Phonetic Distance between Dutch Dialects *

John Nerbonne, Wilbert Heeringa, Erik van den Hout
Peter van der Kooi, Simone Otten, Willem van de Vis
nerbonne@let.rug.nl

Abstract

Traditional dialectology relies on identifying language features which are common to one dialect area while distinguishing it from others. It has difficulty in dealing with partial matches of features and with nonoverlapping language patterns. This paper applies *Levenshtein distance*—a measure of string distance — to pronunciations to overcome both of these difficulties. Partial matches may be quantified, and nonoverlapping patterns may be included in weighted averages of phonetic distance. The result accords with traditonal dialectology to a satisfying degree.

1 Introduction

This paper applies a string distance measure—*Levenshtein distance*—to phonetic data in order to obtain a measure of the distance between words in different dialects. The average distance was then interpreted as a measure of the distance between the dialects themselves, which were examined to obtain dialect groups. The transcriptions of one hundred different words as these are pronounced in a sample of twenty Dutch dialects were compared, and average phonetic distances between dialects were then calculated and compared to determine which dialect variants are closest. The results reconstruct perfectly the traditional division of Dutch dialects into Lower Saxon, Frisian, Franconian and Flemish.

The paper is structured as follows: the next section reports on traditional dialectology and Kessler (1995), who applied Levenshtein distance to Irish Gaelic dialectology, and is followed by a summary of data sources and manipulation. We then turn to an extended presentation of Levenshtein distance and a briefer synopsis of hierachical agglommerative clustering, which was used to determine groups once the average phonetic distance had been determined. We close with a summary, including a discussion of potential problems and interesting directions for further work.

*We received useful advice and criticism from Jo Daan, Ger de Haan, Jack Hoeksema, Cor Hoppenbrouwers, Brett Kessler, Hermann Niebaum, and Alan Swanson. Johan Dijkhuis helped in the project as well.

2 Dialectology

A dialect is a language variant peculiar to a limited region.[1] Although the variant may be distinctive at any level of linguistic analysis, it always includes pronunciation difference, which may or may not correlate with differences in morphology or more abstract linguistic levels. We focus exclusively on pronunciation here. Dialectology is pursued for its intrinsic interest, and then also for the record of cultural history it provides, including migrations, contacts with other peoples, and internal cultural divisions. But knowing how language diversity is distributed geographically may also be of use to language learners, publishers, broadcasters, educators and language planners.

The primary tool of traditional dialectology has been the *isogloss*, the delineation of a concrete language variation on a map. There is a well-known example of a Dutch variation in pronunciation of final [n] in words such as *lopen*, pronounced [lo:pə̃n], [lo:pə̃].[2] If we were to plot its occurrence on a map we could divide one set of occurrences from another with a line on the map. This isogloss would show that in the northeast of the Netherlands, the [n] is retained, while it is dropped elsewhere. Language variants distinguished by many isoglosses emerge then as relatively distinct *dialects*.

But dialectologists recognize that the method of isoglosses does not result in the delineation of "dialects" satisfactorily:

1. It can say little systematically about degrees of overlap in language features, e.g. the pronunciations [lo:pə], [lo:pn̩], [lo:pm̩] of *lopen*, in which the nasality of the final vowel is lost, the schwa is elided, and the nasal is assimilated in place. Since, as Trudgill (1983, p. 51) suggests, "[...] isoglosses usually mark transition zones rather than discrete breaks [...]", varying degrees of overlap are common, not exceptional.

2. Although dialects are supposed to emerge when several isoglosses coincide geographically, this doesn't always happen. In fact, exact mappings of different linguistic features are often at odds with each other, and thus do not jibe with the notion of dialect as arising from accummulation of isoglosses. See (Veith, 1994), (Herrgen, 1994) for examples.

3. Although the method of isoglosses highlights distinctions in a way useful to experts, dialects must ultimately be identified by a dialectologist, rather than through an objective procedure. This is necessitated by the existence of transitional areas and non-coinciding isoglosses noted above.

[1] See (Niebaum, 1983) for a general introduction to dialectology and (Goossens, 1977) for an introduction to Dutch dialects.

[2] Dr. Jo Daan cautions that the (non)appearance of the final [n] is morphologically conditioned in some dialects.

Especially the first two shortcomings suggest that a numerical weighting is needed if the notion "dialect" is to be applied to larger areas. The weighting should reflect (i) degree of overlap and (ii) the number or importance of shared features, but it should allow a more sensitive comparison of features whose geographic distribution does not coincide perfectly.

