Cross-Domain Authorship Attribution with Federales

Notebook for PAN at CLEF 2019

Hans van Halteren

Centre for Language Studies, Radboud University Nijmegen
P.O. Box 9103, NL-6500HD Nijmegen, The Netherlands
hvh@let.ru.nl

Abstract This paper describes the system with which I participated in the Cross-Domain Authorship Attribution task at PAN2019, which entailed attributing fan-fiction texts from a specific "fandom" (texts in a given fictional world) on the basis of training material from other fandoms. As underlying system I used a combination of 5 or 3 (depending on language) feature sets and two author verification methods. In reaction to the genre differences, I added a second round of attribution, this time in-genre, for those authors for whom enough target fandom texts could be identified in the first round. On the training dataset, attribution quality was well over the baseline scores, but with ample space for further improvement. On the test dataset, gain over the baseline scores was lower, indicating some kind of overtraining. In general, performance showed that more work is needed on (automated) hyperparameter selection.

1 Introduction

The Cross-Domain Authorship Attribution task in PAN2019 is the attribution of so-called "fan-fiction" texts in four languages (English, French, Italian and Spanish; see [10] for a full description of the task).[1] Fans of popular fiction books or series, e.g. Sherlock Holmes or Harry Potter, write their own stories in the corresponding fictional worlds. The stories in such a world are dubbed a "fandom". The training material for the PAN2019 task was divided into 20 "problems", five for each language, in which a number of texts from a specific fandom are to be attributed to nine known authors or "none of these", in all cases on the basis of seven known texts from each author, stemming from other fandoms. The problems were to be solved in isolation, i.e. it was not allowed to use information from one problem in solving the others. This severely limited the compilation of a background corpus. However, three baseline systems were

[1] In this paper, I will focus on my own approach. I refer the reader to the overview paper and the other papers on the PAN2019 Cross-Domain Authorship Attribution task for related work. Not only will this prevent overlap between the papers, but most of the other papers, and hence information on the current state of the art, are not available at the time of writing of this paper.

also provided, one of which (the Impostor baseline[11]) was supported with a set of 5,000 texts per language, from various (unknown) fandoms and authors, which could serve as a background corpus. The unknown texts for each problem also contained texts from other authors than the nine given ones. The number of such texts was 20%, 40%, 60%, 80% and 100% of the number of texts by the target authors. Although it seemed likely that the test material would have a similar composition, this was not specified.[2]

Authorial style might or might not have been adapted to the style of the emulated book(s), but it is likely that the language use was influenced by topic and general genre. In authorship attribution, it is well-known that the difficulty of the task is much higher if unknown texts are from a different genre than the known texts. In the current task, we seemed to be somewhere in between, as all texts were fiction. However, the fact that all unknown texts were from the same fandom might have pushed the language use in a similar direction for all authors, adding an additional confounding factor.

My approach for this task[3] built on earlier work on authorship and other text classification tasks, which used to be published under the name Linguistic Profiling, which because of ambiguity of that term has now been replaced by the working title "Feature Deviation Rating Learning System" (henceforth *Federales*). Although the full name implies a specific learning technique, the acronym already indicates my predilection for combination approaches. Which form of combination was used in this task is described below (Section 3).

On top of the basic verification system, I wanted to add some way to deal with the genre differences. A first idea was to model each fandom and learn how a specific author behaved in relation to these fandom models. If relative behaviour were consistent between fandoms, correction factors could be applied to target fandom measurements, hopefully leading to a better attribution. However, there was insufficient material outside the known texts to follow this strategy.[4] A second option was to ignore all features that appeared to be affected by genre (i.e. fandom). In a small pilot study on English, it turned out that so many features were removed that attribution quality went down rather than up.[5] In the end, I did not attempt to apply corrections to the feature measurements. Instead, I took the authors for which a sufficient number of texts was identified with (relatively) high confidence, and used those texts as known texts for a verification model within the target fandom. For those authors, the in-genre models were then used, whereas the other authors had to be attributed with the cross-genre model, albeit with

[2] And therefore I did not use this expectation in building my system.

