Comparison of Dutch Dialects

Master Thesis

by

Martijn B. Wieling

University of Groningen June 2007

Supervisors
Prof.dr. John Nerbonne
Prof.dr. Gerard Renardel de Lavalette

Ze is niet meer
Waar ze was,
Maar altijd
Waar wij zijn.

In liefdevolle herinnering aan Ankie Wieling

** 16 juli 1950*
† 31 december 2004

Abstract

Contemporary Dutch dialects are compared using the most recent Dutch dialect source available: the Goeman-Taeldeman-Van Reenen-Project data (GTRP). The GTRP consists of phonetic transcriptions of 1876 items for 613 localities in the Netherlands and Belgium gathered during the period 1980 – 1995. In this study three different approaches will be taken to obtain dialect distances used in dialect comparison.

In the first approach the GTRP is analysed using the Levenshtein distance as a measure for pronunciation difference. The dialectal situation it represents is compared to the analysis of a 350-locality sample from the *Reeks Nederlands(ch)e Dialectatlassen* (1925 – 1982) studied by Heeringa (2004). Due to transcriptional differences between the Netherlandic and Belgian GTRP data we analyse data from the two countries separately.

The second approach consists of using Pair Hidden Markov Models to automatically obtain segment distances and to use these to improve the sequence distance measure. The improved sequence distance measure is used in turn to obtain better dialect distances. The results are evaluated in two ways, first via comparison to analyses obtained using the Levenshtein distance on the same datasets and second, by comparing the quality of the induced vowel distances to acoustic differences.

In the final approach we propose two adaptations of the regular Levenshtein distance algorithm based on psycholinguistic work on spoken word recognition. The first adaptation follows the idea of the Cohort Model which assumes that the word-initial part is more important for word recognition than the word-final part. The second adaptation follows the idea that stressed syllables contain more information and are more important for word recognition than unstressed syllables. Both algorithms are evaluated by comparing them to the results using the regular Levenshtein distance on several data sets.

Samenvatting

Nederlandse dialecten worden vergeleken op basis van de meest recente dialectverzameling momenteel beschikbaar: de Goeman-Taeldeman-Van Reenen-Project data (GTRP). De GTRP bestaat uit fonetische transcripties van 1876 elementen voor 613 locaties in Nederland en België verzameld gedurende de periode 1980 – 1995. In dit onderzoek worden drie verschillende methodes gebruikt om dialectafstanden te bepalen.

In de eerste methode wordt de GTRP geanalyseerd met behulp van de Levenshtein afstand. De Levenshtein afstand wordt hierbij gebruikt als een maat om uitspraak verschillen te meten. De resultaten worden vergeleken met eerdere resultaten (Heeringa, 2004) op basis van 350 plaatsen uit de *Reeks Nederlands(ch)e Dialectatlassen* (1925 – 1982). Door transcriptieverschillen tussen de Belgische en Nederlandse GTRP data, analyseren we de data van de twee landen afzonderlijk.

De tweede methode bestaat uit het gebruiken van Pair Hidden Markov Modellen voor het automatisch leren van segmentsafstanden. Deze segmentsafstanden worden gebruikt voor het bepalen van verbeterde woordafstanden die op hun beurt weer gebruikt worden om betere dialectafstanden te bepalen. De resultaten worden op twee manieren geëvalueerd. Ten eerste worden de resultaten vergeleken met de resultaten die verkregen zijn door gebruik te maken van het Levenshtein algoritme. Ten tweede wordt de kwaliteit van de geleerde klinkerafstanden vergeleken met akoestische klinkerafstanden.

In de laatste methode worden op basis van psycholinguistisch onderzoek met betrekking tot het begrijpen van spraak twee aangepaste versies van het Levensthein algoritme geïntroduceerd om dialectafstanden te meten. De eerste aanpassing volgt het idee van het Cohort Model. Hierin wordt verondersteld dat het initiële gedeelte van het woord belangrijker is bij woordherkenning dan het laatste gedeelte van het woord. De tweede aanpassing volgt het idee dat beklemtoonde lettergrepen meer informatie bevatten dan de onbeklemtoonde lettergrepen. Beide algoritmes worden geëvalueerd door de resultaten te vergelijken met de resultaten van het reguliere Levenshtein algoritme voor verschillende data sets.

Acknowledgements

First and foremost I would like to thank my main supervisor, John Nerbonne. He was always ready to answer any of my questions and has been invaluable as a co-author of the papers we have written on the basis of this research. I also am thankful for the useful comments and questions of my second supervisor, Gerard Renardel de Lavalette.

I am grateful to Greg Kondrak of the University of Alberta for providing the original source code of the Pair Hidden Markov Models and Fokke Dijkstra of the Donald Smits Centre for Information Technology of the University of Groningen for assisting me in parallelising the PairHMM source code. I am thankful to Peter Kleiweg for making the L04 software package available which was used to obtain dialect distances using the Levenshtein algorithm and to create the maps in my thesis. I thank the Meertens Instituut for making the GTRP dialect data available for research and especially Boudewijn van den Berg for answering our questions regarding this data.

I would also like to thank Therese Leinonen en Wilbert Heeringa. They were invaluable as co-authors of the papers we have written and cooperation with them was very pleasant.

Last but not least, I would like to express my warm gratitude for the support of my family and my love Aafke during this project.

Maximas tibi gratias ago!

Contents

1 Introduction

In the Netherlands and the northern part of Belgium (Flanders) the official language is Dutch. However, when travelling through this area one will come across many regional variants of the Dutch language (dialects). Dialects tend to be similar to nearby dialects, while they are generally more dissimilar to dialects spoken in a more distant region. For example, consider the word 'stones'. The standard Dutch pronunciation of this word is [stenə]. However, in the north of the Netherlands (Groningen) one will often hear [stɑin], while in the south of the Netherlands (Limburg) [stɛin] can be heard. As another example, consider the Dutch word 'to drink' which is pronounced as [drɪ ŋkə]. In the north of the Netherlands this is pronounced as [drɪ ŋʔŋ], while the pronunciation [dɾɪ ŋkə] in the south of the Netherlands resembles standard Dutch much more.

