Authorship Attribution using Filtered N-grams as Features

Manan Singh and Kavi Narayana Murthy

Abstract Authorship attribution is the problem of assigning an author to a document of unknown authorship, given a candidate set of authors and their sample documents. As a text classification task, this requires features that can capture the writing styles of authors. In this work, we compare the filtered n-grams with the traditional or unfiltered n-grams as features for authorship attribution. Filtered n-grams are the n-grams formed after filtering out from the text certain kinds of tokens. We explore the filtered n-grams formed after the removal of noun groups and verb groups. We hypothesize that the remaining text should still be enough to capture the writing style. Moreover, this removal makes possible the construction of new n-grams which would have been missed otherwise. In our experiments, we find that filtered n-grams improve the performance. In the feature ablation study, we confirm that this improvement is due to the new n-grams which are possible only after filtering.

Keywords N-grams · filtered n-grams · writing style · authorship attribution · text classification features

1 Introduction

Authorship attribution is the task of assigning an author to a document of unknown authorship, given a set of candidate authors along with their sample documents [1]. The problem has its origins in the mysteries of authorship disputes such as the controversies over Shakespearean authorship, but in the modern world, it finds its application in various business and professional needs such as plagiarism detection and text forensics [2]. From a data mining perspective, this is a text classification task which involves extracting effective features and using them with a classifier.

M. Singh (✉) · K. N. Murthy
School of Computer and Information Sciences, University of Hyderabad, Hyderabad, Telangana, India
e-mail: manan.pqrs@gmail.com

K. N. Murthy
e-mail: knmuh@yahoo.com

K. A. Reddy et al. (eds.), *Data Engineering and Communication Technology*,
Lecture Notes on Data Engineering and Communications Technologies 63,
https://doi.org/10.1007/978-981-16-0081-4_38

For textual data, a widely used method to extract features is to represent the document in terms of frequencies of N-grams. Traditionally, N-grams are constructed linearly, but Sidorov [3] discusses few alternative nonlinear ways of constructing n-grams to build more effective features for authorship attribution.

The simplest kind of nonlinear construction is skip-grams, i.e., to skip some tokens while constructing the n-grams. As an example, from the text "how are you now", some possible skip-grams are "how you", "are now", "how now", and "how you now", and "how are now". While this sounds simple, we must note that the number of all possible skip-grams grows exponentially with the text size. Also, direct linguistic interpretation is difficult [3].

A more sophisticated method is to follow the branches in the syntax tree of a sentence to construct n-grams, giving us what are known as Syntactic n-grams. Syntactic n-grams have experimentally been shown to be superior in performance to linear n-grams, and highly effective for authorship attribution [4], the only downside being the dependence on an accurate syntactic parser, and the parsing complexity associated with parsing each sentence in the corpus.

Another method is to remove or filter out tokens of certain kind before forming n-grams. Sidorov [3] has coined the term "Filtered n-grams" for this, but we found very little work in the literature on applying this idea. Content words, such as nouns and verbs, are indicative more of the subject matter and less of the style of individual authors. Therefore, removing noun phrases, verb phrases, etc., from texts before constructing n-grams will hopefully give us smaller number of more useful features for authorship attribution task. This will not require complete parsing of each sentence, and thus, will be computationally less expensive too. In this paper, we explore the effectiveness of filtered n-grams so constructed for the authorship attribution task. We demonstrate that filtered n-grams outperform the traditional or the unfiltered n-grams as features for authorship attribution.