[3] I also participated in the Author Profiling task. The differences in handling the two tasks were such that I preferred to describe the other task in a separate paper[7]. The main difference is that here I used verification, and for profiling comparison. Despite that, there will obviously be some overlap between the papers.

[4] As it is very unlikely that the same authors will be active in the same combinations of fandoms, the known texts were usually from various fandoms. The only other source was the set of 5,000 sets by unknown authors per language, but this as well was too fragmented for proper modeling. I hope the organisers will make more data available after the workshop, so that I can investigate author behaviour over various fandoms.

[5] It must be said that this pilot was in the initial phases of the work on the task. It may well be that the approach would be viable in the circumstances provided by the final system. Again, future work is a definite option.

Stiles	NNP	Je	PP1CSN0
had	VBD	sais	VMIP1S0
been	VBN	que	CS
the	DT	le	DA0MS0
one	CD	jour	NCMS000
to	TO	venu	VMP00SM
say	VB	les_autres	PI0CP00
they	PRP	maisons	NCFP000
needed	VBD	seront	VMIF3P0
to	TO	derrière	SP
get	VB	toi	PP2CS00
out	IN	,	Fc
of	IN	professeurs	NCMP000
town	NN	et	CC
first	RB	élèves	NCCP000
.	.	.	Fp

Figure 1. Example POS taggings for English and French

the additional knowledge that some competing authors could be ruled out on the basis of the in-genre verification.

An additional complication of the shared task was that the final evaluation was in the form of a blind test using TIRA[15]. My main approach on earlier projects was a careful investigation of (the known texts of) each individual problem, followed by a manual combination of the most promising components and system settings. Given that there was insufficient time to fully automate this procedure, I was forced here to choose components (and hyperparameters) for all the (probably varied) test data at once, without having access to the known texts or even knowledge of the involved genres. Moreover, in the development of the current software, I had not yet reached the state of an integrated system, which was obviously needed for TIRA. In the end, complications in building the integrated system took away much time that could have been used on better system tuning.

In the next sections, I first describe the features I used (Section 2), the verification techniques (Section 3) and how all this fitted together into a full system (Section 4). After this I continue with performance measurements (Section 5) and a short discussion to conclude the paper (Section 6).

2 Feature Extraction

For English, I used both surface and syntactic features. The simplest type was that of the character n-grams, with n from 1 to 6, which were counted directly on the raw text. For all other types, I analysed all texts with the Stanford CoreNLP system[12].

From the POS tagging, of which an example is shown on the left side of Figure 1, I extracted token n-grams, with n from 1 to 3. Each position in the n-gram was taken by the word itself, the POS tag or the word's group. The latter was looked up in a list of word groups deemed relevant for authorship attribution, which I had available only

Table 1. Feature types.

Feature type	Example	Explanation
Character n-grams (lex)	CG1_.	a period
	CG6_ed˜to˜	word ending in *ed* followed by *to* and more space
Token n-grams (lex)	WF_and	the word *and*
	T3_PGW_VBG_GrpADVgely_at	*-ing*-participle, followed by an adverb ending in *-ly* and the word *at*
Token n-grams (abs)	WFV_midfnn	a mid-IDF-range singular common noun
	T3V_PWW_DT_LIDFNN_ofIN	a determiner followed by a low-IDF-range singular common noun and the word *of* as a preposition
Syntactic n-grams (lex)	SCWV_DT_alldt	the word *all* used as a determiner
	SCFFCCLV_NP_NPDT_DT(aDT)_ NPPO_PP(ofIN)	a noun phrase with determiner *a* and a postmodifying prepositional phrase with preposition *of*
Syntactic n-grams (abs)	SCFFCCLV_S_SU_NP(HIDFNNP)_ V_VP(GrpVrepx)	a sentence in which the subject is a high-IDF-range proper noun and the main verb a reporting verb
	SRFC_VP_AVB_MD_ A_AVP_MVB_VB	a verb phrase which consists of a modal verb, an adverb phrase and a main verb

for English and which contained reporting verbs and various types of adverbs.[6] The token n-grams were generated in two forms. The lexical form (lex) included the words themselves. In the abstract form (abs) the words were replaced by an indication of their IDF value (low, middle or high) if their IDF exceeded a certain value.[7] In this way, the abstract features were supposed to reflect the language use of the author rather than the topics discussed.