Although the Netherlands and the northern part of Belgium (Flanders) only cover about 60.000 square kilometres, there are a large number of dialects in that region. In 1969, Daan and Blok published a map of the Netherlands and Flanders showing the Dutch dialect borders. They identified 28 dialectal regions in the Dutch-speaking language area which are shown in Figure 1.1 and Table 1.1. Their map was based on a survey of 1939 in which people judged the similarity of nearby dialects with respect to their own dialect.

Obtaining perceptual dialect distances is a time consuming task and does not always yield consistent results. For instance, inhabitants of region A may judge the dialect spoken by inhabitants of region B much more different than the other way around. Fortunately, computational methods have been developed to objectively compare dialects to each other.

A popular method in dialectology is the Levenshtein distance, which was introduced by Kessler (1995) as a tool to computationally compare dialects. The Levenshtein distance between two strings is determined by counting the number of edit operations (i.e. insertions, deletions and substitutions) needed to transform one string into the other. For example, the Levenshtein distance between [stenə] and [stɛin] is 3 as illustrated below.

stenə	subst. e/ɛ	1
stɛnə	insert i	1
stɛinə	delete ə	1
stɛin		
		3

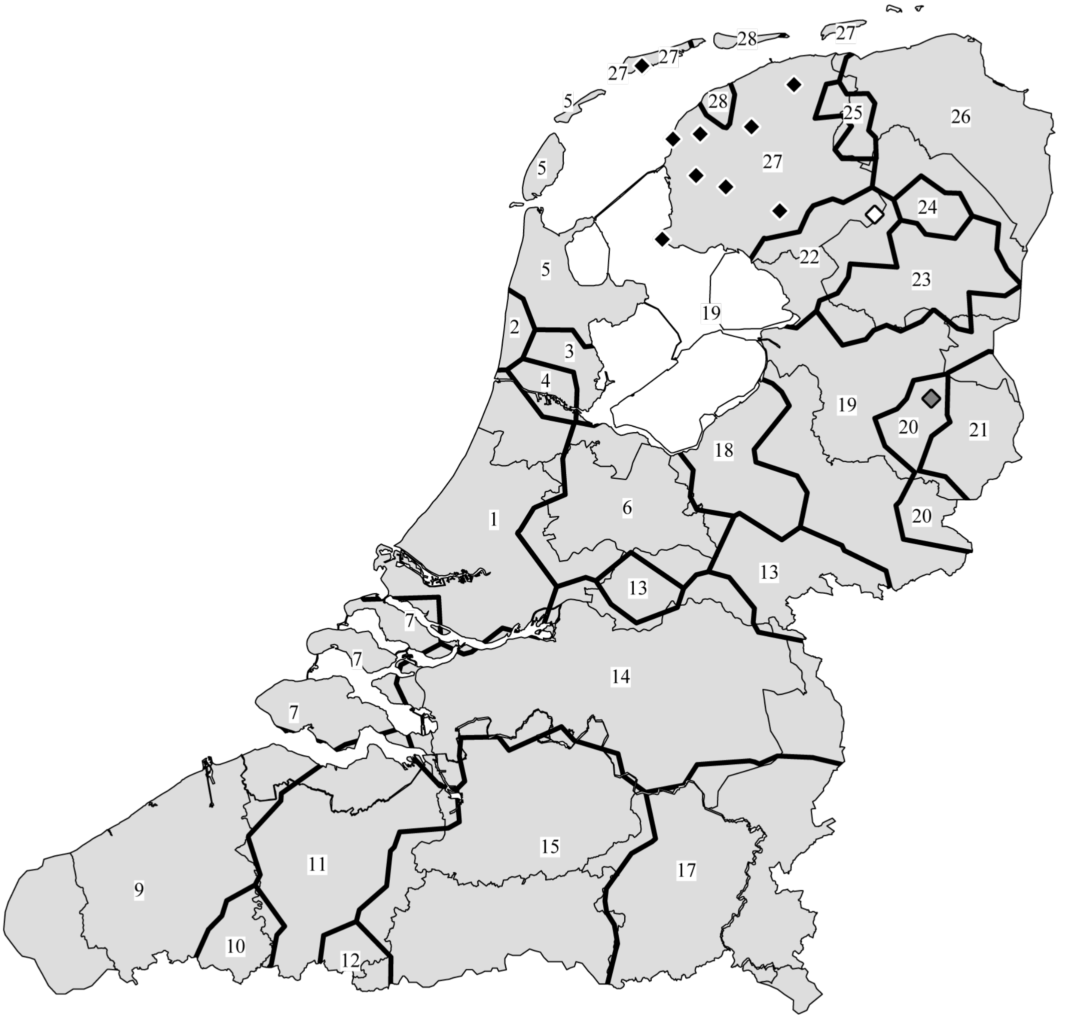

Figure 1.1. Locations of the 28 dialectal groups as distinguished in the map of Daan and Blok (1969). Provincial borders are represented by thinner lines and dialect borders by thicker ones. The numbers are explained in Table 1.1. Diamonds represent dialect islands. The black diamonds represent Frisian cities, which belong to group 28. The white diamond represents Appelscha, where both the dialect of group 22 and group 27 is spoken. The grey diamond represents Vriezenveen which contrasts strongly with its surroundings. Image courtesy of Heeringa (2004).

