

Text Classification for Italian Proficiency Evaluation

Alfredo Milani[1], Stefania Spina[2], Valentino Santucci[2], Luisa Piersanti[1], Marco Simonetti[3], and Giulio Biondi[4(✉)]

[1] Department of Mathematics and Computer Science, University of Perugia, Perugia, Italy
milani@unipg.it, luisa.piersanti@studenti.unipg.it
[2] University for Foreigners of Perugia, Perugia, Italy
{stefania.spina,valentino.santucci}@unistrapg.it
[3] Liceo Artistico "G. Marconi", Foligno, Italy
[4] Department of Mathematics and Computer Science, University of Florence, Florence, Italy
giulio.biondi@unifi.it

Abstract. NLP technologies and components have an increasing diffusion in mass analysis of text based dialogues, such as classifiers for sentiment polarity, trends clustering of online messages and hate speech detection. In this work we present the design and the implementation an automatic classification tool for the evaluation of the complexity of Italian texts as understood by a speaker of Italian as a second language. The classification is done within the Common European Framework of Reference for Languages (CEFR) which aims at classifying speakers language proficiency. Results of preliminary experiments on a data set of real texts, annotated by experts and used in actual CEFR exam sessions, show a strong ability of the proposed system to label texts with the correct language proficiency class and a great potential for its integration in learning tools, such systems supporting examiners in tests design and automatic evaluation of writing abilities.

Keywords: Learning systems · Language proficiency classifier · Text classifier · PoS

1 Introduction

The recent increasing diffusion of NLP components for texts classification, such as those applied for the detection of sentiment polarity [4,13], trends and hate speech detection [25] in online communication, is due to the maturity of service technologies, such as text parsing, filtering [20], PoS tagging and the availability of powerful machine learning tools, efficiently integrated in programming environment or libraries [6,22]. Moreover, a vast amount of data is available on the web to be exploited by automated tools, either in structured [5] or semi-unstructured [14] form for language-related purposes, e.g. determining objects'

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11619, pp. 830–841, 2019.
https://doi.org/10.1007/978-3-030-24289-3_61

similarity [12]. While most applications mentioned above work at a somewhat "gross grain" level of linguistic analysis (e.g. polarity, hate speech etc.) not many applications are available for more refined classification of text. Moreover, due to the relevant populations movements in late year and the consequent educational actions for integration of newcomers, there is a great interest in Europe for teaching second language and for all the related supporting learning technologies [11]. Languages teaching is an area in which being able to rank [8] a text, with respect to the language skills and ability required for writing/reading it, is a crucial task in order to meet the teaching goals. During teaching a too easy or too difficult text can prove useless or impossible to understand for the learners; on the other hand, the appropriate classification of a text is essential in assessing or certifying language proficiency. The Common European Framework of Reference for Languages (CEFR) [1] is a standard aiming at classifying speakers language proficiency and it is typically used for assessing second language learners (L2). The CEFR certification consists in six classes of increasing difficult A1, A2, B1, B2, C1, C2, the same letter pairs corresponding to three level of proficiency *beginner*, *intermediate* and *advanced*. The proficiency in a language typically depends on elements such as the use of a basic or advanced vocabulary of term, the complexity of the syntactic structure, the use of idiomatic phrases, the use of synonyms and conceptually close words [15] to avoid repetitions, and so forth. While it is easy for an experienced examiner to assess the level of proficiency necessary for understanding a given text, it is not easy to provide a deterministic procedure for assessing a text, because the relative weight of the linguistic elements is varying depending on the context [3,16,24]. The general approach proposed in this paper is to extract from the text the linguistic features upon which the proficiency class is believed to depend, and instead of directly encoding a classification procedure, we are using the examiners experience, embedded in the labeling of pre-classified exams texts, to train different types of classifiers. In Sect. 2 the general architecture of the system is introduced while, in the next section, the process of extracting the linguistic features relevant for proficiency classification is presented. The classifiers used in the implementation are briefly recalled in Sect. 4. The data set used in the experiments and results obtained are then presented and discussed. Conclusions are finally drawn in Sect. 6.

