How the Word Adjacency Network (WAN) works

Paul Brown
School of Humanities, De Montfort University, UK

Mark Eisen
Intel Labs, Philadelphia, USA

Santiago Segarra
Computer Science and Electrical & Computer Engineering, Rice University, USA

Alejandro Ribeiro
Department of Electrical and Systems Engineering, University of Pennsylvania, USA

Gabriel Egan
School of Humanities, De Montfort University, UK

Abstract

The Word Adjacency Network (WAN) method is a newly invented method for attributing the authorship of texts based on internal evidence about the proximities of high-frequency words in those texts. The method has been the subject of mathematically rigorous explanations in scientific journals and here is offered instead a relatively non-technical account for humanist readers unfamiliar with advanced mathematical nomenclature. The description will refer to the operation of the algorithm as it is implemented in a newly completed open-source, open-access version that the authors have made available for free unlimited download on the WorldWide Web.

Correspondence:
Gabriel Egan, School of Humanities, De Montfort University, UK.
E-mail:
mail@gabrielegan.com

This essay will describe the algorithm underlying a new method of computational authorship attribution, the Word Adjacency Network (WAN) method, and introduce an Open Source implementation of this algorithm as a Python script freely available for any investigator to experiment with. A mathematically rigorous description of the WAN method and accounts of its application to various problems in authorship attribution has been provided in a series of previous publications (Segarra *et al.*, 2015; Segarra *et al.*, 2016; Eisen *et al.*, 2018), and the specific purpose here is to offer a relatively non-technical account of the underlying algorithm to help dispel misconceptions about it and encourage its use by other investigators.

At the heart of the WAN method is a consideration of language in terms of word frequencies. In English, 'the' is the most frequently used word at about one in sixteen of all words, and the word 'and' comes second at about one in thirty words, and 'to' comes third at

Digital Scholarship in the Humanities, Vol. 37. No. 2, 2022.
doi:10.1093/llc/fqab002 Advance Access published on 8 October 2021

about one in forty-two words. (If we subsume under one heading all the various ways of conjugating 'to be'—as 'was', 'am', 'are', 'is', and so on—then it would take second place in this rank order.) These three words will be used to begin our illustration of the WAN method and then we will switch to the four words 'and', 'in', 'one', and 'with' that conveniently cluster within an excerpt from a Shakespeare play, enabling us to develop our illustration to consider some of the associated complexities.

The most-frequent words in modern English are the so-called function words that serve grammatical and structural purposes that in other languages (and in older forms of English) are more often served by inflection. The 100 most-common words in English, most of them function words, comprise around half of all that is spoken and written. Although all English writers rely heavily on this small set of most-common words, writers vary one from another in how often they use each one. The preferences for favouring certain words and avoiding others enable us to tell one person's writing from another's solely from this internal evidence. That is, because the particular preferences are different for each writer, we can use them for authorship attribution in cases where other evidence is not available or is contested.

The foundational work in this field, based on manual counting of words, was Frederick Mosteller and David L. Wallace's demonstration that the securely attributed writings of the American Founding Fathers James Madison and Alexander Hamilton show distinguishable and consistent preferences for using or avoiding thirty function words and that these preferences may be analysed to attribute anonymous works of which either man might be the author (Mosteller and Wallace, 1963). Subsequent studies enabled by automated counting have repeatedly shown that function-word frequencies can reliably distinguish authorship, and the success rate of this approach can be precisely calculated by applying it to works for which the authorship is already securely established from external evidence and seeing how many correct attributions the approach is capable of 'predicting'.

Here and throughout this essay, we will put 'prediction' and its cognate words in 'scare' quotation marks to signal special usage, as we do not mean them in the everyday sense of forecasting a future event but rather in the scientific sense of reaching a conclusion, derived from a process of calculation, that can be checked against reality in order to judge the validity of the process leading to that conclusion. Scientific models are frequently judged on their accuracy in such 'predictions'. Successful 'predictions' in over 90% of cases are now common in authorship attribution using function words. David L. Hoover's systematic evaluation showed that John Burrows's 'Delta' method of authorship attribution using function words and other high-frequency words is exceptionally accurate and relatively indifferent to changes in language use over time and differences in genre, and Shlomo Engelson Argamon's analysis confirmed the utility of function words in authorship attribution (Hoover, 2004; Argamon, 2018). Function-word frequency has been used to identify authorship in texts as varied as the Latin 'Consolatio' attributed to Marcus Tullius Cicero (Forsyth *et al.*, 1999), the 'Book of Mormon' (Jockers *et al.*, 2008), and the anonymized judgements of the US Supreme Court (Jockers *et al.*, 2019). The reader may multiply these examples many times by browsing the back issues of this and similar journals.

Aside from their mere frequencies of use, do function word occurrences contain useful evidence that has hitherto been neglected? One obvious place to look is in the ways that function words follow one another across a text. Alexis Antonia, Hugh Craig, and Jack Elliott used this approach by analysing what they called skip N-grams, meaning they would 'find the first instance of one of the listed words, then move to the next of them, ignoring any intervening unlisted word' (Antonia *et al.*, 2014, p. 151). This approach captures the attribute of successiveness—that one particular word is likely to be followed by another word—but in skipping over the intervening words, it ignores the ways in which certain words cluster together or are spaced out across a text. Can we capture numerically the authorial preferences for putting certain words near other words and for keeping other pairings apart? As half of all writing consists of repetitions of the 100 or so most-common words, recording the distance, measured in intervening words, from

each function word to every other function word is technically possible but generates large amounts of data. There are, depending on the edition used, around 36,000-word tokens in *Hamlet*, of which around 18,000 tokens will belong to these 100 most-common types. (We use 'tokens' here in the sense of total occurrences including repetitions, so that Hamlet's speech 'Words, words, words' is three 'tokens' but only one word 'type'.) Recording the distances between each pair of 18,000 tokens would generate around 320 million data points, and for whole authorial canons there would be billions of data points.

If we care only about the general tendencies regarding the proximities of word types, we need not count the billions of distances between every pair of tokens. We want to know how often the word 'the' tends to appear near the word 'and', irrespective of the locations of each instance. The significance of the distance falls off sharply as the distance rises, as we would be highly interested in occasions when 'the' falls within a few tokens of 'and' but scarcely interested at all when these words are 50 or 100 tokens apart. Whether their proximities are consciously or unconsciously controlled by writers, no one supposes that such distant pairings are shaped by the creative mind. Loosely, the close proximities that we are interested in can be considered as collocations in the sense that the two words appear together but not necessarily next to each other, as they are in bigrams. But our proximities are not collocations in the stricter sense familiar to corpus linguists, which requires that the words appear together more frequently than we would expect by chance. As in the related notion of keyness, the measurement of probability in the corpus linguistics notion of collocation serves to isolate exceptional usages, whereas the WAN method is concerned with unexceptional word usages that are nonetheless measurable and demonstrably distinctive of authorship.