1	Dialect of Zuid-Holland
2	Dialect of Kennemerland
3	Dialect of Waterland
4	Dialect of Zaan region
5	Dialect of northern Noord-Holland
6	Dialect of the province of Utrecht and the Alblasserwaard region
7	Dialect of Zeeland
8	Dialect of region between Holland and Brabant dialects
9	Dialect of West Flanders and Zeeuws-Vlaanderen
10	Dialect of region between West and East Flanders dialects
11	Dialect of East Flanders
12	Dialect of region between East Flanders and Brabant dialects
13	Dialect of the river region
14	Dialect of Noord-Brabant and northern Limburg
15	Dialect of Brabant
16	Dialect of region between Brabant and Limburg dialects
17	Dialect of Limburg
18	Dialect of the Veluwe region
19	Dialect of Gelderland and western Overijssel
20	Dialect of western Twente and eastern Graafschap
21	Dialect of Twente
22	Dialect of the Stellingwerf region
23	Dialect of southern Drenthe
24	Dialect of central Drenthe
25	Dialect of Kollumerland
26	Dialect of Groningen and northern Drenthe
27	Frisian language
28	Dialects of het Bildt, Frisian cities, Midsland, and Ameland Island

Table 1.1. Dialectal regions in map of Daan and Blok (1969) shown in Figure 1.1.

The corresponding alignment is:

s	t	e		n	ə
s	t	ɛ	i	n	
		1	1		1

The Levenshtein distance has successfully been used to measure linguistic distances in Irish (Kessler, 1995), Dutch (e.g., Nerbonne et al., 1996; Heeringa, 2004), Sardinian (Bolognesi and Heeringa, 2005), Norwegian (e.g., Heeringa, 2004) and German dialects (Nerbonne and Siedle, 2005). Furthermore, the Levenshtein distance has been shown to yield results that are consistent (Cronbach's $\alpha = 0.99$) and valid when compared to dialect speakers judgements of similarity ($r \approx 0.7$; Heeringa et al., 2006). A detailed explanation of the Levenshtein distance can be found in Section 2.3.1.

A *conditio sine qua non* for computational dialect comparison, is the availability of dialectal material in digital form. Unfortunately these digital datasets are relatively scarce. The *Reeks Nederlands(ch)e Dialectatlassen* (RND; Blancquaert and Peé, 1982) created during the period 1925 – 1982 was the first broad-coverage Dutch dialect source available and was digitised in part by Heeringa (2001) to make computational dialect comparison possible. In 2003 another digital Dutch dialect source became available, the Goeman-Taeldeman-Van Reenen-Project data (GTRP; Goeman and Taeldeman, 1996; Van den Berg, 2003). The GTRP is an enormous collection of Dutch dialect data, including transcriptions of over 1800 items from over 600 localities, all collected over a relatively brief, and therefore, unproblematic time interval (15 years, 1980 – 1995). The GTRP complements the RND as a more recent and temporally more limited set (see also Taeldeman and Verleyen, 1999).

Heeringa (2004; Chapter 9) provides an in-depth aggregate analysis of the RND, visualising the dialectal situation it represents. Even though data from the GTRP has been used in several dialect studies (e.g., Goossens et al., 1998; Goossens et al., 2000; De Schutter et al., 2005; De Wulf et al., 2005), none of these have provided a similar, aggregate analysis of this data. Hence, the main purpose of this thesis is to provide the first aggregate analysis of the GTRP data.

We will use the regular Levenshtein distance to analyse the GTRP data in the same way as it was done for the RND by Heeringa (2004). Because language changes over time, for instance due to migration (Kerswill, 2006), we will compare the dialectal situation it represents to the RND, in particular to the 350-locality sample studied by Heeringa (2004), identifying areas of convergence and divergence. The results of the aggregate analysis of the GTRP and comparison to the RND are presented in Chapter 2.

The Levenshtein distance regards segments in a binary fashion, either as same or different. This is a clear simplification; not all sounds are equally different. For instance, the sounds /p/ and /b/ sound more similar than the sounds /p/ and /m/. Although there have been many attempts to incorporate more sensitive segment differences, they failed to show significant improvement (e.g., Heeringa and Braun, 2003; Heeringa, 2004). Instead of using segment distances as these are (incompletely) suggested by phonetic or phonological theory, we can also attempt to acquire these automatically. In Chapter 3, we will obtain these segment distances by training Pair Hidden Markov Models (PairHMMs). The PairHMM is special version of a Hidden Markov Model and was introduced to language studies by Mackay and Kondrak (2005). They used the PairHMM to calculate similarity scores for word pairs in orthographic form. We will investigate if using PairHMMs to obtain dialect distances in the GTRP improves the results as compared to the regular Levenshtein distance approach. Additionally we will evaluate the quality of the trained segment distances by comparing them to acoustic differences.

Inspired by psycholinguistic work on spoken word recognition which states that the importance of a sound (segment) depends on its position within a word, we will investigate a novel position-dependent approach to obtain dialect distances. In Chapter 4 we will propose two adaptations of the regular Levenshtein distance algorithm based on phonological theory. The first adaptation follows the idea of the Cohort Model (Marslen-Wilson and Welsh, 1978; Marslen-Wilson, 1987) which assumes that the word-initial part is more important for word recognition than the word-final part. This can be modelled by assigning edit operations at the start of the alignment a higher cost than edit operations at the end of the alignment, for example:

s	t	e		n	ə
s	t	ɛ	i	n	
		4	3		1

The second adaptation follows the idea that stressed syllables contain more information and are more important for word recognition than unstressed syllables (Altman and Carter, 1989). This can be modelled by giving edit operations involving stressed syllables a higher cost than edit operations involving unstressed syllables.

We will evaluate the results of the position-dependent approach by comparing them to results obtained using the regular Levenshtein algorithm on the GTRP data as well as on a Norwegian dataset for which perceptual dialect distances are available.

