COMP90051 Statistical Machine Learning Project 1 Report

Alice Johnson, Marvin Lai, Matthew Farrugia-Roberts

Semester 2, 2019

Introduction

For our class project we develop a system for automated authorship attribution of Twitter messages ('tweets'). We are given a dataset of 328,932 tweets of known authorship, and 35,437 anonymous tweets to be attributed to one of 9,297 authors as part of a class Kaggle competition.[1]

Existing work on tweet authorship attribution[1,6,7] frames the problem as either (1) supervised multi-class classification, training models on labelled (known-author) tweets to predict the label (author) of test tweets, or (2) an author profiling task, collating all tweets from an author into a single profile and attributing each anonymous tweet by finding the closest profile under some distance metric. Moreover, two broad classes of features are established as effective: 'Static' features are hand-crafted features capturing various stylometric aspects of writing; and 'dynamic' features are lower level patterns automatically determined from data.

Dynamic features are ideal for their simplicity and for their robustness to our informal, non-standard and multilingual text. We explore various dynamic feature classes including byte, character, word, and flexible pattern n-grams.

Our dataset is unique in having an extremely large number of authors with few training tweets per author: Over 90% of our authors have fewer than 50 tweets, and these tweets make up over 70% of the dataset. 50 is the fewest tweets-per-author explored in existing work (to our knowledge). Multi-class classification algorithms may struggle to generalise after seeing so few examples for most classes.

After seeing initially promising results from a profile-based baseline, we elect to focus on deeply exploring profile-based methods, in the hope that these will scale more capably to our 'extreme' dataset. We explore a wide range of profile-based models from recent literature. Furthermore, we reformulate existing distance metrics to make them computationally tractable on our large dataset, and we introduce a new distance metric of our own design.

Feature classes

We explore the appropriateness of various dynamic feature classes for our authorship attribution task, including character, byte, and word n-grams, for $n = 2, 3, 4, 5, 6$.

We also explore *flexible pattern* n-grams, dynamic feature classes capturing stylometric information such as patterns in function-word use not captured by regular word n-grams[7]. Flexible patterns are word n-grams where words appearing above a certain frequency in the corpus ('high-frequency words', or HFWs) are retained, but words appearing below a certain frequency ('content words', CWs) are conflated. A flexible pattern n-gram is a sequence of n HFWs, each separated by zero or more CWs.

We optionally *pre-process* tweets, tokenising at word and punctuation boundaries and normalising infrequent tokens (e.g. dates, times) before extracting n-gram features. We compare this with extracting n-grams directly from raw text.

Learners

We explore several profile-based models for authorship attribution. Each model defines an author 'profile', and a distance metric d between these profiles and new tweets. We learn profiles for a set $\mathcal{A}$ of candidate authors from a corpus of tweets, and then predict the author of each new tweet t as $\arg\min_{a\in\mathcal{A}} d(a, t)$. The models are as follows.

Common N-Gram (CNG) The CNG model[3] defines an author's profile as the normalised frequencies of the L most common n-grams across all of the author's tweets, where L is a hyper-parameter. d_{cng} measures distance between author a and tweet t as

$$d_{cng}(a,t) = \sum_{x\in X_a\cup X_t} \left(\frac{2\cdot(P_a(x) - P_t(x))}{P_a(x) + P_t(x)}\right)^2$$

where X_a is the set of n-grams in author a's profile (i.e. their L most frequent n-grams), X_t is the set of n-grams in t, $P_a(x)$ is the normalised frequency of n-gram x in a's tweets (or 0 if $x \notin X_a$), and $P_t(x)$ is x's normalised frequency in t.

This sum over all n-grams in $X_a \cup X_t$ is expensive to compute for every author, for every test tweet. We exploit the sparsity of our n-gram features by using an equivalent[2] formulation in terms of a sum over only $X_a \cap X_t$:

$$d_{cng}(a,t) = \sum_{x\in X_a\cap X_t} \left(\frac{2\cdot(P_a(x) - P_t(x))}{P_a(x) + P_t(x)}\right)^2 - 8\cdot|X_a\cap X_t| + C$$

where $C = 4\cdot(L + |X_t|)$ is a constant. We can efficiently compute this sum (and $|X_a \cap X_t|$) using an inverted index.

