Recognition of arthropod species names using bigram-based classification

Jennien Raffington
University of Guelph Biodiversity Institute of Ontario
Dirk Steinke
University of Guelph Biodiversity Institute of Ontario
Dan Tulpan (dtulpan@uoguelph.ca)
University of Guelph https://orcid.org/0000-0003-1100-646X

Research article

Keywords: machine learning, classification, species names, arthropod, bigram

Posted Date: May 11th, 2020

DOI: https://doi.org/10.21203/rs.3.rs-26532/v1

Recognition of arthropod species names using bigram-based classification

Jennien Raffington [1]*, Dirk Steinke [1,2]*, Dan Tulpan [3,4]*§

[1] University of Guelph, Centre for Biodiversity Genomics, Guelph, ON, Canada

[2] University of Guelph, Integrative Biology, Guelph, ON, Canada

[3] University of Guelph, Centre for Genetic Improvement of Livestock, Department of Animal Biosciences, Guelph, ON, Canada

[4] University of Guelph, Department of Computer Science, Guelph, ON, Canada

*These authors contributed equally to this work

§Corresponding author

Email addresses:

JR: charles@darwin.co.uk

DS: jane@darwin.co.uk

DT: johnsmith@darwin.co.uk

Abstract

Background

The task of recognizing species names in scientific articles is a quintessential step for a large number of applications in high-throughput text mining and data analytics, such as species-specific information collection, construction of species food networks and trophic relationship extraction. These tasks become even more important in fast-paced species-discovery areas such as entomology, where an impressive number of new arthropod species are discovered each year. This article explores the use of two-character n-grams (bigrams) in machine learning models for arthropod species name recognition. This particular method has been previously applied successfully to the task of language identification [1] but the application to species name identification had yet to be explored.

Results

Arthropod species names, regular English words used in scientific publications and person names were collected from the public domain and bigrams were extracted and used as classifier features. A number of learning classifiers spanning 7 algorithmic categories (tree-based, rule-based, artificial neural network, Bayesian, boosting, lazy and kernel-based) were tested and the highest accuracies were consistently obtained with LIBLINEAR [2], Bayesian Logistic Regression [3], the Multilayer Perceptron [4], Random Forest [5], and the *LIBSVM* [6] classifiers. When compared with dictionary-based external software tools such as GNRD [7] and TaxonFinder [8], our top-3 classifiers were insensitive to words capitalization and were able to correctly classify novel species names that are absent in dictionary-based approaches with accuracies between 88.6% and 91.6%.

Conclusions

Our results suggest that character bigram-based classification is a suitable method for distinguishing arthropod species names from regular English words and person names commonly found in scientific literature. Moreover, our method can also be used to reduce the number of false positives produced by dictionary-based methods.

Data availability: All datasets are publicly available at:

http://animalbiosciences.uoguelph.ca/~dtulpan/papers/specrec2020

Keywords

machine learning, classification, species names, arthropod, bigram

Background

With recent advancements in technology and scientific methods, research findings are produced at rates unseen before. There are approximately 28,000 scientific journals with over 2.5 million articles published every year [9]. The majority of these journals has been digitized providing access to online articles, nevertheless there is a considerable amount of valuable research results that is hard to find and to connect with other results. To this extent, natural language processing tools have been developed to help wade through the vast amount of available information.

Species name identification is a process utilized to link information from different publications [10]. This is useful when attempting to extract information about a specific organism or taxon and can be formulated under the Named Entity Recognition (NER) framework, which focuses on finding and classifying named entities in text [11]. Recognition of species names from a text corpus has been attempted by various methods that could be grouped in three categories: dictionary-based, rule-based and machine learning-based.

A dictionary-based approach involves creating a large list of words relevant to the knowledge domain and then matching text against that dictionary. This approach is particularly powerful and useful when large collections of species names are available from well-trusted taxonomic sources such as the Catalogue of Life [12] and the Entomology Society of America's Common Names of Insects Database [13]. The dictionary method is advantageous because it guarantees a high percentage of true positives. A curated dictionary will only contain species names that have been vetted and checked ensuring that only pertinent text is extracted. However, this approach requires extensive memory due to the size of the required libraries and it is unable to

identify misspelled words, optical character recognition errors or species names that are absent from the dictionary.

The rule-based approach uses a set of rules such as capitalization, word variants, phrases and patterns to match text [14]. The advantage of rule-based methods is speed because capitalization and punctuation can identify putative species names quickly. It allows for fast filtering of large amounts of data, so more arduous processes can be completed on smaller subsets. This can help to overcome disadvantages of the method that may omit particular fields in the text due to their increased variability, therefore leading to false negatives.

The machine learning approach generally involves building a model to detect and classify specific entities. These tools can be generic or specific to a given ontology and they do not depend on a dictionary. The strength of the machine learning approach consists in its ability to recognize species names that may be misspelled, may not match any rules or may not be included in a dictionary. It widens the scope of what can be found but some models might be susceptible to biases towards their training sets.

In practice, most tools represent hybrid combinations of two of the three approaches. For instance, TaxonGrab [15] is a tool that incorporates both, the dictionary and the rule approaches to identify taxonomic names. Similarly, LINNAEUS [16] uses a combination of rules and a dictionary to identify species names in biomedical literature. Both, NetiNeti [10] and Organism-tagger [17] utilize a combination of rules and machine learning approaches to identify species names. Solr-Plant [18] is part of a subset of tools that focus on plant name identification and utilizes an Apache Solr-based system that incorporates the Smith-Waterman alignment algorithm [19]. Whatizit [20] is another suite of text mining tools that finds mentions of organisms in databases and

COPIOUS [21] is a corpus of biodiversity entities that can be used to train text analysis tools.