This thesis will be concluded in Chapter 5 with a general discussion of the results and some suggestions for further research.

2 Dialect pronunciation comparison using the Levenshtein distance

Abstract[*]

Contemporary Dutch dialects are compared using the Levenshtein distance, a measure of pronunciation difference. The material consists of data from the most recent Dutch dialect source available: the Goeman-Taeldeman-Van Reenen-Project (GTRP). This data consists of transcriptions of 1876 items for 613 localities in the Netherlands and Belgium gathered during the period 1980 – 1995. In addition to presenting the analysis of the GTRP, we compare the dialectal situation it represents to the Reeks Nederlands(ch)e Dialectatlassen (RND), in particular to the 350-locality sample studied by Heeringa (2004), noting areas of convergence and divergence. Although it was not the purpose of this research to criticise the GTRP, we nonetheless note that transcriptions from Belgian localities differ substantially from the transcriptions of localities in the Netherlands, impeding the comparison between the varieties of the two different countries. We therefore analyse the developments in the two countries separately.

2.1 Introduction

The Goeman-Taeldeman-Van Reenen-Project (GTRP; Goeman and Taeldeman, 1996) is an enormous collection of data collected from the Dutch dialects, including transcriptions of over 1800 items from over 600 localities, all collected over a relatively brief, and therefore, unproblematic time interval (15 years, 1980 – 1995). The GTRP is the first large-scale collection of Dutch dialect data since Blancquaert and Peé's *Reeks Nederlands(ch)e Dialectatlassen* (RND; 1925 – 1982), and complements it as a more recent and temporally more limited set. The GTRP provides a rich and attractive database, designed by the leading experts in Dutch dialectology, who likewise collaborated in obtaining, transcribing, and organising its information. The GTRP rivals the RND in being fully available digitally (Van den Berg, 2003) and being designed with an eye toward contemporary questions in phonology, morphology and variationist linguistics (Van Oostendorp, 2007). We present the GTRP and the RND in more detail in Section 2.2.

[*] A slightly different form of this text was accepted to appear in *Taal en Tongval* (2007) as: M. Wieling, W. Heeringa, and J. Nerbonne. An Aggregate Analysis of Pronunciation in the Goeman-Taeldeman-Van Reenen-Project Data.

The present chapter provides an aggregate analysis of the pronunciation variation in this collection, using the same techniques for analysis which Nerbonne et al. (1996) first applied, and which Heeringa (2004) lays out in full detail. The aggregate analysis proceeds from a word-by-word measurement of pronunciation differences, which has been shown to provide consistent probes into dialectal relations, and which correlates strongly ($r > 0.7$) with lay dialect speakers' intuitions about the degree to which non-local dialects sound "remote" or "different" (see Heeringa, 2004: Chapter 7; and Heeringa et al., 2006 for rigorous discussions of the consistency and validity of the measures). The aggregate analysis differs from analyses based on a small number of linguistic variables in providing a global view of the relations among varieties, allowing more abstract questions to be posed about these relations. We sketch the necessary technical background for the measurement of pronunciation differences in Section 2.3 below.

For various technical reasons, we restrict our analysis to 562 items in the GTRP, which is nonetheless notably large compared to other analyses. We present the results of this analysis in Sections 2.4.1 and 2.4.2 below.

A second, related goal of this chapter is to examine what has changed between the time of the RND and that of the GTRP. For this purpose we focus our attention on 224 localities which are common to the GTRP and the RND varieties analysed by Heeringa (2004). To allow interpretation to be as exact as possible, we also focused on the 59 words which were common to the GTRP and the RND. Since the two projects differed in methodologies, especially transcription practice, we approach the comparison indirectly, via regression analyses. We are able to identify several areas in which dialects are converging (relatively), and likewise several in which they are diverging. The results of the comparison are the subject of Section 2.4.3 below.

It was not originally a goal of the work reported here to examine the GTRP with respect to its selection and transcription practices, but several preliminary results indicated that the Belgian and the Dutch collaborators had not been optimally successful in unifying these practices. We follow these indications up, and conclude in Section 2.4.1 that caution is needed in interpreting aggregate results unless one separates Dutch and Belgian material. We further suggest that these problems are likely to infect other, non-aggregating approaches as well. At the end of Section 2.4.2 we discuss some clues that fieldworker and transcription practices in the Netherlands may be confounding analyses to some degree. Also Hinskens and Van Oostendorp (2006) reported transcriber effects in the GTRP data.

2.2 Material

In this chapter two Dutch dialect data sources are used: data from the Goeman-Taeldeman-Van Reenen-Project (GTRP; Goeman and Taeldeman, 1996) and data from the *Reeks Nederlands(ch)e Dialectatlassen* (RND; Blancquaert and Peé, 1925 – 1982) as used by Heeringa (2004).

2.2.1 GTRP

The GTRP consists of digital transcriptions for 613 dialect varieties in the Netherlands (424 varieties) and Belgium (189 varieties; see Figure 2.1 for the geographical distribution). All data was gathered during the period 1980 – 1995, making it the most recent broad-coverage Dutch dialect data source available. The GTRP is moreover available digitally, making it especially useful for research. For every variety, a maximum of 1876 items was narrowly transcribed according to the International Phonetic Alphabet. The items consisted of separate words and word groups, including nominals, adjectives and nouns. A more specific overview of the items is given in Taeldeman and Verleyen (1999).

The recordings and transcriptions of the GTRP were made by 25 collaborators, but more than 40% of all data was transcribed by only two individuals who created reliable transcriptions (Goeman, 1999). In most cases there were multiple transcribers operating in a single region, ranging from 1 (Drenthe) to 13 (Zuid-Holland). In general the dialectal data of one variety was based on a single dialect speaker.