For our purposes, there exists a window of interest around each function word and we want to know how often two function words fall within the same window. Because of the way computers are constructed, it is natural for software to process a text by moving through it one word at a time from beginning to end. Considering just our top-three most-frequent words in English ('the', 'and', and 'to'), we may simply project our window forward from the location of each word that we consider in turn, like this:

> where the body is *and* [go with us *to the*] king (*Hamlet* 4.2.24–25).

We show the words without punctuation and reduced to lower case because, for our purposes, these features are irrelevant. In this example, 'and' is the function word we have alighted upon and the square brackets represent a five-token window projected forward from it. This forward-projection captures the fact of one word appearing near another without privileging the order of appearance since, as will be seen when we show how the data are stored, the method by which we record that in this example 'to' follows 'and' allows the same phenomenon to be read as 'and' preceding 'to'. Moreover, this forward-projection prevents us from double-counting, since when we find an occurrence of 'and' followed four tokens later by an occurrence of 'to' we want to record this as one occurrence of these two words in close proximity and not count it a second time by subsequently alighting on 'to' and recording that 'and' appears four tokens earlier. For now, let us assume that our window is five tokens wide, although the software implementation of the method will allow us to vary this number at will.

A further sophistication is to distinguish how far along our five-word window the proximate function words appear. In the example above, the words in the window's leftmost three positions (Positions 1, 2, and 3) are 'go', 'with', and 'us', none of which is one of the three function words we are interested in, but in Position 4 we find 'to' and in Position 5 we find 'the'. The first of these, 'to', is nearer to 'and' than the second, 'the', and these relative proximities should be differently scored in our method. We can achieve this by weighting the window positions so that a match in Position 1 counts more than a match in Position 2, which counts more than a match in Position 3, and so on. For reasons that we will explain shortly, the WAN method uses the descending scale of weights 1.0, 0.75, 0.56, 0.42, and 0.32 for Positions 1 through 5.

Since we are interested in the authorial tendency to place each of our function words near to each of the

Table 1. A matrix representing the proximities of the function words 'the', 'and', and 'to'

↓from to→	the	and	to
the	0.18	0.54	0.28
and	0.51	0.36	0.13
to	0.29	0.55	0.16

others, the obvious data structure to hold this information is a table of rows and columns, called a matrix in mathematics and computer science. In our matrix, the rows and columns are labelled with the function words and each row reports upon the frequency with which we found each of the function words in the five-word windows projected forward from the function word given in the heading for that row, as shown in Table 1.

In Table 1, the scores have been normalized so that each row adds up to the number 1. Thus, the first row, for the word 'the', can be vocalized as 'After "the", the preferences for following it with each of the three words of interest are 18% for "the", 54% for "and", and 28% for "to"'. This does not mean that more than half the times that 'the' was used in the text the word 'and' was found within five tokens after it, as we omit occasions when none of the three words appeared shortly after 'the'. Rather, the numbers are relative preferences: they tell us how much one of the words is preferred over the other two on those occasions when any of them fell within our five-word window. Moreover, the numbers are weighted scores in which occurrence in the window's Position 1 counted more than occurrence in Position 2, which counted more than occurrence in Position 3, and so on.

Table 1 is a way of representing what is called a Markov chain, which we can also visualize using what is called a state-transition diagram, as in Fig. 1. In the state-transition diagram version, the three words of interest are represented by ovals (known as nodes) and the authorial habit of following a word with one of the others is represented as an arrow-headed line (known as an edge) from the first word to the second, which is annotated with the score (known as the weight) representing how often this second word follows the first. Table 1 and Fig. 1 contain the same

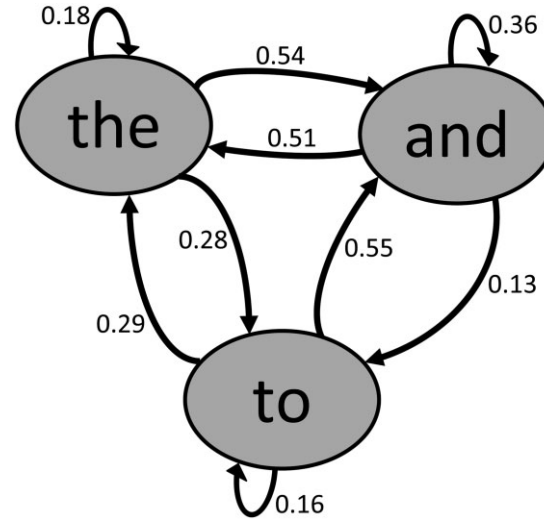

Fig. 1 A state-transition diagram representing the Markov chain shown in Table 1.

information represented in different ways, but for manipulation inside a computer the form given in Table 1 is more convenient because computer memory cells are easily arranged in such a tabular form.

We have been referring to the words whose occurrences in a text we want to count as function words, but more neutrally, we may call them words of interest, as the algorithm that counts them merely compares strings of characters and takes no account of their grammatical significance. The algorithm for creating our matrix of the kind shown in Table 1 may now be stated as follows:

(1) create a square matrix in which each row represents one of the words of interest and each column represents one of the words of interest and fill it with zeroes;
(2) move through the text one word at a time, comparing each word with the list of words of interest;
(3) in the event of a match in this comparison, open a window after the word of interest that takes in the next five words of the text;
(4) move through the five-word window one word at a time, comparing each word with the list of words of interest;
(5) in the event of a match in the window, raise the value of the corresponding cell in the matrix at

the row representing the word found in the text and at the column representing the word found in the five-token window. The amount to raise the cell by is 1 for Position 1 in the window, 0.75 for Position 2, 0.56 for Position 3, 0.42 for Position 4, or 0.32 for Position 5; and

(6) when the entire text has been processed, normalize the matrix by dividing each cell's value by the sum of the values in that cell's row, unless that row's sum is zero (in which case do nothing).

The resulting matrix is the Word Adjacency Network that gives the WAN method its name.