[1] https://www.kaggle.com/c/whodunnit, ranking 3rd of 162 teams in public and private evaluation (team name `the_shrunken_stardust`).

[2] See notes on reformulating CNG in appendix A.1.

Source Code Author Profile (SCAP) In SCAP[2] an author's profile comprises the set of the L most common n-grams across the author's tweets. d_{scap} is then defined in terms of the overlap of this set with that of the test tweet:

$$d_{scap}(a,t) = 1 - |X_a \cap X_t|/L$$

As d_{scap} is already in terms of only $X_a \cap X_t$, we can efficiently compute it for many authors using an inverted index.

Recentered Local Profile (RLP) RLP[5] builds profiles from the L n-grams with the highest absolute 'recentered' normalised frequency, $RP_a(x) = P_a(x) - E(x)$ where $P_a(x)$ is the normalised frequency of n-gram x in a's tweets, and $E(x)$ is the normalised frequency of x in all tweets. Defining $RP_t(x)$ similarly, d_{rlp} is a cosine distance over $X_a \cup X_t$:[3]

$$d_{rlp}(a,t) = 1 - \frac{\sum_{x \in X_a \cup X_t} RP_a(x) \cdot RP_t(x)}{\sqrt{\sum_{x \in X_a \cup X_t} RP_a(x)^2 \cdot \sum_{x \in X_a \cup X_t} RP_t(x)^2}}$$

As with CNG, this calculation is prohibitively expensive at our scale. An exact formulation in terms of only $X_a \cap X_t$ is not possible, so we approXimate RLP (XRLP) instead[4]:

$$d_{xrlp}(a,t) = 1 - \frac{\sum_{x \in X_a \cap X_t} RP_a(x) \cdot P_t(x) - \sum_{x \in X_a} RP_a(x) \cdot E(x)}{\sqrt{\sum_{x \in X_a} RP_a(x)^2 \cdot \sum_{x \in X_t} RP_t(x)^2}}$$

The sum over $X_a \cap X_t$ can be computed using an inverted index, the sums over X_a are independent of t and can thus be pre-computed, and the sum over X_t is a constant.

Smooth P_a Cross Entropy (SPaCE) We present a new model for profile-based authorship attribution. SPaCE defines a profile using the *smoothed* normalised n-gram frequencies from the author's tweets, including for unseen n-grams. We interpret these normalised frequencies as a probability distribution over the set of all n-grams, and use the cross entropy between the probability distributions of t and a (plus a per-author offset capturing author prolificacy) as our distance metric:

$$d_{space}(a,t) = -\ln(P(a))/N_t - \sum_{x \in X_t} P_t(x) \ln(P'_a(x))$$

$P(a)$ is the proportion of corpus tweets by a, N_t is the total number of n-grams in t, and $P'_a(x)$ is the smoothed probability of x. $P'_a(x)$ is defined in terms of $P_a(x)$ using either (i) add-k smoothing; (ii) linear interpolation with $E(x)$ by α; or (iii) linear interpolation with $E(x)$ by $\exp(-N_a/K)$, an amount decaying exponentially with N_a, the total number of n-grams in a's tweets. K, α, and k are hyper-parameters.

While smoothing destroys the sparsity of profiles, it's still possible to efficiently compute d_{space} for many authors.[5]

[3] This is a corrected version of the formulation in [5], based on [4].
[4] See derivation of XRLP in appendix A.2, including addendum.
[5] See notes on computing SPaCE in appendix A.3.

Ensemble We create a simple ensemble in an attempt to combine multiple dynamic feature classes in a single model. We use SCAP as a base learner for its computational simplicity, and attribute tweets to the author selected by the most base models (i.e. by unweighted relative majority vote).

Experiments

In this section, we detail our experimental setup for tuning and comparatively evaluating each combination of learner and dynamic feature class, and we report our results.

Data split We are unable to effectively use our unlabelled tweets to compare the accuracy of different learner/feature class combinations, since the public leaderboard scores are derived from a small number of tweets. In response, *we create our own, larger validation dataset* by randomly partitioning our labelled dataset in two, as follows:

- **Validation data**: 69,838 tweets (20% of labelled data) reserved for final evaluation of each learner/feature class combination, to be used for final model selection.
- **Reduced training data**: Remaining 259,094 tweets (80%), to be split further for use training and tuning each learner/feature class combination.