The Global Name Recognition and Discovery (GNRD) tool [7] identifies species names from unstructured text by looking up the words in multiple species name repositories. The GNRD tool is part of a suite of web tools that perform multiple tasks related to biological species names (Global Architecture). It encompasses tools that parse, index and resolve text. The Biodiversity Heritage Library uses GNRD to analyse any new or updated information on their pages. Until recently GNRD utilized TaxonFinder [8], a dictionary-based approach to detect Latin scientific organism names at all ranks including kingdom, phylum, class, order, family, genus, species and subspecies in plain text.

N-grams have been previously used to automatically identify the language or dialect used in a text corpus [1]. They were built for different languages since n-gram frequencies tend to be unique for each language or dialect. The n-gram method has been applied to several other tasks such as authorship attribution [7], detection of malicious code [8] and file type identification [9].

This manuscript introduces an approach that uses character bigrams extracted from two-word combinations to train, test and validate machine learning classifiers with the purpose of identifying arthropod species names in the presence of regular English words and person names that appear in scientific publications. We used the frequencies of character bigrams to build 15 models utilizing machine learning classifiers spanning seven algorithmic categories (tree-based, rule-based, artificial neural network, Bayesian, boosting, lazy and kernel-based) and we tested the models on three data sets corresponding to three classification problems described in detail in the Methods section.

Results and Discussion

In this work, we investigate the ability of character n-grams-based classifiers to distinguish arthropod species names from regular English words and person names by solving the three problems described in the Methods section. Classification accuracies are summarized in **Error! Reference source not found.**.

Classification accuracy

For all three problems (see Methods), the LIBLINEAR classifier outperformed the other classifiers with accuracy values of 97.53% (P1), 94.70% (P2) and 91.31% (P3). Random Forest ranked second for problems P1 (96.88%), P2 (93.55%) and P3 (90.98%), while the MLP and SVM classifiers consistently ranked among the top 5. The Bayesian Logistic Regression classifier ranked 3rd and 4th, respectively on P1 and P2, while it could not be executed on P3 due to its binary class applicability limitation. The Decision Table [22] rule-based classifier's accuracy ranked among the lower half of the models for all three classification problems. The J48 method performed well in both two-class problems but ranked 11th with an accuracy of 77.67% for the three-class problem. The poorer performers across the board were AdaBoost (ranked last on all problems) and Lazy IBK, which seem to struggle when applied on the three datasets.

The three-class problem proved to be more difficult for all models and we observed a significant overall decrease in performance ranging between 5.9% for Random Forest and 33.6% for AdaBoost.

Classifying two-word phrases that represent people names, species names and English words (P3) proves to be more difficult than differentiating species names and English words (P1) or species names and persons names (P2). Different languages have

different n-gram frequencies [23] and require independently built classifiers to distinguish among them. Species names tend to be based on Latin and ancient Greek while the text in a scientific article is largely English. Their n-grams frequencies are different, and some n-grams are more prevalent in one of the two categories, which explains the higher classification accuracy for solving problem P1. This also explains why n-grams are good features for classifiers applied to language identification [24]. Moreover, each language has its own n-gram frequency distribution. Similarly, person names originating from different cultures and language backgrounds have particular n-gram frequency distributions. As a majority of first and last names typically used in English-based languages have either a Latin or Greek origin, the frequency of certain bigrams is high in both (**Error! Reference source not found.**), the SCI and PEO class instances. In contrast, there is less overlap between sets of high frequency bigrams in SCI and ENG class instances, which could explain why the overall classifier accuracies are higher when solving P1 compared to P2.

Comparison with other species name identification tools

While we attempted to compare our results with nine external tools such as NetiNeti [10], TaxonGrab [25], LINNAEUS [16], SpeciesTagger [26], Organism-tagger [17], Solr-Plant [18], Whatizit [20] and COPIOUS [21], we could only perform comparisons with the Global Name Recognition and Discovery (GNRD) [7] and TaxonFinder [8], the other tools were either not available or did not function as described in the accompanying documentation. The two selected tools are both dictionary-based and therefore the results are expected to be perfect for the recognition of species names, while our method does not use any type of dictionary.

Table 1. *Performance comparisons with GNRD and TaxonFinder*

Method	Validation dataset classes	Num. Correctly Predicted Instances	Prediction Accuracy [%]
LIBLINEAR	V_SCI + V_ENG + V_PEO	1377	91.8
MLP	V_SCI + V_ENG + V_PEO	1332	88.8
Random Forest	V_SCI + V_ENG + V_PEO	1326	88.6
LIBSVM	V_SCI + V_ENG + V_PEO	1284	85.6
GNRD	V_SCI	500	100
GNRD	V_ENG	500	100
GNRD	V_PEO	442	88.4
GNRD	V_SCI + V_ENG + V_PEO	1442	96.1
taxonfinder	V_SCI	495	99.0
taxonfinder	V_ENG	500	100
taxonfinder	V_PEO	493	98.6
taxonfinder	V_SCI + V_ENG + V_PEO		99.2

We first compared our results (**Table 1**) with the Global Names Recognition and Discovery (GNRD) tool using the validation dataset. With an overall accuracy of 96.1%, GNRD identified all species names from the V_SCI class because all species names were present in the GNRD dictionary, but it was able to recognize them only if they were capitalized. While GNRD correctly identified all English words, it misidentified 58 person names out of 500 as species names (false positives). We also tested TaxonFinder on the three validation datasets and, overall, TaxonFinder outperformed GNRD with a total accuracy of 99.2%. TaxonFinder also outperformed GNRD when tested against person names with a total accuracy of 98.6%, while GNRD obtained only 88.4%. TaxonFinder misidentified only seven person names. A closer look at the misidentified person names in both GNRD and TaxonFinder results (e.g. Annamaria, Petunia, Jappa, Fiscella, Cabello, Dacosta, Sabota and Basia)

revealed that parts of the names are either substrings or terms of some species names such as *Anomius annamariae*, *Petunia caesia*, *Jappa kutera*, *Cronia fiscella, Cabello eugeni, Dacosta australis, Calendulauda sabota and Pyrinia sabasia.*

In comparison, our top four classifiers solved three-class problems with accuracies ranging from 85.6% to 91.8% on the validation datasets, which is consistent with the performance obtained when applied on the combined training and testing dataset (*SCI+ENG+PEO*) with 9,000 two-word bigrams.