Kessler (1995) advocates the use of a string distance metric, Levenshtein distance, as a means of calculating the distance between the pronunciations of corresponding words in different dialects. He calculated this distance for pairs of pronunciations of words in many Irish-speaking towns. The measure provides solutions, or, at least avenues toward solutions, for all of the problems above.

1. The measure is sensitive to degrees of overlap.

2. Given the employment of such a measure, one may include *all* the data available, including that which does not follow other isoglosses neatly. The method is robust enough to do this.

3. One may analyse the averages statistically, which may lead to an objective identification of dialects (and indeed does in the cases in which it has been used thus far).

The present paper reports on applying Kessler's method to Dutch dialects.

Of course, there exist alternative attempts to specify a notion of distance between dialects. Hoppenbrouwers and Hoppenbrouwers (1988) propose counting the frequency with which phonetic features are realized, and Babitch and LeBrun (1989) use relative lexical frequency. The present proposal is based on a measure of pronunciation difference of corresponding words. This is preferable to frequency-based approaches because it compares the pronunciation of corresponding words directly, and thus provides an implementation of traditional dialectology. The frequency-based approaches work to some degree because differences in pronunciation lead to differences in frequency, but using frequency directly is relatively insensitive—sounds may share frequency without corresponding.[3]

3 Data and Representation

Our first objective was to gather data for twenty dialects in Dutch and to make it electronically manipulable. This section focuses on the dataset used and how it was digitized. The coding of the phonetic transcription had to be consistent and unambiguous, and it had to be easily encoded and manipulated electronically.

[3] In this context, it is worth noting that neither of the works cited raises the issue of whether the differences they find are statistically significant.

3.1 Source and Manipulation of Data

bang	kippen	bloemen	mijn	vriend
spinnen	werk	brood	schip	splinter
timmerman	vinger	fabriek	bier	glazen
vier	twee	kersen	drie	ik
vastenavond	spinnewebben	paddestoel	pet	breder
breed	breedst	duivel	beesten	keelpijn
bezem	steel	neen	geroepen	peer
geld	beschermen	vrouw	zwemmen	sterk
bed	optillen	metselaar	weddenschap	boterham
jaar	water	potten	maart	paard
pater	zwaluwen	kaarten	kaas	motor
avond	dag	jongetje	kar	rozen
dopen	kindje	dochtertje	rijkdom	mond
weg	liedje	schaduw	kelder	ossebloed
voeder	broer	bergen	Italië	spuwen
vuur	deur	naaien	brouwer	bakken
eikels	eik	groen	hooi	boompje
pastoor	huis	koe	uier	melk
kruisen	kruiwagen	Duitsers	geslagen	blauw
sneeuw	eeuwigheid	stad	soldaten	gebonden

Table 1: The words whose pronunciation was compared in twenty dialects.

Our intention was to compare the pronunciation of about one hundred common words in twenty Dutch dialects. The Reeks Nederlands Dialectatlassen (Blacquart and others, 1925–1982) (henceforth RND) is a series of atlases containing the transcriptions of the pronunciation of 141 sentences and other multiple-word utterances in Dutch dialects (including Belgian Flemish). The material in RND was carefully designed by mid-century dialectologists to represent the range of Dutch dialect variation—it was by no means "arbitrarily chosen." Out of these items we picked the one hundred common words shown in Table 1. We attempted to include a variety of sounds in selecting these 100 items. Early choices were modified when it appeared the word was redundant (e.g. 'spin' *spider* and 'spinneweb' *cobweb* were both in the original list, which seemed redundant). The RND also includes accompanying maps with isoglosses of words and phrases, but all material is available only in a printed form: there is no electronic version. Therefore, we had to find an encoding which preserved all information present in RND.

The dialectologists who collaborated on the RND tried to transcribe consistently, but it was compiled over a period of thirty years, starting in 1936. Goossens (1965) identifies several unreliable points. The RND is the work of

Figure 1: The locations of the dialects studied.

many phoneticians and phonologists, who tried to maintain consistency in transcription, but there are differences in transcription which seem to reflect transcribers as much as they do phonetic reality. In several volumes in RND a slightly different notation has been used. This is particularly evident in matters of phonetic detail as transcribed in diacritics (see (Goossens, 1965)). The long period of compilation led occasionally to phonetically incommensurable data which was subsequently excluded from analysis. For example, before World War Two one encounters the word *Duitser*, German, while after the war the term of abuse *mof* surfaces. Nonphonetic variation of this kind has been ignored, but we note its potential significance. We were not in a position to try to correct for such effects—and simply digitized exactly what we found.