2 A Text Proficiency Classification Architecture

The workflow of the classification process can be divided into four phases, namely *data set transformation and cleaning*, *extraction of basic linguistic structures*, *extraction of more complex linguistic features* and *classifier training*.

2.1 Data Set Transformation and Cleaning

In the first phase, the appropriate features for the experiments are extracted from the corpus of annotated texts provided by the *Centro Valutazione Certificazioni Linguistiche of University for Foreigners of Perugia* used to test the

text comprehension skill of L2 Italian speakers. In the corpus, each text is annotated with its corresponding CEFR level i.e. A1, A2, B1, B2, C1, C2. For the preliminary experiments, texts of levels B2 and C2 have been used. Some basic pre-processing have been performed, in order to overcome some problems of the TINT software [23], which was used in the subsequent phases.

2.2 Extraction of Basic Linguistic Structures

In this phase, the previously mentioned software Tint is used to elaborate the records. Tint is a natural language processing (NLP) pipeline, developed on the basis of Stanford CoreNLP [21] and specifically adapted to Italian language. Tint offers basic NLP features [23], such as tokenization, sentence splitting and lemmatization, as well as advanced ones with, among others, PoS tagging and dependency parsing. The outputs of this phase are single tokens, sentences and dependency trees information; all of these structure are used for the extraction and computation of appropriate linguistic features [17,19,29] (Table 1).

Table 1. Sample PoS tagging output

Word	PoS TAG	Description
John	SP	Proper noun
is	VA	Auxiliary verb
preparing	V	Verb
the	RB	Determinative article
exam	S	Common noun
of	E	Preposition
Computational	SP	Proper noun
Linguistics	SP	Proper noun
.	FS	Sentence boundary punctuation

2.3 Classifier Training

The architecture is parametric with respect to three different types of classifiers, namely Decision Trees, Random Forests and Support Vector Machines (SVM). Their performance have experimentally analyzed and compared. In the implementation we have used the classifiers' versions available in the Python *scikit-learn* open-source package, which offers a variety of tools for classification, regression and clustering tasks.

3 Linguistic Features Extraction

In order to perform the classification task, it is necessary to build a training set containing the linguistic features of the original corpus which are relevant for assessing language proficiency in the language. The choice has been inspired by the work done for the READ-IT project [10], an SVM classifier aimed at assessing the readability of text in Italian language for people with scarce ability in reading and writing or mild intellectual disability. The features, we have focused on, can be grouped in four main areas:

- Raw text features
- Lexical features
- Morpho-syntactic features
- Syntactic Features

Raw text features express simple quantitative characteristics, such as the mean word length and sentence length of the text. Lexical features look at the relative frequency of words, regardless of their syntactic role and dependencies. For this purpose, three collections [9] are considered: fundamental (F), high usage (HU) and high availability (HA) words. The internal distributions, i.e. the percentage of lemmas belonging to each of the three corpora w.r.t. to the whole text, are calculated, as well as the lemmas within the first 100 tokens in the text and the percentage of basic lemmas in the text. Morphological and syntactic features are also taken into account: lexical density, Part of Speech tags distribution, distribution of verbal moods. Finally, other features which are clues of text complexity are extracted from the dependency parse-tree, such as the maximum depth among all the parse trees, and the number and the depth of subordinate phrases.

4 Classifiers for Text Features

In this section, the classifiers used in the experiments are briefly recalled. Three popular classifying algorithms were considered [26], i.e. Decision Trees, Random Forests and Support Vector Machines. During the training phase, the Nested Cross Validation procedure has been chosen [27,28].

