially when the size of the training set grows (training examples are vectors in a highly multidimensional feature space), and its sensitiveness to the local structure of the data (Jianliang & Yongcheng, 2004).

New nearest neighbor algorithms have been recently proposed mainly with the purpose of reducing the number of distance evaluations actually performed, thus trying to make kNN computationally tractable even for large data sets. For instance, (Wang & Wang, 2007) presents a fast kNN algorithm that reduces the cost of similarity computing in order to raise the classifying speed and applicability of kNN.

In Naive Bayes and Rocchio classification we have to estimate corresponding parameters: priors and conditional probabilities and centroids. In kNN we do not need to estimate any parameters but simply memorize all examples in the training set and then compare a test document to them. For this reason, kNN is also called memory-based learning or instance-based learning.

The kNN algorithm is known because of its strong consistency results. As the amount of data approaches infinity, the algorithm is guaranteed to yield an error rate no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data) (Manning et al., 2008). kNN is guaranteed to approach the Bayes error rate for a certain value of k (where k increases as a function of the number of data points). The k-nearest neighbor methods may be improved by using proximity graphs (Toussaint, 2005).

5.3.1. Choosing the class for an unclassified document

To make a decision on a number of unclassified documents, we measure their similarity with all the documents that have already been classified. The unclassified documents are then ranked according to their similarity scores. Appropriate classes for the documents may be assigned in the following ways:

- If we choose $k = 1$, the class is predicted to be the class of the closest training sample. This is called the nearest neighbor algorithm.
- If we choose $k > 1$, then all the documents which ranks are smaller than or equal to k will be included in the ranked list. We can then use different means to find a class for our document, like:
  - we may assign the document to the most common class amongst its k nearest neighbors (if we are dealing with a binary, i.e. two-class, classification problem, it is helpful to choose k to be an odd number as this avoids tied votes).
  - we may estimate the probability of membership in a class c as the proportion of the k nearest neighbors in c. This is commonly referred as the probabilistic version of the kNN classification algorithm.
  - for the individual classes, we may sum the distances to all the documents in which the class occurs, and then choose the class corresponding to the highest accumulated distance (remember that we are using cosine distance).
  - etc.

If we decide to use either the basic "majority voting", the probabilistic method or the sum of distances based classification, those classes with more frequent examples will tend to dominate the prediction of a new vector. This is actually a drawback as they tend to come up in the k nearest neighbors when the neighbors are computed due to their large number (it is important to remind that the available data is certainly imbalanced). To overcome this problem, we compensate this possible imbalance by introducing a slightly modified classification method for our kNN based approach.

Typically, the implementation of these versions of the algorithm starts by computing the distances from the test sample to all stored vectors of the training data set. Next, all these training samples are sorted according to these distances thus ranking the nearest k training samples regardless of their corresponding class. If we look at those already labeled classes instead (neighbors are taken from a set of documents for which the correct classification is known), we could then identify specific top k neighbors for each class c (i.e. k nearest neighbors labeled as c, thus resulting in an overall list composed of $k \times C$ elements). Finally, by computing distance to each class as the average distance between the test sample and those top k class specific neighbors, we will then manage to compensate any possible imbalance in the distribution of the training data among the defined classes.

The best class for kNN classification can then be derived from:

$$c_{kNN} = \arg\max_{c \in C} \text{score}(c, d) = \arg\max_{c \in C} \frac{1}{k} \cdot \sum_{d' \in S_k(c,d)} \text{sim}(\vec{v}(d'), \vec{v}(d)) \quad (13)$$

where $S_k(c, d)$ is the c class specific set of d's k nearest neighbors. As could be derived from Eq. (13), it may also be useful to weight the contributions of the neighbors so that nearer ones contribute more to the average than more distant ones (Tan, 2005). This classification method is weighted by taking into account not only the distance from the test sample to the c set of k nearest neighbors, but also the class compactness particularly for high values of k.

5.3.2. Parameter selection

The parameter k in kNN is typically defined by using some previous experience or specific knowledge about the classification domain. Normally, 1NN is found to be not very robust. The accuracy of the kNN algorithm can be severely degraded by the presence of noisy or irrelevant features (also if the feature scales are not consistent with their importance). 1NN means that the classification decision for each test document only relies on the class of a single training document, whose label could eventually be incorrect or atypical. kNN for $k > 1$ are more robust as larger values of k tend to reduce the effect of noise on the classification (although also make boundaries between classes less distinct).

As an alternative, a good value of k can be assigned heuristically via cross validation technique or empirically via bootstrap method (Hall, Park, & Samworth, 2008). In our experiments, instead of applying any of these parameter selection methods, we tried different k values, thus finally selecting the optimal k as the value which was used when obtaining the best performance.