2 Related Work

The earliest reported case of computer-assisted authorship attribution is the work on the Federalist Papers in 1960 [5]. It used Bayesian statistical analysis of frequencies of a few common words (e.g., "upon", "and", etc.) to classify the disputed papers. By the end of twentieth century, researchers had applied several common pattern recognition algorithms and identified many effective and relevant features for the task. Features such as word length, syllable length, sentence length, distribution of parts of speech, function words, type–token ratio, vocabulary distributions, hapax legomena (i.e., the number of words occurring only once), etc., were found effective [6], and pattern classification techniques such as discriminant analysis, cluster analysis, principal component analysis, and neural networks had been applied [7–10]. Holmes [6, 11] provides a survey of the work done on authorship attribution before twenty-first century. The first decade of twenty-first century saw remarkable diversity in novel techniques such as Burrows' delta measure [12], n-gram-based metrics [13],

compression-based methods [14], and graph of stopwords [15]. Several studies [1, 16, 17] survey the methods before 2010. Post-2010, with the boom of the digital era, and spurt in new form of writings such as blogs, emails, tweets, and programming code, applications have become more diverse. Researchers are exploring short texts [18, 19], newer feature-sets such as specific character n-grams [20], and neural methods such as LSTM and CNN [21, 22]. An assortment of techniques has been applied, ranging from topic models [23], and language models [24] to syntactic n-grams [3]. Many interdisciplinary approaches have also been introduced, for example, based on complex networks [25], cellular automata [26], graph theory [27], and probability theory [28]. A recent ACM computing survey on computational stylometry [29] summarizes the state of the art and mentions the list of publicly available datasets, various classes of features, and various classification techniques.

Despite the diversity in techniques for authorship attribution, the general approach is that of feature extraction followed by pattern recognition or classification. In the next subsection, we describe the features widely used in the authorship attribution literature.

2.1 Features for Authorship Attribution

The various features have been classified into lexical, syntactic, semantic and application-specific categories [16, 29]. These features are described briefly below.

Lexical Features

Lexical features are extracted by processing the text at word or character level. At the word level, features include average word length (measured in terms of characters), average sentence length (measured in terms of tokens), the distribution of word and sentence lengths, average number of syllables per word, frequencies of words, and frequencies of n-grams of words.

A writer can also be distinguished by his repertoire of words. Some measures have also been developed to capture the vocabulary richness of an author. Type–token ratio is the number of types (i.e., unique word forms) divided by the number of tokens in the author's corpus. A writer with a richer vocabulary will use more unique words, and his type–token ratio will be higher than an average writer. This ratio can differ among authors, thus serving as a discriminating feature. The number of words occurring only once or twice is called Hapax Legomena and Hapax Dislegomena, respectively, and can also be indicative of vocabulary richness.

Features solely based on character-level processing have also been investigated. Simplest ideas include frequencies of punctuations, digits, lowercase or uppercase letters, etc. Much more researched is the idea of character-level n-grams [13, 16]. Frequencies of different character or byte-level n-grams have been used as features. The advantages include minimal text preprocessing and being less prone to spelling mistakes. The choice of n is highly problem-specific and also language-dependent.

A large n leads to n-grams that may capture more lexical and contextual information, whereas n-grams with smaller n may capture only subword or syllable-like information.

Syntactic Features

At a level higher than that of tokens, information about syntactic structure of the sentence has also been used to extract features. Feature-sets are constructed from the output of taggers, chunkers, and parsers. Simple features include noun phrase counts, verb phrase counts, length of verb phrases, counts of sequence of POS tags, etc. Advanced features use the paths in the parse trees of sentences to build n-grams. These are called syntactic n-grams [4]. In syntactic n-grams, the n-grams are constructed by following the paths in the syntax tree of a parsed sentence. Firstly, a sentence is parsed and its parse tree or a dependency parse is obtained. In these trees, nodes are words. To construct an n-gram, n words in the path downwards from a node, e.g., from the root, are chosen. Depending on different starting nodes and different paths below them, multiple kinds of n-grams can be constructed. Syntactic n-grams have shown to outperform the conventional n-grams in the authorship attribution task [4]. The downside of syntactic features, of course, is their dependence on the availability of accurate parsers, and the complexity of parsing.

Semantic Features

Very less work has been reported on the usage of semantic features. Authorship attribution is more concerned with the style of the author, rather than the content or theme of the texts. Hence, very less work is focused on capturing the semantics of the text. Also, capturing higher level semantics as features is not very easy, and techniques are very prone to vagueness and subjective interpretations.