A particular modification to this algorithm might be advantageous. For instance, when studying plays we might want to consider each character's speeches separately and so discount a word near the end of one speech matching with a word near the beginning of the next, as would be the case with our example '. . . go with us *to the* king. | HAMLET *the* body is with the king' that begins in one speech and ends in another. To discount such matches we would modify Step 3 above to '. . . open a window taking in the next five words of the text, or as many as are left in the current speech'. In applying the WAN method to texts other than plays, such as novels, the same kind of modification would allow us to treat sentences or paragraphs as the units of construction and disallow matches across a sentence or paragraph boundary. This option is available in the Open Source Python script that accompanies this essay. For technical completeness in the case of our algorithm, we should also add to Step 3 a rule about shortening the window commensurately as we come within the last five words of the text.

Each row in a WAN is what is known in mathematics as a probability distribution, since by embodying the actual preferences shown by a text for following a word of interest by each of the other words of interest a row can be treated as a 'prediction' of what another text by the same author will do regarding the same proximities. These 'predictions' will be reliable, of course, only if the preferences we are concerned with are stable authorial traits. If each author varies substantially in these habits from work to work, or across time, or when writing in different genres, then the 'predictions' will be inaccurate. That the traits are stable is not something we should assume about authorship. Rather we may test this empirically by creating WANs for different authors' works, and sampling across time and across genres the accuracies of the 'predictions' they enable us to make.

We need then to be able to compare one WAN with another and measure their difference. The accepted mathematical means for comparing two probability distributions is the measurement called Kullback–Leibler divergence (Kullback and Leibler, 1951). Since a WAN is a series of probability distributions—one for each word of interest, each row—we can measure the Kullback–Leibler divergence between each row in one WAN and the corresponding row in another and sum the divergences for the entire series of rows. (As will be explained shortly, we weight each Kullback–Leibler divergence by what is called the Limit Probability of the word for which the row is the probability distribution.) The resulting sum is known as the relative entropy between the two WANs. The notion of entropy as a feature of language was invented by the founder of Information Theory, Claude Shannon, and its full explanation is beyond the scope of this essay; an explanation appears in Hugh Craig and Brett Greatley-Hirsch's *Style, Computers, and Early Modern Drama* (Craig and Greatley-Hirsch, 2017, pp. 48–49). The first consideration is that two WANs can meaningfully be compared only if they represent the proximities for the same set of words of interest as they are found in the two texts. That is, the labels for the rows and columns as shown in Table 1 must be the same for the two WANs: we have to be predicting the proximities of the same words of interest.

The notion of relative entropy draws on the idea that, as a series of probability distributions, a WAN 'predicts' how often we should expect to find in another text—one not used to make the WAN—the authorial habit of certain words following other words in close proximity. Given two WANs representing the habits found in two texts, the relative entropy between them is an expression of how accurate would be the 'predictions' about habits of word placement made from one when used to anticipate the actual habits found in the other. Since it concerns 'predictions' about one text based on the WAN for the other, this relative entropy calculation will necessarily come out differently when calculated in the opposite direction, using the second WAN to make 'predictions' about

the first text. Which direction to use in a particular application is a matter for the investigator(s) to decide.

In our published applications of the WAN algorithm to problems of authorship attribution, we consistently generated a WAN for a suspect text—typically, a play whose authorship we wanted to attribute—and a WAN for an entire set of sole-authored well-attributed plays by a candidate author. We then calculated the relative entropy, represented by the letter *H*, that expresses how far the WAN for the suspect text diverges from the expectations represented by the WAN for the candidate author's profile, which calculation we may represent symbolically as *H*(*text*, *profile*). The smaller the value of *H*, the more that the text is like the profile in its placement of the words of interest in proximity to one another. If there exists a set of candidates for the authorship of text, it is reasonable to generate a profile for each, based on his combined body of sole-authored well-attributed works, and calculate *H*(*text*, *profile*) for each candidate's profile. The candidate for whose profile the resulting *H* has the lowest value is the likeliest of those candidates to be the author of the text.

Before proceeding to the calculation for relative entropy, we should explain the sequence of weightings—1.0, 0.75, 0.56, 0.42, and 0.32—used for the five positions in the window opened up after each word of interest that is found in the text. These are approximate values for a single constant, 0.75, raised in turn to the power of 0, 1, 2, 3, and 4. That is, 1.0 is 0.75 to the power of 0, 0.75 is 0.75 to the power of 1, 0.56 is 0.75 squared, 0.42 is 0.75 cubed, and 0.32 is 0.75 to the power of 4. (For ease of explanation, these decimal fractions are shown here only to their first two decimal places; the actual calculation uses the default precision of the computer language employed, typically at least fifteen decimal places.) This series of numbers is a descending power scale and it reflects our decreasing interest in matches as the matched word moves further to the right in our window and hence further away from the word in our text that caused us to open the window. The starting value 0.75, the 'decay parameter', was derived experimentally as the one yielding the greatest accuracy in application of the algorithm to questions of authorship attribution and it may easily be varied (Segarra *et al.*, 2015, p. 5469).

A final complexity to be considered before turning to the calculation of the relative entropy between two WANs is the notion of Limit Probabilities. It is a substantial complication and we will take several pages to explain it. We started by noting that 'the' is the most common word in English, and that 'and' and 'to' are the second and third most common in this rank order. If we apply our method to the 100 most common words in English the words at the bottom end of this rank order will be used much less often—about 100th as often according to the principle known as Zipf's law—than the words at the top of the rank order. Although the phenomenon is most easily demonstrated with function words, any set of words of interest will likely appear at differing frequencies in any two texts. We want the words used more often to matter more in our calculations than the words used less often, but rather than just adopting the rank order and frequencies in which our words appear in English we want to adopt their rank order and frequencies in the actual texts under consideration. That is, we want our weighting of how much importance we attach to 'the', 'and', and 'to' in Fig. 1 to arise from the actual uses of 'the', 'and', and 'to' in the text under examination and hence in the Markov chain representing it. The calculation of Kullback–Leibler divergence requires us to calculate this weighted importance, which is called the Limit Probability, for each word of interest.

To picture how we achieve this, imagine a person taking a series of hops from each node, each word, in Fig. 1 to one of the other nodes/words, and making the decision of which node/word to hop to next using the probabilities given by the edges in the picture. That is, of all the times the hopper lands on 'the' she next hops to 'and' 54% of the time, she next hops to 'to' 28% of the time, and she hops off 'the' only to land back on 'the' 18% of the time. She records how long she spends on each node/word as a proportion of all the hops made. After a great many hops, these three proportions will converge on three numbers, one for each node/word, and will not substantially change as she makes further hops. These three numbers are the Limit Probabilities for 'the', 'and', and 'to' and are used in the relative entropy calculation.