6. Experimental results and discussion

6.1. Experimental set-up

Our main goal is to identify the algorithm that best computes class boundaries and reaches the highest classification accuracy. In our experiments for comparing the performance of the different approaches, a Leave-One-Out cross validation (LOO-CV) method was used. The idea of this method is to use $N - 1$ observations for training (where N is the number of data points) and only 1 data point for testing. This procedure is repeated N times and each observation is used once as the testing data.

6.2. Baseline approach: class-based vs corpus-based feature selection

As introduced in Section 4.2.1, our experiments covered the comparison of the class-based and corpus-based keyword selection approaches.

The corpus-based approach implies the selection of a common feature set for all classes with the top N most representative or indicative terms. The class-based approach instead implies the selection of the most important words for each particular class. In this case, to preserve the balance between classes, N/M words for each specific class were selected, where M is the number of classes. For our classification task M equals to 3, where the first class contained test persons yielding a lower verbal intelligence, the second class contained participants yielding an average verbal intelligence and the third class contained participants yielding a higher verbal intelligence. Then, we composed our feature vector by concatenating all the class-specific features, thus resulting into a vector comparable to the N-dimensional vector corresponding to the corpus-based approach.

However, when using the class-based approach, a particular word may be included in various class-specific subsets (i.e. a word that is important to not only one single class but to several classes). To avoid using duplicate features, we only used the intersection between all the class-specific subsets. Therefore, the dimension of the resulting feature vector in these cases had to be necessarily lower than N. For simplicity, we will better refer to the number of features per class (i.e. $F = N/3$) rather than to the final dimensions of the vectors. Consequently, if we report, for instance, about 50 words or features per class, this means that we are using a 150-dimensional corpus-based vector. In this case for the class-based approach, 150 is the maximum number of dimensions. To definitely determine its value, it is necessary to check the possible intersection.

Of course, the higher the value of F, the more significant the intersection between class-specific word subsets, and also the bigger the difference with respect to the corpus-based vector dimensions. Analysing the corpus, 2210 different words were extracted from all the monolog transcripts. Table 2 shows how the intersection evolved according to F. Considering the size of the vocabulary, the observed difference is significant.

Fig. 6 presents the accuracy results obtained using either the corpus-based or the class-based feature selection methods. The results were obtained using the NB approach for different dimensions of the feature vector. Confidence intervals of 95% are also shown in the figure.

As it can be derived from the figure, the class-based approach clearly outperformed the corpus-based one regardless of any difference about the used dimension. Although the observed differences were not statistically significant in any case, it is interesting to pinpoint the result for the 155-dimensional value. At this point the class-based approach reached the top performance while the difference with the corpus-based alternative also turned to be the biggest one thus becoming almost significant.

From a different point of view, we may also try to analyse the minimum dimensionality required by the class-based approach to outperform the corpus-based one. The corpus-based approach obtained a maximum accuracy of 51,79%. As can be observed in Fig. 6, this performance was reached with dimensionality equal to or higher than 110. Also derived from this figure, we may check that the class-based approach obtained a better performance of 57,14% (though not statistically significant) using "only" 20 features per class. The class-based feature selection, by definition, focuses on finding the most crucial or indicative class keywords. On the other hand, the corpus-based one simply tends to find general keywords concerning all classes. This clearly tips the balance in favor of the class-based approach particularly when we use a reduced set of features. This is important as there may be a significant gain in classification time when a small number of features is used.

By confirming these differences with additional statistical evidence (i.e. more data), we may also conclude that the class-based feature selection improved the performance of the corpus-based one for the NB approach not only in terms of accuracy but also in terms of time. Similar results were already confirmed in Özgür et al. (2005).

When using the corpus-based approach, most features (i.e. words) tend to be selected from the prevailing classes so that rare classes are not well represented. In contrast, when using the class-based approach all the classes are represented equally well as for their representation class specific features are used. Thus, the class-based approach achieved consistently higher accuracies than the corpus-based approach.

Similar differences between the class-based and corpus-based methods have been consistently observed throughout all of our experiments. Therefore, in the next sections we will only focus on the class-based versions.

6.3. Comparison between approaches: Rocchio "Wins"

In this section we compare the results that were obtained using different approaches. Before proceeding with this comparison, we need first to assign the optimal configuration (i.e. k value) for the kNN approach.

Fig. 7 presents classification results corresponding to several k values. As expected, 1NN was found to be not very robust. Optimal performance may be reached by using $k = 3$ in combination with dimensionality of 155. However, if we keep increasing the value of k, which is typically more robust as it helps to reduce the effect of noise on the classification, then the results apparently start to be affected by sparse data bias.

As a result of the initial k-means clustering, only 13 samples were defined to be part of the least populated class. Therefore, starting with 1NN we checked out up to $k = 12$ values leaving one sample out for testing (the LOO approach was applied). For clarity, Fig. 7 presents classification results only with some values of k.