The dialects were selected from the RND with an eye to obtaining an even spread across the Netherlands and Flemish Belgium. Choices for particular dialects were arbitrary, arising from individual taste and history. A full list of dialects used can be found in Table 2, and a map showing where they are spoken is in Figure 1.

3.2 Encoding

Starting with the data present in RND, a computer-readable notation had to be specified which allowed for electronic processing of the transcription. We

Province	*Place-name*	*Identification*	
Friesland	Kollum	k.07.1	B.35
	Leeuwarden	k.07.1	B.56
	Nes	k.07.1	B.2
Groningen	Groningen	k.08.1	C.108
	Winschoten	k.08.1	C.161
Drenthe	Assen	k.08.1	G.4
	Roswinkel	k.08.1	G.39
Overijsel	Ommen	k.06.1	G.112
	Almelo	k.06.1	G.173
	Steenwijk	k.06.1	F.60
Gelderland	Aalten	k.04.1	N.9
	Putten	k.04.1	F.124
Noord-Holland	Schagen	k.05.1	E.9
Zuid-Holland	Delft	k.09	E.198
Utrecht	Soest	k.09	E.164
Noord-Brabant	Gemert	k.03.2	L.207
	Oosterhout	k.03.1	L.63
Limburg	Venray	k.03.2	L.210
Flemish Belgium	Lebbeke	k.02.1	I.264a
	Mechelen	k.02.1	K.330

Table 2: The dialects used and their volume numbers (k.NN[.N]) and identifying numbers as found in the RND in the Groningen *Letterenbibliotheek*.

further decided to retain all the phonetic detail given in RND, deciding that this would be the most generally useful (including to other users). Specifying a consistent and unambiguous notation turned out to be a subtle problem. The main demand on our code was using only the first, reliable, 128 ASCII codes (to be precise: ASCII 32 to ASCII 126, inclusive). This maximizes the probability of correct transport between platforms (e.g. Unix and DOS).

The strategy of using semi-standard computer codes for IPA ((Association, 1949; Burnage, 1990)) ran into the problem that the symbols used in RND do not conform to the IPA. Programs to translate RND to IPA would be easy to write if the correspondences were clear, but they were not. We therefore abandoned this strategy, reasoning that translating into this would introduce errors, and we sought a coding scheme which followed the RND as closely as possible.

The strategy of sticking to visually close ASCII equivalents as closely as possible has the drawback that it results in unusual codings, where we risked data-entry error. For example, the [ŋ] has no visually close representative. We considered representing it as two characters in square brackets ([ng]), but this seemed to conflate segments and diacritics, where multicharacter

Non-standard symbols			Diacritic symbols		
RND	Code	Description	RND	Code	Description
ŋ	N	capital n	ˑ	.	point
ɲ	J	capital j	ː	:	colon
ʋ	W	capital w	ˁ	‘	reversed apostrophe
ʃ	S	capital s	,	,	comma
ʒ	Z	capital z	)	1	one
ɡ	G	capital g	)	2	two
ʔ	?	question mark	(	3	three
ɛ	E	capital e	-(	4	four
ɑ	A	capital a	˔	{	left brace
ɔ/ 2	)	left parenthesis	˕	}	right brace
ʊ	O	capital o	˖	>	greater than
ø/ ω	0	null	˗	<	smaller than
œ	Y	capital y	˅	5	five
ʌ/ Ω	U	capital u	˄	6	six
ə	@	commercial at	˜	~	tilde
a	a	small a	ˈ	7	seven
∩	#	hash	’	’	apostrophe
∪	$	dollar	.	8	eight
æ	Q	capital q			

Table 3: The encoding scheme.

representation is necessary. We settled on the mapping in which all symbols which had a close equivalent in ASCII were used, and which additionally made use of Table 3.

We note several further problems and the solutions adopted (where appropriate).

- Word boundaries are not marked (consistently). ‘De kippen’ *the chickens* appeared as [d@kIp@n]. We coded the one word we wanted (here, ’chicken’ as [kIp@n]).

- As a special case of boundaries, we encountered cases in which similar adjacent segments were recorded only once.

Het	is	een	mooie	dag	geweest
It	PERF-AUX	a	nice	day	be-PRT

  ‘It’s been a nice day’

  In this example the [x] (‘g’) of *dag* and of *geweest* merge. The resulting pronunciation of *dag geweest* in Gemert is [daxəwIst]. Since we were comparing words, not phrasal phonology, we recorded [dax] for *dag*.

