Vowel and Consonant Classification through Spectral Decomposition

Patricia Thaine and **Gerald Penn**
Department of Computer Science
University of Toronto
{pthaine,gpenn}@cs.toronto.edu

Abstract

We consider two related problems in this paper. Given an undeciphered alphabetic writing system or mono-alphabetic cipher, determine: (1) which of its letters are vowels and which are consonants; and (2) whether the writing system is a vocalic alphabet or an abjad. We are able to show that a very simple spectral decomposition based on character co-occurrences provides nearly perfect performance with respect to answering both question types.

1 Introduction

Most of the world's writing systems are based upon *alphabets*, in which each of the basic units of speech, called *phones*, receives its own representational unit or letter. The vast majority of phones are consonants or vowels, the former being produced through a partial or full obstruction of the vocal tract, the latter, through a stable interval of resonance at several characteristic frequencies called *formants*. In the course of deciphering an alphabet, one of the first important questions to answer is which of the letters correspond to vowels, and which to consonants, a problem that has been studied as far back as Ohaver (1933). Indeed, if there is disagreement as to whether a phonetic script is an alphabet or not, a near-perfect separation of its graphemes into consonants and vowels would be very important evidence for confirming the proposition that it was.

A well-publicized, recent attempt at classifying the letters of an undeciphered alphabet as either vowels or consonants was by Kim and Snyder (2013), who used a Bayesian approach to estimating an unobserved set of parameters that cause phonetic regularities among the distributions of letters in the alphabets of known/deciphered writing systems. By contrast, the method proposed in this paper is based on a very simple spectral analysis of letter distributions within only the writing system under investigation, and it requires no training or parameter tuning. It is furthermore based on a newly confirmed empirical universal over alphabetic writing systems that is interesting in its own right, is crucial to our method's numerical stability.

Spectral analysis of vowels and consonants dates back to at least Moler and Morrison (1983), which performs very poorly. Our method can be regarded as both a simplification and improvement to Moler and Morrison (1983). On average, our method correctly classifies 97.45% of characters in any alphabetic writing system.

Another notable antecedent is Goldsmith and Xanthos (2009), who discovered essentially the same method for vowel-consonant separation in the context of spectrally analyzing phonemic transcriptions. While the premise that someone would have phonemically transcribed a text without knowing by the end which phones were vowels or consonants may seem far-fetched, Goldsmith and Xanthos (2009) draw some important conclusions for a subsequent analysis of vowel-harmonic processes that we shall not investigate further here. Goldsmith and Xanthos (2009) also cite Sukhotin (1962), whose method we evaluate below, as a precedent for their own study, possibly influenced by Guy's (1991) English gloss of Sukhotin's work, which misrepresents Sukhotin's (1962) intention as seeking to classify letters in a substitution cipher as vowels or consonants. Sukhotin's (1962) study, which was originally written in Russian, is in fact about the written form (*bukv*) of plaintext letters, not of ciphers nor of the sounds of speech. Sukhotin begins his study by posing the research question of whether, given the well-known separation of the sounds of speech into vowels and consonants, there are similar classes for letters (*podobnyh klassah k'bukvam*). The distinction be-

Proceedings of the First Workshop on Subword and Character Level Models in NLP, pages 82–91,
Copenhagen, Denmark, September 7, 2017.

	_*h	t*e	h*_	_*a	f*t	a*_	c*t
t	1	0	0	0	0	1	0
h	0	1	0	0	0	0	0
e	0	0	1	0	0	0	0
f	0	0	0	1	0	0	0
c	0	0	0	1	0	0	0
a	0	0	0	0	1	0	1

Table 1: The binary matrix, A, for the string 'the fat cat'. Viewed as an adjacency matrix, it represents a bipartite graph.

tween written letters and phones is particularly salient in Russian, which, unlike English, has written letters that simply cannot be classified as vowels or consonants in any context or in isolation.[1]

Sukhotin (1962) made an earlier attempt at our study of writing systems, not at Goldsmith and Xanthos's (2009) study of phoneme clustering. In the present paper, we consider two applications of our method to the problem of classifying an alphabetic writing system as either an abjad (one with letters only for consonants) or a vocalic alphabet (one with letters for vowels as well).

2 A Spectral Universal over Alphabets

A *p-frame* (Stubbs and Barth, 2003) is a bit like a trigram context, except it considers one preceding and one succeeding element of context, rather than two preceding elements. The string 'the fat cat', for example, contains these, among other p-frames at the character level: '_*h', 't*e', 'h*_', '_*a', where '_' represents a space.