The observed differences were found to be statistically significant for the top performance dimensionality (i.e. $F = 155$) when comparing the best configuration (i.e. $k = 3$) with all the others for $k > 5$. No statistically significant differences where observed between the best k and any $k \leqslant 5$ configurations.

Table 2
Dimension differences between class-based and corpus-based approaches.

	# of features per class (3 classes)									
	50	100	150	200	250	300	350	400	450	500
Corpus-based	150	300	450	600	750	900	1050	1200	1350	1500
Class-based	150	289	393	486	557	601	737	858	992	1102
Difference	0	11	57	114	193	299	313	342	358	398
Rel. diff. (%)	0	3.7	12.7	19	25.7	33.2	29.8	28.5	26.5	26.5

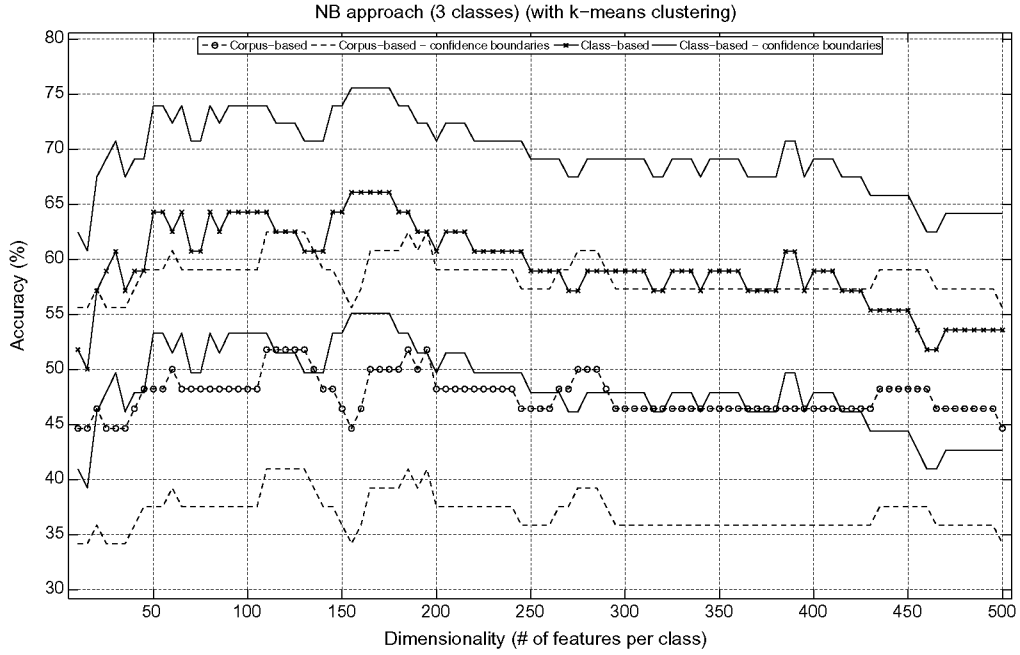

Fig. 6. Baseline approach: Class-based vs corpus-based feature selection.

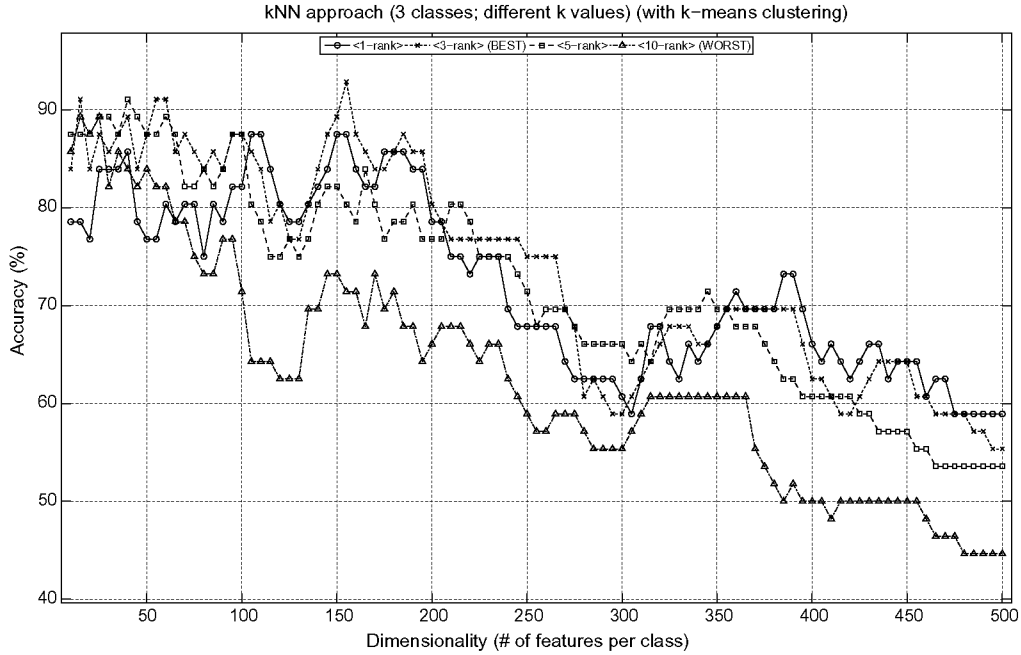

Fig. 7. kNN results for different k values.

