CLG Authorship Analytics: a library for authorship verification

Erwan Moreau[1,2*] and Carl Vogel[1]

[1*]School of Computer Science and Statistics, Trinity College Dublin, College Green, Dublin, 2, Ireland.
[2]Adapt Centre, Trinity College Dublin.

*Corresponding author(s). E-mail(s): moreaue@tcd.ie;
Contributing authors: vogel@tcd.ie;

Abstract

The task of authorship verification consists in detecting whether two texts have been written by the same person. This paper describes the *CLG Authorship Analytics* software, which implements several individual methods as well as a stacked generalization system for authorship verification. The approach relies primarily on ensemble learning methods, i.e. repeatedly sampling the data in order to capture the invariant stylistic patterns. The approach is tested through a series of experiments designed to test the ability of the system to generalize, depending on various parameters. The code and results of the experiments are publicly available.[*]

Keywords: Stylometry, authorship verification, Genetic learning, Stacked generalization

1 Introduction

The task of authorship verification consists in detecting whether two texts have been written by the same person. Literary studies, e.g. identifying the anonymous author of a text, are the most traditional application of the task of authorship verification; but it also has direct applications in forensics, e.g. finding the author of anonymous threats [1, 2], and potentially in various

[*]https://github.com/erwanm/clg-authorship-experiments.

problems with social media, e.g. unmasking a user who posts messages under several identities, or detecting identity theft when an account has been hacked.

authorship verification is the cornerstone of a range of authorship applications, because solving this question would contribute greatly to solving most author identification problems [3]. In particular, the traditional task of author identification, which consists in predicting the author of a text among a set of known authors, can be expressed as multiple authorship verification problems. Importantly, authorship verification focuses on distinguishing a known author from *any other author*, as opposed to only discriminating among a set of known authors. This makes it usable not only for the traditional closed-set classification problem of author identification but also for the open-set variant of the problem, where the author can be one of the known authors *or some new unknown author*. This makes authorship verification much more general in theory and also much more useful in real-world applications such as forensics applications, where problems are usually open-ended. In terms of methods, this also means that authorship verification focuses on the most challenging part of any authorship problem, which is the problem known in Machine Learning (ML) as the *bias-variance tradeoff*: an authorship verification system must be able to distinguish features which are characteristic of an author's style (the general author's stylistic patterns) against features which are only characteristic of the available documents by this author (for instance when the document addresses a specific topic or belongs to a specific genre).

The authorship verification problem can be considered under different angles, in particular in a supervised or unsupervised context. In the former case, authorship verification is seen as a one-class classification problem; this setting is usually favored because it provides more reliable results in general, provided the tested data belongs to the same distribution as the training data. However, this condition is not easy to satisfy in most real applications, either by lack of adequate training data or simply because the type of data to process cannot be predicted. On the other hand, the unsupervised approach is a harder problem to tackle, but is by definition more robust and adaptable.

CLG Authorship Analytics is an open-source software[1] for solving authorship problems, with a particular focus on *authorship verification*. While the system is intended to allow both supervised and unsupervised learning, the latter is not very developed in the current version. The system originated from the authors' participation in the PAN shared tasks from 2013 to 2015 [4–6].[2] It is based on the prototype which was ranked second at the 2015 shared task, with significant improvements in efficiency and genericity. The system allows various setups, ranging from applying a single unsupervised method to training a meta-classifier with genetic learning, using distributed computing. Additionally to the large range of options which can be used with the existing methods, the system is designed to facilitate the integration of new methods at different levels: new categories of features or specific similarity measures can be added,

[1] Available at https://github.com/erwanm/clg-authorship-analytics.
[2] https://pan.webis.de/clef15/pan15-web/authorship-verification.html

new verification methods can be implemented and used in combination with the existing ones, etc.

The approach relies primarily on ensemble learning methods to address the challenging bias-variance tradeoff of authorship verification. By definition ensemble methods require repeatedly sampling the data in order to capture the invariant patterns, and this makes the system resource-intensive, in particular when training models with genetic learning. Additionally the numerous options makes the system fairly complex and somewhat cumbersome. This is why a detailed user guide has been written to accompany the present article.[3] This user guide details the technical aspects of the software, so that readers interested in testing or using the system may follow it in parallel to the explanations provided in this paper.

The article is organized as follows: We briefly review the state of the art in section 2; after describing the system in section 3, some new experiments intended to test the ability of the system to solve authorship verification problems under various conditions are presented in section 4 and analyzed in section 5. The focus of these experiments is not performance per se, but rather the study of the conditions which make the system perform well or not. In particular, these experiments are designed to measure how well the system generalizes to new unknown authors. For the sake of reproducibility, the code and results of the experiments are made available at https://github.com/erwanm/clg-authorship-experiments.

2 Related work

Various methods have been proposed in the literature to address author verification. In particular ensemble methods have proved quite sucessful at tackling the challenge of capturing the features relevant to the author's style and discarding the ones which are not [7–10, 21]. Author verification has also been the focus of several iterations of the PAN shared tasks,[4] e.g. [11–14]. These events contribute datasets and evaluation measures, thus allowing the community to compare different methods on the same basis and in turn boosting the development of author verification methods. The organizers also host a github.com project page[5] which indexes the source code of a large number of participating systems, thus providing a rich collection of state of the art authorship verification systems.

Recent years have seen the development of "big data" author verification methods. Until recently, most methods used for author verification were still relying on traditional supervised ML [15], even though the state of the art in NLP had already moved towards deep (neural) representations. This observation motivated the PAN organizers to collect a very large dataset, in order to allow neural methods, which need a high volume of training data, to show

[3] Available at https://erwanm.github.io/clg-authorship-experiments/user-guide-part1.html.
[4] https://pan.webis.de/.
[5] https://github.com/pan-webis-de.

their strength in the task of authorship verification [14]. This resulted in a new neural networks-based methods being proposed, e.g. [16–20].

The approach presented in this paper belongs to the "traditional" category of authorship verification methods, but a recent neural system is evaluated for the sake of comparison ([19], see details in Section section 4.3).