Given a sufficiently long corpus, C, in the alphabet, Ω, let A be the binary matrix of dimension $m \times n$, where n is the number of different letter types in Ω and m is the number of different p-frames that occur in C (see Table 1), in which $A_{ij} = 1$ iff letter i occurs in p-frame j in C.

Every m by n matrix A has a singular-value decomposition into $A = U\Sigma V^T$. Usually, we are interested in Σ, a diagonal matrix containing the *singular values* of A, but we will be more concerned here with the n by n matrix V, the columns of which, the *right singular vectors* of A, are eigenvectors of A^TA. V is also *orthonormal*, which means that the inner product of any two right singular vectors, $v_i \cdot v_j$, is 0 unless $i = j$, in which case the inner product is 1 (Strang, 2005).

If the rows and columns of U, Σ and V are permuted so that the singular values of Σ appear in decreasing order, then the first two right singular vectors are the most important, in the sense that they provide the most information about A. Let x and y be these two vectors; they are columns of V, and so they are rows of V^T, as shown in Figure 1. Empirically, each x_i is proportional to both the frequency of the i^{th} letter in C and the frequencies of the p-frame contexts in which the i^{th} letter occurs. Again empirically, each y_i ends up being proportional to the number of contexts that the i^{th} letter shares with other letters.

Because V is orthonormal, $\sum_i x_i y_i = 0$. Since their sum centres around zero, for some of the letters $i \in \Omega^+$, $x_i y_i$ is positive, and for other $i \in \Omega^-$, $x_i y_i$ is negative. The spectral universal we have empirically determined is that these two subsets of Ω almost perfectly separate the vowels and consonants of the writing system utilized by C. A moment's reflection will confirm that the p-frame distributions of vowels are probably very different from the p-frame distributions of consonants (Sukhotin, 1962), but the best thing about this universal is its inherent numerical stability. Table 2 shows the sums over these two sets for 15 alphabetic writing systems, expanded to 12 decimal places.

This calculation presumes a foreknowledge of what the vowels and consonants are, but if we were to order all of the letters in Ω by their value y_i, define a separator $y = b$, and then vary the parameter b so as to maximize the sum $|\sum_{i:y_i>b} x_i y_i| + |\sum_{i:y_i \leq b} x_i y_i|$, $b = 0$ attains the maximum value. This is again trivial to prove in theory, but because the differences between vowel and consonant p-frames are the most important differences among all of the possible separators, empirically we may observe that $y = 0$ separates the vowels from the consonants. In other words, the actual values that the y_i attain are irrelevant; all that matters is their signs.

None of this provides any guidance as to which subset/sign contains the vowels and which, the consonants. Borrowing from the general idea behind Sukhotin's algorithm (Guy, 1991), we will assume that the most frequent letter of any alpha-

[1] These are the front and back "yer" that respectively mark the presence or absence of palatalization. Sukhotin (1962) knew about the special status of these letters, too; when his method classifies the "front yer" as a vowel, he expresses some satisfaction because the "front yer" did represent a vowel at an earlier stage in Russian writing.

$$A = U\Sigma V^\top = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

Figure 1: Singular Value Decomposition of A.

Language	$\lvert\sum \mathbf{x_{vowels}} \cdot \mathbf{y_{vowels}}\rvert$	$\lvert\sum \mathbf{x_{consonants}} \cdot \mathbf{y_{consonants}}\rvert$
Danish	0.461778253515	0.461778253515
Dutch	0.478014338904	0.478014338904
English	0.484420669972	0.484420669972
Finnish	0.471723103373	0.471723103373
French	0.482759327181	0.482759327181
German	0.440663056154	0.440663056154
Greek	0.447065776857	0.447065776857
Hawaiian	0.432782088536	0.432782088536
Italian	0.467317672843	0.467317672843
Latin	0.4656326487	0.4656326487
Maltese	0.496082609138	0.496082609138
Portuguese	0.463359992637	0.463359992637
Russian	0.491165538014	0.491165538014
Spanish	0.478974310472	0.478974310472
Swedish	0.430570626024	0.430570626024

Table 2: Inner products of x and y (Figure 1) for 15 different writing systems, accurate to 12 places.

bet is a vowel,[2] (Vietnamese is the singular exception that we have found to this rule) and thus label the subset that contains it as the vowel container[3]. This yields Algorithm 1, which we evaluate in Table 3.[4] [5]

3 Evaluating the Vowel Identification Algorithm

Kim and Snyder (2013) report token-level accuracies with a macro-average of 98.85% across 503 alphabets, with a standard deviation of about 2%. Token-level accuracies are somewhat misleading, as the hyperbolic distribution of letters in all naturally occurring alphabets makes it very easy to inflate accuracies even when the class of many (rare) letters cannot be determined. Furthermore, if the classified or readable portions of corpora were at issue, then these token accuracies should have been micro-averaged, not macro-averaged, and, more importantly, they should have been smoothed by an n-gram character model to produce a more meaningful estimate.