Fig. 8 allows to compare the results of the NB approach, the Rocchio approach and the kNN approach with $k = 3$.

A first important result that we can derive from Fig. 8 is that both Rocchio and kNN are clearly outperforming the NB approach, although the top performance is defined for different dimensionalities in each case. The kNN performance had a maximum accuracy of 92.86% for 155-dimensionality, while Rocchio just required 15 features per class to improve it up to 95.6%. Both results denoted a statistically significant difference when compared to the NB top performance, 66.07% also for 155-dimensionality. However, we did not observe any significant difference between Rocchio and 3NN (naturally, Rocchio was also significantly outperforming any $k > 5$ approach).

As it typically occurs in TC tasks, most of the learning takes place with a small yet crucial portion of features (i.e. keywords) for a class. This is evident in the steeper learning curves that reach the top performance at relatively low dimensionality. Therefore, we may conclude that the class-based feature selection approach is shown to be successful in quickly finding the most crucial or indicative class keywords.

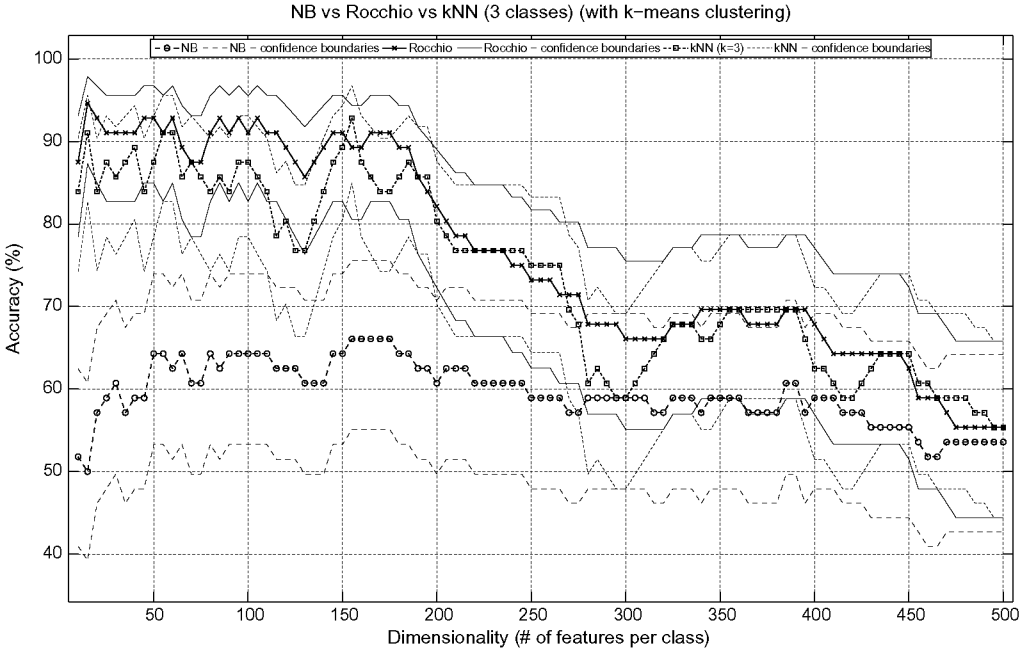

Fig. 8. Comparison between approaches: Rocchio wins.