Our classifiers can also be used as a post-processing step to further improve the prediction of dictionary-based species name recognition tools and reduce the number of false positives. When we filtered the GNRD results using the LIBLINEAR and the MLP bigram-based classifier for the V_PEO dataset, only 13 people names remained miss-classified, while 40 were correctly classified as non-species names, thereby improving the accuracy of GNRD by 9% to a total of 97.4%.

Runtime analysis

Classifier runtimes were measured on an HP Notebook – 14-cf0018ca equipped with a 4-core Intel Core i5-8250U CPU (base frequency of 1.6 GHz, 4MB Cache), 8 GB DDR4-2400 SDRAM, a 256 GB PCIe NVMe M.2 SSD and running Windows 10 (64-bit). For each classification problem, the run time for the classifiers (**Error! Reference source not found.**) varied between 10 milliseconds and 9.08 minutes (545 seconds). Overall, in all three scenarios the Decision Table classifier had the longest runtime with values ranging from 132.42 seconds to 545.10 seconds. The fastest models in all three scenarios were the lazy classifiers, taking between 0.01 seconds and 0.02 seconds. Runtime was not an indicator of performance since both, the fastest and the slowest models, ranked in the bottom half of the 15 models for accuracy for all three classification problems.

Conclusions

Our results suggest that bigram-based classification is a suitable method for distinguishing arthropod species names from regular English words and person names commonly found in scientific literature. To this extent, we proposed three classification problems and constructed three training/testing datasets including 3000 two-word phrases from each category and three more validation datasets with 500 two-word phrases each. We considered 15 classifiers spanning seven generic categories. The LIBLINEAR classifier outperformed all the other classifiers on all three classification tasks with respect to prediction accuracy, while Random Forest, Bayesian Logistic Regression, Multi-Layer Perceptron and LIBSVM ranked in the top 5. The least performant classifiers on the three proposed problems were AdaBoost and Lazy IBK. We also observed that the prediction of all 15 classifiers produced the highest accuracy when applied on the SCI-ENG (P1) binary problem but decreased significantly when applied to the 3-class problem (P3). Moreover, we noticed that their accuracies decreased less stringently when trying to distinguish between arthropod species names and person names (P2: SCI-PEO). We hypothesize that their decrease in accuracy when applied to P2 compared to P1 could be due to the presence of an increased number of high frequency bigrams in the SCI and PEO datasets compared to the SCI and ENG datasets. With respect to execution time, neither the fastest, nor the slowest classifiers were top performers. When compared with two dictionary-based software packages (GNRD and TaxonFinder) on a validation dataset, our bigram-based classifiers maintained their performance, which was comparable but slightly lower than that of GNRD and TaxonFinder.

Moreover, our approach can be used to improve the performance of other species name identification approaches by reducing the number of false positive identifications.

When applied on the GNRD dictionary-based method, our approach improved its overall detection accuracy by 9% to a total of 97.4%.

In the future, we plan to explore the use of 3- and 4-character gram features on the same classification problems and estimate the trade-off between the practical application and increased computing cost of such methods versus the gain in accuracy. We will also investigate how increasing the size of the training dataset will impact the overall prediction accuracy of the models. Moreover, hybridizing the n-gram based classification method with other approaches such as part-of-speech identification and capitalization rules could lead to significant improvements for arthropod species name identification in non-structured text.

Methods

Datasets

We prepared a comprehensive dataset for training the classifiers, which include the following types of information (classes): (i) **SCI:** 3000 two-word phrases representing arthropod species names, (ii) **ENG:** 3000 two-word phrases including two consecutive English words commonly used in scientific literature published in English, and (iii) **PEO**: 3000 two-word phrases representing first and last person names. We selected a well-balanced and sufficiently large number of class instances for the training of basic machine learning models to be able to use them for predictive purposes, while at the same time being able to properly use prediction accuracy for their evaluation. To validate our results, we prepared one more dataset corresponding to the same types of information, each including 500 two-word phrases for each class and we affixed the prefix “V_” to each class to help the reader distinguish between validation and testing/training results (**V_SCI**, **V_ENG** and **V_PEO**). The data from the validation

dataset was not used in the classifiers' training and testing. The arthropod species names were collected from BugGuide - a community site for entomologists who share information and photos of arthropod species [16]. Arthropod species names were scraped from BugGuide and curated using a Python script, resulting in a collection of 11,720 unique two-word phrases, from which we randomly selected 3000. Sets of two consecutive English words were gathered by pre-processing sentences from five research papers and each word was checked against an English dictionary to ensure its correctness. The person names were created from combinations of first and last names obtained from GitHub repositories [27, 28].

The dataset instances were processed and the frequency of all 676 two-letter combinations (bigrams) corresponding to the standard 26 letter English alphabet were calculated for each dataset entry. Therefore, each two-word phrase is represented by a row of 676 bigram frequencies corresponding to all entries in the three categories.

Classification problems

The goal of this work was to identify ML algorithms capable to distinguish arthropod species names in a scientific publication. The two major challenges specific to this task were to distinguish between arthropod species names and regular English words or person names, respectively, which represent major confounders in a text corpus.

To address these challenges, we defined three classification problems, two of which are two-class problems and one is a more generic three-class problem.

The first problem (**P1**) is defined such that, given a set of two-word species names of arthropod species (SCI) and groups of two consecutive English words commonly encountered in the English literature (ENG), classify them into the corresponding categories. The second problem (**P2**) considers a set of two-word arthropod species names (SCI) and two-word phrases representing first and last person names (PEO) and

focuses on classifying them into the 2 categories. The third problem (**P3**) focuses on distinguishing between all three classes: species names, English words, and person names.