The calculation of Limit Probabilities is best visualized using the representation in Fig. 1, but is

Table 2. Multiplication of a three-row, three-column matrix by itself

a	b	c		a	b	c		a' = a × a + b × d + c × g	b' = a × b + b × e + c × h	c' = a × c + b × f + c × i
d	e	f	×	d	e	f	=	d' = d × a + e × d + f × g	e' = d × b + e × e + f × h	f' = d × c + e × f + f × i
g	h	i		g	h	i		g' = g × a + h × d + i × g	h' = g × b + h × e + i × h	i' = g × c + h × f + i × i

computationally more easily achieved using the matrix in Table 1. In this form, the equivalent to taking each hop is multiplying the matrix by itself and the simulation of many hops is achieved by repeating this multiplication many times. Another way to express this is to say that we raise the matrix to a certain power. When multiplying any two matrices, each cell in the resulting matrix contains the sum of the products of the cells in the same row in the first matrix multiplied by the cells in the same column in the second matrix. Table 2 visualizes this process.

In Table 2, the nine cells in the three-by-three matrix that is to be multiplied by itself are labelled *a* through *i* and the multiplications and additions required to produce the resulting matrix are shown in terms of these labels, with the resulting cells labelled *a'* (vocalized as '*a* prime') through *i'* (vocalized as '*i* prime'). For example, the cell *h'* is in the bottom row and the middle column of the resulting matrix, and as can be seen the numbers to be multiplied and summed to produce *h'* are drawn from the bottom row of the first matrix (cells *g*, *h*, and *i*) and the middle column of the second matrix (cells *b*, *e*, and *h*). This method of combining rows and columns is replicated throughout the multiplication.

When the matrix multiplication is repeated many times—when we raise the matrix to a large power—the values converge so that every row in the resulting matrix contains the same set of values. By 'converge', we mean that when the matrix is again multiplied by itself the numbers in it do not appreciably change. Using the labelling in Table 2, this means that in the resulting matrix the same number appears in each of cells *a'*, *d'*, and *g'*, and another number is the same in cells *b'*, *e'*, and *h'*, and another number is the same in cells *c'*, *f'*, and *i'*. For our example expressed in Table 1, the number in cells *a'*, *d'*, and *g'* is the Limit Probability for the word 'the', the number in cells *b'*, *e'*, and *h'* is the Limit Probability for the word 'and', and the number in cells *c'*, *f'*, and *i'* is the Limit Probability for the word 'to'.

Table 3. A matrix representing the normalized WAN for the proximities of 'and', 'in', 'one', and 'with' in Hamlet lines 1.2.11–13

↓from to→	and	in	one	with
and	0.00	0.27	0.30	0.43
in	0.40	0.40	0.00	0.20
one	0.34	0.14	0.26	0.26
with	0.21	0.41	0.31	0.07

Let us take a second concrete example using a small play extract we have explored previously (Segarra *et al.*, 2016, pp. 235–36). This is how lines 1.2.11 – 13 of *Hamlet* look once we have removed capitalization, lineation, and punctuation so that they are a simple string of tokens:

> *with one* auspicious *and one* dropping eye *with* mirth *in* funeral *and with* dirge *in* marriage *in* equal scale weighing delight *and* dole

We have here italicized for clarity all the occurrences of the four function words 'and', 'in', 'one', and 'with', which in this extract repeatedly occur within five words of each other. This is of course why we chose those function words for this illustration: even in this tiny textual sample they occur sufficiently often to make a usefully illuminating WAN.

After running the above six steps of our algorithm for this small extract from *Hamlet* and these four function words (our words of interest), we arrive at the normalized WAN shown in Table 3.

As before, each row sums to 1 because it represents for each word of interest—'and' in the case of the first row, 'in' in the second row, 'one' in the third row, and 'with' in the fourth row—the frequencies at which that word is followed within five words by each of the words of interest (including itself), if it is followed by any of them. That last qualification is necessary because, as we noted earlier, the matrix represents the weighted-by-proximity answer to the question 'if

Table 4. A matrix representing the Limit Probabilities for the WAN in Table 3

	and	in	one	with
and	0.24	0.32	0.20	0.24
in	0.24	0.32	0.20	0.24
one	0.24	0.32	0.20	0.24
with	0.24	0.32	0.20	0.24

it is followed by any of the four, how often is it followed by "and", how often by "in", and so on?'; we are not recording cases where it is followed by none of them.

To produce our Limit Probabilities, we multiply the matrix in Table 3 by itself 100 times to produce the matrix shown in Table 4, where we have removed the labels 'from' and 'to' because the matrix no longer represents the 'from'/'to' relationships and instead contains the Limit Probabilities for each of the four words, indicating how often our hypothetical node-hopper would spend on each node in our Markov chain on account of that node having the number and strengths of incoming edges it has. These Limit Probabilities are used in our calculation of the relative entropy between two WANs because they reflect the different significances each word has in our WAN.

We see in Table 4 that in every column the number in each cell is the same as the number in the other cells in that column and that the rows still sum to 1. If we had examined the results of the matrix multiplication after each of its 100 iterations, we would have seen these numbers gradually converge across the iterations, and indeed they reached convergence to two decimal places long before the 100 multiplications were complete. In this case, 100 iterations were enough for the numbers to converge down as far as the fifteenth decimal place although for clarity we here represent only the first two decimal places. The significance of the convergence of these numbers is that they indicate that no matter which node our node-hopper starts on—whether it is 'and', 'in', 'one', or 'with' in the leftmost column—she will still, after a large number of hops, have spent the same proportion of time on each node as she would if she had started elsewhere. The time spent on the different nodes depends on the weights of the edges between them, not on which one she happens to start on.

Do all WANs necessarily have this quality that after a large number of hops the proportions of time spent on the different nodes will converge on a stable series of numbers that cease changing as we continue hopping? Might not a WAN have features that upset this process? In fact, at least two problems can emerge in principle and need to be prepared for even though they are rarely found in real-world language samples. Suppose that in Table 3, the row for the word 'and' was full of zeroes, representing a text in which 'and' was never followed within five words by 'and', or 'in', or 'one', or 'with'. This would be equivalent to a Markov chain in which the node 'and' has no outgoing edges but still has incoming edges as other words are still followed by 'and'. For our node-hopper, the node 'and' would represent a trap: once she had landed there, she could never escape as no more hops to other nodes would be possible.