4.1 Nested Cross Validation

Nested Cross Validation is used to tune the hyper-parameters of the classifiers and fit a model using different splits, to avoid information used for performance tests leak into the hyper-parameters optimization process; such procedure reduces over-fitting problems, since different data is used for the two tasks. Therefore, Nested Cross Validation takes into account Cross Validation technique and the Hyper-parameters optimization. Cross Validation is one of the *Resampling Techniques* that repeatedly trains a new model, each time with a different part of the training set in order to encounter each record at least once

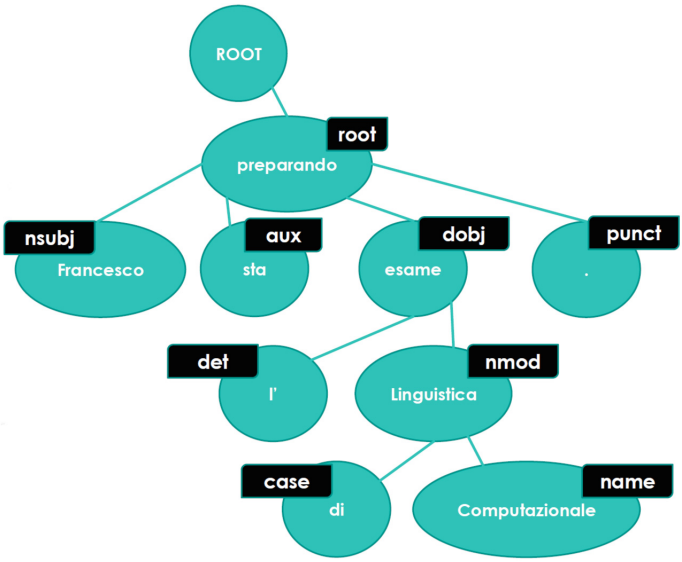


Fig. 1. The dependency parsing tree for a sample sentence.

in the training set. Resampling is an expensive method but luckily, nowadays we are able to perform it with little effort in resources and time. Stratified K-Fold Cross Validation in particular, splits initially the whole data set in two parts: training set and test set. The training set has to be fitted; for k times, it is split into k equal parts where *k-1* pieces represent the training subset and only one subset is the validation set. Because it is stratified, each set has almost an equal distribution of records of each class as the entire data set. At each iteration the validation set is different. The model is fitted in the training part and then, it is tested on the validation set. The result obtained, i.e. a measure of accuracy, will be averaged with the other results returned by the other cross validation rounds. At the end of the process, the model is fitted on the training part and it is ready to make the final evaluation: making predictions on the test set (Figs. 1 and 2).

Hyper-parameters are that kind of parameters of a learning function that need to be set before the learning process begins. On the contrary, parameters are learnt during the learning process. Hyper-parameters optimization is a problem concerned with finding the optimal sequence of hyper-parameters of a learning algorithm in order to obtain the optimal classification model with the best scoring function value: in this project, `f1 macro` has been chosen as scoring function. There are various hyper-parameters optimization techniques; in this work, Grid Search is employed. For each hyper-parameter, i.e. the number of estimators in a random forest, a vector containing the candidate values it can take is created. Then, the Cartesian Product between the candidate values of all the parameters to be optimized is computed and all the combination of