3 Approach

3.1 General design

The task of authorship verification consists essentially in answering the following question: given two distinct texts, have these texts been written by the same person? We opt for a general interpretation of this question, formally defined as follows: given two sets of texts $\{a_1, .., a_I\}$ and $\{b_1, .., b_J\}$, knowing that all the a_i texts were authored by A and that all the b_j texts were authored by B, are A and B the same person? Naturally this definition includes the most simple case where a single text a_1 is compared against a single text b_1. It also includes the PAN 13-15 variant of the question where a set of texts $\{a_1, .., a_I\}$ written by a "known author" is compared against a single document b_1 (the "unknown" or "questioned" document). We also adopt the same probabilistic representation as PAN, defining the output of a verification problem as a numerical value in the interval [0, 1] which represents the probability that the answer is positive (same author). The intended interpretation is for 0 to mean "different author" with maximum certainty, for 1 to mean "same author" with maximum certainty, and any intermediate value describes the likelihood of a positive answer. Again this is the most general representation, since this numerical value can easily be converted to a boolean answer by thresholding it at 0.5.

As a post hoc analysis, the organizers of PAN 2014 evaluated a meta-model based on all the individual systems answers. They observed that *" this meta-classifier was better than each individual submitted method while its ROC curve clearly outperformed the convex hull of all submitted approaches. This demonstrates the great potential of heterogeneous models in author verification, a practically unexplored area."* [11].

This idea of combining the strengths of multiple heterogeneous systems forms the basis of our supervised approach: many individual systems are trained, then their predictions are used to train a meta-model which combines them optimally to predict the final answer (stacked generalization). In this approach, the final model performs theoretically at least as well as any individual learner, and potentially better. However this has two main disadvantages:

- There is a high risk of overfitting the model due to the optimization of a possibly very large number of parameters based on a limited training dataset.

- The system is complex and computation-heavy, since it requires training multiple different models and minimizing overfitting by repeatedly sampling the data.

It is worth noting that every possible individual learner can be used as an independent unsupervised authorship verification system, but the system is not particularly adapted for this usage for two reasons:

- An individual model depends on a set of hyper-parameters with many possible values. This is because the individual models are intended to be optimized and used as independent heterogeneous learners in a supervised context. The choice of these values can have an important impact on the performance of the system, but there is no indication of which one to choose in an unsupervised context.
- The individual models can (and usually do) deliver multiple features as output (see also below). Depending on the method, some of the features may be used individually as an answer to the verification problem, but in general this output is not directly interpretable.

3.2 Common parameters

We call *observation types* the different types of raw features used by the individual models.[6] An observation can be any observable feature in a text. The most common type is n-grams, but observation types are unrestricted. A document can be represented as a collection of n-grams (e.g. bag-of-words or bag-of-characters) of a single observation type.

Different types of observations can be taken into account, e.g. words bigrams, character 5-grams, POS trigrams, etc. The different kinds of observations are distributed among different categories (called *observations families* in the system):

- Token-based n-grams. Examples: `T` represents unigram tokens, `TT` represents bigrams, etc. A skip-gram is represented with a `S`, e.g. `TST` is a skip-gram of length 3 where the middle token is skipped; the result is a bigram which contains the 1st and 3rd token.
- Character-based n-grams. Examples: `CC` represents character bigrams, `CSSCC` skip-grams of length 5 where the 2nd and 3rd chars are skipped; the result is a trigram which contains the 1st, 4th and 5th token.
- Part-Of-speech-based n-grams, which can combine tokens (`T`), POS tags (`P`), lemmas (`L`) and skip (`S`).
- Class-based observations, which represent specific mappings of words to predefined classes, for example:
  - Morphology classes, such as "number", "punctuation", "capital first letter";

[6] To prevent any confusion, we reserve the word "features" for the information fed to the supervised learning algorithms.

– Token length classes, where the tokens are classified depending on their length.

Every observation type also has two parameters which control (1) the minimum frequency for an observation to be taken into account, and (2) whether they should be extracted when they overlap several text units, typically sentences. Additionally, the token-based features accept an option which specifies which words to take into account as predefined vocabulary. This causes all the words which are not in the vocabulary to be replaced by a placeholder symbol. It can be used to count only patterns involving frequent words (often called stop words), in order to avoid content words which are less likely to be good indicators of an author style. Note that this is different from the minimum frequency option, which discards full observations, as opposed to some of the words in an observation.

3.3 Strategies

A *strategy* is a method to produce indicators about the answer to a verification problem. It is possible for a verification strategy to return only one feature value which answers the question directly, but in general the output is a set of features, the meaning of which depends on the strategy. This set of features is intended to be used as input to a learning algorithm. A strategy can have multiple hyper-parameters which can be optimized in the supervised context using genetic learning (see section 3.4.3). Three strategies described below are currently implemented in the system. Every strategy receives as input a set of observation types, each corresponding to a particular representation of the input documents.

3.3.1 Basic strategy

As its name suggests, the *basic* strategy is a simple method. It computes the similarity between two documents according to a similarity measure for every input observation type:

- The only similarity measures currently available in the system are cosine and minmax.[7]
- An output feature is produced for every input observation type, containing the value of the corresponding similarity score.

3.3.2 General Impostor strategy

This method is described fully in [10], used in previous PAN workshops by [22] and in a modified form by [23] and [24]. It repeateadly compares portions of the tested documents against each other and against other external (portions of) external documents which represent the *impostors*. If the

[7]Minmax is defined as $\frac{\sum_i min(x_i,y_i)}{\sum_i max(x_i,y_i)}$, where (x_i, y_i) represent the frequency of the observation i in the two documents.

similarity between the tested documents is significantly higher than the similarity obtained between a tested document and an impostor, then the tested documents are likely to be from the same author.

Randomness plays a crucial role in this method: at every run, the portions of document, the impostor document, the observation type and the subset of observations are picked randomly. Additionally the method takes several parameters, the most important of which are listed below:

- Number of iterations.
- Preselection of the most similar impostors documents. This option allows a subset of impostors documents to be used instead of the full set in order to compare roughly comparable documents.
- Proportion of observations to select at every iteration.
- The similarity measure to use.
- Method to count the most similar features. Available methods include the original one defined in [10] as well as two variants.

The choice of the set of impostors documents is an important parameter. Different options have been proposed in the literature. A common option is to use the results of some Google queries formed by randomly picking words from the set of input documents as impostors.[8] In the experiments presented below (see section 5), we opt for using all the training documents as impostors. While this option is not ideal since the documents obviously include precisely the documents to be compared, it is a reasonable simplification if the training set is diverse enough in terms of authors and if the number of iterations is large enough to prevent the occasional wrong comparison from having a significant effect on the output features.

