Sociolinguistics and Programming

Fariha Naz and Jacqueline E. Rice
Department of Math and Computer Science
University of Lethbridge
Alberta, Canada
Email: [fariha.naz, j.rice] @uleth.ca

***Abstract*—This paper focuses on the use of machine learning techniques for the analysis of computer programs in order to acquire information about an author's gender. There are few existing studies that address the relationship between linguistics and programming; however, in many areas where language is analyzed it is possible to mine important information about the users of that language associated with set of attribute or coding style. In this work we use open source implementations of machine learning algorithms, specifically, nearest neighbor (K^*), decision tree (J48), and Bayes classifier (Naïve Bayes). These algorithms were applied to C++ programs which were associated with sociolinguistic information about the program authors. Our goal was to classify the programs according to the gender of the author. As indicated by our initial results we have been able to achieve precision of 72.3%, recall of 72%, and f-measure of 71.9% which demonstrates that we can predict the gender of the authors of C++ programs.**

I. Introduction and Motivation

In the field of sociolinguistics it is known that individual differences in the use of a language within a society can affect or reflect social factors. Linguistic variables correlate with social variables such as age, socio-economic status, gender, ethnicity, and region to create sociolinguistic variation [1]. However, very few researchers have applied this analysis to the field of computer programming. We are thus interested in answering the following question: do social factors impact the development of C++ programs? To begin to answer this question here we report on our efforts to categorize C++ programs based on the gender of the programmers.

Coding is a "deliberate action across cultural and technological fields" [2]. While the syntax of a computer program is quite strictly determined by the programming language, choices left up to the programmer include the use of different numbers and types of loops, datatypes, keywords, operators, and comments. Everything in software can affect comprehension, including the code, documentation, comments, and structure [3]. Thus examining the language use at this micro level can offer insight into how concepts are communicated in all of these components. Many data analysis problems may be posed as machine learning problems [4], [5]. For this reason, we propose to use the machine-learning techniques that are part of the WEKA tool suite [6] to perform supervised learning (or classification) of documents written in the C++ programming language. In doing so we treat C++ programs as text documents and convert these programs into a numerical representation using tf-idf [7]. To classify these programs according to the gender of the author, we use techniques such as nearest neighbor, decision tree, and Naïve Bayes (NB) classification models that may be reused to categorize future data.

978-1-4673-7788-1/15/$31.00

This research moves towards an improved understanding of different coding styles and their relationships with sociolinguistic variables. Different patterns may be identified and analyzed on the basis of differences in usage of various features (or attributes). These features include comments, keywords, loops, and operators. As a possible outcome of this work, new integrated development environments (IDEs) could be developed to aid in communication between programmers between different sociolinguistic groups. For instance, miscommunication could involve missing, extra, or incorrect information, which could be facilitated and/or minimized by the IDE design [8]. Another contribution could be in detecting plagiarism of programs within groups of programmers [9].

This paper is composed of the following sections: Section II provides the background of this study including a brief overview of sociolinguistics, software linguistics, and machine learning; section III addresses the literature discussing gender differences in text documents and authorship analysis of computer programs; section IV discusses our methodology; and sections V,VI, and VII discuss our results, limitations, and conclusions.

II. Background

A. Sociolinguistics

The field of sociolinguistics focuses in part on the identification of different social factors and their correlation with variants in the use of natural language within a society [1]. The social factors include age, gender, degree of education, experience, ethnicity, and socio-economic status (SES). Thus, the area of sociolinguistics provides a way to analyze whether linguistic variability correlates with social variability. In a society people belonging to different social classes or age groups are distinguishable based on features of their accent and grammar use. For example, people from North America, Britain, and USA are distinguishable based on their varying use of different English words in terms of their pronunciations.

To identify sociolinguistic differences within a society based on the use of a natural language, first the society is identified by its contribution toward the knowledge of a particular

issue or language [10]. People from this society who are willing to provide their expertise, that is, information on their knowledge of the language, are then selected. Data is collected from participants in order to investigate variability in the usage of the linguistic features (variables) under investigation. The data is analyzed to determine the range of variants for each linguistic variable and the ways in which this linguistic variability correlates with social variability. In the area of sociolinguistics text documents written in various natural languages have been analyzed by different researchers [4], [11], [12]. In particular, the interpretation of gender differences have been investigated using various techniques including machine learning and statistical analysis, as well as a combination of these approaches.