Feature engineering Across our experiments with each learner/feature class combination, we observe (1) character n-grams are most effective for $n = 4, 5, 6$, and with raw text input; (2) byte n-grams perform indistinguishably from character n-grams; (3) word n-grams perform best for $n = 2$, where they benefit slightly from our pre-processing; and (4) flexible pattern n-grams perform best with $n = 2, 3$. Figure 1 exemplifies some of these relationships. For brevity, we report only results from learners trained on these high-n character n-grams and low-n word/flexible pattern n-grams.

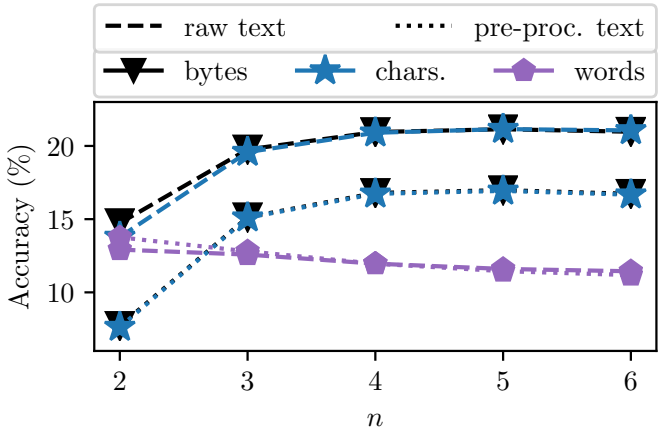

Figure 1: Effect of pre-processing and n on SCAP, fixed $L = 300$.

Hyper-parameter tuning For each learner/feature class combination, we tune our hyper-parameter using grid search optimisation on the reduced training data, as follows:

We tune the profile length L for SCAP using an 8-fold cross-validated grid search over the reduced training data.

Evaluating each configuration of CNG, and XRLP is more computationally expensive, so for these models we tune L using holdout validation rather than 8-fold cross validation (we train on 87.5% of the reduced training data, and select the L giving the highest accuracy on the other 12.5%).

SPaCE models are our most computationally expensive to evaluate, since smoothing removes the sparsity of profiles. To tune the hyper-parameter for each smoothing method (k for method i, α for method ii, and K for method iii) we perform a grid search using holdout validation on the reduced training data, evaluating on 1000 tweets (0.3%). Due to time constraints, we only tune on character n-grams.

For our ensemble, we try various combinations of features, and use holdout validation to select the best combination (seven SCAP base models using character 2–6-grams, word 2-grams, and flexible pattern 2-grams, respectively).

Model selection After tuning each learner/feature class combination as above, we re-train the tuned configurations on the entire reduced training set, and measure accuracy with our validation data. Table 1 summarises our results.

Acc. (%)		Feature class					
		character			word	flex. patt.	
Models	n	4	5	6	2	2	3
CNG		25.4	26.2	26.4	20.5	13.8	12.5
SCAP		21.7	22.2	22.1	14.4	9.7	8.5
XRLP		18.6	18.4	17.4	12.4	9.4	10.0
SPaCE i		27.5	28.3	28.0	—	—	—
SPaCE ii		**32.7**	32.2	30.9	—	—	—
SPaCE iii		32.5	31.5	30.1	—	—	—
Ensemble		—23.4—					

Table 1: Tuned model accuracy on held-out validation data.

We re-train our best performing model/feature class combination (SPaCE ii, character 4-grams) on the entire labelled dataset for submission, achieving a public score of **34.7%** accuracy, and **35.0%** accuracy on the private dataset.

Critical analysis

SPaCE We observe that SPaCE models outperform all other models for character-level n-grams. Minimising d_{space} corresponds to maximising tweet (log) likelihood assuming tweets are sequences of n-grams drawn independently from their author's n-gram probability distribution. It's somewhat surprising to see this level of performance, given the naivety of this assumption. However, SPaCE uses a much richer profile representation than other methods, with smoothing providing effective regularisation. This may help it to make finer grained distinctions between authors.