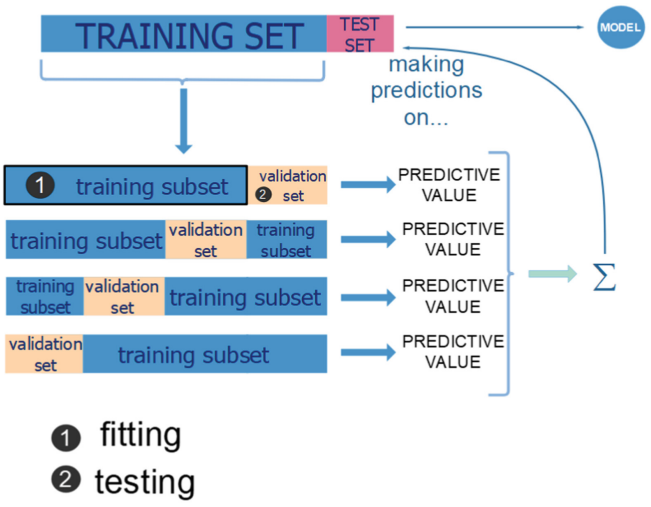


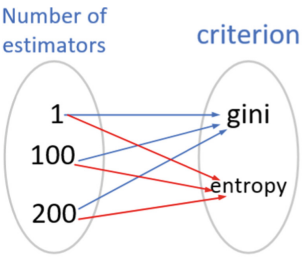
Fig. 2. Phases of cross validation.

parameters in the product, i.e. in the grid, are tested [2]. For each parameter of the learning function, only one of the values inside the corresponding vector will be chosen and assigned to it.

In order to assess the quality of a certain sequence of hyper-parameters in the whole data set, cross validation is executed for each sequence of hyper-parameters. It is clear that setting many values for the hyper-parameters means a critical increase in the complexity of the problem. Nested Cross Validation is more advanced than the classical CV: in fact at each round of the outer cross validation, a Grid Search CV is executed on the training subset. Grid Search CV is essentially the research of the best configuration of the model's hyper parameters, through an internal Cross Validation. For each possible hyper parameters configuration, an internal cross validation is executed on the training subset. The configuration with the lowest test error will be chosen for that outer Cross Validation round and it will make predictions on the test part of the current outer cross validation round (Fig. 3).

4.2 Decision Trees

Decision Trees are data structures characterized by a single root node, internal nodes and leaves nodes. For each node n, except the root, it is defined a unique parent node p [7]. Leaves nodes have no descendants, while internal nodes have both ancestors and descendants. The structure of a Decision Tree embeds a set of rules. For each node, we evaluate a decision; basing on the outcome, one of the edges that start from that node will be followed and a new decision node evaluated. This procedure is iterated until is reached a leaf node. In other words, a path, that begins from the root and ends in a leaf, represents a series



{(1,gini),(1,entropy),(100,gini),
(100,entropy),(200,gini),(200,entropy)}

Criterion / no. estimators	100	200
gini	{gini,100}	{gini,200}
entropy	{entropy,100}	{entropy,200}

Fig. 3. Combinations of parameters obtained by grid search.

of evaluations and decisions where the leaf represents the final classification decision, i.e. the label to assign to the data item under evaluation. There is a unique path from any node to the root. In a leaf node, there is a class label and the subset of instances that belongs to that class and that respect the rules imposed in the path followed in order to reach the leaf. The label assigned to the leaf is the one related to the maximum number of instances there. The best case is when the leaves have only records of one class with Gini impurity almost equal to 0. At the beginning, when the decision tree still needs to be built, we have a set of data items, i.e. a set of vectors feature values, represented by X and the corresponding class labels y, where each vector is assigned a class label. To build the decision tree, there are many techniques. For instance, at each step we could choose a rule of splitting. The vectors that follow it, will be grouped together with their classes in a node. The others will be grouped in another node. This procedure is called recursive binary splitting because it incrementally grows the decision tree by making two child nodes for each parent node. It is furthermore, a greedy procedure because at each time, for each possible splitting rule, the best split with minimal Gini impurity is chosen [18] (Fig. 4).