B. Machine Learning

In recent years approaches such as statistical analysis, machine learning, and pattern recognition have been used in the analysis of different kinds of data ranging from textual data (documents) [5] to molecular data (genes or proteins) [13]. Machine learning algorithms learn what trends are present in the given data and then can be used to make decisions for the analysis of future data.

There are various types of machine learning algorithms [14], [15]. The assignment of datasets based on pre-defined class labels is called supervised learning (or classification). For example, values such as *male* and *female* may represent two different classes, and then the data instances that are present in the datasets are associated with these class labels. Here we have carried out supervised learning using the gender of the authors of C++ programs(as our data instances) as pre-defined class labels.

Supervised learning algorithms require a particular data format [16] to carry out the task of analyzing data in order to extract useful information. Term Frequency-Inverse Document Frequency (tf-idf) is one of the techniques used in the area of automatic text retrieval [7] to convert text documents to an appropriate data format that can be used to analyze text documents. Using tf-idf the dataset is transformed to a form of numerical representation of documents on the basis of identified features. The tf-idf is a "vector-space model which represents an object as a vector of weighted indexing term, and define object similarity in terms of those vectors." [17]. The tf-idf vector is composed of the tf part and the idf part. Term Frequency (tf) deals with a single sample and counts the occurrences of a specific feature. Inverse Document Frequency (idf) counts the occurrences of the same feature within the entire dataset [17]. The product of tf and idf gives the tf-idf score of a feature.

C. Classification Algorithms

In this work we used three supervised learning algorithms: nearest neighbor (K*), decision tree (J48), and Bayes classifier (Naïve Bayes). Each of these are implemented as part of the open-source WEKA software [6].

K* Algorithm:

The K* algorithm is described in the WEKA software as finding the "nearest neighbor with generalized distance function" [15]. This algorithm uses the entropic distance measure to calculate the distance between similar instances in the dataset. The distance is computed between new/test instance x and the existing/training instances y_i. The test instance is labeled with the associated class label on the basis of the closest k-nearest existing instances, y_i [18] as in the equation:

$$K^*(y_i, x) = -\log P^*(y_i, x),$$

where $i \in \{1, 2, ...k\}$ and the probability of all possible paths from (x) to (y_i) is represented as P. The number of important neighbor instances can be found in the "sphere of influence" [19].

J48 Algorithm:

The J48 algorithm is also available as part of WEKA. Using J48, a flow chart-like tree structure, which is referred to as a decision tree, is constructed recursively on the basis of important attributes that accurately partitions instances into distinct classes. The tree is composed of a root node, internal nodes, branches, and leaf nodes. To classify test tuples on the basis of an associated class label a path is traced from the root to a leaf node.

The J48 algorithm works on a "divide-and-conquer" technique [15]. This algorithm computes the probability distribution of instances in the given dataset after learning from instances and analyzing the distribution of classes among instances. To determine the "best" attribute, which extracts the maximum amount of information from a set of instances to make a prediction, a gain ratio feature selection method is incorporated.

Naïve Bayes Algorithm:

In this study, a simple Bayesian classifier is used, which is referred to as a Naïve Bayes. Bayesian classifiers compute the probability of a given instance. Naïve Bayes (NB) is widely used to categorize data with two-classes, for instance, assigning a dataset with class labels of *male* and *female* is a two-class problem. This algorithm is implemented as a "standard probabilistic naïve bayes classifier" [15] in WEKA and is based on the assumption of "class conditional independence" [20]. This means attributes that are associated with a given class label are independent of each other. Thus, the probability of attributes is computed as shown in the following equation:

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)},$$

where $P(C_i|X)$ is the maximum probability of an instance, *X*, among all class labels. $P(X|C_i)$ is the probability of *X* associated with a specified class. $P(C_i)$ is the probability of

instance *X* belonging to a particular class C_i. $P(X)$ is the probability of an instance *X* within the dataset, and is often constant.