Clark and Hannon [30] have reported a classifier based on synonym-based features. They assume that the author's style is reflected in his choice among synonyms. For words with large number of possible synonyms, different authors will have different preferences for the particular synonym they use. The frequency for each word is weighted by the number of its synonyms. Thus, words with larger synonym set will be given more weight, and the weighted features help in distinguishing authors.

Application-specific Features

The features mentioned above are application or domain independent. For newer and specific forms of texts such as emails, blog posts, tweets, online forums, HTML texts, etc., application-specific features can be extracted. These can include email signatures, URL counts, use of indentation, abbreviations, etc. If the text is in HTML format, then HTML-specific features such as HTML tag distribution, font properties, and their counts can also be used [29].

Once important features have been identified and extracted, then any classification algorithm can be applied for authorship attribution.

3 Methodology

This section describes the methodology that we use in this work to compare filtered n-grams with unfiltered n-grams.

3.1 Filtered N-grams

In this work, we explore the filtered n-grams formed after removing noun groups and verb groups from the text. We use the chunker available in the pattern python library [31] to identify the NP and VP chunks (base noun/verb phrases that do not contain other noun/verb phrases [32]). Table 1 shows some illustrative examples from the literature dataset that we have used. After removing the NP and VP chunks, the

Table 1 Illustrative Examples of Filtered N-grams

Text type	Text representation	Sample N-grams formed
Without filtering	The impetus was heightened by those little events of the day which had roused her discontent with the actual conditions of her life We were lying in the darkness of the shadow of the wall of the great crater	The impetus, day which, had roused, of her life; were lying, of the, great crater
Chunked text	[The impetus]/NP [was heightened]/VP by [those little events]/NP of [the day]/NP which [had roused]/VP [her discontent]/NP with [the actual conditions]/NP of [her life]/NP We]/NP [were lying]/VP in [the darkness]/NP of [the shadow]/NP of [the wall]/NP of [the great crater]/NP	
NP Removed	was heightened by of which had roused with of were lying in of of of	**Traditional**: had roused, which had; were lying **New**: by of which, with of; lying in of, in of
VP Removed	The impetus by those little events of the day which her discontent with the actual conditions of her life We in the darkness of the shadow of the wall of the great crater	**Traditional**: The impetus, of her life; of the, great crater **New**: impetus by, which her; we in the
NP & VP removed	by of which with of in of of of	**Traditional**: - **New**: by of which, with of; in of, of of of

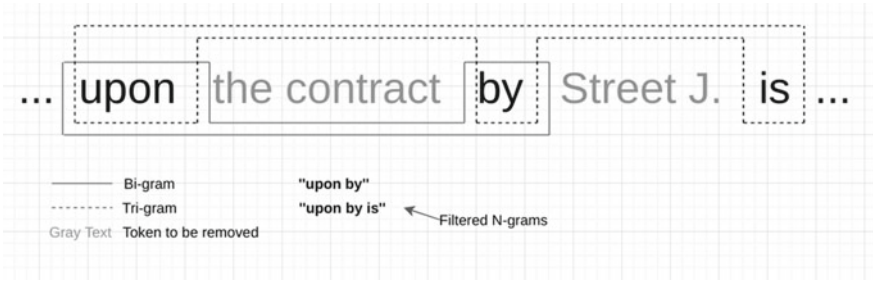

Fig. 1 An Illustration of Filtered N-grams

sentences are stripped off the theme-specific words. The skeletons that remain are hopefully sufficient to discriminate between the writing style of different authors.

Also, note that the filtered n-grams contain a combination of traditional and new n-grams. The new n-grams are those which would have been missed had we not filtered the text. Examples are "upon by", "with of", "by of which", etc. (See Fig. 1 for an illustration.) Later, in our experiments, we also present a feature ablation study in which these new n-grams are ablated from the feature set, and the performance is found to get degraded, thus signifying their importance.