The datasets used for P1 and P2 contain 6000 instances each, with 3000 instances in each class, while the dataset for P3 contains 9000 instances, with 3000 instances in each class. In each dataset, the instances are represented by bigrams in the forward direction.

Classification methods

Machine learning models were trained and tested using Weka [29]. Ten-fold cross validation [30] was used to train and test each model. The following Weka classifiers were used in this study: tree-based (Random Forest, J48), Bayesian (Bayesian Logistic Regression, Naïve Bayes, Bayes Net, Complement Naïve Bayes, Naïve Bayes Multinomial, Naïve Bayes Updateable), ANN (MLP Classifier), kernel-based (LIBSVM using a radial basis function kernel, LIBLINEAR), lazy (Lazy K*, Lazy IBK), rule-based (Decision Table) and boosting (AdaBoost). The default settings for each classifier were used. All dataset files were saved in CSV format and converted to the Weka ARFF-format.

Tree-based classifiers

Random Forest [5] is an increasingly popular ensemble-based classification method that uses a group of decision trees and bootstrap aggregations to make predictions calculated as the average prediction of each individual tree.

$$A_{k,pred} = \frac{1}{T}\sum_{t=1}^{T} A_{k,t,pred} \quad (1)$$

The data is sampled with replacement multiple times and individual decision trees are trained on each sample. The final prediction is the average of the individual tree predictions shown in Equation 1. Random Forest is a robust classification approach

capable to handle missing data. Nevertheless, Random Forest can be computationally expensive for large problem sizes, due to generation of a large number of decision trees.

The *J48* classification method is an implementation of the ID3 algorithm, which in turn represents an improvement of the C4.5 algorithm developed by Ross Quinlan [31]. The algorithm is used to create decision trees by iterating over the features of a dataset. In each iteration the algorithm selects a new feature and calculates its entropy (H) or information gain (IG). It then selects the attribute with the smallest H or IG and splits the set of data instances based on the selected attribute and organizes them as a tree. The algorithm continues to split each data subset using the same principles until all instances are labelled. A new instance is then classified by traversing the tree based on the values of its features until a leaf node is reached. The class of the leaf node becomes the class of the instance.

Kernel-based classifiers

LIBLINEAR is an open source library for large-scale linear classification that supports two popular binary linear classifiers: Logistic Regression (LR) [32] and linear Support Vector Machines (SVM) [33]. For a given set of instance-label pairs (x_i,y_i), where i=1..n and label y_i={-1,1}, LR and SVM solve an unconstrained optimization problem with three loss functions (f_1, f_2 and f_3) as described in Equation (2):

$$\min_w \frac{1}{2} w^T w + C * \sum_{i=1}^{n} f_k(w; x_i, y_i) \quad (2)$$

where C is a positive penalty parameter, k=1..3, $f_1 = \max(1 - y_i w^T x_i, 0)$, $f_2 = \max(1 - y_i w^T x_i, 0)^2$ and $f_3 = \log(1 + e^{y_i w^T x_i})$. The loss functions f_1 and f_2 are used by SVM while f_3 is used by LR.

LIBSVM [6] is based on kernel Support Vector Machines – widely used supervised machine learning approaches successfully applied to solve both classification and regression problems. SVMs rely on the use of dimensions separability and build

multidimensional hyperplanes to group and separate similarly labelled instances. SVMs can operate on both categorical and numerical variables and use kernel functions to map data instances on a new space defined by support vectors composed of instance coordinate combinations. SVMs are particularly effective when applied to non-linearly separable problems using multidimensional data. Nevertheless, SVMs require more computing time than similar methods when presented with very large datasets and tend to produce poor results for noisy data and overlapping instance labels. The difference between LIBSVM and LIBLINEAR consists in the use of kernel transforms in LIBSVM while they are not used by LIBLINEAR, which makes it faster in certain classification scenarios [34]. The LIBSVM algorithm in this paper used a radial-basis function kernel.

Artificial neural network (ANN) classifiers

The *Multi-Layer Perceptron (MLP)* classifier is one of the most popular types of neural networks consisting of a feedforward architecture with three or more layers of artificial neurons (nodes): an input layer, an output layer and one or more hidden layers. Each node sums up the weighted inputs forming an activation potential, which is further processed by an activation function (e.g. sigmoid). The input weights represent connection strengths among nodes, and they get continuously updated during the training phase. The activation function acts as a threshold allowing only strong signals with sufficiently high activation potentials to produce an output. At each iteration, an input vector x is processed and the MLP outputs the vector $y(x, w)$, where w is a vector of adapted weights. The error between calculated and expected outputs is estimated and minimized iteratively using a backpropagation approach [35] that adapts the weight values during each iteration until an acceptable error level is reached.

Bayesian classifiers

Bayesian classifiers are probabilistic models that use Bayes Theorem [36] to relate inputs with outputs. The Bayes Theorem is based on a formula for conditional probabilities (Equation 3).

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \quad (3)$$

The learning modules of Bayesian classifiers are tasked with constructing a probabilistic model including all data features and uses the model to predict the class of new data instances. The Bayesian classifiers used in this study are: Naïve Bayes [37], Bayesian Logistic Regression [3], Bayesian Networks [38], Complement Naïve Bayes [39], Naïve Bayes Multinomial [40], Naïve Bayes Updateable [37].

Lazy classifiers

Lazy classifiers operate by delaying generalization beyond the training data until a new instance is encountered [41]. When a new instance is presented to the classifier, a set of similar instances is retrieved from the training set and the new instance is labelled based on its similarity with the extracted instances. Lazy classifiers are able to solve several problems concurrently because target functions are approximated locally within the system. The disadvantage is the space required to store training data when large training sets are involved. Although the training of the system can be fast, the evaluation phase is usually slower. An example of a lazy classification method is the *K* algorithm* [42], which applies cluster analysis and the final classification of a new instance is based on the cluster that has the nearest mean. The *IBK algorithm* [43] is another example of lazy learner, which relies on a k-nearest neighbour classifier where the final classification is based on the majority vote of the k nearest neighbours.