Vowel/consonant classification is better viewed as a letter-type, not letter-instance, classification problem, in which progress is evaluated according to the percentage of letter types that are correctly classified. Semivowels or whatever ambiguous classes one wishes to define should ideally be distinguished as extra classes, or at the very least disregarded. For a level comparison with our base-

[2] Note that we treat ò,ó,ô, and o, for example, as four distinct vowels.

[3] Out of the 26 alphabets we examine, this assumption only fails for Vietnamese, whose most frequent letter is n. This is mainly due to the large number of diacriticized vowels in Vietnamese that we treat discretely.

[4] In this and the subsequent experiments, the following writing systems were withheld as an evaluation set to prevent overfitting: Aramaic, Farsi, Hungarian, Serbian, Urdu, and Vietnamese.

[5] Each corpus was sampled from a combination of Wikipedia, Project Gutenberg and BBC World Service web pages, and consists of between 14316 and 706422 characters (median=164757). All punctuation was removed, and all letters were downcased.

Algorithm 1 Vowel and consonant classification algorithm

$num_{words} \leftarrow 0$
$num_{letters} \leftarrow length(letters)$
$contexts \leftarrow list\ of\ num_{letters}\ empty\ lists$
$frames_{keys} \leftarrow [\,]$
$frames_{values} \leftarrow [\,]$
$letters_{count} \leftarrow list\ of\ zeros\ of\ size\ num_{letters}$
$A \leftarrow [\,]$
$A_{weighted} \leftarrow [\,]$
function VOWELCONSONANTCLASSIFICATION(V, $most\_freq\_letter$)
    $coordinates \leftarrow zip(V[0], V[1], letters)$
    $cluster_1 \leftarrow triples\ where\ V[1]\ value\ >\ 0$
    $cluster_2 \leftarrow triples\ where\ V[1]\ value\ <\ 0$
    $vowels \leftarrow cluster\ that\ has\ most\_freq\_letter$
    $consonants \leftarrow cluster\ that\ does\ not\ have\ most\_freq\_letter$
    return $vowels, consonants$
end function
function ALGORITHM1($corpus$, max)
    for all $word \in corpus$ **do**
        $word \leftarrow ['\_'] + list(word) + ['\_']$
        $num_{words} +=\ 1$
        if $num_{words} > max$ **then**
            break
        end if
        $MakePFrames(word)$ # Calculates A and $A_{weighted}$
    end for
    $index_{most\_freq\_letter} \leftarrow index\ of\ max(letters_{count})$
    $most\_freq\_letter \leftarrow letters[index_{most\_freq\_letter}]$
    $U,\ s,\ V \leftarrow SVD(A)$
    $vowels, consonants \leftarrow VowelConsonantClassification(V, most\_freq\_letter)$
    return $vowels, consonants$
end function