The choice of smoothing method is critical. We see additive smoothing (i) outperformed by interpolation smoothing (ii, iii). Additive smoothing corresponds to MAP estimation of profiles with a prior distribution (over n-gram distributions) concentrated about the uniform n-gram distribution, while interpolation methods take corpus-level frequency information into account. Since authors' n-gram distribution are indeed highly non-uniform, interpolation smoothing is theoretically more appropriate.

CNG, SCAP, XRLP Recent works show RLP outperforms CNG on large documents with few authors[5], and SCAP outperforms CNG when there is limited training data per author[2]. In contrast, we see CNG outperforming both SCAP and (X)RLP. XRLP may be under-performing due to our small (per author) dataset—profiles based on recentered frequencies may be unreliable when computed from noisy n-gram counts. CNG's under-performance in [2] with profiles shorter than L may be due to a flaw in the distance metric, which our reformulation implicitly overcomes (we correct for short profiles by using a constant offset term, effectively assuming all authors have at least L n-grams).

Features We see character n-grams enabling greater accuracy compared to word and flexible pattern n-grams. Our lack of data (per author) may be responsible; The same amount of text from a given author yields more character n-grams than word or flexible pattern n-grams, possibly leading to a more discriminating learned profile.

We further observe word-based models consistently outperforming flexible pattern-based models. Flexible pattern n-grams are similar to word n-grams in their number of n-grams per tweet, but they sacrifice content information about an author's text by retaining only HFWs, thereby striking a different position on a style/content information trade-off. While word n-grams are not necessarily superior in general, it seems that content is more salient in our task.

Character, word and flexible pattern n-grams are individually incomplete representations of text. Combining the features in the ensemble model must capture more information about authorship, suggesting that our feature classes are somewhat orthogonal. Future work may investigate more sophisticated methods for combining multiple dynamic feature classes into a more robust model.

References

[1] M. Bhargava, P. Mehndiratta, and K. Asawa. "Stylometric analysis for authorship attribution on twitter". In: *International Conference on Big Data Analytics*. 2013.

[2] G. Frantzeskou et al. "Effective identification of source code authors using byte-level information". In: *Proceedings of the 28th international conference on Software engineering*. 2006.

[3] V. Kešelj et al. "N-gram-based author profiles for authorship attribution". In: *Proceedings of the conference pacific association for computational linguistics, PACLING*. 2003.

[4] R. Layton. *A tutorial on Local n-grams for Authorship Attribution*. 2014. URL: https://github.com/robertlayton/authorship_tutorials/blob/master/LNGTutorial.ipynb (visited on 09/09/2019).

[5] R. Layton, P. Watters, and R. Dazeley. "Recentred local profiles for authorship attribution". In: *Natural Language Engineering* (2012).

[6] A. Rocha et al. "Authorship attribution for social media forensics". In: *IEEE Transactions on Information Forensics and Security* (2016).

[7] R. Schwartz et al. "Authorship attribution of micro-messages". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 2013.

Appendix

A.0 Inverted index computation

Consider computing a sum $s(A)$ of some function f over the elements in the intersection of two sets A and B, for many A in some family $\mathcal{A}$, with each $|A| \gg |B|$. That is,

$$s(A) = \sum_{x \in A \cap B} f(x) \qquad \text{(for each } A \in \mathcal{A})$$

The following naive algorithm (1) runs in $\mathcal{O}(|\mathcal{A}| \cdot |B|)$ time. However, if each element of B is an element of only a small number of $A \in \mathcal{A}$, many iterations of the inner loop will contribute nothing to the accumulators (because $x \notin A$).

Algorithm 1 Naive computation of $s(A)$ for $A \in \mathcal{A}$

```
s ← a mapping from sets A to empty accumulators s[A]
for A ∈ 𝒜 do
    for x ∈ B do
        if x ∈ A then
            s[A] ← s[A] + f(x)          ▷ s[A] defaults to 0
        end if
    end for
end for                                 ▷ Now, s[A] contains s(A) for each A ∈ 𝒜
```

A well-known idea from information retrieval is to use an index (or 'inverted index') to allow us to iterate over only the sets A that will contribute to an accumulator for a particular $x \in B$. Constructing this index takes as much time as the previous algorithm, but construction time is offset by faster computation of sums over many different Bs.

The idea is to precompute, for each (potential) element x, the set of those A containing x. These are the As for which x will contribute to the sum $s(A)$. Then, for each $x \in B$, we iterate through these so-called 'posting lists' instead of $\mathcal{A}$.