Rule-based classifiers

Decision Table is a rule-based classification algorithm that uses different sets of conditions to determine a final classification of a data instance [22]. Relying on a Decision Table Majority (DTM) representation that encompasses a schema (set of features) and a body (labelled instances), a Decision Table provides a constant average classification time making it a good candidate for real-time applications. Moreover. it is easy for humans to understand and can be used as an initialization mechanism for selecting feature subsets for more complex classifiers. On the downside, decision tables have a higher than average results variability and are sensitive to noisy data.

Boosting classifiers

AdaBoost is a boosting type of algorithm introduced in 1995 by Freund and Schapire [44]. Initially, AdaBoost assigns equal weights to the data features and iteratively applies a basic classifier on the data. In each iteration, the algorithm augments the values of the weights for incorrectly classified instances, therefore forcing the base classifier to focus on the hard-to-classify instances of the training set and boost its performance.

Experimental setup and performance metrics

All 15 classifiers were evaluated with Weka [29] using a 10-fold cross validation approach [30]. All tests were performed using the default settings for each classifier. Since classes are balanced and include 3,000 items each for training and 500 each for validation, we report the accuracy of correctly predicted instances (%) and the execution time (seconds). N-gram extraction was achieved using the Python 3 *re* library.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable

Availability of data and materials
The datasets generated and/or analysed during the current study are available in a data repository hosted on University of Guelph servers at the following URL: http://animalbiosciences.uoguelph.ca/~dtulpan/papers/specrec2020. In addition, the data has also been submitted as a ZIP Additional file 1 archive to be hosted on BMC servers.

Competing interests
The authors declare that they have no competing interests.

Funding
This study was enabled by support from the Canada First Research Excellence Fund to the "Food from Thought" research program at University of Guelph.

Authors' contributions
JR implemented code in Python, performed the Weka analyses and interpreted the data. DS and DT designed the study and assisted with data analyses and interpretation. All authors contributed to writing the manuscript. All authors read and approved the final manuscript.

Acknowledgements
We would like to acknowledge the Canada First Research Excellence Fund and the "Food from Thought" research program at University of Guelph for providing funding for this prject. We would also like to acknowledge Dr. Luiza Antonie from the School

of Computer Science at University of Guelph for valuable discussions, feedback and comments that helped us improve the manuscript.

References

1. Jauhiainen T, Lui M, Zampieri M, Baldwin T, Lindén K. Automatic Language Identification in Texts: A Survey. J Artif Intell Res. 2018;65:1–103.

2. Fan R-E, Chang K-W, Hsieh C-J, Wang X-R, Lin C-J. LIBLINEAR: A Library for Large Linear Classification. J Mach Learn Res. 2008;9:1871–4.

3. Genkin A, Lewis DD, Madigan D. Large-scale bayesian logistic regression for text categorization. Technometrics. 2007;49:291–304.

4. Suykens JAK, Vandewalle J. Training multilayer perceptron classifiers based on a modified support vector method. IEEE Trans Neural Networks. 1999;10:907–11.

5. Breiman L, Leo. Random Forests. Mach Learn. 2001;45:5–32.

6. Chang CC, Lin CJ. LIBSVM: A Library for support vector machines. ACM Trans Intell Syst Technol. 2011;2.

7. Global Names Recognition and Discovery. https://gnrd.globalnames.org/. Accessed 12 Jan 2020.

8. Leary P. taxonfinder.org | Find scientific names in text. 2014. http://taxonfinder.org/. Accessed 10 Apr 2020.

9. Ware M, Mabe M. The STM Report: An overview of scientific and scholarly journal publishing. 2015. www.markwareconsulting.com. Accessed 16 Jan 2020.

10. Akella LM, Norton CN, Miller H. NetiNeti: discovery of scientific names from text using machine learning methods. BMC Bioinformatics. 2012;13:211.

11. Sang EFTK, De Meulder F. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In: Proceedings of the Seventh

Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4. Edmonton, Canada: Association for Computational Linguistics; 2003. p. 142–147.

12. Roskov Y., Ower G., Orrell T., Nicolson D., Bailly N., Kirk P.M., Bourgoin T., DeWalt R.E., Decock W., Nieukerken E. van, Zarucchi J., Penev L. eds. Catalogue of Life - 2019 Annual Checklist : Search all names. 2019. http://www.catalogueoflife.org/annual-checklist/2019/. Accessed 27 Apr 2020.

13. Common Names of Insects Database | Page 2 | Entomological Society of America. https://www.entsoc.org/common-names?title=&field_scientific_name_value=&tid=&tid_1=&tid_2=&tid_3=&tid_4=&page=1. Accessed 27 Apr 2020.

14. Wang X, Matthews M. Distinguishing the species of biomedical named entities for term identification. In: BMC Bioinformatics. BioMed Central; 2008. p. S6.

15. Koning D, Sarkar IN, Moritz T. TaxonGrab: Extracting Taxonomic Names From Text. Biodivers Informatics. 2005;2:79–82.

16. LINNAEUS. http://linnaeus.sourceforge.net/. Accessed 12 Jan 2020.

17. The OrganismTagger System | semanticsoftware.info. https://www.semanticsoftware.info/organism-tagger. Accessed 12 Jan 2020.

18. Sharma V, Restrepo MI, Sarkar IN. Solr-Plant: Efficient extraction of plant names from text. BMC Bioinformatics. 2019;20:263.