Language	(Moler and Morrison, 1983)				Sukhotin's Algorithm			Algorithm 1		
	NC	P	R	A	P	R	A	P	R	A
Abkhaz	4	1.00	0.67	0.94	1.00	1.00	**1.00**	1.00	1.00	**1.00**
Afrikaans	18	0.71	0.36	0.31	0.93	0.81	0.88	1	0.81	**0.91**
Czech	23	1.00	0.63	0.68	1.00	0.94	**0.98**	1.00	0.94	**0.98**
Dutch	11	1.00	1.00	**1.00**	0.83	1.00	0.96	1.00	1.00	**1.00**
Danish	26	0.67	0.67	0.56	0.88	0.93	0.91	1.00	0.93	**0.97**
English (Middle)	4	1.00	1.00	**1.00**	1	0.90	0.96	1	0.90	0.96
English (Modern)	5	1.00	1.00	**1.00**	0.71	1.00	0.92	1.00	1.00	**1.00**
English (Old)	19	0.86	0.67	0.64	1.00	1.00	**1.00**	1.00	1.00	**1.00**
Finnish	3	1.00	0.89	**0.96**	0.89	1.00	**0.96**	0.89	1.00	**0.96**
French (Modern)	29	0.43	1.00	0.60	1.00	0.79	**0.89**	1.00	0.79	**0.89**
Inuktitut	6	1.00	1.00	**1.00**	0.95	0.95	0.95	1.00	0.95	0.97
Italian	17	0.90	0.90	0.86	0.91	0.67	0.82	1.00	0.93	**0.97**
German	13	1.00	0.88	0.93	0.73	1.00	0.89	0.88	1.00	**0.96**
Greek (Ancient)	3	0.83	1.00	0.95	1.00	1.00	**1.00**	1.00	1.00	**1.00**
Greek (Modern)	3	1.00	1.00	**1.00**	1.00	1.00	**1.00**	1.00	1.00	**1.00**
Hawaiian	5	0.90	0.90	0.92	0.83	0.91	0.90	1.00	1.00	**1.00**
Hungarian	14	0.44	0.80	0.71	0.94	0.94	0.94	1.00	1.00	**1.00**
Latin	3	1.00	1.00	**1.00**	1.00	1.00	**1.00**	1.00	1.00	**1.00**
Maltese	2	1.00	1.00	**1.00**	0.83	1.00	0.96	1.00	1.00	**1.00**
Portuguese	24	0.88	1.00	0.92	1.00	0.88	**0.94**	1.00	0.88	**0.94**
Russian	5	1.00	1.00	**1.00**	1.00	1.00	**1.00**	1.00	1.00	**1.00**
Serbian	25	0.89	0.89	0.85	0.90	0.69	0.88	1.00	0.82	**0.95**
Spanish	16	0.86	0.86	0.86	0.91	1.00	0.97	1.00	1.00	**1.00**
Swedish	6	1.00	1.00	**1.00**	0.89	1.00	0.96	0.80	1.00	0.93
Tagalog	4	1.00	0.94	**0.97**	0.95	1.00	**0.97**	1.00	0.89	0.95
Vietnamese	40	0.04	0.07	0.02	0.71	0.67	0.87	0.94	1.00	**0.99**

Table 3: Algorithm 1 evaluated with type-level accuracies. Corpora were sampled from the same sources as in Table 2, but with between 25738 and 968298 characters (median = 177529). The best accuracies are highlighted. Algorithm 1 incorrectly classifies several infrequent vowels (ë,ï,œ and ù) as consonants in Modern French. P, R, and A stand for Precision, Recall, and Accuracy, respectively. NC is the number of letters not classified by Moler and Morrison's (1983) algorithm; they are not necessarily semivowels. Unclassified letters are not included in the calculation of their method's precision, recall, and accuracy, however; their results are even worse when NC letters are treated as false negatives.

lines (most are interested in vowel vs. non-vowel; Kim and Snyder (2013) experimented with distinguishing nasals as well), ambiguous letters such as English 'y' have been manually identified and discarded altogether in Table 3.

It is impossible to determine the type accuracy of Kim and Snyder's (2013) method, because they only made the raw counts of words in their corpus available[6] (not the code, nor the resulting classifications). It is also impossible to reproduce their evaluation, since they did not provide their parameter settings. In addition, their ground truth classification of graphemes into vowels and consonants was remarkably ambitious. They treated all semivowels as consonants, for example — even tokens where they act as vowels. The "front yer" palatalization marker in Russian Cyrillic was called a consonant, for example, and yet the "back yer" that blocks palatalization is called a vowel. With such arbitrary labellings of graphemes that simply should have been left out of the classification, a controlled comparison of even token accuracy is perhaps beside the point. For what it is worth, however, we could use the correct

[6] http://pages.cs.wisc.edu/~ybkim/data/consonant_vowel_acl2013.tgz.

grapheme classifications in the 20 writing systems that constitute the overlap between the 503 that they sampled and the 26 that we did, and Algorithm 1's macro-averaged token-accuracy on these is 99.93%, whereas Sukhotin's is 96.05%.

An even greater cause for concern with this corpus is the sampling method that created it. Kim and Snyder's (2013) use of a leave-one-out protocol to evaluate their method on each of their 503 writing systems at first seems reasonable — every known writing system should be pressed into the service of analyzing an unknown one. But all of these samples are Biblical, and many of them (the English, Portuguese, Italian and Spanish samples, for example, or the French and German samples) are the same verses translated into different languages. It is not reasonable in general to expect that a sample of unknown writing would necessarily be a translation of a text from a known writing system. The overlap in character contexts between transliterated proper names and cognates makes for a very charitable transfer of knowledge between writing systems.