D. Cross Validation Technique

In a situation where data is limited, the technique of cross-validation can be used to reduce the risk of overfitting [14], [15]. The reason that overfitting can occur is that the generalization from the given training tuples was not learned accurately due to the small size of a dataset. Thus, there is a risk of incorrect predictions being "learned" as correct by the model [16], [21]. The k-fold cross validation technique is commonly used to mitigate the risk of overfitting of a dataset.

Cross validation is a technique in which a dataset is partitioned using a fixed number of folds (k) [15]. Leave-one-out cross validation (LOOCV) is *n*-fold cross validation and can be applied using WEKA, where n=k and *n* is the total number of instances that are part of the dataset. There are *n* turns in which a single instance is left out of the training dataset and becomes a test dataset. The remaining instances (tuples) are used for training the model. After the completion of *n* turns, evaluation metrics of all runs (or iterations) can be used as the final estimates to demonstrate the predictive ability of the developed model.

E. Evaluation Metrics

The predictive ability of a model (or classifier) is identified by analyzing the confusion matrix and various evaluation metrics including precision, recall, and f-measure. A confusion matrix represents how well a model identifies instances associated with specific class labels [14]. Using a confusion matrix, the values of various evaluation metrics can be computed. For instance, Table I illustrates the confusion matrix resulting from attempting to classify programs by the author's gender, where we have restricted gender to male or female.

TABLE I
2X2 CONFUSION MATRIX.

Gender	Female	Male	Total
Female	TP	FN	P
Male	FP	TN	N
Total	P'	N'	P+N

In the above confusion matrix, computer programs from female programmers are positive instances, while those from male programmers are negative instances [14]. True positives (TP) are instances that are correctly labeled as female-written programs. True negatives (TN) are instances that are correctly labeled as male-authored programs. False positives (FP) are instances that are misclassified as female-written. False negatives (FN) are instances that are misclassified as male-written.

In supervised learning of data, precision and recall evaluation metrics are widely used. Recall provides the number of data instances that are correctly labeled as a specific class label; however, there is no information about data instances that are mislabeled by the model. Precision gives the number of data instances that are labeled as a specific class and actually belonged to that specific class. However, sometimes a model may acquire more precision then recall. Thus, some researchers in this area and in information retrieval areas use another evaluation metric, f-measure [14], [15], [21]. This metric is appropriate because it represents the harmonic mean of both precision and recall. F-measure is the combination of precision and recall, and is calculated as, (2*Precision*Recall)/(Precision+Recall).

III. RELATED WORK

A. Software Linguistics

In 1982 Misek-Falkoff [8] identified a new domain that she called "Software Linguistics" as a result of her investigation to identify the relationship between software and linguistics. She observed that by using a set of rules for a natural language, a text written in that language may be analyzed using syntactic and semantic structures of a language. For example, natural language analysis may be performed by examining each part of a word or a sentence. Misek-Falkoff, further suggested that similar techniques would be applicable towards the analysis of computer programs.

B. Gender Identification

More recently Argamon et al. [4] explored gender differences in French literature by performing classification of text documents using SVM$^{\text{light}}$ [5]. A model was developed to discriminate between male- and female-authored documents on the basis of word distribution, usage, and frequencies. Argamon et al. observed that female authors of texts used more personal pronouns and negation, whereas male writers used more determiners and quantifiers. Common function words such as articles were used equally by males and females; however, personal pronouns and emotional language were more commonly used in texts written by female authors. The model was able to assign gender labels to unclassified data with an accuracy of 90%; moreover, the model allowed the extraction of the features that played the highest role in the classification process.

Argamon et al. [11] also investigated gender differences in English literature, utilizing the British National Corpus (BNC). This dataset was analyzed on the usage of parts-of-speech. A machine learning technique called the EG algorithm was used to select a small set of the most useful features from a list of over 1000 features. The EG algorithm was able to identify 50 features that played the most important part in the distinction of male- and female- authored texts. Argamon et al. concluded that the findings in the French texts replicated similar analysis from the English texts. This cross-linguistic similarity illustrated the usefulness of the machine learning techniques across datasets from different languages.