The CoreNLP system also yielded a dependency analysis. However, as the dependency structure was less amenable to variation studies than a constituency structure, I first transformed the trees, aiming for a representation similar to that in the TOSCA Project[1].[8] Apart from restructuring the tree, the transformation program lexicalized the analysis by percolating the head words upwards. As an example the parse of the (English) sentence in Figure 1 is shown in Figure 2. From these transformed trees, syntactic features were derived, namely slices from the trees representing presence of constituents, dominance and linear precedence, as well as full rewrites. Again a lexical and an abstract form were generated, this time with the word (or IDF indication) concatenated with the POS tag.[9] An overview of the feature types, with examples for English, is given in Table 1.

[6] The list was created during work on author recognition on texts from the British National Corpus[2] and was still under development.

[7] IDFs were calculated on the texts in the British National Corpus.

[8] Unfortunately, the parser(s) from that project are (currently) unavailable.

[9] Unfortunately, there is no space for a more extensive description, but this will follow in future work.

```
ROOT:ROOT(<UNHEAD>) -> [ UTT NOFUpunc ]
   UTT:S(be) -> [ SU V CS ]
      SU:NP-NPRPNP00(Stiles) -> [ NPHD ]
         NPHD:NNP(Stiles) -> Stiles
      V:VP(be) -> [ AVB MVB ]
         AVB:VBD(have) -> had
         MVB:VBN(be) -> been
      CS:NP-NPRPNP01(one) -> [ NPDT NPHD NPPO ]
         NPDT:DT(the) -> the
         NPHD:CD(one) -> one
         NPPO:SBAR(say|GrpVrepd) -> [ S ]
            S:S(say|GrpVrepd) -> [ A V A ]
               A:TO(to) -> to
               V:VP(say|GrpVrepd) -> [ MVB ]
                  MVB:VB(say|GrpVrepd) -> say
               A:SBARc(need) -> [ S ]
                  S:S(need) -> [ SU V A ]
                     SU:NP-PRPNP00(they) -> [ NPHD ]
                        NPHD:PRP(they) -> they
                     V:VP(need) -> [ MVB ]
                        MVB:VBD(need) -> needed
                     A:Sx(get) -> [ A V A A ]
                        A:TO(to) -> to
                        V:VP(get) -> [ MVB ]
                           MVB:VB(get) -> get
                        A:AVP(out) -> [ AVHD AVPO ]
                           AVHD:IN(out) -> out
                           AVPO:PP(of|town) -> [ PREP PCOMP ]
                              PREP:IN(of) -> of
                              PCOMP:NP-NPRPNP00(town) -> [ NPHD ]
                                 NPHD:NN(town) -> town
                        A:AVP(first) -> [ AVHD ]
                           AVHD:RB(first) -> first
   NOFUpunc:.(.) -> .
```

Figure 2. Example syntactic analysis for English

Table 2. Number of features used for the problems in the training data, per language and feature type.

Problem	Number of unknown texts	Char n-gram	Tok n-gram (lex)	Tok n-gram (abs)	Syn n-gram (lex)	Syn n-gram (abs)
English	137-561	120-235K	119-331K	122-323K	179-426K	114-249K
French	38-430	78-214K	146-611K	130-531K		
Italian	46-196	83-146K	174-390K	155-340K		
Spanish	112-450	121-215K	274-718K	246-635K		

For all three Romance languages, no appropriate syntactic parser was available.[10] Instead, I POS-tagged the texts with FreeLing[14]. An example for French is shown on the right side of Figure 1. From this output I extracted only surface features, leading to three different feature sets. Furthermore, the token n-grams were less involved, as no word groups were present, nor IDF statistics, so that the abstract form replaced all words of open word classes.

For each problem, the system took all known and unknown texts, plus the 5,000-text background corpus, and extracted all features which occurred in at least three texts. The numbers of features for each problem and feature type (for the training data) are indicated in Table 2. The large ranges within each language were caused mostly by the greatly varying sizes of the unknown text sets, from 38 texts for problem 10 (French) to 561 for problem 1 (English).

3 Learning Techniques

Apart from a combination of feature types, I also used a combination of learning techniques.