Across the 26 writing systems that we have evaluated, our samples are all different texts from several genres. Our method requires no training, so all of the samples can be used for evaluation, but it also cannot avail itself of transfer across writing systems. On these samples, Algorithm 1 achieves a macro-averaged type accuracy of 97.45% and a macro-averaged token accuracy of 99.39% with a standard deviation of 1.67%. Performance is very robust in the realistic context of low transfer. On the same samples, Sukhotin's algorithm has a macro-averaged type accuracy of 94.34%.

Moler and Morrison (1983)'s algorithm is less accurate than Algorithm 1. Moler and Morrison (1983) claim that their method is intended for "vowel-follows-consonant" (vfc) texts, where the proportion of vowels following consonants is greater than the proportion of vowels following vowels. Yet every writing system in our corpus is vfc, and still it performs poorly. Instead of using a binary adjacency matrix representing which letters occur within which p-frames, they calculate the number of times every possible letter pair occurs. They run SVD on the resulting matrix and use the second right and left singular vectors to plot the letters. The plot is divided into four quadrants, where letters in the fourth quadrant are classified as vowels, those in the second quadrant as consonants, and those in the first or third quadrants as "neuter," [*sic*] meaning unclassified (see NC on Table 3). Our plots, on the other hand, are split into half planes with a crisp, numerically stable separation at the x-axis between the putative vowels and putative consonants, leaving no letter unclassified unless it falls on $y = 0$, which would only occur with completely unattested letters. Given the computational power and the number of electronic multilingual sources available at the time, Moler and Morrison (1983) had no workable means of thoroughly evaluating their method.

Another important concern is stability as a function of length — many undeciphered writing systems are not well attested in terms of the number or length of their surviving samples. Our spectral method performs robustly at the 97.45% level for sparse samples down to a minimum of about 500 word types or 4000 word tokens. It is possible that below this threshold Sukhotin's algorithm would still be preferable.

Goldsmith and Xanthos (2009) only evaluate their method on one collection of written words, sampled from Finnish,[7] and they obtain the same result as we do below, with our algorithm only misclassifying the grapheme 'q'.[8] This should come as no surprise, because their method is an algebraically very close variant of ours — they compute eigenvectors on the Gram closure of our grapheme/context matrix (which they call F) instead of a singular value decomposition directly.

It may nevertheless come as a surprise that their method is so similar to ours. Their motivation consists of a lengthy discussion of graph cuts, along with a reference to Fiedler vectors, the name of the second eigenvector (the correlate to our $\vec{y}$) of a graph's Laplacian matrix, which is known to relate to the graph's algebraic connectivity. Neither Goldsmith and Xanthos (2009) nor we explicitly calculate the Laplacian matrix of a graph, and if this would-be graph happened to have more than one connected component, the Fiedler vector would not be uniquely well-defined on its Lapla-

[7] This is offered with the apology that Finnish is orthographically transparent, thus almost qualifying as a phonemic transcription.

[8] Goldsmith and Xanthos's (2009) explanation for this is a "problem of threshold," but our study has found that the numerical stability of the threshold is extremely accurate. Instead, the problem is the relative disconnectedness of 'q' from other graphemes owing to its sparsity, as the discussion in this paragraph will elaborate upon.

cian matrix in general.[9] Vowels and consonants rarely if ever separate into perfectly disjoint contexts; among our corpora the most disjoint is Vietnamese, in which vowels and consonants share exactly 100/645 p-frames. Out of curiosity, we evaluated our algorithm on the matrices from all 26 writing systems with their inter-CV/VC links removed. Performance degrades (macro-averaged accuracy: 89.08%) — which implies that this method is not merely computing an overall minimum graph cut — but not so badly that partitions could merely be ignoring either all of the vowels or all of the consonants. The explanation found in Goldsmith and Xanthos (2009) therefore does not account for the robustness or generality of our collective approach. Our own determination of this method, along with this universal, was entirely experimental.

A final difference to our approach is that Goldsmith and Xanthos (2009) use bigram contexts instead of p-frames, although they are aware that this choice is arbitrary. Empirically, p-frames work better than bigrams (macro-averaged type accuracy: 89.06%) as well as trigrams with two preceding elements (96.24%).