3.3.3 Universum Inference strategy

This strategy follows the idea described in [8]: in this paper, a large corpus containing several "categories" (the input documents in our task) is split into small chunks. A chunk of category A is compared against many chunks from other categories and from category A as well, picked randomly. It was shown that a reliable measure of the category homogeneity can be derived from examining how different the level of similarity is between comparing A to A and comparing A to some distinct category X.

We adapted the approach to the case of smaller documents in the following way. Let A and B be two documents, the following process is repeated N times with different random subsets:

1. The two documents are split into three parts randomly: A_1, A_2, A_3 and B_1, B_2, B_3; one of the thirds is split again into two parts: $X_3 = X_3', X_3''$.
2. The pieces of text are re-organized under three categories, each containing two parts and all the parts being of similar size: $C_A = \{A_1, A_2\}$; $C_B = \{B_1, B_2\}$; $C_{mixed} = \{A_3' \cup B_3', A_3'' \cup B_3''\}$.

[8]The system includes a script to this effect.

3. The three categories are compared in the same way as in [8], that is, both against itself (using the two parts belonging to this category) and against each other category (picking one of the two parts randomly).

In other words, different portions of A and B are mixed together repeatedly, then we compare the similarity obtained between A and A, B and B, A and B, mixed-AB and A, mixed-AB and B, mixed-AB and mixed-AB. If A and B have the same author, the resulting similarity scores should be all similar. The hyper-parameters specify the number of iterations, the proportion of observations to pick at every iteration, the similarity measure, as well as several parameters which determine the calculations used to obtain the output features.

3.4 Supervised learning

3.4.1 General principle

A strategy returns a set of features for every verification problem. A problem can be labelled, i.e. provided with the gold-standard answer indicating whether the two groups of documents are actually from the same author or not. Naturally, a set of labelled verification problems can be used to train a supervised model, using the features returned by the strategy to represent every problem as an instance. The most simple option would be to treat the task as binary classification, but since we are also interested in quantifying the level of the confidence of the system (see section 3.1), the supervised setting is implemented as a regression task:

- During training, the problems are provided with the answer as 0 (different author) or 1 (same author). The model is trained to predict this numerical value based on the features returned by the strategy for multiple such problems.
- When the model is applied, the predicted value for the answer may vary between 0 and 1. A value close to 0 (resp. 1) indicates probably different (resp. same) author, and a value around 0.5 indicates uncertainty.

Thus a regression model is trained (or applied) to the features which have been computed for all the input problems (instances for the model). As learning algorithms, we use the Weka [25] (version 3.8.5) implementation of SVM regression (with polynomial or RBF kernel) [26] as well as decision trees regression [27], with variants depending on their parameters.

Optionally, a second model can be trained in order to evaluate the confidence of the model in each answer, and possibly replace it with the special value 0.5, meaning that the case is unanswered in the PAN evaluation methodology. This classification “confidence model” can use any of the available features, as well as the score computed with the first regression model. In the learning stage, the model which was trained is applied to the instances. Depending on the configuration, the instances can be split up so that the second model is based on unseen instances, but then less instances are used to train each model. This option is not used in the experiments presented in this paper.

3.4.2 Evaluation

The system supports three evaluation measures:

- Accuracy, i.e. the proportion of problems predicted correctly. The system actually uses the modified accuracy c@1, which was introduced in [28] in order to take into account the possible absence of answer (represented as the special value 0.5), i.e. cases where the system chooses to answer "I don't know". Such cases are less penalized than wrong predictions by c@1. Since we do not use the confidence evaluation system in these experiments (see above), the system is very unlikely to predict exactly 0.5 so this measure is practically equivalent to accuracy in our case.
- Area Under the ROC curve (AUC), which can be interpreted as the probability that a true negative case obtains a score lower than a true positive case.
- The product of two above scores, which was the official evaluation measure at PAN15 [12]. It is worth noting that this evaluation score is mathematically lower than accuracy or AUC (it is equal only in the case of perfect performance). In particular, the random baseline corresponds to a performance of 0.5 for both accuracy and AUC but to 0.25 for their product.

3.4.3 Genetic algorithm

As mentioned above, a verification strategy may be applied to any set of observation types and may accept a number of specific hyper-parameters. These input parameters are provided to the strategy in the form of a *configuration*, which represents every possible option as a name-value pair: $C = \{p_1 \mapsto v_1, \ldots, p_n \mapsto v_n\}$. Naturally the total number of possible configurations, i.e. all the combinations of name-value pairs, can be extremely high. The genetic algorithm is used to explore the space of all the possible configurations efficiently and discover the best performing ones for a particular training set. In other words, this process is essentially hyper-parameter tuning with an extremely large number of parameter values.

The genetic algorithm works with configurations as "individuals": each configuration describes the hyper-parameters of a strategy. A *multi-configuration* associates multiple values to one parameter:

$$MC = \{p_1 \mapsto \{v_1^1, \ldots, v_{m_1}^1\}, \ldots, p_n \mapsto \{v_1^n, \ldots, v_{m_n}^n\}\}$$

In theory, a set of multi-configurations can be used to describe the set of meaningful combinations of parameters, in a way similar to a disjunctive normal form. For the sake of simplicity we use a single multi-configuration by strategy, leaving the selection of relevant combinations to the genetic algorithm. A multi-configuration is the input of the genetic algorithm:

- The first generation of configurations is initialized randomly: N configurations are selected among the possible combinations represented by the multi-configuration.
- After the first one, every generation is obtained based on the individual performance of the configurations from the previous generation:
  - The "breeders" are selected in a way such that the probability of a configuration being selected is proportional to its rank by performance.
  - For every new configuration, two parents are selected randomly among the breeders and every parameter is assigned the value of either one of the parents value, with a small possibility of mutation.

Additionally, the algorithm allows for a proportion of the new generation to be selected fully randomly, and for a proportion of the best previous configurations to be cloned to the next one (elitism).

The convergence of the algorithm is assessed automatically: at the end of every generation, the mean performance over a window of the last n generations is calculated. If this value does not increase anymore over a fixed number of windows, the stop criterion is met. The user can specify various parameters in order to adapt the process to the task at hand and the available computing resources: the multi-configuration which defines the search space, the parameters which control the genetic algorithm, and the stop criterion. This way the genetic process can be configured to favor speed or a more exhaustive exploration of the search space.