3.1 Feature Value versus Profile Range Comparison

The Federales system built on the Linguistic Profiling system, which had been used in various studies, such as authorship recognition[4][5], language proficiency[8], source language recognition[6], and gender recognition[9]. The approach is based on the assumption that (relative) counts for each specific feature typically move within a specific range for a class of texts and that deviations from this typical behavior indicate that the deviating text does not belong to the class in question. If the frequency range for a feature is very large, the design of the scoring mechanism ensures that the system mostly ignores that feature.

[10] This was not because no parsers existed, but because there was no time to build tranformers from the parser output to the structure I wanted for the syntactic features.

For each feature, the relative counts[11] for all samples in the class are used to calculate a mean and a standard deviation.[12] The deviation of the feature count for a specific test sample is simply the z-score with respect to this mean and standard deviation, and is viewed as a penalty value. Hyperparameters enable the user to set a threshold below which deviations are not taken into account (the *smoothing threshold*), a power to apply to the z-score in order to give more or less weight to larger or smaller deviations (*deviation power*), and a *penalty ceiling* to limit the impact of extreme deviations. When comparing two classes, a further hyperparameter sets a power value for the difference between the two distributions (*difference power*), the result of which is then multiplied with the deviation value. The optimal behaviour in cases where a feature is seen in the training texts for the class but not in the test sample, or vice versa, is still under consideration. In the current task, features only seen in the training texts were counted as they are, namely with a count of 0 in the test sample; features only seen in the test sample were compared against the lowest mean and standard deviation from among the training features, which should correspond more or less to the scores for hapaxes in the training texts.

The penalties for all features are added. A set of benchmark texts is used to calculate a mean and standard deviation for the penalty totals, to allow comparison between different models. For verification, the z-score for the penalty total is an outcome by itself; for comparison between two models, the difference of the z-scores can be taken; for attribution within larger candidate sets (as in the current task), the z-scores can be compared. In all cases, a threshold can be chosen for the final decision.

Even though optimal settings for one author were often bad settings for another author, I started with the fallback strategy of using a single, basic choice for the hyperparameters, namely no smoothing threshold, no deviation or difference power (i.e. power 1), and a penalty ceiling of 10. The next step, automated tuning, did not take place anymore, due to lack of time.

3.2 Support Vector Regression

As a second learning method to assign feature vectors to classes, I used Support Vector Regression as provided by the libsvm package[3]. For each author, vectors for known texts of the author were given class 1 and vectors for texts by other authors with class -1. svm-scale was used with its default settings. Here too, a single, simple list of hyperparameters was used: ν-regression with an RBF kernel, and the defaults for cost ($c = 1$) and gamma ($\gamma = 1/number\_of\_features$); only ν was set other than the default (0.1 versus a default of 0.5). To correct for a bias towards positive or negative examples because of different numbers of texts in the two classes, the hyperparameter w was used, with weights that exactly compensated for class size. By choosing regression, I received a score rather than a decision on the class, which could then be used in further processing.

[11] I.e. the absolute count divided by the corresponding number of items, e.g. count of a token in a text divided by that of all tokens within the text, or a character n-gram count divided by the number of characters in the text.

[12] Theoretically, this is questionable, as most counts will not be distributed normally, but the system appears quite robust against this theoretical objection.

3.3 Sample Score Combination

After all ten/six individual component runs, the component scores for each were normalized by factoring in a linear model which predicted the component score on the basis of the average component scores for the models and test samples in question, plus the number of feature comparisons made during scoring. The adjusted component score was the deviation from the predicted value. Finally, all adjusted component scores were normalized to z-scores with regard to all observed adjusted component scores for a model and with regard to all observed adjusted component scores for a test sample. These normalized component scores allow for an intuitive interpretation and provide comparability between different author models. The latter goal was needed here both for selecting a common threshold for verification and for score combination of the various components. In addition to the two individual normalized component scores (wrt model and wrt sample), their sum was also calculated. Which of these three normalized scores was used, depended on the phase in the attribution (cf. Section 4).

4 Complete System

After the normalized scores (henceforth simply "scores") for each feature set and learner became available, the system needed to combine these to determine an attribution for each text. This was done in several phases.