- Hand-written atlases proved difficult to read and slowed coding time significantly. Fine graphic distinctions are easy to miss, such as the difference between the open o [ɔ] and the schwa [ə].

- Some data were missing. It was difficult to determine the reason for this. Sometimes the field linguist seems to have forgotten to transcribe completely—and an item ends halfway. In other cases, entirely different items are recorded. In exactly these cases, we noted the data as *missing*, and coded a vertical bar with whitespace on either side: " | ".

- Sometimes we find phrases where we expect words. *keelpijn* 'sore throat' appears as *pijn in de keel* 'ache in the throat' (Oosterhout). We retained these, reasoning that this strategy made the list more useful, but we modified the distance metric to prevent this decision from carrying too much weight.

4 Levenshtein Distance

We want to know how closely dialects in the Netherlands and north Belgium resemble each other phonetically. To do this, we compare their words and impose a measure of phonetic distance on them. Several distance measures exist which compare words of the same length (Kruskal, 1983, p. 1), but Levenshtein distance relaxes this restriction. This section explains Levenshtein distance and an algorithm for its efficient computation.

4.1 String Operations

Fundamental to the idea of Levenshtein distance is the notion of string-changing operations. To determine the extent to which two strings differ from each other, you check what operations could change one string to another. The operations available are:

Substitutions One character is replaced by another.

Indels Insert a letter in the string or delete a letter from the string ('indel' is a portmanteau for *in*sertions and *del*etions).

Next, values are assigned to the operations. Substitutions count two, and indels one. Because a substitution is always equal in effect to a combination of an insertion and a deletion, it counts just as the sum of these separately. We illustrate two sets of operations which change 'industry' to 'interest':

		0	.	.	.	j	.	.	.	m
			i	n	d	u	s	t	r	y
0										
.	i									
.	n									
.	t									
i	e									
.	r									
.	e									
.	s									
n	t									

Figure 2: Sample `dist` array

industry	delete d	1
inustry	delete u	1
instry	subst. s/y	2
instrs	insert e	1
insters	insert e	1
insteres	delete s	1
interes	insert t	1
interest		
Total cost		8

industry	delete y	1
industr	delete r	1
indust	subst. r/d	2
inrust	subst. e/u	2
inrest	insert t	1
intrest	insert e	1
interest		
Total cost:		8

The examples illustrate that it is possible to change one word to the other in many ways. In this case the analyses are equal in cost, but in general they need not be. We are interested in the set of operations with the least cost which change w_1 into w_2. This is Levenshtein distance, $dist(w_1, w_2)$.

Given that there are many (actually, infinitely many) different sets of operations which map w_1 to w_2, it is not obvious how to determine the least set, even less how to determine it efficiently. The *Levenshtein algorithm* accomplishes this, however.

4.2 Levenshtein Algorithm

The words are compared on the basis of their characters. In order to speak generally, we define a weight function which assigns a cost to substituting one character for another, and also to inserting and deleting characters. The cost of substitutions is two, and the cost of insertions and deletions is one, and we examine more discriminating weight-cost functions in connection with phonetic symbols to which diacritics have been added in Section 4.3.

To compute the distance between two words, we use a two-dimensional array `dist` of size $(l(w_1) + 1, l(w_2) + 1)$ where $l(w)$ is the length of word w (see Figure 2).

The cell `dist`$[0,0]$ gets the value 0. The other cells in row 0 (`dist`$[0,j]$) are filled with the value of index j, and the cells in column 0 (`dist`$[i,0]$) get the value of index i. We continue by assigning values to the rest of `dist`, row by row. We begin with row 1, and within each row, we always begin with column 1. We call the current column number j and the current row number i. For each cell in the array, we always have to look at three possibilities (to obtain a minimum):

1. *Deletion* of the j-th character from word1:
   We determine weight(word1[j],ϵ) and add it to the value in the cell above the current one: `dist`$[i-1,j]$. The sum is assigned to the temporary variable `above`.

2. *Substitution* of the j-th character of word1 by the i-th character of word2:
   We look up weight(word1[j],word[i]), and take the sum of this + the value in the element above and to the left of the current cell, i.e. `dist`$[i-1,j-1]$. The sum is kept in the temporary variable `aboveleft`.

3. *Insertion* of the i-th character in word2:
   We take weight(ϵ,word2[i]) and add this to the value in the cell to the left of the current cell, i.e. `dist`$[i,j-1]$. The sum is retained in the temporary variable `left`.