Algorithm 2 runs in $\mathcal{O}(\sum_{A \in \mathcal{A}} |A \cap B|)$ time (not counting constructing p), with no unnecessary iterations.

This enhancement applies to our situation. For profile-based authorship attribution methods, we must compute $\arg\min_{a \in \mathcal{A}} d(a, t)$ for family of authors $\mathcal{A}$ and tweet t, over a large number of tweets. Moreover, author profiles often contain only a small number of the many possible n-grams that occur in test tweets, and so the sets of n-grams involved are sparsely overlapping. Thus, where the distance metric contains a sum over an intersection, we can calculate this component quickly using an inverted index we compute at training time.

Algorithm 2 Indexed computation of $s(A)$ for $A \in \mathcal{A}$

```
p ← a mapping from x ∈ B to sets p[x] = {A ∈ 𝒜 | x ∈ A}
s ← a mapping from sets A to empty accumulators s[A]
for x ∈ B do
    for A ∈ p[x] do
        s[A] ← s[A] + f(x)              ▷ s[A] defaults to 0
    end for
end for                                 ▷ Now, s[A] contains s(A)
```

A.1 Reformulating CNG

The CNG method's distance metric is expressed as a sum over $X_a \cup X_t$, not an intersection. The inverted index method does not apply directly in this situation. However, using the set identities[6]

$$A \cup B = (A \cap B) + (A \cap B^c) + (A^c \cap B) \tag{1}$$

$$A \cap B^c = A - (A \cap B) \tag{2}$$

we can reformulate d_{cng} in terms of $X_a \cap X_t$. First, define $F_{a,t}(x) = \left(\frac{2 \cdot (P_a(x) - P_t(x))}{P_a(x) + P_t(x)}\right)^2$ for brevity. Then,

$$\begin{aligned} d_{cng}(a,t) &= \sum_{x \in X_a \cup X_t} \left(\frac{2 \cdot (P_a(x) - P_t(x))}{P_a(x) + P_t(x)}\right)^2 \equiv \sum_{x \in X_a \cup X_t} F_{a,t}(x) \\ &= \sum_{x \in X_a \cap X_t} F_{a,t}(x) + \sum_{x \in X_a \cap X_t^c} F_{a,t}(x) + \sum_{x \in X_a^c \cap X_t} F_{a,t}(x) \quad \text{by (1)} \end{aligned}$$

But, if $x \in X_a \cap X_t^c$ then $x \notin X_t$, so $P_t(x) = 0$, and $F_{a,t}(x)$ simplifies to $\left(\frac{2 \cdot P_a(x)}{P_a(x)}\right)^2 = 4$. Similarly, if $x \in X_a^c \cap X_t$, then $x \notin X_a$, so $P_a(x) = 0$ (in CNG, profiles include only normalised frequencies for n-grams in the top L for each author, X_a; all other frequencies are forgotten). In this case, $F_{a,t}(x) = 4$ also. So,

$$\begin{aligned} d_{cng}(a,t) &= \sum_{x \in X_a \cap X_t} F_{a,t}(x) + \sum_{x \in X_a \cap X_t^c} 4 + \sum_{x \in X_a^c \cap X_t} 4 \\ &= \sum_{x \in X_a \cap X_t} F_{a,t}(x) + \sum_{x \in X_a} 4 - \sum_{x \in X_a \cap X_t} 4 + \sum_{x \in X_t} 4 - \sum_{x \in X_a \cap X_t} 4 \\ &\qquad \text{by (2)} \\ &= \sum_{x \in X_a \cap X_t} F_{a,t}(x) + 4 \cdot (|X_a| + |X_t| - 2 \cdot |X_a \cap X_t|) \\ &= \sum_{x \in X_a \cap X_t} \left(\frac{2 \cdot (P_a(x) - P_t(x))}{P_a(x) + P_t(x)}\right)^2 - 8 \cdot |X_a \cap X_t| + C \end{aligned}$$

[6] $A + B$ is set union for mutually exclusive sets A and B. A sum over $A + B$ equals the sum over A plus the sum over B. $A - B$ is set difference for $B \subset A$. A sum over $A - B$ equals the sum over A minus the sum over B.

where $C = 4 \cdot (|X_a| + |X_t|)$. Assuming all authors use at least L n-grams throughout the training data, $|X_a| = L$, and so this term is an additive constant which will not affect $\arg\min_{a \in \mathcal{A}} d_{cng}(a, t)$. Even if an author has fewer than L n-grams in their profile, it may be preferable to treat C as constant, so as to avoid biasing d_{cng} disproportionately in favour of authors with few tweets. We can understand this as implicitly 'padding out' profiles with zero frequencies for all unseen n-grams before truncating profiles to the top L most-frequent n-grams.