3.2 Dataset

The dataset used in our experiments comprises of 50 english literature books of 5 authors - 10 books per author. The authors and their book titles are mentioned in Table 2. It is a subset of the 100 books dataset used by Rozz et al. [33]. All the books are in the public domain, and their electronic versions can be downloaded from the online repository of Project Gutenberg. To avoid the effect of different text-lengths upon feature frequencies, all the books were truncated to 20 thousand tokens which was found to be the length of the shortest book in the set.

3.3 Classification and Evaluation Procedure

We have used a fivefold cross-validation procedure throughout our experiments. The dataset is divided into five folds or groups, and five iterations of training and testing are performed with a different group being used as the test set each time. In each iteration, the training and test sets comprise of 40 and 10 documents, respectively. The split is stratified; i.e., the ratio of classes is balanced in each set. The training set is used to build the model, and then the authors of the test set are predicted. The average classification accuracy over the fivefolds is reported in the end.

The classification pipeline which we have used is as follows:

Table 2 List of authors and their books used in this work. *Source* [33]

Author	Book titles
Bernard Shaw (1856–1950)	Arms and the Man, Caesar and Cleopatra, Candida, Cashel Byron's Profession, Heartbreak House, Major Barbara, Man and Superman, Pygmalion, The Devil's Disciple, The Philanderer
Charles Dickens (1812–1870)	A Christmas Carol, A Tale of Two Cities, David Copperfield, Dombey and Son, Great Expectations, Little Dorrit, Oliver Twist, Our Mutual Friend, The Life and Adventures of Nicholas Nickleby, The Pickwick Papers
George Eliot (1819–1880)	Adam Bede, Daniel Deronda, Felix Holt The Radical, Impressions of Theophrastus Such, MiddleMarch, Romola, Scenes of Clerical Life, Silas Marner, The Essays of George Eliot, The Mill on the Floss
Herbert George Wells (1866–1946)	Ann Veronica, In the Days of the Comet, Tales of Space and Time, The Country of the Blind and Other Stories, The First Men in the Moon, The Food of the Gods and How It Came to Earth, The Invisible Man, The Island of Doctor Moreau, The Time Machine, The War of the Worlds
Oscar Wilde (1850–1900)	A House of Pomegranates, An Ideal Husband, A Woman of No Importance, Intentions, Lady Windermere's Fan, Lord Arthur Savile's Crime and Other Stories, The Duchess of Padua, The Importance of Being Earnest, The Picture of Dorian Gray, Vera

- *Document Preprocessing*: The texts are lower-cased, tokenized, and chunked into noun phrases (NP), verb phrases (VP), etc. The NP and VP chunks are removed depending upon the experiment.
- *Feature Representation*: The preprocessed documents are converted into a document-term matrix in which rows represent documents, and columns represent the filtered n-grams. Values in this matrix are the frequencies of the filtered n-grams in the documents. Each row is the feature representation of the corresponding document.
  While forming n-grams, size range from 2 to 4 is used. This means all the bi-grams, tri-grams, and 4-grams that are possible to be formed from the training set are formed, and their frequencies in each document are computed. The n-grams having document frequency less than 5 are removed. These are the n-grams that are too infrequent, and unnecessarily increase the feature space dimension, and hence, are removed. The reason for not using the n-grams outside the size range from 2 to 4 is because (i) uni-grams cannot be used to differentiate between filtered and unfiltered n-grams as there are no new uni-grams formed after filtering, and (ii) n-grams of size bigger than 4 are too infrequent. N-grams of size-range 2 to 4 seem enough to capture most of the patterns of writing style.
- *Feature Subset Selection*: We select the N most differential features. These are the features which are used more by one author but less by others. To identify these, a

numeric value denoting the differential capacity is calculated for each feature, and the *N* features with highest values are picked. To calculate the differential values, the following procedure is followed for each feature. Firstly, its frequencies per author are calculated. Then, its highest frequency among authors is multiplied with the number of authors. Lastly, from this value, the feature's actual total frequency in the documents is subtracted. The resulting value is considered as a numeric measure of the differential capacity of the feature. After calculating this value for each feature, the top *N* differential features are selected.