Figure 2 shows example classifications by Algorithm 1 of six different writing systems. Each letter is plotted at its (x_i, y_i) coordinate, but the classification is made using only y_i. It is worth noting that semivowels and other trouble-makers consistently fall very close to the $y = 0$ threshold. Maltese is particularly important, as it uses a vocalic alphabet with a Semitic language. Our correct handling of this case, and converse cases such as Farsi, demonstrates that we are responding to properties of alphabetic writing systems, and not of linguistic phylogeny.

4 Distinguishing Abjads from Vocalic Alphabets

Some writing systems assign syllabic or larger phonetic values to individual graphemes. Those that do not are sometimes called *alphabetic* writing systems, which is confusing because not all of them are true alphabets. There is another kind of alphabetic writing system called an *abjad*, which expresses only consonants. Arabic writing and writing systems based upon Arabic writing (whether or not the underlying language is related to the Arabic language) are the prototypical abjads; the rest (e.g., Hebrew, Aramaic) express Hatto-Semitic languages. Abjads express words in languages that have vowels, but the vowels must be inferred from context, unless, in anomalous genres, they are expressed through optional diacritics (Daniels and Bright, 1996).

We can use the spectral method presented in Section 2 to classify an alphabetic writing system as either an abjad or a true, vocalic alphabet. This is a different kind of classification problem than that of Section 3, as we are attempting here to classify the structure of entire writing systems rather than the phonetic values assigned to individual graphemes. We will consider two algorithms for distinguishing abjads from vocalic alphabets:

4.1 Algorithm 2: Divergence

This variant begins by provisionally assuming that the writing system under investigation is a vocalic alphabet, and applying Algorithm 1 to it, which involves the calculation of the aforementioned matrix, A, and the classification of every letter as a consonant or vowel. There is a related matrix W, for which W_{ij} is the number of times letter i occurs in the context of p-frame j. W is not binary. We will label the rows of W as $\hat{v_i}$ or $\hat{c_j}$ according to whether i and j are labelled as vowels or consonants by Algorithm 1. Algorithm 1 still uses A in assigning the labels, not W.

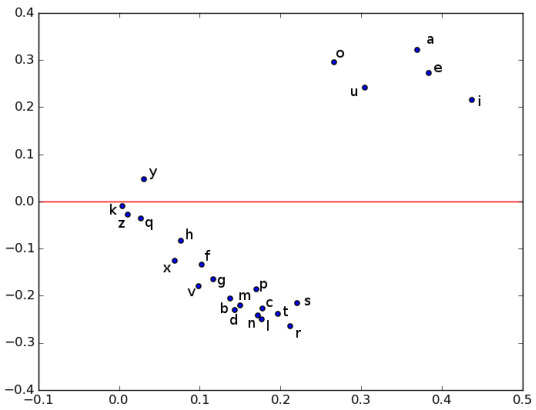
We can view each row of W as a discrete distribution over p-frame contexts. In recognition of this, Algorithm 2 calculates:

$$N = \sum_{\hat{v_i},\hat{v_j}} |D|(\hat{v_i}||\hat{v_j}) - \sum_{\hat{v_i},\hat{c_j}} |D|(\hat{v_i}||\hat{c_j}),$$

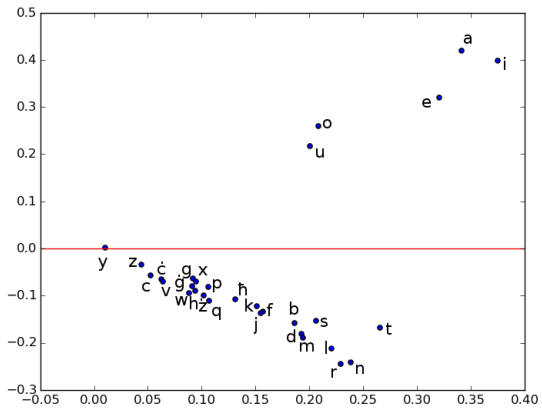
where $D(p||q)$ is the Kullback-Leibler divergence of p and q. We use $|D|$ to represent the absolute-value of each element-wise calculation of $\hat{v_i} \log \frac{\hat{v_i}}{\hat{v_j} \text{or} \hat{c_j}}$. The distributions of putative vowels tend to be more dissimilar to one another in abjads than in true alphabets. The distributions of putative vowels are more similar to that of putative consonants in abjads than in true alphabets. Values of N are shown for 30 writing systems in Table 4.