The disadvantage of using a genetic algorithm, especially with a vast set of possible configurations, is the risk of overfitting. We use cross-validation inside the genetic algorithm: every generated configuration is evaluated using k-fold cross-validation, and the resulting performance is used as the fitness function by the genetic algorithm. We use several other techniques in order to keep overfitting to a minimum:

- The partitioning for the k-fold cross-validation is randomly (re-)generated at every generation.
- The system allows to chain multiple stages of genetic learning with different parameters, in particular different values of k and different values for the size and number of windows in the stop criterion. At every new stage, the previously selected set of configuration is used as first generation, and re-evaluated under the new parameters. This allows the process to check and progressively refine the optimal configurations and/or adjust the trade-off between the precision of the process and the required computing power.
- At the end of the last stage, the N best configurations are re-evaluated using a 10x2 cross-validation setting,[9] in order to control the influence of the cross-validation partitioning on the performance variance.

The genetic learning process is applied to every strategy independently, resulting in a set of N best configurations for each strategy.

[9]Repeat 2-fold cross-validation 10 times with a different split every time.

3.5 Meta-model (stacked generalization)

Training the meta-classifier consists in finding an optimal way to combine the results obtained by the N best individual strategy configurations returned for each of the strategies (e.g. if three strategies are trained, there are $N \times 3$ configurations available as features for the meta-training; in the experiments below we use $N = 50$). Among these, the strategy configurations which prove useful are selected (in the experiments, the meta-model usually selects between 50 and 70 individual models out of 150 available), the other ones are discarded (feature selection). The method to combine them is also selected during this stage. This process results in a set of M best meta-models, each model being a combination of a selection of strategy configurations. The initial N best individual strategy configurations are also re-evaluated, to allow selecting a single strategy model instead of the best meta-model. In the below experiments we also use this option in order to study the performance of the meta-model in relation to individual models.

The predicted scores of the N best strategy configurations for every strategy are used as features by the meta-classifier. A subset of strategy configurations, together with a method to combine them, forms a *meta-configuration*. The genetic learning algorithm is used once again at this stage in order to find the optimal meta-configuration(s).

In a meta-configuration, an additional parameter indicates how the strategy predictions are combined. In order to avoid another potential source of overfitting, we restrict this combination to the simplest methods: the algorithm can select only the arithmetic mean, geometric mean or the median. The system allows using a regression algorithm; while this might provide better results, it would also increase the risk of overfitting.

The cross-validation setting is quite complex because it needs to balance the limited amount of instances with the need to assess the model on fresh instances at three different levels. The computational complexity is also an issue that must be taken into account. Importantly, the instances used to train the individual strategy models cannot be reused to train the meta-model: the selected configurations have been optimized during the previous stages, therefore their predicted scores otained on these instances are unrealistically accurate. This would result in the trained meta-classifier expecting higher quality features than those actually obtained from fresh instances, causing a large performance drop on the test set. For these reasons, the system applies a method which does not correspond to a proper nested cross-validation setting but is somewhat similar: at the outermost level, the training data is split into two subsets A and B. The following process is run independently twice, once with A as training set and B as test set and once with the opposite training/test assignment:

1. The strategy genetic training uses the outermost training set only (50%). The genetic process uses cross-validation internally at every generation, then returns the set of N best models for every strategy.

2. The outermost test set (50% of the full data) is split further into meta training set (25%) and validation set (25%).
3. The meta-model is trained using only the meta training set. Again, the genetic process uses cross-validation internally at every generation.
4. The best meta-models, as well as the best individual strategy models, are finally evaluated on the validation set, i.e. the last unseen 25% instances. This evaluation uses a variant of bagging (bootstrap aggregation):[10] the models are evaluated 20 times against a different random half of the instances (thus one can manually control for variance).

But there are still two issues left:

- The last evaluation stage is done on a small proportion of instances, so the results might not be very reliable.
- The resulting models from the two outermost cross-validation runs are not comparable together. Moreover, it is possible that one of the runs would perform better than the other in average because of the different subsets the two use for training and testing.

This is why an additional stage of evaluation of all the resulting models is carried out on the whole dataset, using our variant of bagging again. This unusual methodology relies on the fact that the meta-models have not *directly* seen the outermost training instances used for the strategy training stage. In other words, this is a pragmatic workaround for the two issues above. Testing proved that the performance on these instances was, in general, not overevaluated compared to the actual fresh instances of the validation set. There is an obvious risk of bias with this final evaluation method, thus it might be preferable to rely on the validation set results, at least when the number of instances is large enough.

4 Experimental setup

4.1 Data

The Diachronic Corpus for Literary Style Analysis (DCLSA), proposed by [29], is made of 554 books written by 22 American literary authors from the mid 19th to early 20th century. The books were collected mostly from Project Gutenberg[11], supplemented with books from the Internet Archive[12]. [29] indicates that books from the latter source were scanned using Optical Character Recognition (OCR), causing various OCR errors which were corrected using a semi-automatic method.

We opted for an English literature corpus because the experiments are aimed at exploring how well the system distinguishes authors by varying parameters such as the size of the documents or the number of verification

[10]Strictly speaking, the process differs from bagging because the instances are sampled without replacement.
[11]https://www.gutenberg.org/.
[12]https://archive.org/.

problems used as training data. This corpus offers a quite diverse collection of authors and multiple long texts for most authors, thus allowing the selection of subsets of data along different dimensions. Literary texts are notably less challenging for stylometry systems than user-generated content such as social media texts, since the authors are professional writers who are able to maintain stylistic consistency. Since we are interested in comparing performance between different settings, high quality texts are more suitable in order to minimize variations due to the texts themselves. The corpus was originally compiled for the purpose of studying changes in the authors' style across time, but we do not consider this aspect in this paper.

The experiments presented below in section 5 are made with limited portions of the full books. The size of the portions is counted in number of lines, knowing that all the documents are formatted with standard-size lines: in average a non-empty line contains 10.3 words or 56.6 characters, with 99% of the lines made of less than 76 characters. Thus a 100 lines document, as used in most of the experiments, corresponds to approximately 1,030 words or 5,700 characters. All the books except one are more than 1,000 lines long, ranging from 230 lines to 30,000 with an average of 10,344 lines. Contiguous portions of text are extracted randomly from a document, after discarding empty lines in order to maintain a consistent document size. The random selection is done only once in the first experiment (see section 5.1).