Now, we take the minimum of the three values, `above`, `aboveleft` and `left`, and the current cell takes it as value:

$$\texttt{dist}[i,j] \leftarrow min(\texttt{above, above-left,left})$$

Once we've traversed the entire array, and computed values for all cells, then the distance—the least cost of operations mapping from word1 to word2—is found in the cell `dist`$[l(\text{word2}),l(\text{word1})]$. This is the Levenshtein distance between the strings. It is easy to see that it corresponds to the value of a set of operations which will map the first word into the second. To note further that these sets of operations correspond to paths through the matrix, examine the matrix in Figure 3.
The path of filled cells here corresponds to the sequence of operations listed with values on the right. Any such path must correspond to a mapping from the first word to the second.

Finally we note that paths arise only by adding minimally to minimal-cost cells. This guarantees that the least distance is computed.[4] The algorithm in pseudo-code is shown in Figure 4.

[4]The path through the matrix shown above is misleading in the sense that it does not show that link from one cell to an adjacent cell builds on the minimal value of the first. This can only be done if *all* the values are filled in.

		i	n	d	u	s	t	r	y
	0								
i		0							
n			0						
t			1						
e			2						
r			3	4	5				
e					6				
s						6			
t							6	7	8

industry	subst. i/i	0
industry	subst. n/n	0
intdustry	insert t	1
intedustry	insert e	2
interdustry	insert r	3
interustry	delete d	4
interstry	delete u	5
interestry	insert e	6
interestry	subst. s/s	6
interestry	subst. t/t	6
interesty	delete r	7
interest	delete y	8
Total cost		8

Figure 3: Correspondence between paths in the matrix and sets of operations

If we simply used this distance, it would tend to bias measurements so that changes in longer words would tend to contribute more toward the average phonetic distance (since they tend to involve more changes). This may be legitimate, but since words are a crucial linguistic unit we chose to stick to average word distance. This involves the computation of *relative distance*, which we get by dividing the absolute distance by the length of the larger word.

4.3 Variants and Implementations

We furthermore explored a variant of distance measures in which the `weight` function was made sensitive to the role of diacritics. We were motivated to explore this option because we suspected that the difference between [ə] and [ə̃] might be overvalued (if it is the same as the difference between [ə] and [s], for example). In that version, the substitution of one letter for another involving only a change in diacritic was valued at 0.2 (rather than 2). Although this affected the determination of distances and relative distances considerably, it had no effect on the final relative dialect distances assigned. The less discriminating measure was every bit as effective in determining dialects.

The algorithm was implemented both in C and in Pascal and is available at the dialect web site (`http://thok.let.rug.nl/dialects/`).

5 Dialect Distance

For each pair of the twenty variants studied, their average phonetic distance was computed. Missing words in one or both of the dialects were ignored,

	Almelo	Assen	Delft	Gemert	Groningen	Kollum	Lebbeke	Leeuwarden	Mechelen	Ness	Ommen	Oosterhout	Putten	Roswinkel	Schagen	Soest	Steenwijk	Venray	Winschoten
Aalten	68	66	80	68	63	74	89	73	90	74	63	67	51	59	77	82	68	76	76
Almelo		41	82	84	73	73	90	73	90	76	65	81	74	36	78	90	67	84	43
Assen			68	74	63	59	88	64	94	66	55	77	63	40	75	79	60	78	55
Delft				67	82	84	83	75	77	79	85	66	77	81	61	75	81	72	87
Gemert					80	83	77	78	75	81	84	56	72	82	72	71	82	66	91
Groningen						84	88	79	88	85	51	79	74	65	77	93	63	78	75
Kollum							98	40	101	39	75	79	68	71	69	80	79	82	75
Lebbeke								88	51	89	91	79	89	91	90	91	89	78	92
Leeuwarden									97	42	75	76	66	71	63	76	75	74	76
Mechelen										90	92	82	93	91	85	87	89	76	93
Ness											80	79	69	72	68	83	83	81	80
Ommen												80	69	54	80	86	46	86	69
Oosterhout													68	81	71	68	81	62	88
Putten														68	70	73	74	77	81
Roswinekl															82	88	60	86	40
Schagen																78	80	70	88
Soest																	87	78	93
Steenwijk																		86	74
Venray																			92
Winschoten																			