Over the long term, any Markov chain containing such a trap will eventually capture the node-hopper and keep her there for the rest of eternity so that the proportion of time spent on the trapping node will creep ever closer to 100%. This trap is known as an 'absorbing state' and it makes impossible the useful interpretation of our Markov chain. The solution to this problem is to say that the run of zeroes in the row for 'and' represents our text's equal indifference to which word follows 'and'. The text is just as likely to follow 'and' with 'and' as it is to follow 'and' with 'in' or 'one' or 'with', since in each case that likelihood is equally zero. Since we care about the expressed preferences for following one word with another, this absence of a preference is just as well represented by putting the same number, $^1/_4$, in each cell in that row. This $^1/_4$ comes from the fact that there are four words under consideration and since the text is equally likely (because not at all likely) to follow 'and' with one of them we can divide certainty (the probability represented by 1) four equal ways. So, in our algorithm we need to add the proviso that in the event of a row being all zeroes we fill each cell with 1 divided by the number of words of interest we are using, in order to represent the absence of a preference. For our node-hopper, this eliminates the trap by turning it into a node from which the outgoing edges, giving the probabilities for her next hop, are evenly divided amongst all the nodes.

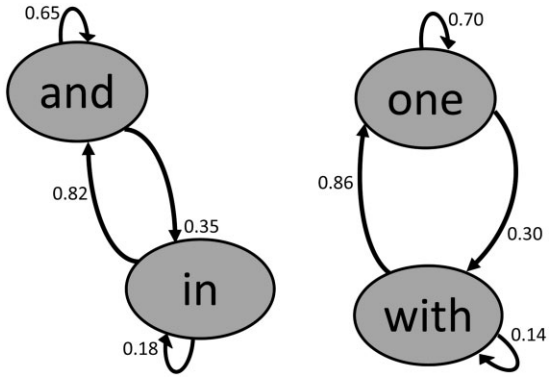

Fig. 2 A state-transition diagram representing the Markov chain shown in Table 5.

Figure 2 shows a second kind of trap we have to consider. In the text from which this WAN was constructed, 'and' is only ever followed by 'and' or 'in' and 'in' is only ever followed by 'in' or 'and', while 'one' is only ever followed by 'one' or 'with' and 'with' is only ever followed by 'with' or 'one'. It would be a strange text that resulted in this WAN, but here is one that would produce it:

> and and and in and and and in and and and in and in
> and and and in and and and in and and and in and in
> the the the the the the the the the the the the the
> the the the the the the the the the the the the the
> one one one with one one one with one one one with
> one one one with one one one with one one one with
> the the the the the the the the the the the the the

We might be tempted to rule out such possibilities using our knowledge of the language, as this text is simply nonsense made of meaningless repetitions. But Shakespeare was capable of writing dramatic lines such as 'Never, never, never, never, never' (*King Lear* 5.3.284), 'Words, words, words' (*Hamlet* 2.2.195), and 'O horror, horror, horror' (*Macbeth* 2.3.62), so we really should rule nothing out as impossible regarding unusual repetitions. Let us see what happens when we attempt to analyse such a strange text.

The normalized matrix for the WAN shown in Fig. 2, derived from our nonsense text, is shown in Table 5. Consider what the WAN in Fig. 2 means for our node-hopper. It is not that she would get stuck for eternity on any one node as with the simple trap of a node with no outgoing edges, but that she would be stuck for eternity in one or other of the two pairings: either the 'and'/'in' pairing or the 'one'/'with' pairing. Which of the two pairings she got stuck in would depend on which node she started on and as the two pairings are isolated that first-step choice would entirely determine the Limit Probabilities that her endless hopping would produce. We can see this in practice if we multiply the matrix in Table 5 by itself 100 times, which produces the matrix in Table 6.

Comparing Table 6 with Table 4, we can see that in Table 6 derived from the WAN shown in Table 5 and Fig. 2, the Limit Probabilities have not converged. In Fig. 4, the column giving the Limit Probability for 'and' has four occurrences of the same number 0.24, but the numbers in the column for 'and' in Table 6 are 0.70, 0.70, 0.00, and 0.00. This represents the difference that whereas in Table 4 it does not matter which node we start our hopping from—starting from 'and' or 'in' or 'one' or 'with' always gives us a Limit Probability for 'and' of 0.24—in Table 6 (representing the WAN in Fig. 2) starting from 'and' or 'in' gives us a Limit Probability for 'and' of 0.70, while starting from 'one' or 'with' gives us a Limit Probability for 'and' of zero, because if we start from 'one' or 'with' we never land on 'and'. That is, the non-converging numbers in

Table 5. Matrix representing the WAN shown in Fig. 2

↓from to→	and	in	one	with
and	0.65	0.35	0.00	0.00
in	0.82	0.18	0.00	0.00
one	0.00	0.00	0.70	0.30
with	0.00	0.00	0.86	0.14

Table 6. A matrix representing the Limit Probabilities for the WAN in Table 5

	and	in	one	with
and	0.70	0.30	0.00	0.00
in	0.70	0.30	0.00	0.00
one	0.00	0.00	0.74	0.26
with	0.00	0.00	0.74	0.26

Table 6 correctly represent the fact that for this particular WAN, derived from this unusual nonsense text, the result of the node-hopping process, that is the Limit Probabilities, is strongly determined by where we start the hopping from.

What can be done about this? The solution is to conduct multiple node-hopping exercises, each starting from a different node, and then combine the results. But how many times should we run the node-hopping exercise for each starting word? The reasonable answer is that we repeat each node-hopping exercise, each of which will give us a different set of Limit Probabilities depending on our starting word, the number of times that this starting word appears in our text. In our nonsense text represented by the WAN in Fig. 2, there are fifty-two occurrences of 'and', 'in', 'one', and 'with', plus thirty-six occurrences of 'the' that in this exercise we are not counting. Of those fifty-two words, twenty are 'and', eight are 'in', eighteen are 'one', and six are 'with'. If we run the node-hopping exercise fifty-two times, we should start on 'and' twenty times, on 'in' eight times, on 'one' eighteen times, and on 'with' six times. To run more node-hopping exercises we need only stick to these proportions, starting on 'and' $^{20}/_{52}$ of the time, on 'in' $^{8}/_{52}$ of the time, on 'one' $^{18}/_{52}$ of the time, and on 'with' $^{6}/_{52}$ of the time. Having thus started from each word in proportion to its frequency in the text, we can safely sum the Limit Probabilities across all the node-hopping exercises since by selecting afresh where to start each exercise we have applied the necessary weighting.