4.3 Random Forests

As stated before, one of the disadvantages of decision trees is their inclination to overfit the data, especially when a tree becomes very depth. Random Forests avoid the problem following the Bagging Algorithm [4]. The idea of the algorithm is to split the training set T of size n in m training sets Ti of the same dimension n'; there could be some elements repeated in the various subsets, that's why it is called "random sample with replacement" [7,18]. Then, m Decision Tree models are fitted with the m samples created before. In classification problems, each

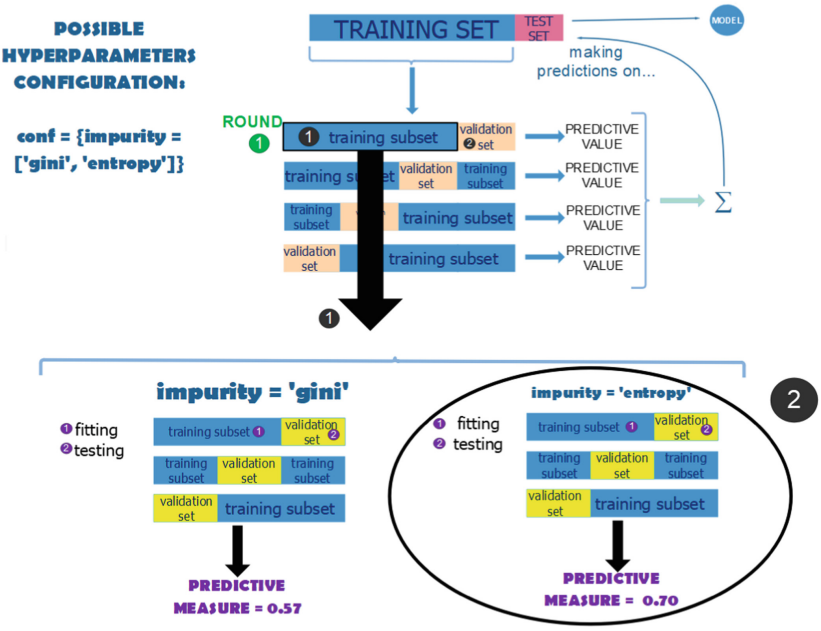


Fig. 4. Nested cross validation.

model will predict a class and the final result will be the one which received the greatest number of votes. The advantage of this method is:

- averaging the results of many trees instead of one, will make the results less sensible to noises.
- training m models with different parts of the training set will make them more independent from each other.

It is suggested to fit at least a few hundred of decision trees. In order to improve the quality of predictions, it is applied not only Bagging Algorithm on the data but also on the features. Feature Bagging at each split makes a "bag" of m features, where usually $m = \sqrt{p}$ p, from a total of p predictors. In the current split, only one of the m predictors contained in that bag will be chosen as node-splitting rule.

4.4 SVM

Support-Vector Machine is a classifier whose goal is to find the optimal hyper-plane that maximizes the margin [18]; for instance, in a binary classification problem, an hyper-plane is the boundary that divides the instances of a class from the other. The optimal hyper-plane is the one that maximizes the separation between the two classes. Hyper-planes are multi-dimensional objects.

The decision boundary could be written as $\overrightarrow{w} \cdot \overrightarrow{x} + b = 0$, where $\overrightarrow{w}$ and b are parameters of the model. Having the following constraints we could predict the class label of a test record:

$$f(\boldsymbol{x_i}) = \begin{cases} 1, & \text{if } \overrightarrow{w} \cdot \overrightarrow{x} + b \geq 1 \\ -1, & \text{if } \overrightarrow{w} \cdot \overrightarrow{x} + b \leq 1 \end{cases}$$

The goal is to maximize the margin:

$$Margin = \frac{2}{\|\overrightarrow{w}\|^2}$$

Sometimes it is not possible to divide into parts the data set through an hyper plane as in the second graph below. However, there's a solution or a "trick" to this problem. The "kernel trick" consists in using a kernel function that allows to map the data set into a higher dimensional space without computing all the new coordinates, in order to have a separable training set. An example is given in Fig. 2.13. In the first example, there is a separable data set. In the second example, it is not possible to divide the data set with an hyper-plane. Applying the kernel trick to the second example we obtain the third example, that is a separable problem. The kernel function in fact, calculating the inner product of each pair of vectors, can bring all the data into a higher dimension, saving up in terms of computational resources. RBF (Radial Basis Function) kernel is one of the most popular.