4.2 Design of the experiments

In order to measure the effect of various parameters and preserve comparability of the results, the documents are split in a controlled way between the training and test set:

1. The 22 authors[13] are randomly split into three groups of approximately equal size:
    - Authors assigned to the training set only;
    - Authors assigned to the test set only;
    - Authors shared between the training and test set.
2. For every author, their books are pre-assigned to either the training or test set:
    - If the author belongs to the *training set only* group (resp. *test set only*) then all their books are pre-assigned to the training set (resp. test set);
    - If the author belongs to the shared group, their books are randomly pre-assigned to either the training and test set.

As a result of this method, a book cannot belong to both the training and test set in any of the experiments. For every experiment, books are randomly picked according to these pre-assignments in order to build a labelled dataset

[13]Note: according to the filenames identifying the authors in the DCLSA, the dataset contains two additional authors with one book each. It is not clear to the author whether this was intended or not. In our experiments these two isolated authors are automatically assigned to the test set.

of verification problems, i.e. a set of of pairs of same/different authors groups. The specific training and test set vary for every experiment (see section 5), but the pre-assignments described above is maintained identical across all the experiments. As a consequence the variations in performance cannot be due to a particular author or book being assigned to either the training or test set. The "shared author" category is used to compare the performance on the test set between authors seen vs. unseen in the training set. An ideal verification system should be able to perform as well whether the involved authors were known from the training or not. All the test set problems containing at least one group where the author was seen in the training set are marked as "author seen in the trainining set".[14] The difference in performance between the cases where the author was seen in the training set versus those where they were not is called the *known author bias* in the analysis presented below.

All the experiments described below follow the same general design: given a particular parameter as target of the experiment, a range of appropriate values for this parameter is studied. An individual experiment is prepared for each such value, where the training and test data are generated randomly but following various constraints (specific to the study) in order to maintain "all other things equal" as much as possible. The results of the experiments are obtained in a standard way, by using the training set for the full training process (including the nested cross-validation splitting described in section 3.5), then calculating the performance of every model on the unseen test set. The product of accuracy and AUC is used to measure performance (see section 3.4.2). Additionally the performance on the training set is calculated in order to measure overfitting, and the test set performance is also calculated separately for the "author seen in the trainining set" instances vs. "author not seen". Finally we also extract not only the top model returned by the training stage (called the *selected* model below), but also the best model for each of the four types (one for each of the three strategies plus one meta-model). This allows the analysis of the differences between types of models, in particular with respect to overfitting.

Ideally the experiments would be performed multiple times in order to minimize the effect of randomness on performance. Unfortunately the complexity of the system makes this difficult, as a single training process take one to three days using 40 cores under the default parameters. This is why each experiment is run only once, but with several values of the varying parameter in order to obtain an accurate picture of the performance variations across values. The code used for the experiments is available at https://github.com/erwanm/clg-authorship-experiments.

4.3 Baseline systems

Experiments are also carried out with a third-party system for the sake of comparison. To this end, a recent neural system has been trained and tested

[14]Note: this includes cases where one author is known from the training set and the other is not.

in the same way as the main system. The *AdHominem/O2D2* system [19][15] is a hybrid neural-probabilistic end-to-end framework, which includes neural feature extraction and deep metric learning, deep Bayes factor scoring, uncertainty modeling and adaptation, a combined loss function, and an out-of-distribution detector for defining non-responses. This system achieved the best performance at the PAN 21 authorship verification task. Thus in theory this system is a competitive baseline, but it is designed to be trained with a high volume of data. This is why two models are used in every experiment:

- The regular *AH* model is trained on the same data as our system.
- The pretrained *AH.pan21* model submitted to PAN 21 (made available by the authors), trained on the PAN official training data [14]: 148,000 same-author and 128,000 different-author pairs across 1,600 fandoms "fanfiction".

We do not use the out-of-distribution detector (O2D2) part of the model, consistently with the exclusion of the confidence evaluation system in our own model (see section 3.4). We originally intended to use the predictions of the Uncertainty Adaptation Layer (UAL), i.e. the output of the model immediately before the O2D2 component is applied. However preliminary tests showed very poor performance with both models, due to the models always predicting a positive case. This happens with both the regular and pretrained model, ruling out an error in the training process for *AH*.[16] Because of this issue, it was decided to use the predictions of the Deep Metric Learning (DML) part, i.e. the first component of the model.

In the following experiments, the performance of the two *AH* models is shown in the main performance graph (first from the left). These systems are excluded from the graphs showing the training/test and the seen/unseen authors difference in performance. Additionally they are excluded from the second experiment about the number of documents by group, because the *AdHominem/O2D2* is designed only for pairs of single documents.

5 Experiments

5.1 Document size

In this experiment we measure the effect of the documents size on the performance of the system. Both the training and test set contain 100 instances (problems). Every problem is made of a pair of single documents (i.e. no group of multiple documents is used to represent an author). Both the training and test set contain an equal number of positive and negative instances. Additionally the books are picked without replacement, so that a book never appears

[15]The authors made their system available at https://github.com/boenninghoff/pan_2020_2021_authorship_verification. Some adaptations were required in order to train and apply the system to our dataset, these modifications are made available at https://github.com/erwanm/pan_2020_2021_authorship_verification.

[16]*AdHominem/O2D2* is a complex system, and despite our best efforts it is possible that an error was made is some other part of the process.

twice neither in the training set nor in the test set. The different sizes of documents are applied to both the training and the test set.