- *Training and Testing*: We use a classification scheme of K-nearest neighbor (K-NN) classifier with $K = 5$ and Euclidean distance. We use $K = 5$ because it is a reasonable choice for our dataset of 50 samples, and we use Euclidean distance because it is a simple yet effective metric of inter-textual distance used in authorship attribution [2]. The implementations found in the Scikit-learn [34] machine learning library are used.

Because K-NN is an instance-based classifier, no explicit training is needed, except that all samples are converted to their feature representations. During the testing-phase, to classify a test-sample, first, its feature representation is compared with that of all the training-samples using Euclidean distance, and the five least distant training-samples are identified as its five neighbors. The test sample is assigned the class to which the majority of its neighbors belong. All the samples in the test set are assigned authors, and these predictions are compared with actual authors to calculate the accuracy.

4 Experiments and Results

4.1 Comparing the Performance of Filtered and Unfiltered n-grams

Using the literature dataset, the fivefold cross-validation procedure, and the classification mechanism described earlier, we compared the performance of filtered and unfiltered n-grams. The three filtering criteria, viz. removal of noun groups, removal of verb groups, and removal of both, were tried. In all the four cases (including the one in which no filtering was done), experiments were carried out with different number of top differential features—starting from 100, doubling till it remains below another power of 10 (e.g., 1000), and then repeating so on. Because the feature set size was around 8000 when both NP and VP chunks were removed, we did not try larger feature sets except a final case in which we used all the features, i.e., without any feature selection. The average classification accuracies over five folds are reported in Table 3.

When no filtering was performed, the best accuracy was 70% with around 800 features. Adding more features did not improve but degraded it. Removal of only

Table 3 Classification accuracies with different number of differential features

	100	200	400	800	1000	2000	4000	8000	All features
Without Filtering	66	68	66	70	70	68	68	68	68
NP removed	70	72	70	74	74	74	74	76	76
VP removed	58	64	60	64	60	60	62	62	62
NP & VP removed	74	76	78	76	76	76	80	80	80

VP chunks did not prove useful with its best accuracy as 64%—much below the unfiltered case. Removal of only NP chunks and removal of both NP and VP chunks showed significant improvements in accuracies, upto 76% in the former, and upto 80% in the latter. There was also an overall increase in accuracy on increasing the number of differential features.

The finding, that when noun groups and verb groups were removed, the performance increased upto 10%, supported our hypothesis of filtered n-grams performing better than the unfiltered ones.

4.2 Feature Ablation Study

To further analyze the increase in performance achieved with filtered n-grams, we performed a feature ablation study. As described earlier too, the filtered n-grams contain a combination of traditional and new n-grams. The new n-grams include the ones only possible to be formed after filtering, such as “in of”, “of of”, “with of”, “of who”, “by of which”, etc. Few illustrative examples were given earlier in Table 1. We hypothesize that the improvement in performance is due to these new n-grams. In our ablation experiments, we removed the new n-grams from the feature-set, and then re-run the experiments. Significant drops in performance were observed.

To distinguish between the traditional and the new n-grams in the feature-set, we maintained beforehand a list of all the n-grams of size from 2 to 4 possible to be formed from the corpus without filtering, and for each feature in the feature-set, checked whether it was present in that list or not.

The feature ablation results corresponding to the best accuracies achieved in Table 3 are shown in Table 4. The average values over the 5 iterations of 5FCV are shown. The accuracy improvements of 76% and 80% upon filtering NP and VP chunks can be observed to drop down to 70% which was also the best accuracy achievable by unfiltered n-grams. This further validates our hypothesis that the 6% and 10% increase in accuracy was due to the new n-grams formed due to filtering. Also, these new n-grams make up a significant portion of the entire feature-set. When only noun groups are removed, around one-third, and when both noun groups and verb groups are removed, around two-third are the new n-grams.