5 Experiments Design and Results

For the experiments, a Nested Cross Validation procedure has been employed [28]. Nested Cross Validation is used to tune the hyper-parameters of the classifiers and fit a model using different splits, to avoid information used for performance tests leak into the hyper-parameters optimization process; such procedure reduces over-fitting problems, since different data is used for the two tasks. The number of inner and outer stratification was set to 5; in the inner loop, a Grid Search was executed to tune the parameter, using *f1 macro* as the scoring function. In addition, for the decision tree and random Forest classifiers, the maximum depth was set to $\log_2 |features|$ to avoid over-fitting. After tuning the hyper-parameters and creating the model, tests were performed on the test set. The chosen metrics were *accuracy*, *macro-averaged f1 score*, *macro-averaged*

Table 2. Classification results

Classifier	Accuracy	F1-score macro	Precision macro	Recall macro
Decision Tree 2C	0,9292	0,9281	0,9265	0,9303
Random Forest 2C	0,9292	0,9278	0,9278	0,9278
SVM 2C	0,9336	0,9318	0,9355	0,9291

precision and *macro-averaged recall*; results are shown in Table 2. Results are very promising, as for each metric a score higher than 0.9 is achieved, In particular, the best overall classifier is SVM, with Decision Trees and Random Forest delivering slightly lower and comparable performance.

6 Conclusions

In this work we have introduced a model for classifying Italian texts according to the CEFR classification system for second language learners used in language teaching and certification. The proposed architecture relies on linguistics features extractions, which make use of corpora of terms of increasing complexity level and morpho-syntactic features measuring the complexity of dependency parse tree. The tree types of experimented classifiers, decision tree, random forest and SVM, all show outstanding precision performance greater than 90% on Italian texts actually used in official CEFR certification exams. The proposed methodology has a great potential of being extended to other languages, with quite straightforward adaptations. The preliminary experiments have been held on texts belonging to two well separated classes B2 and C2, therefore more systematic further experiments, using texts ranging over all the six CEFR classes, are planned as a future development. A great potential for massive language education is also represented by the integration of the classification module in exam design application supporting examiner or automatic language proficiency examination systems.

References

1. Council of Europe Language Policy Portal. https://www.coe.int/en/web/language-policy/home
2. What is underfitting and overfitting and how to deal with it. https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76
3. Bachman, L., Palmer, A.: Language Assessment in Practice. Oxford University Press, Oxford (2010)
4. Biondi, G., Franzoni, V., Li, Y., Milani, A.: Web-based similarity for emotion recognition in web objects. In: Proceedings of the 9th International Conference on Utility and Cloud Computing, UCC 2016, pp. 327–332. ACM, New York (2016). https://doi.org/10.1145/2996890.3007883, http://doi.acm.org/10.1145/2996890.3007883
5. Bizer, C., Heath, T., Berners-Lee, T.: Linked data: the story so far. In: Sheth, A. (ed.) Semantic Services, Interoperability and Web Applications: Emerging Concepts, pp. 205–227. IGI Global, Hershey (2011). https://doi.org/10.4018/978-1-60960-593-3.ch008
6. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606 (2016)
7. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and Regression Trees. The Wadsworth and Brooks-Cole Statistics-Probability Series. Taylor & Francis, Abingdon (1984)