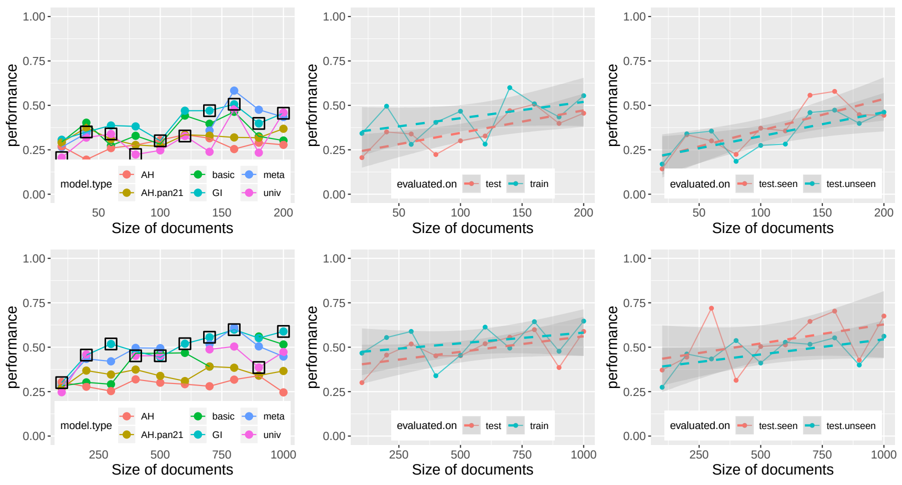

Fig. 1 Performance by document size. Left: performance of the three best strategy learners and meta-learner, with the models selected by the training stage for every size marked with a black square box.[17] Middle: performance of the selected model on the training and test set. Right: performance of the selected model (test set) when separating instances whether the author was "seen during training" or not. The middle and right graphs show the linear regression lines (dashed lines). For the sake of readability the data points are split into two series, with sizes from 20 to 200 by step of 20 shown in the top graphs and sizes from 100 to 1,000 by step of 1000 shown at the bottom.

Fig. 1 shows the performance of the system for different sizes of documents. As expected, the performance tends to increase with the size. However the variations in performance are very large even when reaching large sizes. These variations are very likely due to the different portions of documents being picked randomly at different sizes, making the set of verification problems more or less difficult to solve. The training stage tends to overestimate performance, which is a clear indication of overfitting. While overfitting tends to reduce as the size of the documents increases, the opposite is true about the known author bias. There is too much noise in the observations to draw any solid conclusion, but it is possible that larger documents cause the models to become more specific about their authors.

5.2 Number of documents by group

Recall that a problem is represented as two sets of texts $\{a_1, .., a_I\}$ and $\{b_1, .., b_J\}$, with all the a_i texts authored by A and all the b_j texts authored by B. In this experiment we measure the effect of the number of documents in the

[17]Some model types may not have a value for every size because the system returns only the top 50 models. As a consequence, a model type is ignored if it did not reach the top 50 during training.

group on one side of a verification problem, that is we make the number I of documents a_i vary, while the second group $\{b_1, .., b_J\}$ always contains a single document ($J = 1$). This is an important parameter because when $I > 1$ the system can leverage the similarities and differences between the documents a_i to characterize the style of author A more accurately. Both the training and test set contain 100 instances (problems) equally divided between positive and negative instances. The documents are picked with replacement in the training, allowing a book to appear several times in the training set. However the same book never appears twice in the the test set. All the documents are 100 lines long.

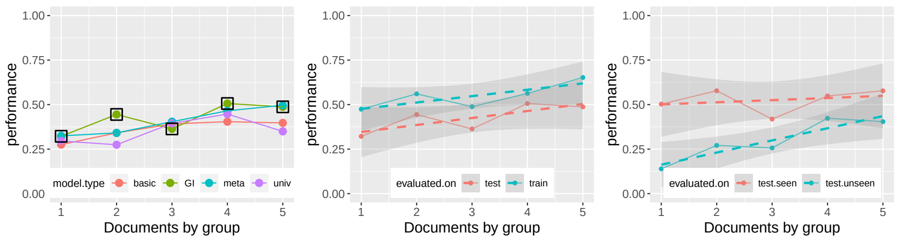

Fig. 2 Performance by group size. See the explanations about the different graphs in the caption of fig. 1.

As expected, fig. 2 shows that the performance increases with the number of documents by group. However there is no observable reduction in overfitting as the group size increases. The difference in performance between author seen vs. unseen instances is important, but decreases significantly when the group size increases. This tends to indicate that the system generalizes better to new author if it is fed with verification problems made of large groups of documents.

5.3 Number of problems in the training set

This experiment examines the role of the number of problems in the training set. A high number of instances is likely to reduce the risk of overfitting. The test set is once again made of 100 instances without any duplicate book, while the training set contains a varying number of instances in which books are picked with replacement. Both the training and test set instances are made of a single document by group and the positive/negative cases are balanced. All the documents are 100 lines long. It is also worth noting that the training process is much longer when the number of instances in the training set is large.

Increasing the number of instances in the training set reduces overfitting, as expected. The training and test performance become quickly much closer to each other, and there is no visible overfitting starting from around 500 instances. Interestingly, this gives the meta model a clear advantage over the individual models: compared to the other experiments where the number of intances is always 100, the meta model is more often selected as the best

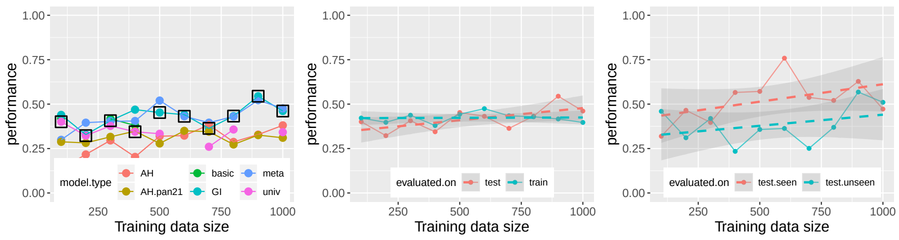

Fig. 3 Performance by number of problems in the training set. See the explanations about the different graphs in the caption of fig. 1.

model by the training stage, and does indeed perform best on the test set most of the time. However the overall performance increases only moderately with more instances. Finally the difference in performance between author seen vs. unseen instances is large, and even appears to increase as the size of the training set increases. It seems that the larger number of training instances causes the model to "specialize" in the authors found in the training set. But the variations in seen/unseen author are very large across the different sizes, so it is unclear whether this observation is a real pattern or due to chance.

5.4 Author diversity in the training set

In this experiment we study how the number of distinct authors in the training set impacts the performance of the verification system. This makes it possible to measure how author-wise diversity in the training set influences the ability of the model to generalize to unseen authors found only in the test set. The experiment setting is as follows: the test set contains 100 instances split equally between positive and negative cases, with no duplicate book. The training set also contains 100 instances but can contain duplicate books. The problems are all defined with groups made of single documents and the documents are 100 lines long. In order to guarantee a minimum of document-level diversity in the training set when the variable number of authors is low, the training set authors are first ranked by decreasing order of their number of books, and then added to the training set in order. This means that more distinct books are added at every increment for the low values in number of authors than for the high values.