Our analyses are conducted on a subset of the GTRP items. Because the Levenshtein distance is used to obtain dialect distances, we only take single words into account (like Heeringa, 2004). Unfortunately, word boundaries are not always clearly identified in the transcriptions (primarily for Belgian dialect varieties), making segmentation very hard. For this reason, we restrict our subset to items consisting of a single word. Because the singular nouns are (sometimes, but not always) preceded by an article ('n) these will not be included. The first-person plural is the only verb form not preceded by a pronoun and therefore the only verb form which is included. Finally, no items are included where multiple lexemes are possible.

The GTRP was compiled with a view to documenting both phonological and morphological variation (De Schutter et al., 2005). Because our purpose here is the analysis of variation in pronunciation, we ignore many items in the GTRP whose primary purpose was presumably the documentation of morphological variation. If we had included this material directly, the measurements would have confounded pronunciation and morphological variation. Differently inflected forms of one word (e.g., base and comparative forms of an adjective) are very similar and therefore are not both selected in the subset to keep the distance measurement focused on pronunciation.

The following forms are included in the subset:

- The plural nouns, but not the diminutive nouns (the singular nouns are preceded by an article and therefore not included)
- The base forms of the adjectives instead of the comparative forms
- The first-person plural verbs (the transcriptions of other verb forms include pronouns and were therefore not included)

The complete list of the 562 remaining items used in our analysis is displayed in Table 2.1.

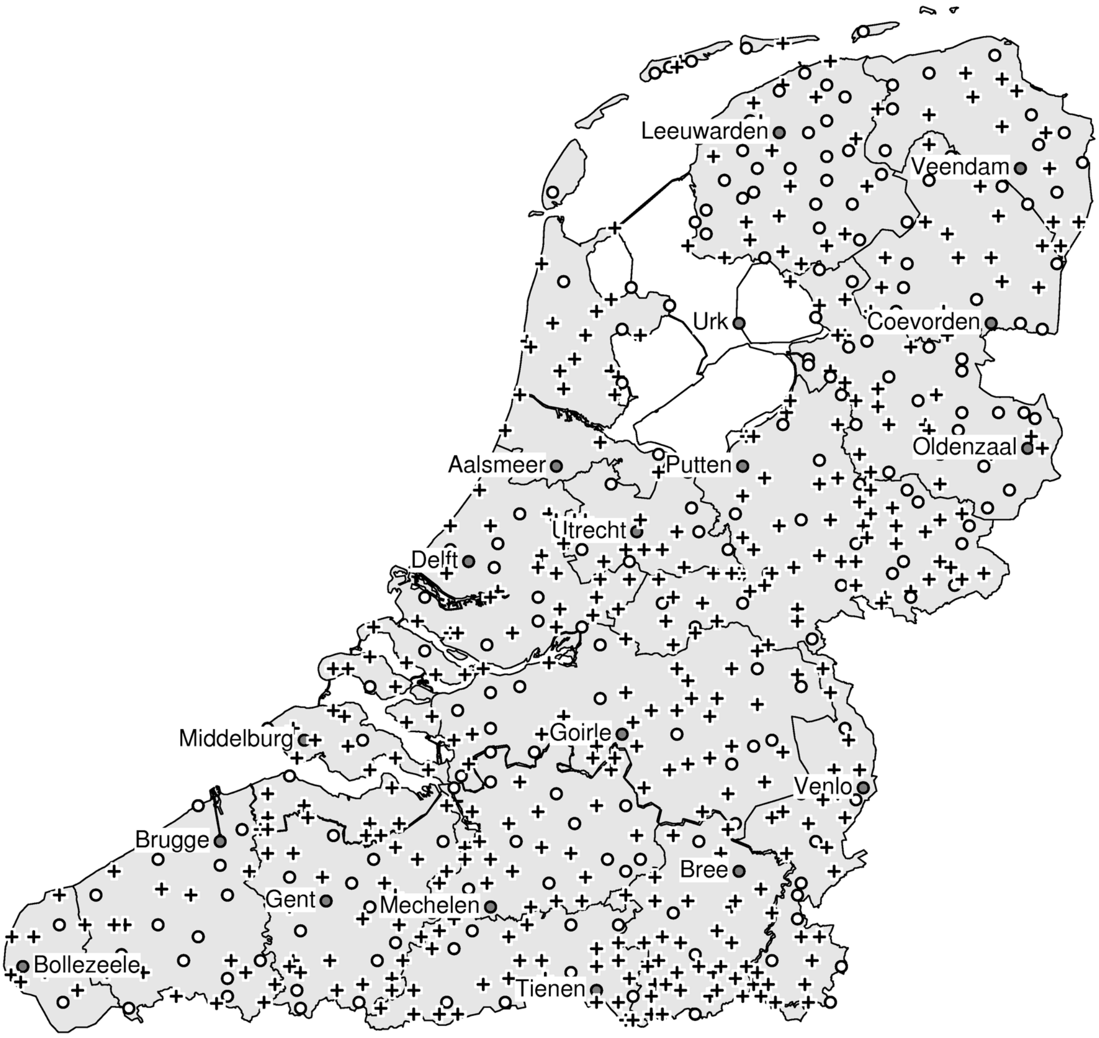

Figure 2.1. The geographic distribution of the 613 GTRP localities. The 224 localities marked with a circle appear both in the GTRP and in the 360-element sample of the RND studied by Heeringa (2004). Localities marked by a '+' occur only in the GTRP. See the text for further remarks.