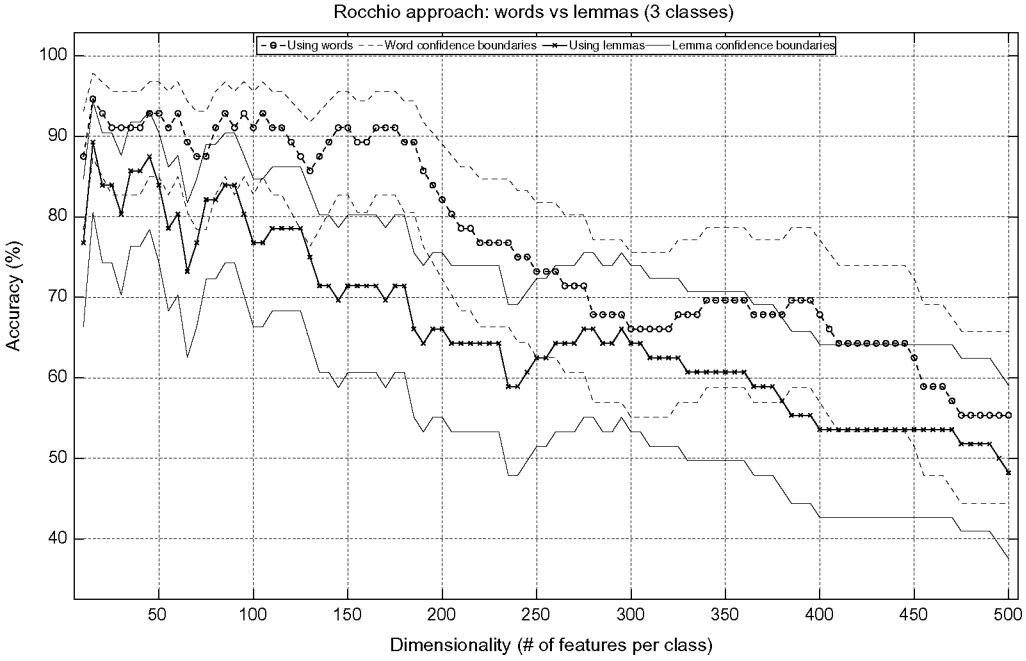

Fig. 9. Classification results using words vs lemmas.

Another visible result in Fig. 8, common to all the tested approaches, is the performance decrease as the value of F increases (particularly beyond a 200-dimensional value). As we already introduced in Section 6.2 and proved in Table 2, the higher the value of F, the more significant the intersection between class-specific word subsets. If we expand this interpretation, the more significant the intersection, the less discriminative the class-specific subsets, the more likely to include words that are not really indicative of any of the classes, and so the performance decreases.

6.4. Using words vs lemmas

As we introduced in Section 4.2.2, we also tried a word lemmatisation strategy (i.e. to group together those words that are in different forms but with the same lemma). This strategy was implemented as part of the data pre-processing stage during the classification task. Fig. 9 shows the results with and without word lemmatisation for our top performing approach: the Rocchio one.

The main advantage of word lemmatisation is to reduce the dimensionality of the data space. In a TC task, it is basically applied

under the assumption that all the documents belonging to the same category or topic may include these lemmas appearing in different forms, and of course, it makes sense to use them as they refer to words with similar meanings. TC tasks typically rely on this. However, to be successful and thus really enhance system performance, there is another important hypothesis that also needs to be confirmed: each topic or class to be distinguished should be fairly represented by only some class-specific lemmas.

While the former one happens to be true for most of the cases, the latter one, though also successfully applied in typical TC tasks, may reasonably not be true in our case. The main reason for this would be that, from this point of view, all the documents (i.e. monologs) could be regarded as belonging to the same category according to their topic or content: all the documents are about the film which the participants watched. Consequently, we could expect an important number of lemmas to be shared among the participants as they all were talking about the same topic.

This is an important difference with conventional TC tasks where, normally, the topics or classes are well separated according to their conceptualization. In contrast, in our domain we may expect the participants to be identifiable among others not by the concepts or ideas themselves but by the way they express these ideas. Therefore, in this particular case, we may expect lemmas not to have much contribution to category discrimination but the different endings and forms instead. Hence, missing this discriminative information because of lemmatisation (simplifying words with different forms into their more common roots) could have some undesirable consequences in classification and clustering.

Moreover, the fact that all the participants were German native speakers could be particularly critical for this problem. In this regard it is important to remark that German is a very agglutinative language (Olsen, 2000). Compound words or words that consist of more than one lemma (i.e. compounding or word-compounding occurs when a person attaches two or more words together to make one word), can be found very often in the German language. "Donaudampfschifffahrtsgesellschaftskapitänsmütze" (i.e. Danube steamboat shipping company Captain's hat) is a good example of how long these compound words could be (they can be practically unlimited in length, particularly in case of biochemistry).

The meaning of a compound word differs from the meanings of words which it consists of. Lemmatisation of compound words would simply reduce them to their more common lemmas thus loosing this discriminative information.

To what extent this argument could be either true or false is something that can be derived from Fig. 9. In fact, the word-based approaches systematically outperform the lemma-based ones. Confidence boundaries for both cases are also shown in this figure. As we can observe, differences become statistically significant mostly around the same dimensionality range that was previously pinpointed when referring to the top performance for both kNN and NB (particularly at a 155-dimensional value). However, the differences are not statistically significant at $F = 15$, the point at which Rocchio reaches its maximum accuracy for both word-based and lemma-based approaches.

6.5. Tempted to use more classes

Although the three-classes scheme can be found entirely suitable from a practical implementation point of view (i.e. participants yielding a lower, an average and a higher verbal intelligence), we were also interested in analysing the performance of the suggested approaches for a higher number of classes. This would enable a better granularity for the verbal intelligence classification.