N separates the abjads from the vocalic alphabets at about $N = -100$.

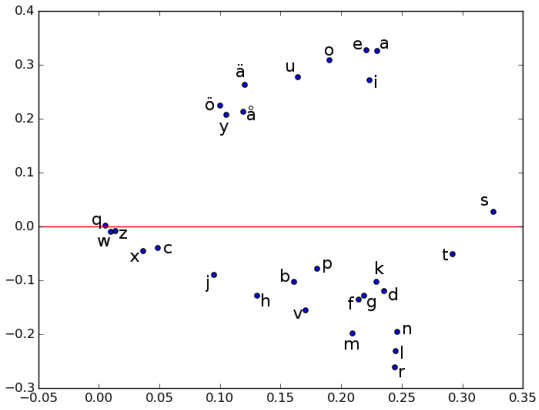
[9] Unless all of the connected components fortuitously had first and second eigenvalues of exactly the same magnitudes, the overall second non-zero eigenvector would not cross all of the components.



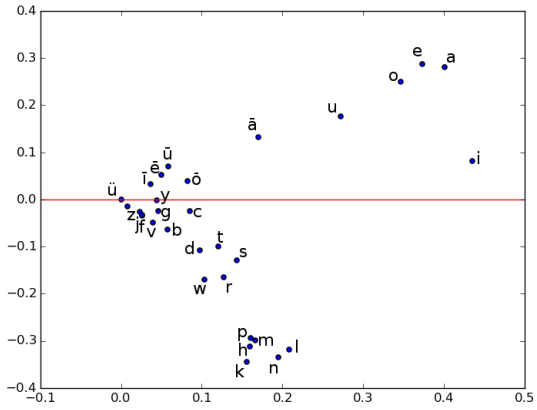
Latin



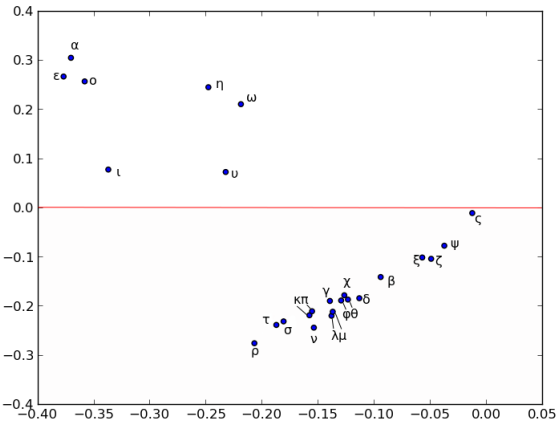
Maltese



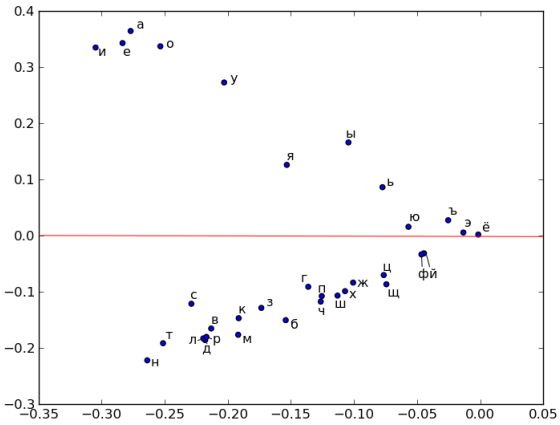
Swedish



Hawaiian



Modern Greek



Russian

Figure 2: x and y for several writing systems.

Language	N
Hungarian	773.7
Tagalog	531.43
Inuktitut	424.12
Vietnamese	359.53
Finnish	240.26
Old English	234.52
Czech	223.96
Spanish	147.44
Russian	135.88
Swedish	121.77
Maltese	104.63
Latin	83.88
Ancient Greek	65.88
Hawaiian	57.29
Middle English	48.21
Serbian	28.07
Modern Greek	20.6
German	20.33
French	16.01
Modern English	-31.05
Portuguese	-53.19
Dutch	-57.18
Afrikaans	-73.52
Italian	-89.94
NVME	-167.63
Farsi	-185.7
Aramaic	-191.23
Hebrew	-207.32
Urdu	-220.01
Arabic	-225.36

Table 4: Values of N for Algorithm 2, calculated over corpora of roughly 5000 words each (min character tokens = 13681, max = 39936, median = 20361). NVME is the Modern English corpus with vowels removed. Abkhaz ($N = -70.94$) is not included because of its small size.