C. Authorship Analysis

Authorship analysis has been carried out mostly with datasets composed of text documents written in natural language. However, there are some studies which investigate authorship analysis of artificial language datasets (computer programs). Krsul and Sappford [9] classified C programs to discover a set of features that might be used to identify authorship. The intent was to use such information in resolving authorship disputes, detecting plagiarism, and in the construction of a programmer's "signature" [9], [22]. The distinction of authors was based on writer-specific features coinciding with small variations in programs by a given author but large variations in the whole dataset. LNKnet [23] software was used to apply machine learning algorithms and statistical analysis of features to refine the feature list in order to discriminate more accurately between programmers. Along with machine learning techniques used to identify authorship, statistical analysis of C programs was carried out using SAS [24]. Krsul and Sappford concluded that distinct authors can be found, even using a small dataset of C programs, because programmers tend to develop programs using style conventions with which they are personally familiar.

IV. Methodology

To employ machine learning methods on the textual data there are a few steps that must be performed including choosing the document representation, learning method, and testing protocol [3], [12]. An overview of our methodology is given in Figure 1.

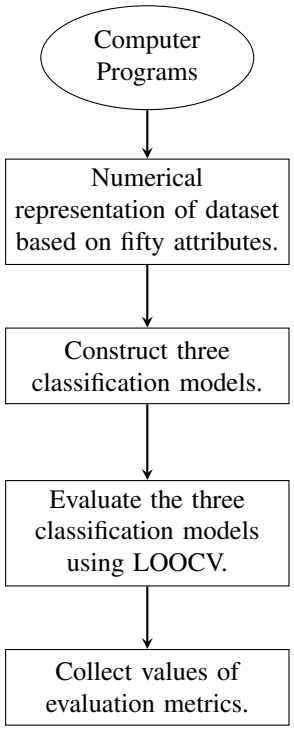

Fig. 1. Overview.

Step 1: Data

Our dataset consisted of C++ programs collected as part of assignments and projects from computer science classes at the University of Lethbridge. The sociolinguistic information was gathered via a survey which was provided to each participant. Male programmers were over-represented in the dataset due to the low number of female students in these courses as shown in Table II. Thus, in our attempt to create a balanced dataset of 100 C++ programs, we needed to oversample the dataset; that is, multiple computer programs written by a single female programmer were used as part of the dataset. Each C++ program was cleaned manually to avoid the misrepresentation of the identified features. This manual cleaning included removing the participant's information to preserve the confidentiality, removal of white spaces and removal of comments that contained features.

TABLE II
Participants

Gender	Participants	Provided Samples	Used Samples
Male	65	240	50
Female	19	64	50
Total	84	304	100

Step 2: Document Representation

To create the numerical representation of computer programs, we used a small list of features such as operators, keywords, loops, and comments [21]. We also applied the term frequency and inverse document frequency (tf-idf) technique, as described in section II-B, to compute the frequency of features in each author's computer program and in the entire dataset. The data instances for our experiments consisted of C++ programs collected from University of Lethbridge students. Each instance is composed of class labels (representing "male" and "female" programmers) and feature vectors. The set of features with its tf-idf score [7] for each program is referred to as a feature vector.

TABLE III
List of Attributes

Type of Attributes	Fifty Attributes
Keywords	#include, #define, using, void, cout, cerr, cin, return, exit, int, float, char, const double, bool, new, break, public, private
Operators	$<$, $->$, $>$, &, &&, +, ++, !, ! =, ==, =, $-$, $--$, $*$, /, |, ||, /=, +=, -=, $*$=,<=, >=
Comments	//*, //, /*
Brackets	{}, ( )
Loops	for, while, switch

Step 3: Supervised Learning Models

In this work we constructed three supervised learning (classification) models: K*, J48, and Naïve Bayes (as described in

section II-C), using an open-source machine learning software, WEKA [6]. These models were created to classify computer programs based on the gender of the programmers.