Table 4: The average relative phonetic distance between twenty Dutch dialects as determined by the Levenshtein measure, where distances are given in hundredths of the distances units calculated directly. The average per-word distance is 56 ($\sigma = 11$), while the average per-dialect distance is 75. Since sample size was 100, standard error is $11/\sqrt{100} = 1.1$, so that distances less than 73 or more than 77 are significant above the .05 level.

```
word_distance(w1,w2:wordtype) :  real;

begin
    dist[0,0] := 0;
    for i:=1 to length_w2 do
        dist[i,0]:=i;
    for j:=1 to length_w1 do
        dist[0,j]:=j;
    for i:=1 to length_w2 do
    begin
        for j:=1 to length_w1 do
        begin
            above:=dist[i-1,j]+weight(w1[j],o);
            aboveleft:= dist[i-1,j-1]+weight(w1[j],w2[i]);
            left:= dist[i,j-1]+weight(o,w2[i]);
            dist[i,j]:= min(left,aboveleft,above);
        end;
    end;
end;
```

Figure 4: Levenshtein algorithm in pseudo-code. The algorithm works dynamically, so that, for each p_1,p_2 prefixes of word1, word2, it determines the least cost of operations mapping p_1 to p_2. The version used here adds a step relativizing the distance measure to the word length (of the longer word).

since we wished to focus on phonetic distance. A penalty for lexical differences might have conflated factors. This results in a measure of average relative phonetic difference for each of the 190 ($= \binom{20}{2}$) pairs of dialects. We present these in a distance matrix in Table 4.

Since each of the 190 dialect comparisons involved (nearly) 100 words, nearly $19,000$ word comparisons were made. The mean per word difference was 56, where σ was 11. The standard error for averages of sets of comparisons of size n is $\sigma/\sqrt{n}$, yielding here 1.1. Thus average dialect distances over 2.8 are significant at the 0.01 level. All of the dialect clusters are significantly different at this level.

6 Clustering Dialects

Given the distance matrix calculated above, it is natural to investigate groups of larger sizes. We sought clusters of phonetically close dialects by applying *hierarchical agglommerative clustering*,[5] which we refer to simply as 'clustering'. For illustrative purposes we use the matrix below, taken from Table 4.

	Assen	Delft	Kollum	Nes	Soest
Assen		68	59	66	79
Delft			84	81	75
Kollum				39	80
Nes					83
Soest					

Naturally, the (i, i) cells are all zero (representing the distance of a dialect to itself), and matrix is symmetric about the diagonal, allowing us to ignore one half.

Clustering is most easily understood procedurally. At each step of the procedure we select the shortest distance in the matrix, and then fuse the two data points which gave rise to it. Since we wish to iterate the procedure, we have to assign a distance from the newly formed cluster to all the remaining points. For this purpose we take a weighted average of the distances from each of the points in the cluster. The weighting is determined by the size of the elements being clustered. If the distance between i and j is minimal, then we form a cluster $C(i,j)$ and calculate the distance from each k to the new cluster:

$$\texttt{dist}(k, C(i,j)) = \frac{|i|}{|i|+|j|}(\frac{1}{|i|}\sum_i \texttt{dist}(\mathrm{i,k})) + \frac{|j|}{|i|+|j|}(\frac{1}{|j|}\sum_j \texttt{dist}(\mathrm{j,k}))$$

where $|k|$ is the number of elements in k. Note that the right-hand side is the sum of two terms, each of which is the product of a weighting ($\frac{|i|}{|i|+|j|}$), and an average distance.

In the distance matrix above Kollum and Nes are closest. If we fuse them, we need to calculate distances from the new cluster to each of the remaining elements. For example, the distance from Assen (a) to the Kollum(k)-Nes(n) cluster is calculated as follows:

$$\begin{aligned}\texttt{dist}(a, C(k,n)) &= \tfrac{|k|}{|k|+|n|}(\tfrac{1}{|k|}\textstyle\sum_k \texttt{dist}(k,a)) + \tfrac{|n|}{|k|+|n|}(\tfrac{1}{|n|}\textstyle\sum_n \texttt{dist}(n,a)) \\ &= \tfrac{1}{2}(\tfrac{1}{1}\textstyle\sum_k \texttt{dist}(k,a)) + \tfrac{1}{2}(\tfrac{1}{1}\textstyle\sum_n \texttt{dist}(n,a)) \\ &= (\tfrac{1}{2}\times 59) + (\tfrac{1}{2}\times 66) \\ &= 62.5\end{aligned}$$

[5] See (Kaufman and Rousseeuw, 1990) or (Aldenderfer and Blashfield, 1984) for a general introduction. The technique is commonly applied in history (Boonstra, Doorn, and Hendrickx, 1990, pp. 143 ff.), but also finds application in psycholinguistics (Woods, Fletcher, and Hughes, 1986, pp. 249 ff.).