8. Chiancone, A., Franzoni, V., Niyogi, R., Milani, A.: Improving link ranking quality by quasi-common neighbourhood, pp. 21–26. IEEE Press (2015). https://doi.org/10.1109/ICCSA.2015.19
9. De Mauro, T., Chiari, I.: Il Nuovo Vocabolario di Base della Lingua Italiana (forthcoming)
10. Dell'Orletta, F., Montemagni, S., Venturi, G.: Read-it: Assessing readability of Italian texts with a view to text simplification. In: Proceedings of the Second Workshop on Speech and Language Processing for Assistive Technologies, pp. 73–83. Association for Computational Linguistics, Edinburgh, July 2011
11. European Commission/EACEA/Eurydice: Key Data on Teaching Languages at School in Europe. Eurydice European Unit, Brussels. Technical report (2017)
12. Franzoni, V., Leung, C.H.C., Li, Y., Mengoni, P., Milani, A.: Set similarity measures for images based on collective knowledge. In: Gervasi, O., et al. (eds.) ICCSA 2015. LNCS, vol. 9155, pp. 408–417. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21404-7_30
13. Franzoni, V., Li, Y., Mengoni, P.: A path-based model for emotion abstraction on Facebook using sentiment analysis and taxonomy knowledge, pp. 947–952. IEEE Press (2017). https://doi.org/10.1145/3106426.3109420
14. Franzoni, V., Mencacci, M., Mengoni, P., Milani, A.: Heuristics for semantic path search in Wikipedia. In: Murgante, B., et al. (eds.) ICCSA 2014, Part VI. LNCS, vol. 8584, pp. 327–340. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09153-2_25
15. Franzoni, V., Milani, A.: PMING distance: a collaborative semantic proximity measure, vol. 2, pp. 442–449. IEEE Press (2012) .https://doi.org/10.1109/WI-IAT.2012.226
16. Franzoni, V., Milani, A., Pallottelli, S., Leung, C., Li, Y.: Context-based image semantic similarity, pp. 1280–1284. IEEE Press (2016). https://doi.org/10.1109/FSKD.2015.7382127
17. Graesser, A., McNamara, D., Louwerse, M., Cai, Z.: Coh-metrix: analysis of text on cohesion and language. Behav. Res. Methods Instrum. Comput. **36**, 193–202 (2004)
18. James, G., Witten, D., Hastie, T., Tibshirani, R.: An Introduction to Statistical Learning: With Applications in R. Springer Texts in Statistics. Springer, New York (2014). https://doi.org/10.1007/978-1-4614-7138-7
19. Kincaid, P., Fishburne, R.P., Rogers R.L.: Derivation of new readability formulas for navy enlisted personnel. Research Branch Report, pp. 8–75. Chief of Naval Training, Millington (1975)
20. Leung, C.H.C., Li, Y., Milani, A., Franzoni, V.: Collective evolutionary concept distance based query expansion for effective web document retrieval. In: Murgante, B., et al. (eds.) ICCSA 2013, Part IV. LNCS, vol. 7974, pp. 657–672. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39649-6_47
21. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: Association for Computational Linguistics (ACL) System Demonstrations, pp. 55–60 (2014)
22. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 26, pp. 3111–3119. Curran Associates, Inc. (2013)
23. Palmero Aprosio, A., Moretti, G.: Italy goes to Stanford: a collection of CoreNLP modules for Italian. arXiv e-prints, September 2016

24. Purpura, J.: Cognition and language assessment. In: The Companion to Language Assessment, vol. III, pp. 1453–1476 (2014)
25. Santucci, V., Spina, S., Milani, A., Biondi, G., Bari, G.D.: Detecting hate speech for Italian language in social media (2018)
26. Shalev-Shwartz, S., Ben-David, S.: Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, Cambridge (2014)
27. Varma, S., Simon, R.: Bias in error estimation when using cross-validation for-model selection. BMC Bioinformatics **7**(1), 91 (2006). https://doi.org/10.1186/1471-2105-7-91
28. Wainer, J., Cawley, G.C.: Nested cross-validation when selecting classifiers is overzealous for most practical applications. CoRR abs/1809.09446 (2018)
29. Xiaobin, C., Meurers, D.: Ctap: a web-based tool supporting automatic complexity analysis. Research Branch Report, pp. 8–75. Chief of Naval Training, Millington (1975)