4.2 Algorithm 3: Vowelless words

For writing systems that conventionally use interword whitespace, we can alternatively apply vowel identification to the task of discriminating abjads from vocalic alphabets by examining the percentage of word tokens with no vowel graphemes.[10] This method, Algorithm 3, is implicit to Reddy and Knight's (2011) 2-state HMM analysis of part of the Voynich manuscript, in which they observed that every word was recognized as an instance of the regular language a^*b. They believed the most likely explanation is that every word was written with several consonants followed by a vowel, and that the Voynich manuscript therefore uses an abjad.

[10]In vocalic writing systems, vowelless words include typographical errors, abbreviations and, in some writing systems, words with semivowels that can occupy a syllabic mora, such as 'y' in English.

Language	V	C
Arabic	**3.75**	0.92
Hebrew	**3.63**	0.2
Urdu	**2.58**	0.22
Farsi	**2.35**	0.13
Aramaic	**1.97**	0.18
NVME	0.19	**0.69**
Abkhaz	**0.63**	0.44
Russian	**0.37**	0.29
Maltese	**0.36**	0.06
Vietnamese	0.25	**0.27**
Modern Greek	**0.14**	0.06
Dutch	**0.13**	0.04
Old English	**0.12**	0.11
Hawaiian	**0.12**	0.4
Middle English	0	**0.12**
Spanish	**0.11**	0.08
German	**0.09**	0.04
Tagalog	**0.07**	0.06
Inuktitut	**0.07**	0.05
Italian	**0.07**	0.04
Serbian	**0.07**	0.02
Portuguese	**0.05**	0.05
Afrikaans	**0.05**	0.04
Czech	**0.05**	0.01
Modern English	**0.05**	0.01
Latin	**0.04**	0.03
Finnish	**0.03**	0.03
Swedish	**0.03**	0.03
French	**0.03**	0.02
Hungarian	**0.02**	0.01

Table 5: Percentages of word tokens with no putative vowels (V) or consonants (C), as determined by Algorithm 3.

From this percentage, a decision boundary also emerges at about 1%, as shown in Table 5. NVME is not correctly classified unless one uses the greater of the percentage of words without a vowel or consonant, but this (Modern English with the Once again, putative vowels and consonants have been determined by Algorithm 1.

5 Conclusion and Future Work

We have shown that a very simple spectral decomposition based on character co-occurrences provides nearly perfect performance with respect to classifying both a letter as vowel or consonant and a writing system as an abjad or alphabet. Algorithm 1 does not resolve other pertinent questions, e.g., distinguishing numbers from letters, or determining which capital letters correspond to which lowercase letters. Our method of vowel/consonant classification is meant to inform existing methods of finding graphemes' corresponding sounds. An additional source for associating sound values to graphemes is comparing letter frequencies between two related languages.

Future research on associating sound values to graphemes could include extending a method similar to Algorithm 1 to other types of writing systems, such as syllabaries.

References

Peter T Daniels and William Bright. 1996. *The world's writing systems*. Oxford University Press.

J. Goldsmith and A. Xanthos. 2009. Learning phonological categories. *Language* 85(1):4–38.

Jacques BM Guy. 1991. Vowel identification: an old (but good) algorithm. *Cryptologia* 15(3):258–262.

Young-Bum Kim and Benjamin Snyder. 2013. Unsupervised Consonant-Vowel Prediction over Hundreds of Languages. In *ACL (1)*. pages 1527–1536.

Cleve Moler and Donald Morrison. 1983. Singular value analysis of cryptograms. *American Mathematical Monthly* pages 78–87.

Merle E Ohaver. 1933. *Cryptogram solving*. Etcetera Press, PO Drawer 27100, Columbus, Ohio 43227.

Sravana Reddy and Kevin Knight. 2011. What we know about the Voynich manuscript. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*. Association for Computational Linguistics, pages 78–86.

Gilbert Strang. 2005. *Linear Algebra and Its Applications, 4th Edition*. Brooks/Cole Publishing Company.

Michael Stubbs and Isabel Barth. 2003. Using recurrent phrases as text-type. *Functions of language* 10(1):61–104.

B.V. Sukhotin. 1962. Eksperimental'noe vydelenie klassov bukv s po- moshch'ju elektronnoj vychislitel'noj mashiny. *Problemy strukturnoj lingvistiki* 234:198–106.