V. Results and Discussion

To develop three models we employed the leave-one-out cross-validation (LOOCV) technique, which is described in section II-D. We used confusion matrices (as reported in section II-E) to perform comparative analysis and calculated evaluation metrics that are associated with each classification model. Table IV shows the confusion matrix for the decision tree classification model. We observe that this model is able to accurately classify 33 computer programs out of 50 programs as female-authored. Similarly, 30 out of 50 computer programs are correctly classified as male-written programs. However, this model misclassifies 17 as male-authored and 20 as female-authored programs.

TABLE IV
J48 Confusion Matrix

Gender	Female	Male
Female	33	17
Male	20	30

Table V shows the confusion matrix for the nearest neighbor model. This model accurately classifies 39 computer programs out of 50 as female-written programs and 33 out of 50 computer programs as male-written programs. However, 11 were mislabeled as male-written and 17 as female-written programs.

TABLE V
K* Confusion Matrix

Gender	Female	Male
Female	39	11
Male	17	33

In Table VI, we observe that the Naïve Bayes (NB) classification model correctly labels 33 samples out of 50 as programs developed by female and male programmers. This model misclassifies 17 samples as male and female-authored programs. We observe these differences in the performance of various supervised learning models due to the differences in their underlying algorithms.

TABLE VI
Naïve Bayes Confusion Matrix

Gender	Female	Male
Female	33	17
Male	17	33

We also perform a comparative analysis on the basis of f-measure to address concerns where precision and recall may not provide an accurate assessment of a model's predictive capabilities. As shown in Table VII, we achieve f-measure of 63% with the decision tree (J48) model because of the small number of correctly classified computer programs. The highest f-measure is achieved by the nearest neighbor (K*) model. This means that the classification models developed in this study are able to correctly predict the gender of approximately 63%-72% of the samples in our dataset. We would expect the predictive performance of these models on a new dataset would lie within this range.

TABLE VII
Performance of the Classification Models

Models	**Precision (%)**	**Recall (%)**	**F-measure (%)**
K*	72.3	72	71.9
J48	63	63	63
NB	66	66	66

VI. Limitations

In this study, there are potential limitations that threats the validity of the work:

1) The data in these experiments was male skewed. To address this we used multiple computer programs written by female programmers. Because of this our female-authored data may be biased towards a particular individual's programming style. This implied that multiple computer programs showed certain coding practices, resulting in similar choices in terms of coding style.
2) In programming courses, there is a possibility of imposing a specific set of coding styles regarding the usage comments and data types. This could skew the style of newly beginning programmers. However, there is still the possibility for a great deal of individuality in a program, especially by the end of the semester.
3) In this study, a large number of participants belonged to the computer science courses that were offered at the University of Lethbridge. Thus, the computer programs utilized in our study were developed exclusively by participants with little experience in comparison to industrial programmers with experience in developing software in C++ programming language.

VII. Conclusion and Future Work

In this work, we categorized computer programs on the basis of the gender of the computer programmers. We used dataset composed of C++ programs written by male and female programmers. We developed three classification models: nearest neighbor (K*), decision tree (J48), and Bayes classifier (Naïve Bayes). We concluded that for a limited dataset of C++ programs it is possible to utilize machine learning techniques to differentiate between male and female programmers. We are able to achieve 72.3% of precision, 72% of recall, and

71.9% of f-measure which established that social factors such as gender are reflected in the use of the C++ programming language.

We are planning to extend this preliminary work by applying attribute selection algorithms to distinguish the top-ranked features out of 50 features. This way we would be able to identify features that are prevalent in *male* and *female* written computer programs and observe the impact on the evaluation metrics of the models. In addition, we plan to expand this work to examine different sociolinguistic variables, including the first spoken language of programmers, the first learned artificial (or programming) language, or the years of experience in the C++ programming language to observe their effects on the artificial language use by programmers.

ACKNOWLEDGMENT

This research was funded in part by a grant from the University of Lethbridge Research Fund (ULRF). We would also like to extend our appreciation to all of the students and faculty who agreed to participate and helped us to carry out this research. Many thanks to committee members Dr. Kevin Grant, Dr. Inge Genee, and Dr. Javid Sadr for their continuous support towards this research.