Fig. 10 presents benchmarking results for 4 classes instead of 3 (as it was shown in Fig. 8). From a practical point of view, these classes may correspond to the following levels of verbal intelligence: *poor*, *average*, *high* and *very high* respectively.

In this regard it seems to be important to remark that working with a higher number of classes, like 5 or more, was practically infeasible because of sparse data problems (i.e. k-means resulted into unpopulated classes).

As for three classes, the Rocchio approach showed the highest accuracy again (i.e. 87.5% at $F = 15$). The optimal dimensionality remained to be the same as for three classes (i.e. $F = 15$). Regarding the comparison between Rocchio and kNN, the observed differences also remained to be not statistically significant.

Additionally, both Rocchio and kNN clearly outperformed the NB algorithm, once again by a significant margin. For these two,

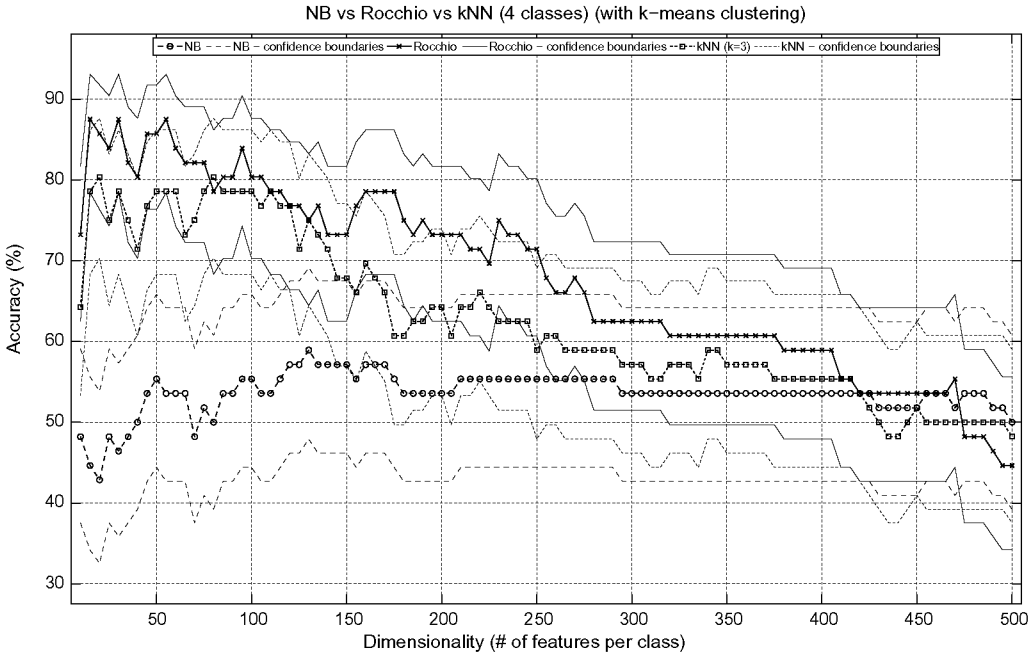

Fig. 10. Tempted to use more classes: 4 classes.

Table 3
Confusion matrix corresponding to Rocchio's top performance using 4 classes ($F = 15$).

		Prediction outcome			
		a	b	c	d
Actual value	a = Poor	4	0	1	0
	b = Average	1	14	1	0
	c = High	1	0	22	1
	d = Very high	0	1	1	9

another effect starts to become evident: the top performance region, previously observed for dimensionality values up to 200, now turns to be narrower, locating its limit approximately around a value of 100. As we simply increase the number of classes, it seems to be evident that the number of terms or features that are really indicative of each particular class becomes smaller, thus affecting the performance.

From a general point of view, the resulting performance can still be deemed to be satisfactory as the error rate is only roughly 7% higher than with three classes. If we look at the confusion matrix, presented in Table 3, we may check that by adopting the four classes into three by grouping *high* and *very high* classes, predictability would be more similar reducing the gap roughly to the half (i.e. 94.64% for three classes and 91.07% for four classes adopted into three).

Finally, it may be interesting, particularly from a practical point of view, to have a look at the upper-right and lower-left corners of the matrix: 0 errors. This means there was not any critical errors like regarding a lower verbal intelligence individual as a higher verbal intelligence one and vice versa.

7. Conclusions and future work

This work showed that verbal intelligence may be recognized by computers through language cues. The achieved classification accuracy can be deemed as satisfying for a number of classes that is reasonably high enough to enable its integration into SLDSs. To our knowledge, this is the first report of experiments attempting to automatically predict verbal intelligence.