19. Smith TF, Waterman MS. Identification of common molecular subsequences. J Mol Biol. 1981;147:195–7.

20. Whatizit. http://www.ebi.ac.uk/webservices/whatizit/info.jsf. Accessed 12 Jan 2020.

21. Argo. http://argo.nactem.ac.uk/test/. Accessed 12 Jan 2020.

22. Kohavi R. The power of decision tables. In: Lecture Notes in Computer Science

(including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer Verlag; 1995. p. 174–89.

23. Keselj V, Keselj V, Peng F, Cercone N, Thomas C. N-gram-based author profiles for authorship attribution. In: Proceedings of the conference pacific association for computational linguistics (PACLING). 2003. p. 255–264.

24. Muhammad R, Nawab A, Stevenson M, Clough P. Detecting Text Reuse with Modified and Weighted N-grams. In: Proceedings of the Sixth International Workshop on Semantic Evaluation. Montreal: Association for Computational Linguistics; 2012. p. 54–8.

25. TaxonGrab. https://sourceforge.net/projects/taxongrab/. Accessed 12 Jan 2020.

26. SPECIES - Organism Name Identification in the Scientific Literature. https://species.jensenlab.org/. Accessed 12 Jan 2020.

27. Tarr D. dominictarr/random-name · GitHub. https://github.com/dominictarr/random-name/blob/master/first-names.txt. Accessed 14 Jan 2020.

28. American Registry for Internet Numbers E. arincli/last-names.txt · GitHub. https://github.com/arineng/arincli/blob/master/lib/last-names.txt. Accessed 14 Jan 2020.

29. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The WEKA data mining software. ACM SIGKDD Explor Newsl. 2009;11:10.

30. Stone M. Cross-Validatory Choice and Assessment of Statistical Predictions. 1974.

31. Salzberg SL. C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993. Mach Learn. 1994;16:235–40.

32. Hastie T, Tibshirani R, Friedman J. Linear Methods for Regression. In: The

Elements of Statistical Learning. 2009. p. 43–99.

33. Ben-Hur A, Horn D, Siegelmann HT, Vapnik V. Support Vector Clustering. J Mach Learn Res. 2001;2:125–37.

34. Houvardas J, Stamatatos E. N-gram feature selection for authorship identification. In: Artificial Intelligence: Methodology, Systems, and Applications. Springer Verlag; 2006. p. 77–86.

35. Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature. 1986;323:533–6.

36. Bayes T. LII. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, F. R. S. communicated by Mr. Price, in a letter to John Canton, A. M. F. R. S. Philos Trans R Soc London. 1763;53:370–418.

37. John GH, Langley P. Estimating Continuous Distributions in Bayesian Classifiers. In: Philippe Besnard, Steve Hanks, editors. Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI1995). San Francisco: Morgan Kaufmann Publishers Inc.; 1995. p. 338–345.

38. Friedman N, Geiger D, Goldszmit M. Bayesian Network Classifiers. Mach Learn. 1997;29 2/3:131–63.

39. Rennie JDM, Shih L, Teevan J, Karger DR. Tackling the Poor Assumptions of Naive Bayes Text Classifiers. In: Fawcett T, Mishra N, editors. In Proceedings of the Twentieth International Conference on International Conference on Machine Learning (ICML'03). Washington: AAAI Press; 2003. p. 616--623.

40. McCallum A, McCallum A, Nigam K. A comparison of event models for Naive Bayes text classification. In: In Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics (EACL'03) - Volume 1. Association for Computational Linguistics; 1998. p. 307–14.

41. Galván IM, Valls JM, Lecomte N, Isasi P. A lazy approach for machine learning algorithms. IFIP Int Fed Inf Process. 2009;296:517–22.

42. Cleary JG, Trigg LE. K*: An Instance-based Learner Using an Entropic Distance Measure. In: Machine Learning Proceedings 1995. Elsevier; 1995. p. 108–14.

43. Aha DW, Kibler D, Albert MK. Instance-based learning algorithms. Mach Learn. 1991;6:37–66.

44. Freund Y, Schapire RE. Experiments with a New Boosting Algorithm. In: Saitta L, editor. Proceedings of the Thirteenth International Conference on International Conference on Machine Learning (ICML'96). Morgan Kaufmann Publishers Inc.; 1996. p. 148–56.

Figures

Figure 1 - Training and testing classification accuracies for all 15 classifiers applied on problems P1, P2 and P3.

The figure depicts a bar plot of prediction accuracy for all the machine learning models applied in this study on the 3 classification problems. *Note: the Bayesian Logistic Regression method could not be applied on non-binary classification problems such as P3.*

Figure 2 - Venn diagram representation of top 100 high frequency bigrams for SCI, ENG and PEO instances from the training set.

The 3 set intersection diagram depicts the overlapping and unique bigrams commonly appearing in top 100 high frequency bigrams occurring in the datasets including arthropod species names, person names and English words.

Figure 3 - Average runtimes for all 15 classifier methods applied to the 3 classification problems: P1, P2 and P3.

The figure depicts the average runtimes for the classifiers used in this study. Each classifier was executed three times on each problem and the results were averaged out and reported in the table included at the bottom of the figure.

Tables

Table 1 - Results comparison with two external tools: Global Names Recognition Discovery (GNRD) and TaxonFinder.

The table includes the number of correctly predicted instances and the prediction accuracy for our top 4 classifiers, GNRD and TaxonFinder applied on the 3 individual validation datasets and the combined dataset.