We can efficienty compute the remaining terms in this formulation of d_{cng} (the sum over $X_a \cap X_t$, and $|X_a \cap X_t|$ itself) using algorithm 2.

A.2 Approximating RLP

RLP's d_{rlp} does not permit a reformulation in terms of only $X_a \cap X_t$. However, we can approximate d_{rlp} efficiently.

Computing $d_{rlp}(a, t)$, as formulated, requres computing three sums, S_1, S_2, and S_3:

$$\overbrace{\sum_{x \in X_a \cup X_t} RP_a(x) RP_t(x)}^{S_1} \quad \overbrace{\sum_{x \in X_a \cup X_t} RP_a(x)^2}^{S_2} \quad \overbrace{\sum_{x \in X_a \cup X_t} RP_t(x)^2}^{S_3}$$

Using identity (1), we can re-write S_1:

$$\begin{aligned} S_1 &= \sum_{x \in X_a \cup X_t} RP_a(x) \cdot RP_t(x) \\ &= \sum_{x \in X_a \cap X_t} RP_a(x) \cdot RP_t(x) + \sum_{x \in X_a \cap X_t^c} RP_a(x) \cdot RP_t(x) \\ &\qquad + \sum_{x \in X_a^c \cap X_t} RP_a(x) \cdot RP_t(x) \end{aligned}$$

If $x \in X_a \cap X_t^c$, then $x \notin X_t$. In that case, $P_t(x) = 0$, and so $RP_t(x) = P_t(x) - E(x) = -E(x)$. Meanwhile, if $x \in X_a^c \cap C_t$, then $x \notin X_a$. This does not mean that $P_a(x) = 0$, but, for large L, it's likely that $RP_a(x) \approx 0$ (recentered normalised frequencies significantly far from 0 are likely to be in X_a, by definition). Thus we can (approximately) simplify S_1 to:

$$\begin{aligned} S_1 &\approx \sum_{x \in X_a \cap X_t} RP_a(x) \cdot RP_t(x) + \sum_{x \in X_a \cap X_t^c} RP_a(x) \cdot (-E(x)) \\ &\qquad + \sum_{x \in X_a^c \cap X_t} 0 \cdot RP_t(x) \\ &= \sum_{x \in X_a \cap X_t} RP_a(x) \cdot RP_t(x) - \sum_{x \in X_a \cap X_t^c} RP_a(x) \cdot E(x) + 0 \\ &= \sum_{x \in X_a \cap X_t} RP_a(x) \cdot RP_t(x) + \sum_{x \in X_a \cap X_t} RP_a(x) \cdot E(x) \\ &\qquad - \sum_{x \in X_a} RP_a(x) \cdot E(x) \qquad \text{by (2)} \\ &= \sum_{x \in X_a \cap X_t} RP_a(x) \cdot (RP_t(x) + E(x)) - \sum_{x \in X_a} RP_a(x) \cdot E(x) \\ &= \sum_{x \in X_a \cap X_t} RP_a(x) \cdot P_t(x) - \sum_{x \in X_a} RP_a(x) \cdot E(x) \end{aligned}$$