Some of the most popular TC algorithms were applied to this task: NB, Rocchio and kNN. NB models are typically expected to perform well for TC tasks despite the conditional independence and the positional independence assumptions. However, the performance of NB approach was significantly worse than with the other approaches: kNN and Rocchio. This suggests that this probabilistic classifier was more sensitive to the low number of examples available, mainly resulting into inaccurate probability estimates, than the vector space ones (computing distances to some relevant members or to a prototype of each defined class seems to be more robust against sparse data).

On the other hand, and connecting with those independence assumptions, it is well known that conditional independence does not really hold for text data (even worse considering that our features are highly correlated). Furthermore, we firmly believe that, for this specific task, the position of a term in a document by itself could carry more information about the class than expected, mainly because of the above mentioned peculiarities of our classification task (i.e. it is not only about the words that participants used to denote their intelligence, but also the way they combined them). Therefore, our data is somehow violating these independence assumptions, thus finally explaining why the NB approach performed so poorly. In this regard, it would be very interesting to test a LM based TC approach to better validate this argument.

Using the class-based feature selection approaches has proven to be an essential factor, not only to achieve a better inference performance but also to reduce its computational cost.

Despite typically successful when applied to TC tasks, word lemmatisation was not really helpful for our task. The word-based approaches systematically outperformed the lemma-based approaches, thus pointing out some peculiarities of the classification task. Particularly, these results were found to be mainly explained by two different factors: the same topic for collecting monologs and the use of the German language.

Unlike typical TC tasks, our verbal intelligence prediction task is influenced by the necessary fact that the different categories or classes to be identified are not well separated from a conceptualization point of view. Of course, it might be easier to distinguish people talking about different topics from their everyday life although the results for such a comparison might not be objective. By letting the participants (i.e. people with different interests and hobbies) to discuss their own topics, we would be then recognizing the topics themselves rather than people with different cognitive processes.

On the other hand, the use of German, a very agglutinative language, resulted to be a drawback with regards to word lemmatisation. By lemmatisation of compound words (compounding is a pretty common phenomena in German) we are basically loosing the extra meaning that arises from the combination of the interrelated words. This meaning has proven to be really helpful to correctly discriminate between different levels of verbal intelligence.

In future work, it would also be interesting to examine how well the suggested approaches perform when integrated into existing SLDS. In this regard, it is important to remark that any application involving speech recognition will always introduce noise in the features that we used. This needs to be considered as it will surely reduce the presented accuracies. Testing these approaches with conventional SLDS would allow us to assess whether the accuracies we achieve are high enough or not for our intended application (i.e. dialog system adaptation).

On the other hand, this also suggests the importance of finding some other features that could be more robust when being used in a conventional system. Prosodic features could be a good alternative; so it would be interesting to start working on an multimodal inference framework that could jointly exploit the potential of, among others, this kind of features. As we have already mentioned, the linguistic cues that we have used in this work could pose a problem, for instance, if we want to apply these solutions with the same users but across different domains. In this regard, prosodic features would be found to be advantageous as they would also allow us to explore the possibility of finding topic independent solutions.

Acknowledgement

This work is partly supported by the DAAD (German Academic Exchange Service).

Parts of the research described in this article are supported by the Transregional Collaborative Research Centre SFB/TRR 62 "Companion-Technology for Cognitive Technical Systems" funded by the German Research Foundation (DFG).

For this work, Fernando was granted a fellowship by the Caja Madrid foundation.

References

Baeza-Yates, R. A., & Ribeiro-Neto, B. (1999). *Modern information retrieval*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Bi, Y., Bell, D., Wang, H., Guo, G., & Guan, J. (2007). Combining multiple classifiers using dempster's rule for text categorization. *Applied Artificial Intelligence, 21*, 211–239. <http://dl.acm.org/citation.cfm?id=1392641.1392644>.

Cianciolo, A. T., & Sternberg, T. J. (2004). *Intelligence: a brief history*. Blackwell Publishing.

Dumais, S., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of the*

seventh international conference on Information and knowledge management. CIKM '98 (pp. 148–155). New York, NY, USA: ACM. URL http://doi.acm.org/10.1145/288627.288651.

Goethals, G., Sorenson, G., & Burns, J. (2004). *Encyclopedia of leadership. No. v. 1 in encyclopedia of leadership*. Sage Publications. <http://books.google.es/books?id=kjLspnsZS4UC>.

Hall, P., Park, B. U., & Samworth, R.J. (2008). Choice of neighbor order in nearest-neighbor classification. *Annals of Statistics 36*, 2135. doi:doi:10.1214/07-AOS537.

Hui, G. G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003). Using knn model-based approach for automatic text. In *Proceedings of ODBASE'03, the 2nd international conference on ontologies, database and applications of semantics, LNCS* (pp. 986–996).

Ittner, D. J., Lewis, D. D., & Ahn, D. D. (1995). Text categorization of low quality images. In *Proceedings of SDAIR-95, 4th annual symposium on document analysis and information retrieval* (pp. 301–315).