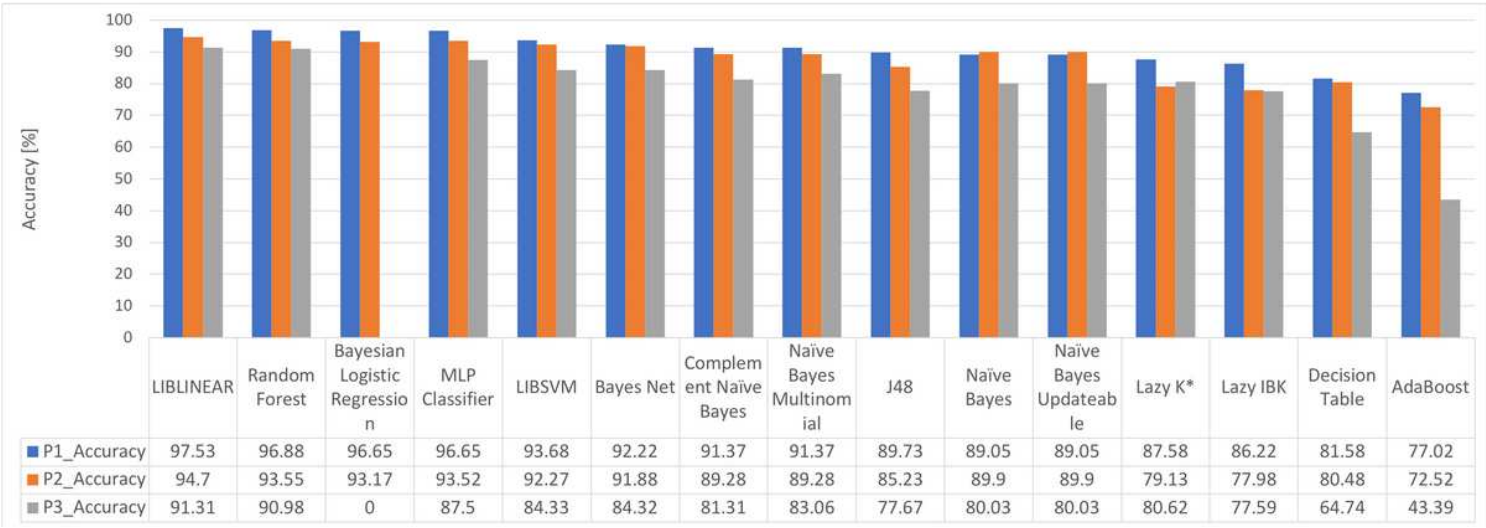
Additional files

Additional file 1 – ZIP archive including all datasets used in this work. Each file is labelled by applying the naming convention used in the manuscript.

All datasets are also made publicly available at:

http://animalbiosciences.uoguelph.ca/~dtulpan/papers/specrec2020

Figures



	LIBLINEAR	Random Forest	Bayesian Logistic Regression	MLP Classifier	LIBSVM	Bayes Net	Complement Naïve Bayes	Naïve Bayes Multinomial	J48	Naïve Bayes	Naïve Bayes Updateable	Lazy K*	Lazy IBK	Decision Table	AdaBoost
P1_Accuracy	97.53	96.88	96.65	96.65	93.68	92.22	91.37	91.37	89.73	89.05	89.05	87.58	86.22	81.58	77.02
P2_Accuracy	94.7	93.55	93.17	93.52	92.27	91.88	89.28	89.28	85.23	89.9	89.9	79.13	77.98	80.48	72.52
P3_Accuracy	91.31	90.98	0	87.5	84.33	84.32	81.31	83.06	77.67	80.03	80.03	80.62	77.59	64.74	43.39

Figure 1

Training and testing classification accuracies for all 15 classifiers applied on problems P1, P2 and P3. The figure depicts a bar plot of prediction accuracy for all the machine learning models applied in this study on the 3 classification problems. Note: the Bayesian Logistic Regression method could not be applied on non-binary classification problems such as P3.