Table 4 Feature Ablation Results (When the new n-grams formed after filtering are removed from the feature set)

Filtering Type	Total no. of features used	No. of traditional n-grams	No. of new n-grams	Percentage of new n-grams (%)	Original accuracy (%)	Accuracy after removing the new n-grams (%)	Decrease in Accuracy (%)
NP removed	8000	5444	2556	31.95	76	70	6
NP and VP removed	8000	2881	5119	63.99	80	70	10

5 Conclusion and Future Scope

In this work, we compared the performance of filtered n-grams with unfiltered n-grams using an English literature dataset. The best performance achievable by unfiltered n-grams was 70%, whereas, using the filtered n-grams, the accuracy improved to 76% when only noun groups were removed from the text, and to 80% when both noun groups and verb groups were removed. This removal of theme-specific words from the text leaves only the skeleton of sentences, and these are enough to capture the writing styles of authors. The n-grams constructed from the remaining text contain some new n-grams which would not have been possible without filtering. These new n-grams help improve the authorship attribution. In the feature ablation study, we confirmed this when we found that the 6% and 10% increase in performance was due to these new n-grams. Thus, in this work, we found filtered n-grams can be effective features for authorship attribution.

Future work can involve exploring other possible filtering criteria, exploring the idea of automatic discovery of optimal filtering criteria, and using filtered n-grams for text classification tasks other than authorship attribution.

References

1. Juola P (2008) Authorship attribution. Foundations and Trends in Information Retrieval. 1(3):233–334
2. Oakes MP (2014) Literary detective work on the computer. John Benjamins Publishing Company, Amsterdam
3. Sidorov G (2019) Syntactic n-grams in computational linguistics. Springer, Cham
4. Sidorov G, Velasquez F, Stamatatos E, Gelbukh A, Chanona-Hernández L (2014) Syntactic n-grams as machine learning features for natural language processing. Expert Systems with Applications 41(3):853–860 (2014)
5. Mosteller F, Wallace DL (1963) Inference in an authorship problem. J American Stat Assoc 58(302):275–309
6. Holmes DI (1994) Authorship attribution. Comput Humanit 28(2):87–106