Jianliang, Y., & Yongcheng, W. (2004). Application of iterative-knn based on knn and automatic retrieval in automatic categorization. *Journal of The China Society For Scientific and Technical Information, 23*, 137–141.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning (ECML)* (pp. 137–142). Berlin: Springer.

Kupietz, M., Belica, C., Keibe, H., & Witt, A. (2010). The german reference corpus dereko: A primordial sample for linguistic research. In Calzolari, Nicoletta et al. (Eds.). *Proceedings of the 7th conference on international language resources and evaluation (LREC 2010)* (pp. 1848–1854).

Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research, 5*, 361–397. <http://dl.acm.org/citation.cfm?id=1005332.1005345>.

Logsdon, A. (2011). Learning disabilities <http://www.learningdisabilities.about.com/>

Manning, C. D., Raghavan, P., & Schtze, H. (2008). *Introduction to information retrieval*. New York, NY, USA: Cambridge University Press.

Mergenthaler, E. (1996). Emotion-abstraction patterns in verbatim protocols: A new way of describing psychotherapeutic processes. *Journal of Consulting and Clinical Psychology, 6*(64).

Miao, Y.-Q., & Kamel, M. (2011). Pairwise optimized Rocchio algorithm for text categorization. *Pattern Recognition Letters, 32*, 375–382. <http://dx.doi.org/10.1016/j.patrec.2010.09.018>.

Moschitti, A. (2003). A study on optimal parameter tuning for rocchio text classifier. In *Proceedings of the 25th European conference on IR research. ECIR'03* (pp. 420–435). Berlin, Heidelberg: Springer-Verlag. <http://dl.acm.org/citation.cfm?id=1757788.1757828>.

Olsen, S. (2000). Ein internationales handbuch zur flexion und wortbildung. In G. Booij, C. Lehmann, & J. Mugdan (Eds.), *Morphologie* (pp. 897–916). Berlin/New York: de Gruyter.

Özgür, A., Özgür, L., & Güngör, T. (2005). Text categorization with class-based and corpus-based keyword selection. In *ISCIS* (pp. 606–615).

Rocchio, J. (1971). *Relevance feedback in information retrieval*. Prentice-Hall Inc. Ch. 14, pp. 313–323.

Salles, T., Rocha, L., Gisele L. P., Mourão, F., Meira, W., Jr., & Gonçalves, M. (2010). Temporally-aware algorithms for document classification. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval. SIGIR '10* (pp. 307–314). New York, NY, USA: ACM. URL http://doi.acm.org/10.1145/1835449.1835502.

Salton, G. (1989). *Automatic text processing: The transformation, analysis, and retrieval of information by computer*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computation Survey, 34*, 1–47. <http://doi.acm.org/10.1145/505282.505283>.

Solka, J. (2008). Text data mining: Theory and methods. *Statistics Surveys, 2*, 94–112 <http://dx.doi.org/10.1214/07-SS016> the statistics surveys <http://www.i-journals.org/ss/> by the institute of mathematical statistics <http://www.imstat.org>.

Tan, S. (2005). Neighbor-weighted *k*-nearest neighbor for unbalanced text corpus. *Expert Systems with Applications, 28*, 667–671. <http://dx.doi.org/10.1016/j.eswa.2004.12.023>.

Toussaint, G. T. (2005). Geometric proximity graphs for improving nearest neighbor methods in instance-based learning and data mining. *International Journal of Computational Geometry and Applications, 15*(2), 101–150.

Vinciarelli, A. (2005). Application of information retrieval techniques to single writer documents. *Pattern Recognition Letters, 26*, 2262–2271. <http://dx.doi.org/10.1016/j.patrec.2005.03.036>.

Wang, Y., & Wang, Z.-O. (2007). A fast knn algorithm for text categorization. In *International Conference on machine learning and cybernetics* (Vol. 6, pp. 3436–3441).

Wechsler, D. (1939). *The measurement of adult intelligence*. Baltimore (MD): Williams & Witkins.

Wechsler, D. (1982). *Handanweisung zum Hamburg-Wechsler-Intelligenztest fuer Erwachsene (HAWIE)*. Separatdr., Bern; Stuttgart; Wien, Huber.

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. SIGIR '99* (pp. 42–49). ACM, New York, NY, USA <http://doi.acm.org/10.1145/312624.312647>.

Yang, Y., & Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In D. H. Fisher (Ed.), *Proceedings of ICML-97, 14th international conference on machine learning* (pp. 412–420). San Francisco, US, Nashville, US: Morgan Kaufmann Publishers. <citeseer.nj.nec.com/yang97comparative.html>.

Zablotskaya, K., Walter, S., & Minker, W. (2010). Speech data corpus for verbal intelligence estimation. In: *Proceedings of LREC'10* (pp. 1077–1080).