To implement this idea in our matrix, we simply have to multiply the Limit Probabilities for each result by the fraction representing the frequency in our text of the word from which we have to start in order to get that result. So, we multiply all the results for starting with 'and' by $^{20}/_{52}$, all the results for starting with 'in' by $^{8}/_{52}$, all the results for starting with 'one' by $^{18}/_{52}$, and all the results for starting with 'with' by $^{6}/_{52}$. Our equivalent to summing all the results for the node-hopping exercises is to add together all numbers in each column, this being the sum of the weighted Limit Probabilities found when we start with 'and', 'in', 'one', and 'with'. This weighting and summing are shown in Table 7. The four Column Totals in Table 7—0.38, 0.16, 0.34, and 0.12—are the LPs for the words 'and', 'in', 'one', and 'with', respectively.

This procedure handles the case of a peculiar nonsense text giving rise to the peculiar WAN shown in Fig. 2, and it works just as well for more peculiar cases, where for example there might be not two closed loops of two words as shown in Fig. 2 but many dozens of closed loops of many words. But what about the more usual real-world text from which the WAN in Table 3 was produced, for which the Limit Probabilities converge as shown in Table 4? In this case, there is no need to apply special weightings that represent what happens when we begin our node-hopping from different starting points, since the results are the same no matter which node we start from. The technical name for such a Markov chain that has no absorbing states and no closed loops is 'ergodic'. It turns out that if we apply the same weighting procedure to such an ergodic case as this, which does not need it, the procedure does no harm. This is shown in Table 8.

As we can see in Table 8, applying the weightings to the Limit Probabilities makes no change to the Limit Probabilities if they have already converged. This is because $^{20}/_{52} + {}^{8}/_{52} + {}^{18}/_{52} + {}^{6}/_{52}$ equals $^{52}/_{52}$ which equals 1, as it must since we started with a pool of fifty-two word occurrences and divided it four unequal ways, one for each of the four word-occurrences' frequencies. No matter what these frequencies and their resulting fractions are, if we multiply a single number—0.24 in the first column, 0.32 in the second, and

Table 7. Matrix showing the Limit Probabilities from Table 6 after weighting to reflect each word's frequency in the text

	and	**in**	**one**	**with**
and	$0.70 \times {}^{20}/_{52}$	$0.30 \times {}^{20}/_{52}$	$0.00 \times {}^{20}/_{52}$	$0.00 \times {}^{20}/_{52}$
in	$0.70 \times {}^{8}/_{52}$	$0.30 \times {}^{8}/_{52}$	$0.00 \times {}^{8}/_{52}$	$0.00 \times {}^{8}/_{52}$
one	$0.00 \times {}^{18}/_{52}$	$0.00 \times {}^{18}/_{52}$	$0.74 \times {}^{18}/_{52}$	$0.26 \times {}^{18}/_{52}$
with	$0.00 \times {}^{6}/_{52}$	$0.00 \times {}^{6}/_{52}$	$0.74 \times {}^{6}/_{52}$	$0.26 \times {}^{6}/_{52}$
Column totals	0.38	0.16	0.34	0.12

Table 8. Matrix showing the Limit Probabilities from Table 4 after weighting to reflect each word's frequency in the text

	and	in	one	with
and	$0.24 \times {}^{20}/_{52}$	$0.32 \times {}^{20}/_{52}$	$0.20 \times {}^{20}/_{52}$	$0.24 \times {}^{20}/_{52}$
in	$0.24 \times {}^{8}/_{52}$	$0.32 \times {}^{8}/_{52}$	$0.20 \times {}^{8}/_{52}$	$0.24 \times {}^{8}/_{52}$
one	$0.24 \times {}^{18}/_{52}$	$0.32 \times {}^{18}/_{52}$	$0.20 \times {}^{18}/_{52}$	$0.24 \times {}^{18}/_{52}$
with	$0.24 \times {}^{6}/_{52}$	$0.32 \times {}^{6}/_{52}$	$0.20 \times {}^{6}/_{52}$	$0.24 \times {}^{6}/_{52}$
Column totals	0.24	0.32	0.20	0.24

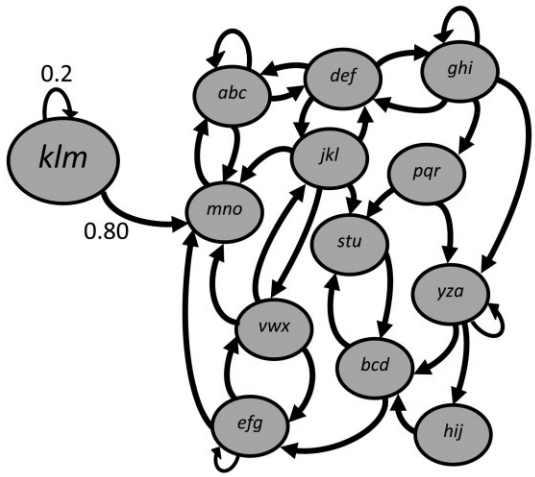

Fig. 3 A Markov chain in which node *klm* has a Limit Probability of zero.

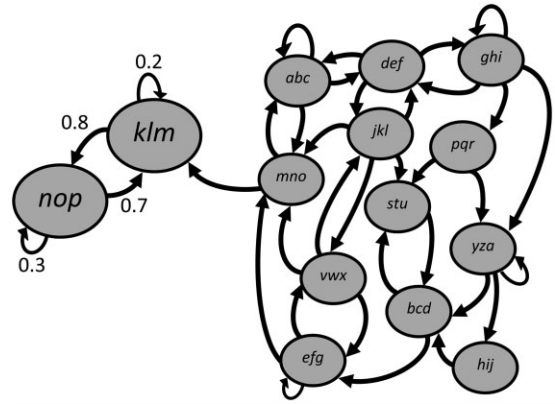

Fig. 4 A Markov chain in which all nodes except *klm* and *nop* have a Limit Probability of zero.

so on—by each of a set of fractions that sum to 1 and then sum the products (as we do in the row 'Column totals') the outcome is the same as multiplying that single number by 1. Thus, our procedure leaves the Limit Probabilities unchanged if they were already the same no matter which word we started with. It is only when the Limit Probabilities differ according to which word we started with, as they do in Table 6, that this process applies a weighting effect.

We have considered two peculiarities that could emerge in our Markov chain that require special attention in our algorithm. The first is a node having no outgoing edges (represented by a row of zeroes in our WAN table) because in our text the word it represents is never followed by one of our words of interest. The second is a closed set of nodes having edges only to each other and no connections to the wider chain. Both those cases are anticipated and corrected for in the algorithm. A third potential problem is the existence of a node on which our random walker will spend virtually no time at all no matter where she starts. Consider Fig. 3 showing a set of nodes labelled *abc* to *hij* that are well-connected with one another and a single isolated node labelled *klm*. If our node hopper starts on *klm*, she will quickly leave it never to return, as it has no incoming edges, and if she starts anywhere else, she will never visit *klm* because it has no incoming edges. Over the long term, our node hopper will spend negligible time on *klm*: its Limit Probability will approach zero and it will play no role in our calculation of relative entropy. This quite reasonably reflects the fact that *klm* is a rare word for which we do not have an established pattern of proximity. For this reason, we are content to ignore such a word and our algorithm does not correct for this situation.