4.1 Phase 1: Cross-Genre Attribution

In the first phase, the system examined the scores produced by the various models so far. On the basis of the sum of all individual scores,[13] a ranking of potential authors was produced. If the attribution task were a closed one, the first ranked author could now be selected. However, the test texts also included texts by other than the nine known authors. This meant that a decision had to be made whether or not to attribute to a known author at all.

For this, a number of statistics were calculated per test sample. For each potential author, these were: the mean of the reciprocal rank, the mean of the score, the mean of the distance to the score of the top-ranked author, and the highest and lowest score reached. For the sample as a whole, these were: the sum of the various best scores and distance in the individual models, plus the highest values for the five author statistics. Finally, the first principal component of a PCA of the seven whole-sample statistics was added.[14]

In the various phases, all of the whole-sample statistics, apart from the reciprocal rank, were used for comparisons against thresholds to determine whether or not to accept the top-ranked author for attribution. The thresholds were set manually per language, in such a way that the average evaluation score over all 45 authors was optimized.[15] My hope was that possibly overtraining in this threshold selection process

[13] To be exact, the normalized scores with regard to both model and sample.

[14] The rotation to calculate this value was based on the known samples and applied to the test samples.

[15] This manual tuning process too should be, but has not yet been, automated.

would be of an acceptable level, seeing that the selected thresholds were chosen for 45 different authors each in five different fandoms.[16]

4.2 Phase 2: Selecting Training Texts for In-Genre Attribution

The next step was to build an in-genre attribution process for those authors for which sufficient in-genre texts had been identified. If at least four texts were attributed to an author in phase 1, these texts were used to build in-genre models, with which the same set of suggested texts were scored.[17]

These scores were then transformed to z-scores with regard to all text scores for the author in question and then sorted from high to low. Texts were removed from the set if a) they had a z-score lower than -2 and/or b) they were in the second half of the ranking and the gap to the next higher z-score was greater than 1. The second criterion was to correct for the situation where another author (known or unknown) was being falsely accepted as well. This correction fails if the authors are too much alike, but also if more samples from the unwanted author were included than by the wanted author.[18]

The texts which were not removed were next used to build full in-genre attribution models. In addition, they were marked for final attribution to their respective authors.

4.3 Phase 3: In-Genre Attribution

On the basis of the texts selected in phase 2, the system now created an attribution model for each author for whom at least three in-genre texts were left. For the other authors, the cross-genre models were reused. With these new models, all unknown texts were processed in the same way as in phase 1, except that now only the normalized scores with regard to the sample were used (and no longer adding those for the models), and other thresholds were chosen.

The texts attributed to any of the in-genre models were marked for final attribution to the corresponding authors. All other texts were marked as NOT belonging to any of these authors and referred to phase 4, where they could be attributed to one of the other authors or remain unattributed.

4.4 Phase 4: Cross-Genre Attribution of Underrepresented Authors

For this final attribution phase, the results of phase 1 were reused, but also using the normalized scores with regard to the sample.

All authors processed in phase 3 were struck from the rankings and the top-ranking remaining author was examined. If his/her mean and/or lowest scores over the individual runs was over selected (language-dependent) thresholds, he/she was selected for final attribution of the text in question. If not, the text remained unattributed.

[16] After receiving the test results, I must conclude that this was not the case, and that author-dependent hyperparameter and threshold selection is indeed needed (cf. Section 6). Furthermore, in the circumstances, reporting on the manually selected thresholds does not seem worthwhile.

[17] For reasons of processing time, support vector regression was not included in this phase.

[18] I hoped the task organisers were not too cruel in their text selection, in which case random selection should be on my side.

Table 3. Results on the training set, compared to three baselines.