7. Holmes DI (1992) A stylometric analysis of mormon scripture and related texts. J Royal Stati Soc Series A (Statistics in Society).155(1), 91–120 (1992)
8. Holmes DI, Forsyth RS (1995) The Federalist Revisited: New Directions in Authorship Attribution. Literary Linguistic Comput 10(2):111–127
9. Matthews RA, Merriam TV (1993) Neural computation in stylometry I: an application to the works of Shakespeare and Fletcher. Literary and Linguistic Computing 8(4), 203–209 (01 1993). https://doi.org/10.1093/llc/8.4.203
10. Merriam TV, Matthews RA (1994) Neural computation in stylometry II: an application to the works of Shakespeare and Marlowe. Literary and Linguistic Computing 9(1), 1–6 (01 1994). https://doi.org/10.1093/llc/9.1.1
11. Holmes DI (1998) The evolution of stylometry in humanities scholarship. Literary Linguistic Comput 13(3), 111–117.https://doi.org/10.1093/llc/13.3.111
12. Burrows J (2002) Delta: a measure of stylistic difference and a guide to likely authorship. Literary Linguistic Comput 17(3):267–287 (09 2002).https://doi.org/10.1093/llc/17.3.267
13. Kešelj V, Peng F, Cercone N, Thomas C (2003) N-gram-based author profiles for authorship attribution. In: Proceedings of the Conference Pacific Association for Computational Linguistics, PACLING. 3:255–264
14. Benedetto D, Caglioti E, Loreto V (2002) Language trees and zipping. Phys Rev Lett 88(4):048702
15. Arun R, Suresh V, Madhavan CV (2009) Stopword graphs and authorship attribution in text corpora. In: 2009 IEEE international conference on semantic computing. pp 192–196. IEEE Computer Society, Los Alamitos, CA, USA
16. Stamatatos E (2009) A survey of modern authorship attribution methods. J Am Soc Inform Sci Technol 60(3):538–556
17. Koppel M, Schler J, Argamon S (2009) Computational methods in authorship attribution. J Am Soc Inform Sci Technol 60(1):9–26
18. Shrestha P, Sierra S, González F, Montes M, Rosso P, Solorio T (2017) Convolutional neural networks for authorship attribution of short texts. In: Proceedings of the 15th conference of the European chapter of the association for computational linguistics: vol 2, Short Papers. pp 669–674. Association for Computational Linguistics, Valencia, Spain
19. Altakrori MH, Iqbal F, Fung BCM, Ding SHH, Tubaishat A (2018) Arabic authorship attribution: An extensive study on twitter posts. ACM Trans Asian Low-Resour Lang Inf Process 18(1)
20. Sapkota U, Bethard S, Montes M, Solorio T (2015) Not all character n-grams are created equal: A study in authorship attribution. In: Proceedings of the 2015 Cconference of the North American chapter of the association for computational linguistics: human language technologies. pp 93–102. Association for Computational Linguistics, Denver, Colorado
21. Hitschler J, van den Berg E, Rehbein I (2017) Authorship attribution with convolutional neural networks and POS-eliding. In: Proceedings of the Workshop on Stylistic Variation. pp. 53–58. Association for Computational Linguistics, Copenhagen, Denmark
22. Alsulami B, Dauber E, Harang R, Mancoridis S, Greenstadt R (2017) Source code authorship attribution using long short-term memory based networks. In: Foley SN, Gollmann D, Snekkenes E (eds) Computer Security—ESORICS 2017. Springer International Publishing, Cham, pp 65–82
23. Seroussi Y, Zukerman I, Bohnert F (2014) Authorship attribution with topic models. Computational Linguistics 40(2):269–310
24. Fourkioti O, Symeonidis S, Arampatzis A (2019) Language models and fusion for authorship attribution. Inf Process Manage 56(6):102061
25. Amancio DR (2015) A complex network approach to stylometry. PLOS ONE 10(8):1–21
26. Machicao J, Corra EA, Miranda GHB, Amancio DR, Bruno OM (2018) Authorship attribution based on life-like network automata. PLOS ONE 13(3):1–21
27. Shalymov D, Granichin O, Klebanov L, Volkovich Z (2016) Literary writing style recognition via a minimal spanning tree-based approach. Expert Syst Appl 61:145–153

28. Zheng L, Zheng H (2019) Authorship attribution via coupon-collector-type indices. J Quantitative Linguistics 1–13 (2019). https://doi.org/10.1080/09296174.2019.1577939
29. Neal T, Sundararajan K, Fatima A, Yan Y, Xiang Y, Woodard D (2017) Surveying stylometry techniques and applications. ACM Comput. Surv. 50(6)
30. Clark JH, Hannon CJ (2007) A classifier system for author recognition using synonym-based features. In: Gelbukh A, Kuri ÁF (eds) MICAI 2007: Advances in Artificial Intelligence. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp 839–849
31. Smedt TD, Daelemans W (2012) Pattern for python. J Mach Learn Res 13(66):2063–2067
32. Bird S, Klein E, Loper E (2009) Natural language processing with Python: analyzing text with the natural language toolkit. O'Reilly Media, Inc.
33. Al Y, Menezes R (2018) Author attribution using network motifs. In: Cornelius S, Coronges K, Gonçalves B, Sinatra R, Vespignani A (eds) Complex Networks IX. Springer International Publishing, Cham, pp 199–207
34. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M (2011) Édouard Duchesnay: Scikit-learn: Machine learning in python. J Machine Learn Res 12(85):2825–2830