Figure 4 illustrates the last potential problem we will consider, in which a pair of nodes present a trap for our node hopper. As in Fig. 3, the nodes labelled *abc* to *hij* are well-connected to one another, but now this set's edge with *klm* is reversed and *klm* forms a

pair with the node *nop*. This is essentially the opposite of the situation depicted in Fig. 3 in that once our node hopper visits *klm* or *nop* she is trapped, endlessly hopping between just these two because there is no edge taking her back to the set *abc* to *hij*. Our node hopper must inevitably fall into this trap, as she either starts on *klm* or *nop* or she starts on a node in the set *abc* to *hij* and is eventually taken to *klm* by the edge that leads to it. Notice that *klm* and *nop* have outgoing edges to each other so they will not be detected by the adjustment described above in which a row of zeroes in the WAN table is replaced by a row of fractions, each being one divided by the number of words of interest.

If a text produces a Markov chain of the kind depicted in Fig. 4, then the Limit Probabilities for the nodes *abc* to *hij* will approach zero, as the longer the node hopping continues the longer the hopper spends trapped on the pair *klm* and *nop*. With all other Limit Probabilities falling to zero, our algorithm will in such a case base its attribution solely on the proximities for the two words represented by the nodes *klm* and *nop*. The situation in Fig. 4 can arise only if in our text the following conditions apply: (1) there is a pair of rare words, represented by nodes *klm* and *nop*, that occur in proximity to each other (hence the existence of edges connecting them to each other); (2) one of them appears at least once after one of the common words represented by nodes *abc* to *hij* (hence the edge into *klm*); and (3) neither of them appears before one of the common words (hence the absence of edges leading back from *klm* or *nop* to the set from *abc* to *hij*).

It is not impossible for a text to meet the conditions that give rise to the Markov chain shown in Fig. 4, but it is a vanishingly rare occurrence in all but the shortest texts. This is one of the reasons why the WAN method requires that the texts it attributes are longer than a few hundred words. One can think of more convoluted situations with several rare words trapping the walker, but these too can be shown to be unlikely occurrences so long as the text used for attribution is sufficiently long. Our algorithm does not attempt to detect and correct for these rare occurrences, since their only effect is an acceptable and precisely quantifiable diminution in the accuracy of the attributions when the method is applied to small textual samples.

We are now in position to write the penultimate step, Step 7, of our algorithm, to deal with the calculation of the Limit Probabilities:

> 7) Make a copy of the WAN, called LP (for Limit Probabilities), and wherever it has a row of all zeroes replace each cell's value with 1 divided by the number of words of interest. Multiply this matrix LP by itself 100 times. Then for each row in the LP multiply the number in each cell by the number of occurrences in the text of the word of interest represented by that row and divide the result by the total number of all occurrences in the text of all the words of interest. Add up the cells in each column to produce the Limit Probability for the word represented by that column.

With our Limit Probabilities calculated, we can now proceed to calculating the Kullback–Leibler divergence, or relative entropy, between two WANs and hence between two texts.

To compare the word-proximity habits of two texts, we need a WAN for each, so we must perform Steps 1 through 6 for each of the texts. Step 7 need be performed only for the first of the two WANs, as only its Limit Probabilities are needed for the calculation of the relative entropy. The final step in our algorithm is the calculation of the relative entropy between two WANs, one for each of two texts, which we will call WAN1 and WAN2:

> 8) Find the matrix cells for which, at the same row and column positions in the two matrices, the WAN1 cell and the WAN2 cell contain non-zero values. For each such pair of cells, deduct from the natural logarithm of the WAN1 cell's value the natural logarithm of the WAN2 cell's value and then multiply this difference by the value in the WAN1 cell and by the Limit Probability of the word represented by this row in WAN1. Sum the result of this calculation for each such pair of cells in WAN1 and WAN2 to give the relative entropy expressed in the units called nats.

The only technical term in Step 8 we have not yet defined is the logarithm. This is in a sense the inverse of raising one number to the power of another

number. We say that 3 to the power of 5 is 243 because $3 \times 3 \times 3 \times 3 \times 3 = 243$. That is, we raise x to the power of y by multiplying x by itself repeatedly so that there are y occurrences of x in the multiplication. We can express this relationship the opposite way around and say that the logarithm of 243 in the base of 3 is 5 because we have to raise 3 to the power of 5 in order to reach 243. We can use any base, so that the logarithm of 10,000 in the base of 10 is 4 because 10 to the power 4 is 10,000, or the logarithm of 256 in the base of 2 is 8 because 2 to the power of 8 is 256. Logarithms in the base of 10 are called common logarithms and logarithms in the base of 2 are called binary logarithms. The natural logarithm referred to in our Step 8 is one that uses as its base the mathematical constant called e (approximately 2.72) and when we use this base our relative entropy is expressed in nats. We could just as well use any base, so for example if we chose base 2, the binary logarithm, then our resulting relative entropy would be expressed in the units called bits.

Where in the above account we refer to a pair of texts this may be on the one hand a single play of unknown authorship and on the other an entire canon of plays known to have been written by one dramatist. A reasonable use of the algorithm is to measure the relative entropy between a WAN made from a play of unknown authorship and, in turn, each of the WANs made from the dramatic canons of the plausible candidates for the play's authorship. Validation investigations in which the supposedly unknown play is in fact one we know the authorship of—which play of course was not one of the plays from which the author's WAN was constructed—tell us that this method is able to 'predict' the correct author in around 90–94% of cases for which we have sufficient text to measure in the sample and in the canons of the candidate authors.

The procedure leading to this claimed 90–94% success rate is known as 'leave-one-out cross validation'. We take a set of securely attributed plays, each by a different author, and we extract one play at random and then build a WAN for each author in the remaining set, based on all their plays. If we find that of all the authorial WANs the one with the lowest relative entropy to the WAN of the extracted play is the one for the true author of the play, then this counts as a successful attribution. The procedure is repeated many times, with a different randomly chosen play extracted each time, and the rate of successful attributions is counted.