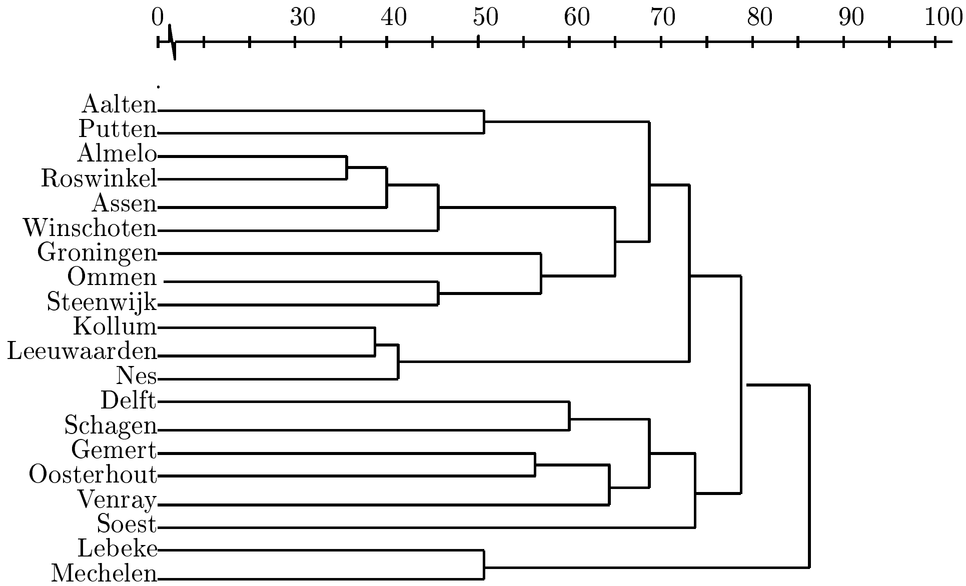

Figure 5: Clusters of Dutch dialects. Branchings appear at mean Levenshtein (phonetic) distance. Note that the last four clusters to emerge (joined beyond 70) correspond to the Lower Saxon, Frisian, Franconian, and Flemish dialect areas.

So the first step just results in the average of the two distances, as should be expected. If we calculate the remaining differences we obtain the distance matrix below (new values are shown in bold, old values in normal font). Notice that the resulting distance matrix is a unit smaller in each dimension.

	Assen	Delft	Kollum & Nes	Soest
Assen		69	**62.5**	79
Delft			**82.5**	78
Kollum & Nes				**81.5**
Soest				

We continue clustering the distance matrix, reducing it in every iteration, until there is nothing left to cluster. The result is a complete, hierarchical grouping of dialect variants. The closest variants show up in a group by themselves, but this group is part of a larger one, which may be integrated into a yet larger one, etc. The results may be displayed in a *dendrogram*, as in Figure 5. This display of results resembles the 'family trees' used in historical linguistics, but note that the length of edges is significant, corresponding to phonetic distance. Unlike family trees, its groupings need not be interpreted genealogically, as indicating a common ancestor (though of course, this may often be the most plausible explanation for the phonetic similarity they demonstrate).

Clustering is an explorative technique, i.e. its results are not guaranteed to be optimal: The optimal clustering would have to be selected by exhaustive search, which is infeasible for large numbers of elements (there are $\frac{1}{n+1}\binom{2n}{n}$ clusterings of a set of n elements, or about 8×10^8 for a group of 20). Nor do results have to be statistically significant, but results can be checked for significance once they are obtained. In fact, all of the clusters distinguished are statistically significant at the 0.05 level, and the larger divisions are very significant. Thus this approach confirms confirms the traditional scholarship dividing Dutch into Lower Saxon, Frisian, Franconian and Flemish. Note that these clusters contain the most elements, and that their distances to each other are also great.[6]

7 Conclusions and Prospects

The purpose of initial work with new methods should be the verification of the methods themselves, which motivates our applying the dialect identification techniques to the well-studied case of Dutch. The introduction of a numerical distance solves problems in traditional methodology, or points to possibilities for their solution. We promised to contribute to solutions to the following problems.

degrees of dialect overlap The technique is sensitive to partial overlap both in the distance metric employed as well as in the weighted averages employed.

comprehensive attention to data The measure may sensibly be applied to *all* the data available, including that which does not follow dialect boundaries or other isoglosses neatly. The method is robust.

objective An objective identification of dialects arises through statistical analysis of the average distances.

The primary result of the paper is confirmation of the methods suggested by Kessler for Irish Gaelic. Naturally the methods are still only promising candidates which should be tested further and refined, but the initial results are most encouraging.