For S_2 and S_3, note the following identity for sets:

$$A \cup B = A + (A^c \cap B) \tag{3}$$

Using (3), and the same kinds of simplifications as for S_1, we can rewrite S_2 (approximately) as

$$\begin{aligned} S_2 &= \sum_{x \in X_a \cup X_t} RP_a(x)^2 \\ &= \sum_{x \in X_a} RP_a(x)^2 + \sum_{x \in X_a^c \cap X_t} RP_a(x)^2 \qquad \text{by (3)} \\ &\approx \sum_{x \in X_a} RP_a(x)^2 + \sum_{x \in X_a^c \cap X_t} 0^2 \\ &= \sum_{x \in X_a} RP_a(x)^2 \end{aligned}$$

and S_3 (approximately) as

$$\begin{aligned} S_3 &= \sum_{x \in X_a \cup X_t} RP_t(x)^2 \\ &= \sum_{x \in X_t} RP_t(x)^2 + \sum_{x \in X_a \cap X_t^c} RP_t(x)^2 \qquad \text{by (3)} \\ &= \sum_{x \in X_t} RP_t(x)^2 + \sum_{x \in X_a \cap X_t^c} (0 - E(x))^2 \\ &= \sum_{x \in X_t} RP_t(x)^2 + \sum_{x \in X_a \cap X_t^c} E(x)^2 \\ &\approx \sum_{x \in X_t} RP_t(x)^2 + \sum_{x \in X_a \cap X_t^c} 0^2 \qquad (*) \\ &= \sum_{x \in X_t} RP_t(x)^2 \end{aligned}$$

where in step $(*)$ we note that n-grams with high $E(x)$ are likely to appear in any given tweet, so $E(x)$ should be small for any $x \notin X_t$. Thus, we define

$$d_{xrlp}(a, t) = 1 - \frac{\sum_{x \in X_a \cap X_t} RP_a(x) \cdot P_t(x) - \sum_{x \in X_a} RP_a(x) \cdot E(x)}{\sqrt{\sum_{x \in X_a} RP_a(x)^2 \cdot \sum_{x \in X_t} RP_t(x)^2}}$$

All components of $d_{xrlp}(a, t)$ are either independent of a or t (and can thus be precomputed) or are a sum over $X_a \cap X_t$ (and can thus be computed efficiently using algorithm 2).

Addendum Returning to step $(*)$, we note that an efficiently computable *and exact* formulation of S_3 is:

$$\begin{aligned} S_3 &= \sum_{x \in X_t} RP_t(x)^2 + \sum_{x \in X_a \cap X_t^c} E(x)^2 \\ &= \sum_{x \in X_t} RP_t(x)^2 + \sum_{x \in X_a} E(x)^2 - \sum_{x \in X_a \cap X_t} E(x)^2 \qquad \text{by (2)} \end{aligned}$$

This leads to a slightly improved version of XRLP. Our experiments reported above *do not* include this enhancement.

A.3 Computing with smoothed distributions

The profile smoothing we employ in the SPaCE method destroys profile sparsity, by design. However, depending on the choice of smoothing method, d_{space} may still permit an efficient reformulation. For all three methods explored in this report, this is the case.

To see why, first consider defining X_a for SPaCE profiles to be the set of all n-grams with non-zero normalised frequencies. If we can express d_{space} in terms of sums over $X_a \cap X_t$, we might hope to compute it efficiently using algorithm 2, as for CNG, SCAP and XRLP—even without truncating profiles to L, many n-grams will be unused by many authors. The seeming difficulty arises because while $P_a(x) = 0$ for $x \notin X_a$, $P'_a(x) \neq 0$ due to smoothing. However, if $P'_a(x)$ is some simple expression of a and x when $x \notin X_a$ (namely an expression independent of either a or x, or factorising into a product of such expressions) then we may still reformulate d_{space} into an efficient form. For many smoothing methods, this will be the case, since smoothing gives each unseen n-gram some simple 'default' probability. Defaults cannot be based on the normalised frequency of the n-gram in the author's tweets, because the latter is zero.

Define $D_a(x)$ to represent this 'default' probabability for a given smoothing method. That is, $D_a(x) = P'_a(x)|_{P_a(x)=0}$. Importantly, we must distinguish $D_a(x)$ and $P'_a(x)$ when, for a particular a and x, $x \in X_a$. In that case, $D_a(x) \neq P'_a(x)$, because $D_a(x)$ represents the value that $P'_a(x)$ *would have* if we had seen some other n-gram every time we saw x in the training tweets. For our smoothing methods:

$$P'_a(x) = \frac{\text{Count}_a(x) + k}{N_a + k|X|} \qquad \to D_a(x) = \frac{k}{N_a + k|X|}$$
(additive smoothing)