aarde
aardig
acht
achter
adem
af
anders
appels
arm
armen
auto’s
baarden
bakken
barsten
bedden
beenderen
beginnen
benen
beren (wild)
best (bijw)
beurzen
beven
bezems
bezig
bidden
bier
bij (vz)
bijen
bijten
binden
bitter
bladen
bladeren
blauw
blazen
bleek
blijven
blind
bloeden
bloeien
blond
blozen
bokken
bomen
bonen
boren
boter
bouwen
boven
braaf
braden
branden
breed
breien
breken
brengen
broden
broeken
broers
bruin
buigen
buiten
dagen
daken
damp
dansen
darmen
deeg
denken
derde
deuren
dienen
diep
dieven
dik
dingen
dinsdag
dochters
doeken
doen
dol
donder
donderdag
donker
doof
dooien
door
dopen
dorsen
dorst
draaien
draden
dragen
dreigen
drie
drinken
dromen
droog
dubbel
duiven
duizend
dun
durven
duur
duwen
dweilen
echt
eeuwen
eieren
eigen
einde
elf
engelen
enkel
eten
ezels
fel
fijn
flauw
flessen
fruit
gaan
gaarne
gal
ganzen
gapen
gebruiken
geel
gehad
geld
geloven
genoeg
geraken
gerst
geven
geweest
gewoon
gisteren
glazen
god
goed
goud
gouden
gras
graven
grijs
groen
grof
groot
haast
haastig
haken
halen
half
handen
hanen
hangen
hard
haver
hebben
heel
heet
heffen
heilig
helpen
hemden
hemel
hengsten
heren
heten
hier
hoeden
hoesten
hol
holen
honden
honger
hoog
hooi
hoop (espoir)
hopen
horen
horens
houden
huizen
jagen
jeuken
jong
jongen
juist
kaas
kaf
kalm
kalveren
kamers
kammen (nom)
kammen (vb)
kanten
karren
kasten
katten
kennen
kermis
kersen
kervel
keuren
kiezen
kijken
kinderen
klaver
kleden
klederen
klein
kloppen
kloppen
knechten
kneden
knieën
koeien
koel
koken
komen
kommen
konijnen
koorts
kopen
koper
kort
koud
kousen
kraken
kramp
kreupel
krijgen
krimpen
krom
kruipen
kwaad
laag
laat
lachen
lam
lammeren
lampen
lang
lastig
laten
latten
leden
ledig
leem
leggen
leren
leugens
leunen
leven
lezen
licht
liederen
liggen
lijken
likken
lomp
lopen
lucht
lui
luiden
luisteren
maandag
maanden
maart
magen
mager
maken
marmer
maten
mazelen
meer
mei
meid
melk
menen
merg
metselen
meubels
missen
modder
moe
moes
moeten
mogelijk
mogen
morgen (demain)
mossels
muizen
muren
naalden
nat
negen
negers
nieuw
noemen
nog
noorden
noten
nu
ogen
om
ons
oogst
ook
oosten
op
open
oud
over
paarden
padden
paden
Pasen
pekel
pellen
peper
peren
piepen
pijpen
planken
pleinen
ploegen (wrktg)
potten
proeven
proper
raar
raden
recht
redden
regen
rekken
ribben
riet
rijden
rijk
rijp
rijst
ringen
roepen
roeren
rogge
rokken
rond
rondes
rood
rook
ruiken
runderen
ruzies
sap
saus
schade
schapen
schaven
scheef
scheel
scheiden
schepen
scheppen
scheren
scherp
schieten
schimmel
schoenen
scholen
schoon
schrijven
schudden
schuiven
schuld
schuren
schuw
simpel
slaan
slapen
slecht
slijm
slijpen
slim
sluiten
smal
smeden
smelten
smeren
sneeuw
sneeuwen
soep
spannen
sparen
spartelen
spelden
spelen
sport (spel)
spreken
springen
spuiten
staan
stallen
stampen
steken
stelen
stenen
sterven
stijf
stil
stoelen
stof (huisvuil)
stokken
stom
stout
straten
strepen
strooien
sturen (zenden)
suiker
taai
taarten
tafels
takken
tam
tanden
tangen
tantes
tarwe
tegen
tellen
temmen
tenen
tien
timmeren
torens
traag
tralies
trams
treffen
treinen
trouwen
tussen
twaalf
twee
tweede
twijfel
twintig
uilen
vader
vallen
vals
vangen
varen
vast
vaten
vechten
veel
veertig
ver
verf
vers
vesten
vet
veulens
vier
vieren
vijf
vijftig
vijgen
vinden
vingers
vissen
vlaggen
vlas
vlees
vliegen
vloeken
vlooien
voegen
voelen
voeten
vogels
vol
volgen
volk
voor
vragen
vreemd
vriezen
vrij
vrijdag
vrijen
vroeg
vuil
vuur
wachten
wafels
warm
wassen
weer
weg
wegen
wegen
weinig
weken
wensen
werken
weten
wieden
wijd
wijn
wijven
wild
willen
winnen
wippen
wit
woensdag
wol
wonen
woorden
worden
wrijven
zacht
zakken
zand
zaterdag
zee
zeep
zeggen
zeilen
zeker
zelf
zes
zetten
zeven
zeventig
ziek
ziektes
zien
zijn
zilveren
zitten
zoeken
zoet
zondag
zonder
zonen
zorgen
zout
zouten
zuchten
zuigen
zuur
zwaar
zwart
zwellen
zwemmen
zwijgen

breder

Table 2.1. List of all 562 words in the GTRP subset. The 59 words in boldface are used for RND-GTRP comparison (see Section 2.4.3). The word *breder* is included in the set used for comparison with the RND, but not in the base subset of 562 words (due to the presence of *breed*).

2.2.2 RND

We will compare the results obtained on the basis of the GTRP with results obtained on the basis of an earlier data source, the *Reeks Nederlands(ch)e Dialectatlassen* (RND). The RND is a series of atlases covering the Dutch language area. The Dutch area comprises the Netherlands, the northern part of Belgium (Flanders), a smaller northwestern part of France and the German county Bentheim. The RND contains 1956 varieties, which can be found in 16 volumes. The first volume appeared in 1925, the last in 1982. The first recordings were made in 1922, the last ones in 1975. E. Blancquaert initiated the project. When Blancquaert passed away before all the volumes were finished, the project was finished under the direction of W. Peé. In the RND, the same 141 sentences are translated and transcribed in phonetic script for each dialect.