Problem	Precision	Recall	F1	SVM	Compression	Impostor
1 (en)	0.803	0.716	**0.751**	0.711	0.682	0.699
2 (en)	0.667	0.633	**0.629**	0.444	0.336	0.396
3 (en)	0.652	0.583	**0.572**	0.491	0.501	0.428
4 (en)	0.378	0.729	0.451	0.277	**0.490**	0.471
5 (en)	0.641	0.580	**0.583**	0.474	0.340	0.272
Overall (en)			**0.597**	0.479	0.470	0.453
6 (fr)	0.742	0.701	0.693	**0.702**	0.691	0.564
7 (fr)	0.519	0.570	0.504	0.499	**0.542**	0.450
8 (fr)	0.628	0.596	**0.658**	0.505	0.492	0.384
9 (fr)	0.653	0.688	0.579	0.593	**0.608**	0.265
10 (fr)	0.410	0.635	0.447	0.442	**0.501**	0.393
Overall (fr)			**0.576**	0.548	0.567	0.411
11 (it)	0.857	0.751	**0.763**	0.651	0.595	0.441
12 (it)	0.636	0.662	**0.631**	0.599	0.508	0.561
13 (it)	0.675	0.830	0.706	0.687	**0.731**	0.543
14 (it)	0.694	0.676	0.680	0.583	**0.780**	0.553
15 (it)	0.841	0.857	**0.840**	0.760	0.712	0.360
Overall (it)			**0.724**	0.656	0.665	0.492
16 (sp)	0.928	0.835	**0.875**	0.767	0.705	0.668
17 (sp)	0.909	0.818	**0.831**	0.581	0.623	0.351
18 (sp)	0.861	0.914	**0.874**	0.713	0.659	0.549
19 (sp)	0.749	0.745	**0.732**	0.559	0.403	0.312
20 (sp)	0.443	0.585	0.417	**0.515**	0.223	0.323
Overall (sp)			**0.746**	0.627	0.523	0.441
Overall			**0.656**	0.578	0.556	0.449

5 Results

The evaluation measure for the PAN2019 attribution task was the open-set macro-averaged F1 score, which was calculated over the training classes without the "unknown" class, but counting false accepts of unknown-class texts against precision[13].[19] Evaluation with the provided python script of my system's results on the training set yielded the scores listed in Table 3. Also in Table 3 are the Macro-F1 scores for three baseline systems, for the description of which I refer the reader to [10].[20]

For the training material, my system outperformed the three baselines, but with substantial variation per problem (and even more so per author, but those results are not shown here for lack of space). It is positive that the largest improvement with regard to the baselines was for English, where overall the system reached a 25% gain in Macro-F1 over the best baseline (SVM). This is the language I have mostly been

[19] As the Macro-F1 was taken as an average over all F1, and not calculated from the Macro-Precision and the Macro-Recall, the Macro-F1 could be lower than the two others.

[20] I did not actually run the baseline systems myself, but these measurements were provided by Mike Kestemont, for which I thank him.

working on so far and this is the language for which the syntactic features could be used. For the Romance languages, my system was much less evolved. For Spanish, the relative performance (19% performance gain over again SVM) was therefore rather satisfying. Apparently, the syntactic structure used for English is reflected sufficiently in the morphology here. Italian was somewhere in between, with 9% gain, this time over compression. But French was apparently a serious problem, with a gain of only 1.5% (over compression) and a best score only for problem 8.

For the test material, the system performed much worse. For English, the Macro-F1 of 0.532 was only 8% over the compression baseline (0.493). For French, my 0.554 was even lower than compression's 0.595. For Italian, the test run went better than the training run, with 0.653 and a 12.5% improvement over the best baseline (0.580, again compression). For all three languages, the best scores were much higher: English 0.665, French 0.705 and Italian 0.717, all by "muttenthaler". Spanish was an exception, with SVM as the best baseline (0.577) and "neri" the best participating system (0.679), so that my 0.652 ended up on rank 3 with 13% over SVM, still lower than the training performance.

6 Conclusion

For the PAN2019 Cross-Domain Authorship Attribution task, I built a system combining various feature types and two classification methods, and furthermore attempting to apply in-genre models when enough training texts could be identified in the cross-genre recognition. Under pressure of the shared task deadline, after losing too much time on building an integrated system that could run on TIRA, automation of the tuning procedure for hyperparameters and thresholds was no longer possible, and suboptimal values had to be used.