It can be observed in fig. 4 that overfitting decreases drastically when the number of authors increases: the performance on the test set and training set converges when reaching the highest number of authors. Curiously though, the performance on authors unseen during training increases only slightly and stays far below the performance on seen authors, even when reaching the highest number of authors. This might indicate that author diversity contributes mostly to a more accurate estimation of performance during training, but not (or very slightly) to a higher performance on the test set. However this interpretation would require more experiments to be confirmed, since it may be a consequence of the important performance variations.

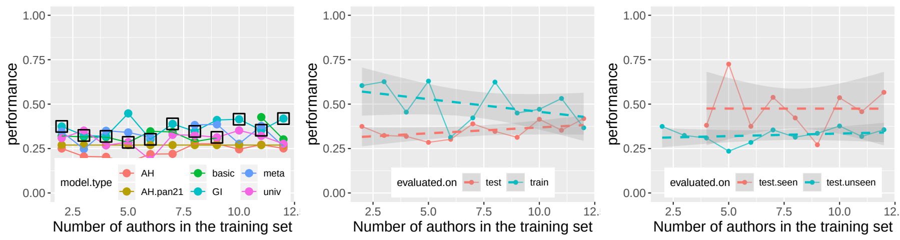

Fig. 4 Performance by number of distinct authors in the training set. See the explanations about the different graphs in the caption of fig. 1.

Model type	Best on training set	Best on test set
GI	69.8%	51.2%
basic	2.3%	11.6%
meta	2.3%	32.6%
univ	25.6%	4.6%

Table 1 Overall statistics by model type.

5.5 Global observations

Table 1 shows how often every type of model is estimated as the best model during the training stage and (2) actually performs best out of the four model types on the test set.

While *GI* is clearly the best individual strategy, the second best strategy on the training set is *univ* even though it almost never performs best on the test set. This is due to overfitting, as observed in the various experiments: *univ* tends to perform well on the training set, but drastically underperforms on the test set. By contrast, the meta-model barely suffers from overfitting and achieves the best performance on the test set almost a third of the time.

The better stability of the *meta* method is also confirmed by a small study of the case where the size of the documents is 100, the number of documents by group is one, the number of training instances is 100 and the number of authors is maximum. This combination of parameters is present in all four experiments. We calculated the standard deviation of the performance on the test set for the four model types: all three individual models have a standard deviation between 0.066 and 0.069, while the meta model has only 0.026. The sample is too small for this observation to be conclusive, but this observation evidences the better stability of the meta-model.

The *AH* baseline models perform decently in most experiments but not as well as the best models, *GI* and the meta-model. As mentioned in section 4.3, this is probably due to the fact that this model was designed for a massive volume of training data and that the pretrained model is too specific to the domain of fanfiction. In particular, the AH model performs especially bad in experiments 1 and 3 when the number of instances or authors is low. It can reasonably be assumed that these models would likely outperform our model if provided with a lot more training data. It is however questionable whether a high volume of training data is a realistic setting in the perspective of authorship verification applications.

6 Conclusion

We described the *CLG Authorship Analytics* system, which implements several individual methods as well as a meta-model for the task of authorship verification. We performed a series of experiments in order to measure the impact of various standard parameters of the authorship verification task on performance. The analysis shows that the size of the documents as well as the number of instances in the training set tend to reduce overfitting but to increase the known author bias. On the contrary, increasing the number of documents by group has little or no effect on overfitting but tends to decrease the known author bias.

Overall, the *General Impostor* method performs best. The results also show that the stacked generalization approach tends to cause less overfitting and more stable performance overall. However our experimemts did not establish clearly the superiority of the meta-model: the best individual strategy, *GI*, performs as well or slightly better most of the time. Despite the variety of the hyper-parameters for each individual strategy and the multiple learners combined in the meta-model (usually 50 to 70), the meta-model could certainly benefit from more having diversity to choose from in the type of models. Future work should focus on adding new and diverse models, in order to improve the performance of the meta-model.

7 Declarations

This work was supported by the ADAPT Centre for Digital Content Technology, funded under the SFI Research Centres Programme (Grant 13/RC/2106) and co-funded under the European Regional Development Fund.

The authors have no relevant financial or non-financial interests to disclose.

References

[1] Lambers, M., Veenman, C.J.: Forensic authorship attribution using compression distances to prototypes. In: Geradts, Z.J.M.H., Franke, K.Y., Veenman, C.J. (eds.) Computational Forensics: Third International Workshop, IWCF 2009, The Hague, The Netherlands, August 13-14, 2009. Proceedings, pp. 13–24. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03521-0_2. http://dx.doi.org/10.1007/978-3-642-03521-0_2

[2] Abbasi, A., Chen, H.: Applying authorship analysis to extremist-group web forum messages. IEEE Intelligent Systems **20**(5), 67–75 (2005). https://doi.org/10.1109/MIS.2005.81

[3] Koppel, M., Schler, J., Argamon, S., Winter, Y.: The "fundamental problem" of authorship attribution. English Studies **93**(3), 284–291 (2012) https://arxiv.org/abs/http://dx.doi.org/10.1080/0013838X.2012.668794. https://doi.org/10.1080/0013838X.2012.668794

[4] Moreau, E., Vogel, C.: Style-based distance features for author verification - notebook for pan at CLEF 2013. In: CLEF 2013 Evaluation Labs and Workshop - Working Notes Papers, Valencia, Spain, p. (2013)

[5] Moreau, E., Jayapal, A., Vogel, C.: Author Verification: Exploring a Large set of Parameters using a Genetic Algorithm - Notebook for PAN at CLEF 2014. In: Cappellato, L., Ferro, N., Halvey, M., Kraaij, W. (eds.) Working Notes for CLEF 2014 Conference, vol. 1180, p. 12. CEUR Workshop Proceedings, Sheffield, United Kingdom (2014)