The recordings and transcriptions of the RND were made by 16 collaborators, who mostly restricted their activities to a single region (Heeringa, 2004). For every variety, material was gathered from multiple dialect speakers.

In 2001 the RND material was digitised in part. Since digitising the phonetic texts is time-consuming, a selection of 360 dialects was made and for each dialect the same 125 words were selected from the text. The words represent (nearly) all the vowels (monophthongs and diphthongs) and consonants. Heeringa (2001) and Heeringa (2004) describe the selection of dialects and words in more detail and discuss how differences introduced by different transcribers are processed.

Our set of 360 RND varieties and the set of 613 GTRP varieties have 224 varieties in common. Their distribution is shown in Figure 2.1. The 125 RND words and the set of 562 GTRP words share 58 words. We added one extra word, *breder* 'wider', which was excluded from the set of 562 GTRP words since we used no more than one morphologic variant per item and the word *breed* 'wide' was already included. So in total we have 59 words, which are listed in boldface in Table 2.1. The comparisons between the RND and GTRP in this chapter are based only on the 224 common varieties and the 59 common words.

2.3 Measuring linguistic distances

In 1995 Kessler introduced the Levenshtein distance as a tool for measuring linguistic distances between language varieties. The Levenshtein distance is a string edit distance measure, and Kessler applied this algorithm to the comparison of Irish dialects. Later the same technique was successfully applied to Dutch (Nerbonne et al., 1996; Heeringa, 2004: 213 – 278), Sardinian (Bolognesi and Heeringa, 2005), Norwegian (Gooskens and Heeringa, 2004) and German (Nerbonne and Siedle, 2005).

In this chapter we use the Levenshtein distance for the measurement of pronunciation distances. Pronunciation variation includes phonetic and morphologic variation, and excludes lexical variation. Below, we give a brief explanation of the methodology. For a more extensive explanation see Heeringa (2004: 121 – 135).

The Levenshtein algorithm provides a rough, but completely consistent measure of pronunciation distance. Its strength lies in the fact that it can be implemented on the computer, so that large amounts of dialect material can be compared and analysed. The usage of this computational technique enables dialectology to be based on the aggregated comparisons of millions of pairs of phonetic segments.

2.3.1 Levenshtein algorithm

Using the Levenshtein distance, two varieties are compared by comparing the pronunciation of words in the first variety with the pronunciation of the same words in the second. We determine how one pronunciation might be transformed into the other by inserting, deleting or substituting sounds. Weights are assigned to these three operations. In the simplest form of the algorithm, all operations have the same cost, e.g., 1. Assume *melk* 'milk' is pronounced as [mɔəlkə] in the dialect of Veenwouden (Friesland), and as [mɛlək] in the dialect of Delft (Zuid-Holland). Changing one pronunciation into the other can be done as follows (ignoring suprasegmentals and diacritics):

mɔəlkə	subst. ɔ/ɛ	1
mɛəlkə	delete ə	1
mɛlkə	insert ə	1
mɛləkə	delete ə	1
mɛlək		
		4

In fact many sequence operations map [mɔəlkə] to [mɛlək]. The power of the Levenshtein algorithm is that it always finds the cost of the cheapest mapping.

A naive method to compute the Levenshtein distance is using a recursive function with all three edit operations as illustrated in the pseudocode below. Note that `INS_COST`, `DEL_COST`, and `SUBST_COST` all equal 1 for the regular Levenshtein algorithm.

```
LEVEN_DIST(empty_string, empty_string) := 0
LEVEN_DIST(string1, empty_string) := LENGTH(string1)
LEVEN_DIST(empty_string, string2) := LENGTH(string2)

LEVEN_DIST(string1 + finalchar1, string2 + finalchar2) :=
   MIN(
         LEVEN_DIST(string1, string2 + finalchar2) + INS_COST,
         LEVEN_DIST(string1 + finalchar1, string2) + DEL_COST,
         IF finalchar1 = finalchar2 THEN
            LEVEN_DIST(string1, string2) // no cost
         ELSE
            LEVEN_DIST(string1, string2) + SUBST_COST
         END
      )
```

This naive computation yields a time complexity of $\mathcal{O}(3^n)$, where n is the length of an input string. The exponential time complexity is caused by the large number of duplicate computations, e.g. the cost of substituting the first characters will be calculated very often.

Fortunately, this problem can be solved by storing intermediate results in a two dimensional matrix instead of calculating them again and again. This approach is called a dynamic programming approach and is illustrated below. The improved algorithm has a time complexity of only $\mathcal{O}(n^2)$.

```
LEVEN_TABLE(0,0) := 0

FOR i := 1 TO LENGTH(string1)
   LEVEN_TABLE(i,0) := i
END

FOR j := 1 TO LENGTH(string2)
   LEVEN_TABLE(0,j) := j
END

FOR i := 1 TO LENGTH(string1) DO
   FOR j := 1 TO LENGTH(string2) DO
      LEVEN_TABLE(i,j) :=
         MIN(
               LEVEN_TABLE(i-1, j) + INS_COST,
               LEVEN_TABLE(i, j-1) + DEL_COST,
               IF finalchar1 = finalchar2 THEN
                  LEVEN_TABLE(i-1, j-1) // no cost
               ELSE
                  LEVEN_TABLE(i-1, j-1) + SUBST_COST
               END
            )
   END
END

LEVEN_DIST := LEVEN_TABLE( LENGTH(string1),
                           LENGTH(string2) )
```