$$P'_a(x) = (1-\alpha)P_a(x) + \alpha E(x) \qquad \to D_a(x) = \alpha E(x)$$
(interpolation smoothing)

$$P'_a(x) = (1-\alpha_a)P_a(x) + \alpha_a E(x) \quad \to D_a(x) = \alpha_a E(x)$$
$$\alpha_a = \exp\left(-\tfrac{N_a}{K}\right) \qquad \text{(decaying interpolation smoothing)}$$

where $\text{Count}_a(x)$ is the number of times x occurs in a's tweets; N_a is the total number of n-grams in a's tweets; X is the set of all n-grams used by all authors, so $|X|$ is the number of distinct n-grams; $E(x)$ is the normalised frequency of n-gram x over all authors' tweets; and k, α and K are our hyperparameters.

Now, noting one final set identity

$$A = (A \cap B) + (A \cap B^c) \tag{4}$$

and focusing on the sum over X_t in the original formula, we may reformulate $d_{space}(a,t)$ as follows:

$$\begin{aligned}
-d_{space}(a,t) - \ln(P(a))/N_t &= \sum_{x \in X_t} P_t(x) \ln P'_a(x) \\
&= \sum_{x \in X_t \cap X_a} P_t(x) \ln P'_a(x) + \sum_{x \in X_t \cap X_a^c} P_t(x) \ln P'_a(x) \quad \text{by (4)} \\
&= \sum_{x \in X_t \cap X_a} P_t(x) \ln P'_a(x) + \sum_{x \in X_t \cap X_a^c} P_t(x) \ln D_a(x) \\
&= \sum_{x \in X_t \cap X_a} P_t(x) \ln P'_a(x) + \sum_{x \in X_t} P_t(x) \ln D_a(x) \\
&\qquad - \sum_{x \in X_t \cap X_a} P_t(x) \ln D_a(x) \quad \text{by (2)} \\
&= \sum_{x \in X_t \cap X_a} P_t(x) \ln \frac{P'_a(x)}{D_a(x)} + \sum_{x \in X_t} P_t(x) \ln D_a(x)
\end{aligned}$$

The first component of the above sum can be efficiently computed using algorithm 2. The second may be efficiently computable, depending on the smoothing method.

For each of our smoothing methods, this sum may either be precomputed, or is independent of a (and therefore unnecessary in computing $\arg\min_{a \in \mathcal{A}} d_{space}(a,t)$):

i. For additive smoothing, $D_a(x)$ is independent of x, so

$$\sum_{x \in X_t} P_t(x) \ln D_a(x) = \ln\left(\frac{k}{N_a + k|X|}\right) \cdot \sum_{x \in X_t} P_t(x)$$

But $\sum_{x \in X_t} P_t(x) = 1$, so the whole sum simplifies to

$$\ln\left(\frac{k}{N_a + k|X|}\right)$$

which can be precomputed per-author at training time.

ii. For interpolation smoothing, $D_a(x)$ is independent of a, so the entire sum is independent of a and can be dropped as an additive constant.

$$\sum_{x \in X_t} P_t(x) \ln D_a(x) = \sum_{x \in X_t} P_t(x) \ln(\alpha E(x))$$

iii. For exponentially decaying interpolation smoothing, $D_a(x) = \alpha_a E(x)$ is independent of neither a nor x. However, it is the product of a factor independent of x (α_a) and a factor independent of a ($E(x)$). Thus,

$$\begin{aligned}
\sum_{x \in X_t} P_t(x) \ln D_a(x) &= \sum_{x \in X_t} P_t(x) \ln(\alpha_a E(x)) \\
&= \sum_{x \in X_t} P_t(x) \ln \alpha_a + \sum_{x \in X_t} P_t(x) \ln E(x) \\
&= \ln \alpha_a \sum_{x \in X_t} P_t(x) + \sum_{x \in X_t} P_t(x) \ln E(x)
\end{aligned}$$

Again, $\sum_{x \in X_t} P_t(x) = 1$, and the second sum can be dropped for optimisation. Thus, the whole sum is equivalent to

$$\ln \alpha_a = \ln \exp\left(-\frac{N_a}{K}\right) = -\frac{N_a}{K}$$

Using these simplifications, it's possible to efficiently compute the minimum $d_{space}(a,t)$ for many authors, despite smoothing destroying the sparsity of profiles.