On the training set, the system outperformed three baselines, but with varying margins. The largest margin was for English, for which my software was the most developed and the features included ones derived from full syntactic analysis trees. On the test set, performance varied. In relation to the baselines, there was a drop for English (+25% for training to +8% for test), French (+1.5% to -7%) and Spanish (+19% to +13%), but an improvement for Italian (+9% to +12.5%). In relation to the best systems (for each language), however, we see bad scores for English (-20%) and French (-21.5%), slightly better scores for Italian (-9%), and acceptable scores for Spanish (-3%).

Given the large variation in relative scores, it would seem that success or failure with the single settings option is largely a matter of luck. The settings may work or they may not. This is clearly not an acceptable situation and automated hyperparameter tuning is a matter of utmost urgency. Once this is in place, I can also turn my attention to the relative contributions of the various feature types and components, as well as the partial in-genre attribution, which would be rather meaningless at the moment.

Obviously, careful analysis of the results is only possible with access to the test data. But I would like to go further and express the hope that the organisers will release even more fandom data, as this would allow a more detailed mapping of the behaviour of authors across (fiction) genres, which was not possible with the current selection. Once

cross-genre regularities and irregularities are understood better, cross-genre authorship attribution should become much more viable.

References

1. Aarts, J., van Halteren, H., Oostdijk, N.: The linguistic annotation of corpora: The TOSCA analysis system. International journal of corpus linguistics **3**(2), 189–210 (1998)
2. BNC Consortium: The British National Corpus, version 3 (BNC XML Edition). URL: http://www.natcorp.ox.ac.uk/ (2007)
3. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology **2**(27), 1–27 (2011)
4. van Halteren, H.: Linguistic Profiling for authorship recognition and verification. In: Proceedings ACL 2004. pp. 199–206 (2004)
5. van Halteren, H.: Author verification by Linguistic Profiling: An exploration of the parameter space. ACM Transactions on Speech and Language Processing (TSLP) **4**(1), 1 (2007)
6. van Halteren, H.: Source language markers in Europarl translations. In: Proceedings of COLING2008, 22nd International Conference on Computational Linguistics. pp. 937–944 (2008)
7. van Halteren, H.: Bot and gender recognition on tweets using feature count deviations, notebook for PAN at CLEF2019. In: Cappellato, L., Ferro, N., Müller, H., Losada, D. (eds.) CLEF 2019 Labs and Workshops, Notebook Papers. CEUR Workshop Proceedings. CEUR-WS.org (2019)
8. van Halteren, H., Oostdijk, N.: Linguistic Profiling of texts for the purpose of language verification. In: Proceedings of the 20th international conference on Computational Linguistics. p. 966. Association for Computational Linguistics (2004)
9. van Halteren, H., Speerstra, N.: Gender recognition of Dutch tweets. Computational Linguistics in the Netherlands Journal **4**, 171–190 (2014)
10. Kestemont, M., Stamatatos, E., Manjavacas, E., Daelemans, W., Potthast, M., Stein, B.: Overview of the Cross-domain Authorship Attribution Task at PAN 2019. In: Cappellato, L., Ferro, N., Losada, D., Müller, H. (eds.) CLEF 2019 Labs and Workshops, Notebook Papers. CEUR-WS.org (2019)
11. Koppel, M., Winter, Y.: Determining if two documents are written by the same author. Journal of the Association for Information Science and Technology **65**(1), 178–187 (2014)
12. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: Association for Computational Linguistics (ACL) System Demonstrations. pp. 55–60 (2014), http://www.aclweb.org/anthology/P/P14/P14-5010
13. Mendes Júnior, P.R., de Souza, R.M., Werneck, R.d.O., Stein, B.V., Pazinato, D.V., de Almeida, W.R., Penatti, O.A.B., Torres, R.d.S., Rocha, A.: Nearest neighbors distance ratio open-set classifier. Machine Learning **106**(3), 359–386 (Mar 2017)
14. Padró, L., Stanilovsky, E.: FreeLing 3.0: Towards wider multilinguality. In: Proceedings of the Language Resources and Evaluation Conference (LREC 2012). ELRA, Istanbul, Turkey (May 2012)
15. Potthast, M., Gollub, T., Wiegmann, M., Stein, B.: TIRA Integrated Research Architecture. In: Ferro, N., Peters, C. (eds.) Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF. Springer (2019)