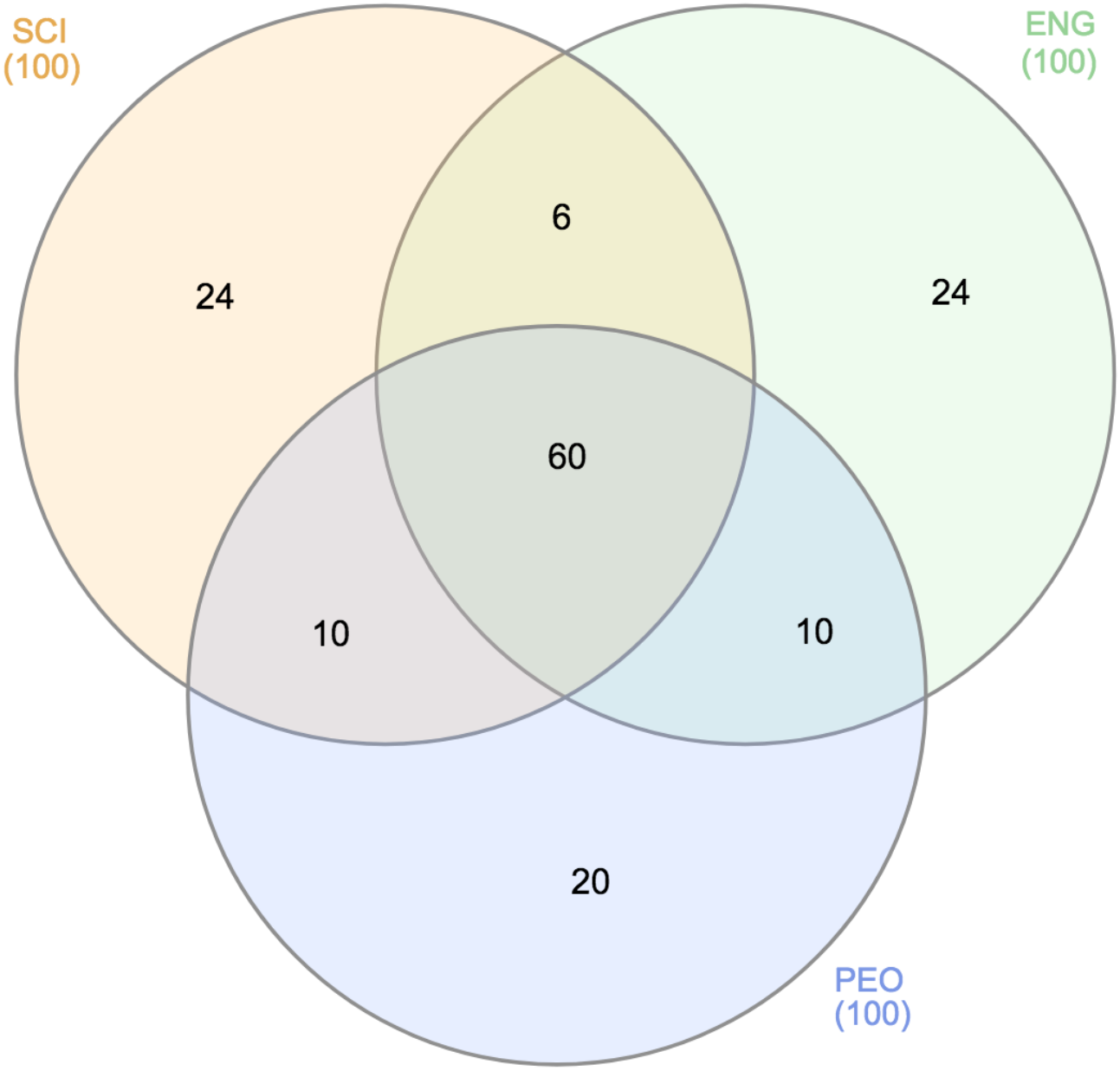
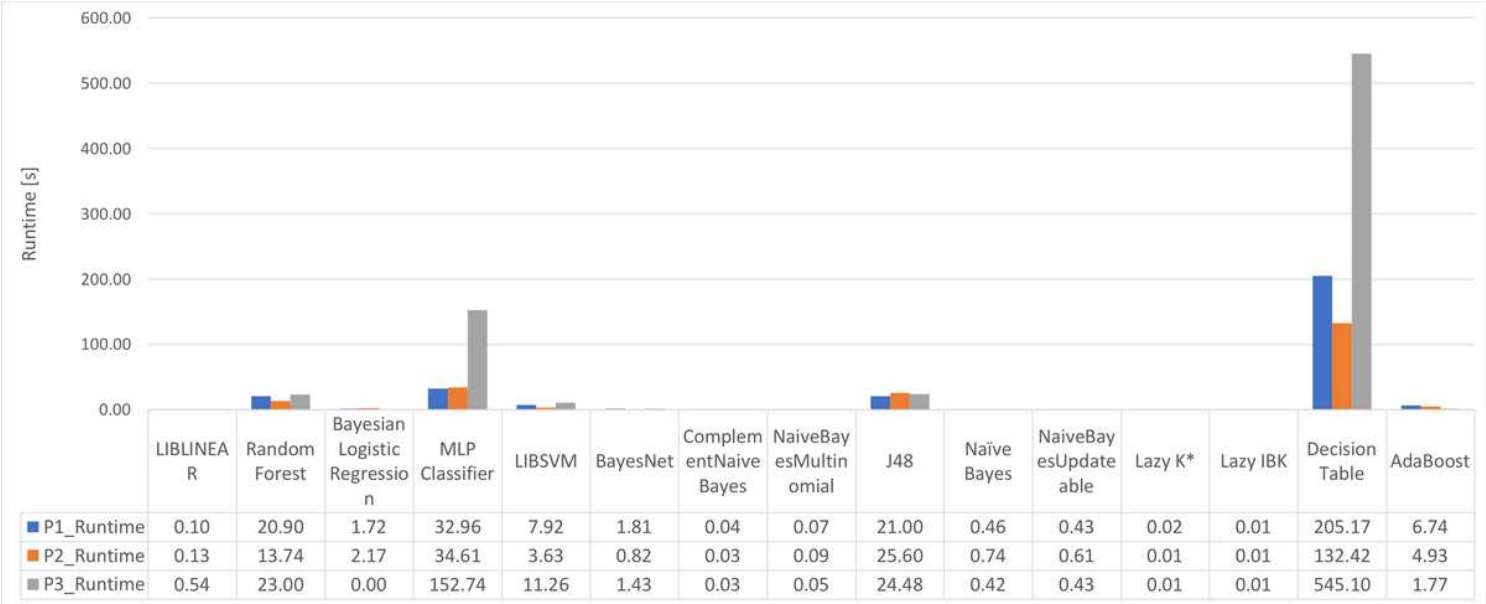


Figure 2

Venn diagram representation of top 100 high frequency bigrams for SCI, ENG and PEO instances from the training set. The 3 set intersection diagram depicts the overlapping and unique bigrams commonly appearing in top 100 high frequency bigrams occurring in the datasets including arthropod species names, person names and English words.



	LIBLINEAR	Random Forest	Bayesian Logistic Regression	MLP Classifier	LIBSVM	BayesNet	ComplementNaive Bayes	NaiveBayesMultinomial	J48	Naïve Bayes	NaiveBayesUpdateable	Lazy K*	Lazy IBK	Decision Table	AdaBoost
P1_Runtime	0.10	20.90	1.72	32.96	7.92	1.81	0.04	0.07	21.00	0.46	0.43	0.02	0.01	205.17	6.74
P2_Runtime	0.13	13.74	2.17	34.61	3.63	0.82	0.03	0.09	25.60	0.74	0.61	0.01	0.01	132.42	4.93
P3_Runtime	0.54	23.00	0.00	152.74	11.26	1.43	0.03	0.05	24.48	0.42	0.43	0.01	0.01	545.10	1.77

Figure 3

Average runtimes for all 15 classifier methods applied to the 3 classification problems: P1, P2 and P3. The figure depicts the average runtimes for the classifiers used in this study. Each classifier was executed three times on each problem and the results were averaged out and reported in the table included at the bottom of the figure.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- Additionalfile1.zip