In our published applications of the WAN method, the only additional complication we have not yet described is a means for choosing the list of words of interest whose proximities the algorithm records. Rather than simply using all of the 100 most common words in the language, it is possible to select the particular words whose proximities most effectively serve to distinguish the various authors in a set of plays of known authorship. In our experiments, we started with a single list of words in the set of securely attributed plays, arranged in rank order from highest to lowest frequency. Taking just the first few most-common words as our words of interest, we ran the validation process many times, always with the same words of interest and extracting a different random play each time. This produced an accuracy score associated with the minimal set of words of interest.

Then we added to our words-of-interest list the next most common word from the rank-order list of those in the securely attributed plays and we reran the validation process, always using the same, new list of words of interest and extracting a different random play each time. If we found that adding this new word to our list of words of interest improved the accuracy score, we retained it and if not, we discarded it. Then we tried this all again with the addition of the next most common word from the rank-order list of those in the securely attributed plays, and so on until adding new words of interest made no appreciable difference to the accuracy score. The resulting list of words of interest represents those that are demonstrably best able to differentiate the authors in our set of securely attributed plays.

Our published work on WANs has been the subject of critiques in articles appearing in the journal *ANQ: A Quarterly Journal of Short Articles, Notes and Reviews* (Rizvi, 2018; Barber, 2020). These we have answered in articles in the same journal, pointing out any misunderstandings and misrepresentations of our work (Segarra et al., 2020a,b). We hope that the above detailed explanation of the WANs method helps scholars who want to understand the method, and those who want to critique it, and those who want to use it in their own research, by providing the technical details in a relatively non-technical idiom.

We have made available an Open Source Python script for the above algorithm by putting it on the WorldWide Web at <http://gabrielegan.com/WAN>. This Python script performs the functions expressed in our algorithm's Steps 1 through 8 as described above, and has an option for stopping the five-word window at the end of a dramatic speech, or at the end of any other syntactic break that the user is able to identify and encode in the digital source texts used. The input to the Python script is three text files: two samples of writing to be compared and a list of words of interest for making the comparison—typically high-frequency function words—and the output is a statement of the relative entropy between the samples of writing.

All three files must be encoded in the American Standard Code for Information Interchange (ASCII) format, which for virtually all potential users is readily available as the subset of American National Standards Institute (ANSI) and Unicode Transformation Format 8 (UTF-8) standards. The script may well work for encodings that use the non-ASCII encodings in the ANSI and UTF-8 standards, but this has not been tested by the authors. It is assumed that the samples of writing comprise words separated by spaces and other conventional pieces of punctuation without any special tagging or mark-up. However, as the script merely compares strings of letters as if these represent words, it will also work correctly with files in which parts of speech have been encoded by appending an abbreviation to each word. Thus, the correct results will be obtained if in the writing samples and the list of words of interest the word 'the' is consistently recorded as *the_det* (to represent that it is a determiner), the word 'and' is consistently recorded as *and_conj* (to represent that it is a conjunction), and so on for other words.

The Python script does not implement the function-word selection process we describe above as this is a matter of how the algorithm is applied, and in a large-scale investigation it would be achieved by automated 'calling' of our software script multiple times by a kind of 'supervisor' software that performed the word-selection process. As we have mentioned before, our function-word selection process was not essential in our application of the WAN method to the authorship claims we investigated and '. . . we could just pick the top 100 most-used words in English and the results would be about the same' (Segarra *et al.*, 2020b, p. 3). This fact should remind us that language is ineluctably a stochastic process, indifferent to the linguistic categories that we apply to particular kinds of words. And this in turn should remind us that Claude Shannon's extraordinary contribution to the field of Information Theory and hence computation was also an extraordinary gift to the study of language itself.

References

Antonia, A., Craig, H., and Elliott, J. (2014). Language chunking, data sparseness, and the value of a long marker list: Explorations with word N-grams and authorial attribution. *Literary and Linguistic Computing*, **29**: 147–63.

Argamon, S. E. (2018). Computational forensic authorship analysis: Promises and pitfalls. *Language and Law/Linguagem e Direito*, **5**(2): 7–37.

Barber, R. (2020). Function word adjacency networks and early modern plays. *ANQ*: *A Quarterly Journal of Short Articles, Notes and Reviews*, **33**: 204–13.

Craig, H. and Greatley-Hirsch, B. (2017). *Style, Computers, and Early Modern Drama: Beyond Authorship*. Cambridge: Cambridge University Press.

Eisen, M., Ribeiro, A., Segarra, S., and Egan, G. (2018). Stylometric analysis of early modern English plays. *Digital Scholarship in the Humanities*, **33**: 500–28.

Forsyth, R. S., Holmes, D. I., and Tse, E. K. (1999). Cicero, Sigonio, and Burrows: Investigating the authenticity of the *Consolatio*. *Literary and Linguistic Computing*, **14**: 375–400.

Hoover, D. L. (2004). Delta prime? *Literary and Linguistic Computing*, **19**: 477–95.

Jockers, M., Nascimento, F.,and Taylor, G. H. (2019). Judging style: The case of *Bush Versus Gore*. *Digital Scholarship in the Humanities*, **35**: 319–27.

Jockers, M. L., Witten, D. M., and Criddle, C. S. (2008). Reassessing authorship of the Book of Mormon using delta and nearest shrunken centroid classification. *Literary and Linguistic Computing*, **23**: 465–91.

Kullback, S. and Leibler, R. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, **22**(1): 79–86.

Mosteller, F. and Wallace, D. L. (1963). Inference in an authorship problem. *Journal of the American Statistical Association*, **58**: 275–309.

Rizvi, P. (2018). Authorship attribution for early modern plays using function word adjacency networks: A critical

view. *ANQ: A Quarterly Journal of Short Articles, Notes and Reviews*, **33**: 328–31.

Segarra, S., Eisen, M., and Ribeiro, A. (2015). Authorship attribution through function word adjacency networks. *Institute of Electrical and Electronics Engineers (IEEE) Transactions on Signal Processing*, **62**(20): 5464–78.

Segarra, S., Eisen, M.,, Egan, G., and Ribeiro, A. (2016). Attributing the authorship of the *Henry VI* plays by word adjacency. *Shakespeare Quarterly*, **67**: 232–56.

Segarra, S., Eisen, M., , Egan, G., and Ribeiro, A. (2020a). A response to Pervez Rizvi's critique of the word adjacency method for authorship attribution. *ANQ: A Quarterly Journal of Short Articles, Notes and Reviews*, **33**: 332–37.

Segarra, S., Eisen, M., Egan, G., and Ribeiro, A. (2020b). A response to Rosalind Barber's critique of the word adjacency method for authorship attribution. *ANQ: A Quarterly Journal of Short Articles, Notes and Reviews Advance Access*, 1–6.