[6] Moreau, E., Jayapal, A., Lynch, G., Vogel, C.: Author Verification: Basic Stacked Generalization Applied To Predictions from a Set of Heterogeneous Learners - Notebook for PAN at CLEF 2015. In: Cappellato, L., Ferro, N., Jones, G.J.F., SanJuan, E. (eds.) CLEF 2015 - Conference and Labs of the Evaluation Forum. CEUR Workshop Proceedings. CEUR, Toulouse, France (2015)

[7] Koppel, M., Schler, J., Bonchek-Dokow, E.: Measuring differentiability: Unmasking pseudonymous authors. Journal of Machine Learning Research **8**, 1261–1276 (2007)

[8] Vogel, C., Lynch, G., Janssen, J.: Universum inference and corpus homogeneity. In: Bramer, M., Petridis, M., Coenen, F. (eds.) Research and Development in Intelligent Systems XXV, pp. 367–372. Springer, ??? (2009). https://doi.org/10.1007/978-1-84882-171-2_29

[9] Koppel, M., Seidman, S.: Automatically identifying pseudepigraphic texts. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP, A Meeting of SIGDAT, a Special Interest Group of The ACL, pp. 1449–1454. ACL, Grand Hyatt Seattle, Seattle, Washington, USA (2013)

[10] 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)

[11] Stamatatos, E., Daelemans, W., Verhoeven, B., Stein, B., Potthast, M., Juola, P., Sánchez-Pérez, M.A., Barrón-Cedeño, A.: Overview of the author identification task at PAN 2014. In: Cappellato, L., Ferro, N., Halvey, M., Kraaij, W. (eds.) Working Notes for CLEF 2014 Conference. CEUR Workshop Proceedings, vol. 1180, pp. 877–897. CEUR-WS.org, Sheffield, UK (2014). http://ceur-ws.org/Vol-1180

[12] Stamatatos, E., Daelemans, W., Verhoeven, B., Juola, P., Lopez Lopez, A., Potthast, M., Stein, B.: Overview of the Author Identification Task at PAN 2015. In: Working Notes Papers of the CLEF 2015 Evaluation Labs. CEUR Workshop Proceedings. CLEF and CEUR-WS.org, Toulouse, France (2015). http://www.clef-initiative.eu/publication/working-notes

[13] Bevendorff, J., Ghanem, B., Giachanou, A., Kestemont, M., Manjavacas, E., Markov, I., Mayerl, M., Potthast, M., Rangel, F., Rosso, P., *et al.*: Overview of pan 2020: Authorship verification, celebrity profiling, profiling fake news spreaders on twitter, and style change detection. In: International Conference of the Cross-Language Evaluation Forum for European Languages, pp. 372–383 (2020). Springer

[14] Kestemont, M., Manjavacas, E., Markov, I., Bevendorff, J., Wiegmann, M., Stamatatos, E., Stein, B., Potthast, M.: Overview of the cross-domain authorship verification task at pan 2021. In: CLEF (Working Notes) (2021)

[15] Stamatatos, E.: Authorship verification: a review of recent advances. Research in Computing Science **123**, 9–25 (2016)

[16] Schaetti, N., Emile-Argand, R.: Author verification in stream of text with echo state network-based recurrent neural models. In: SwissText (2019)

[17] Tyo, J., Dhingra, B., Lipton, Z.C.: Siamese bert for authorship verification. In: CLEF (Working Notes), pp. 2169–2177 (2021)

[18] Manolache, A., Brad, F., Burceanu, E., Barbalau, A., Ionescu, R., Popescu, M.: Transferring bert-like transformers' knowledge for authorship verification. arXiv preprint arXiv:2112.05125 (2021)

[19] Boenninghoff, B., Nickel, R.M., Kolossa, D.: O2d2: Out-of-distribution detector to capture undecidable trials in authorship verification. arXiv preprint arXiv:2106.15825 (2021)

[20] Boenninghoff, B., Kolossa, D., Nickel, R.M.: Self-calibrating neural-probabilistic model for authorship verification under covariate shift. In: International Conference of the Cross-Language Evaluation Forum for European Languages, pp. 145–158 (2021). Springer

[21] Potha, N., Stamatatos, E.: Dynamic ensemble selection for author verification. In: European Conference on Information Retrieval, pp. 102–115 (2019). Springer

[22] Seidman, S.: Authorship verification using the impostors method. In: CLEF 2013 Evaluation Labs and Workshop-Online Working Notes (2013)

[23] Mayor, C., Gutierrez, J., Toledo, A., Martinez, R., Ledesma, P., Fuentes, G., Meza, I.: A single author style representation for the author verification task. In: CLEF 2014 Evaluation Labs and Workshop-Online Working Notes (2014)

[24] Khonji, M., Iraqi, Y.: A slightly-modified gi-based author-verifier with lots of features (ASGALF). In: Cappellato, L., Ferro, N., Halvey, M., Kraaij, W. (eds.) Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15-18, 2014. CEUR Workshop Proceedings, vol. 1180, pp. 977–983. CEUR-WS.org, Sheffield, UK (2014). http://ceur-ws.org/Vol-1180

[25] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. ACM SIGKDD Explorations Newsletter **11**(1), 10–18 (2009)

[26] Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.K.: Improvements to platt's SMO algorithm for SVM classifier design. Neural Comput. **13**(3), 637–649 (2001)

[27] Quinlan, J.R.: Learning with continuous classes. In: Proceedings of the 5th Australian Joint Conference on Artificial Intelligence, pp. 343–348 (1992). Singapore

[28] Peñas, A., Rodrigo, A.: A simple measure to assess non-response. In: Proceedings of the 49th Annual Meeting of the ACL: Human Language Technologies, pp. 1415–1424. Association for Computational Linguistics, Portland, Oregon, USA (2011). http://www.aclweb.org/anthology/P11-1142

[29] Klaussner, C., Vogel, C.: A diachronic corpus for literary style analysis. In: Calzolari, N., Choukri, K., Cieri, C., Declerck, T., Goggi, S., Hasida, K., Isahara, H., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., Piperidis, S., Tokunaga, T. (eds.) Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018. European Language Resources Association (ELRA), Miyazaki, Japan (2018). http://www.lrec-conf.org/proceedings/lrec2018/summaries/864.html

[30] Cappellato, L., Ferro, N., Halvey, M., Kraaij, W. (eds.): Working Notes for CLEF 2014 Conference. CEUR Workshop Proceedings, vol. 1180. CEUR-WS.org, Sheffield, UK (2014). http://ceur-ws.org/Vol-1180