REFERENCES

[1] W. Labov, "The linguistic variable as a structural unit," *Washington Linguistics Review 3*, pp. 4–22, 1966.

[2] G. Cox, A. McLean, and F. B. Berardi, *Speaking Code: Coding as Aesthetic and Political Expression*. MIT Press, 2013.

[3] J. E. Rice, I. Genee, and F. Naz, "Linking linguistics and programming: How to start?(work in progress)," in *Proc. 25th Annual Psychology of Programming Interest Group Conference - PPIG*, 2014.

[4] S. Argamon, J. Goulain, R. Horton, and M. Olsen, "Vive la différence! text mining gender difference in french literature @ONLINE," *Digital Humanities Quarterly*, vol. 3, no. 2, 2009, http://www.digitalhumanities.org/dhq/vol/3/2/000042/000042.html.

[5] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," Universitá Dortmund, Tech. Rep.

[6] G. Holmes, A. Donkin, and I. H. Witten, "Weka: A machine learning workbench," in *Proc. Australia and New Zealand Conf. Intelligent Information Systems*, Brisbane, Australia, 1994.

[7] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 5, no. 24, 1988.

[8] L. D. Misek-Falkoff, "The new field of software linguistics: An early-bird view," *SIGMETRICS performance evaluation review*, vol. 11, no. 2, pp. 35–51, 1982.

[9] I. Krsul and E. Spafford, "Authorship analysis: Identifying the author of a program," *Computers and Security*, vol. 16, no. 3, pp. 233–248, 1997.

[10] W. O'Grady and J. Archibald, *An Introduction Contemporary Linguistic Analysis*, 6th ed. Pearson Education Canada, 2008.

[11] S. Argamon, M. Koppel, J. Fine, and A. R. Shimoni, "Gender, genre, and writing style in formal written texts," *TEXT*, vol. 23, pp. 321–346, 2003.

[12] M. Koppel, S. Argamon, and A. Shimoni, "Automatically categorizing written texts by author gender," *Literary and Linguistic Computing*, vol. 17, no. 4, pp. 401–412, 2002.

[13] P. Pavlidis, I. Wapinski, and W. S. Noble, "Support vector machine classification on the web," vol. 20, no. 4, pp. 586–587, 2004.

[14] J. Han, M. Kamber, and J. Pei, *Data Mining Concepts and Techniques*, 3rd ed. Elsevier and Morgan Kaufmann Publishers, 2012.

[15] I. H. Witten and E. Frank, *Data Mining Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann Publishers, 2005.

[16] C. Hsu, C. Chang, and C. Lin, "A practical guide to support vector classification," 2010.

[17] G. Zweig, P. Mguyen, J. Droppo, and A. Acero, "Continuous speech recognition with a tf-idf acoustic model," *International Speech Communication Association*, September 2010.

[18] S. Vijayarani and M. Muthulakshmi, "Comparative analysis of bayes and lazy classification algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, no. 8, August 2013.

[19] J. G. Cleary and L. E. Trigg, "K*: An instance-based learner using an entropic distance measure," in *12th International Conference on Machine Learning*, 1995, pp. 108–114.

[20] S. M. Kamruzzaman, F. Haider, and A. R. Hasan, "Text classification using data mining," *Proc. International Conference on Information and Communication Technology in Management (ICTM-2005)*, May 2005.

[21] R. P. L. Buse and W. Weimer, "Learning a metric for code readability," *IEEE Transactions on Software Engineering (TSE Special Issue on the ISSTA 2008 best papers)*, vol. 36, no. 4, pp. 546–558, 2010.

[22] S. Burrows and S. M. Tahaghoghi, "Source code authorship attribution using n-grams," *Proceedings of the 12th Australasian Document Computing Symposium, Melbourne, Australia, RMIT University*, pp. 32–39, 2007.

[23] L. Kukolich and R. Lippmann, *LNKnet User's Guide*. RM E32-300, 200 Carleton Street, Cambridge, MA 02142-1324: MIT Lincoln Laboratory, MIT Technology Licensing Office, April 1995.

[24] *SAS/STAT 9.2 User's Guide, 1:ANOVA-FREQ*, 4th ed., 2008.