Prospects for further development are promising. Certainly one could attempt application to more dialects, and we are investigating the correlation between phonetic and geographic distance, as Séguy (1971) investigates

[6] Frisian scholars suggest that Frisian ought to be more distant from Franconian and point out that Kollum is quite close to the Frisian border (to Lower Saxony), and that the other two Frisian sites are regarded as 'stadsfries'. They furthermore regard Frisian as most distant from other dialects, which would mean that the highest dendrogram branching should separate Frisian from all other Dutch varieties. But the RND volume on Frisian notes that Frisian has accepted a great deal from Lower Saxon over the centuries of their contact.

the correlation between geographical and lexical distance. The present application may be criticised for proceeding from hand-picked data (RND), and application to arbitrarily chosen data (perhaps including frequency) would be interesting. Kessler himself modified the Levenshtein measure to use phonetic feature make-up, but our preliminary studies do not indicate significantly different results here. Finally, it would be interesting to apply these techniques to situations in which language genealogies are at issue, as Kruskal, Dyen, and Black (1971) and Batagelj, Keržic, and Pisanski (1992) have done, but this would seem to require some means of excluding similarities due to borrowing.

References

Aldenderfer, Mark S. and Roger K. Blashfield. 1984. *Cluster Analysis.* Quantitative Applications in the Social Sciences. Sage, Beverly Hills.

Association, International Phonetics, 1949. *The Principles of the International Phonetic Association.* London.

Babitch, Rose Mary and Eric LeBrun. 1989. Dialectometry as computerized agglomerative hierarchical cluster analysis. *Journal of English Linguistics*, pages 83–90.

Batagelj, Vladimir, Damijana Keržic, and Tomaž Pisanski. 1992. Automatic clustering of languages. *Computational Linguistics*, 18(3):339–343.

Blacquart, E. et al. 1925–1982. Reeks nederlandse dialectatlassen.

Boonstra, Onno, Peter Doorn, and Fran cois Hendrickx. 1990. *Voortgezette Statistiek voor Historici.* Coutinho, Muiderberg.

Burnage, G., 1990. *CELEX: A guide for users.* Nijmegen.

Goossens, J. 1965. Die niederländische strukturgeographie und die 'reeks nederlandse dialectatlassen'. *Bijdragen en Mededelingen van de Koninklijke Nederlandse Akademie van Wetenschappen*, 29.

Goossens, J. 1977. *Inleiding tot de Nederlandse Dialectologie.* Wolters-Noordhoff, Groningen.

Herrgen, Joachim. 1994. Kontrastive dialektgeographie. In Klaus Mattheier and Peter Wiesinger, editors, *Dialektologie des Deutschen*, volume 147 of *Reihe Germanistische Linguistik.* Niemeyer, Tübingen, pages 131–164.

Hoppenbrouwers, Cor and Geer Hoppenbrouwers. 1988. De featurefrequentiemethode en de classificatie van nederlandse dialecten. *TABU: Bulletin voor Taalwetenschap*, 18(2):51–92.

Kaufman, Leonard and Peter J. Rousseeuw. 1990. *Finding groups in data : an introduction to cluster analysis.* Wiley, New York.

Kessler, Brett. 1995. Computational dialectology in Irish Gaelic. In *Proc. of the European ACL*, pages 60–67, Dublin.

Kruskal, Joseph. 1983. An overview of sequence comparison. In David Sankoff and Joseph Kruskal, editors, *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison.* Addison-Wesley, Reading, Mass., pages 1–44.

Kruskal, Joseph B., Isidore Dyen, and Paul Black. 1971. Some results from the vocabulary method of reconstructing languages trees. In *Lexico-Statistics in Genetic Linguistics.* Yale University, New Haven.

Niebaum, Hermann. 1983. *Dialektologie.* Niemeyer, Tuebingen.

Séguy, Jean. 1971. La relation entre la distance spatiale et la distance lexical. *Revue de Linguistique Romane*, 35:335–357.

Trudgill, Peter. 1983. *On Dialect. Social and Geographical Perspectives.* Blackwell, Oxford.

Veith, Werner H. 1994. Quantitative dialektologie, computerkartographie. In Klaus Mattheier and Peter Wiesinger, editors, *Dialektologie des Deutschen*, volume 147 of *Reihe Germanistische Linguistik.* Niemeyer, Tübingen, pages 193–244.

Woods, Anthony, Paul Fletcher, and Arthur Hughes. 1986. *Statistics in Language Studies.* Cambridge University Press, Cambridge.