An example of this approach for the words [mɔəlkə] and [mɛlək] is illustrated in the table below. To find the sequence of edit operations which has the lowest cost, traverse from the top-left to the bottom-right by going down (insertion: cost must be increased by 1), right (deletion: cost must be increased by 1) or down-right (same symbol substitution: cost must remain equal; different symbol substitution: cost must be increased by 1). The initial value is 0 (top-left) and the Levenshtein distance equals the value in the bottom-right field of the table (i.e. 4). If it is not possible in a certain field to go down, right or down-right while increasing the value from this field with 0 (same symbol substitution) or 1 (other cases) that field is not part of the path yielding the Levenshtein distance. One possible path yielding the Levenshtein distance is marked in bold face in the table below.

		m	ɔ	ə	l	k	ə
	0	1	2	3	4	5	6
m	1	**0**	1	2	3	4	5
ɛ	2	1	**1**	**2**	3	4	5
l	3	2	2	2	**2**	3	4
ə	4	3	3	2	**3**	3	3
k	5	4	4	3	3	**3**	**4**

To deal with syllabicity, the Levenshtein algorithm is adapted so that only vowels may match with vowels, and consonants with consonants, with several special exceptions: [j] and [w] may match with vowels, [i] and [u] with consonants, and central vowels (in our research only the schwa) with sonorants. So the [i], [u], [j] and [w] align with anything, the [ə] with syllabic (sonorant) consonants, but otherwise vowels align with vowels and consonants with consonants. In this way unlikely matches (e.g., a [p] with an [a]) are prevented. In our example we thus have the following alignment (also shown in the previous table illustrating the Levenshtein algorithm):

m	ɔ	ə	l		k	ə
m	ɛ		l	ə	k	
	1	1		1		1

In earlier work we divided the sum of the operations by the length of the alignment. This normalises scores so that longer words do not count more heavily than shorter ones, reflecting the status of words as linguistic units. However, Heeringa et al. (2006) showed that results based on raw Levenshtein distances approximate dialect differences as perceived by the dialect speakers better than results based on normalised Levenshtein distances. Therefore we do not normalise the Levenshtein distances in this chapter but use the raw distances, i.e. distances which give us the sum of the operations needed to transform one pronunciation into another, with no transformation for length.

2.3.2 Operation weights

The example above is based on a notion of phonetic distance in which phonetic overlap is binary: non-identical phones contribute to phonetic distance, identical ones do not. Thus the pair [i, ɒ] counts as different to the same degree as [i, ɪ]. In earlier work we experimented with more sensitive versions in which phones are compared on the basis of their feature values or acoustic representations. In that way the pair [i, ɒ] counts as more different than [i, ɪ].

In a validation study Heeringa (2004) compared results of binary, feature-based and acoustic-based versions to the results of a perception experiment carried out by Charlotte Gooskens. In this experiment dialect differences as perceived by Norwegian dialect speakers were measured. It was found that generally speaking the binary versions approximate perceptual distances better than the feature-based and acoustic-based versions. The fact that segments differ appears to be more important in the perception of speakers than the degree to which segments differ. Therefore we will use the binary version of Levenshtein distance in this chapter, as illustrated in the example in Section 2.3.1. All substitutions, insertions and deletions have the same weight, in our example the value 1.

2.3.3 Diacritics

We do not process suprasegmentals and diacritics. Differences between the way in which transcribers transcribe pronunciations are found especially frequently in the use of suprasegmentals and diacritics (Goeman, 1999). The RND transcribers, instructed by (or in the line of) Blancquaert, may have used them differently from the GTRP transcribers. To make the comparison between RND and GTRP results as fair as possible, we restrict our analyses to the basic phonetic segments and ignore suprasegmentals and diacritics.

2.3.4 Dialect distances

When comparing two varieties on the basis of n_w words, we analyse n_w word pairs and get n_w Levenshtein distances. The dialect distance is equal to the sum of n_w Levenshtein distances divided by n_w. When comparing n_d varieties, the average Levenshtein distances are calculated between each pair of varieties and arranged in a matrix which has n_d rows and n_d columns.

To measure the consistency (or reliability) of our data, we use Cronbach's α (Cronbach, 1951). On the basis of variation of one single word (or item) we create a $n_d \times n_d$ distance matrix. With n_w words, we obtain n_w distance matrices, for each word one matrix. Cronbach's α is a function of the number of linguistic variables and the average inter-item correlation among the variables. In our case it is a function of the number of words n_w and the average inter-word correlation among the n_w matrices. Its values range between zero and one, higher values indicating greater reliability. As a rule of thumb, values higher than 0.7 are considered sufficient to obtain consistent results in social sciences (Nunnally, 1978).

2.4 Results

2.4.1 GTRP data of all varieties

To find the distance between two pronunciations of the same word, we use the Levenshtein distance. The dialect distance between two varieties is obtained by averaging the distances for all the word pairs. To measure data consistency, we calculated Cronbach's α for the obtained distance measurements. For our results, Cronbach's α is 0.99, which is

562 items			
The Netherlands	**1** (1.077.169)	Belgium	**33** (469.155)
Noord-Brabant	**12** (130.324)	Antwerp	**40** (86.257)
Limburg	**15** (80.535)	Belgian Limburg	**38** (110.294)
Goirle (NB)	**39** (2.553)	Poppel (Ant)	**49** (2.687)
1876 items			
The Netherlands	**0** (4.790.266)	Belgium	**27** (2.128.066)

Table 2.2. In boldface total number of distinct phonetic symbols (out of 83) which do not occur in the transcriptions. The total size (number of phonetic symbol tokens) of the dialect data for each region is given between parentheses.

much higher than the accepted threshold in social science (where $\alpha > 0.70$ is regarded as acceptable). We conclude that our distance measurements are highly consistent.

Figure 2.2 shows the dialect distances geographically. Varieties which are strongly related are connected by darker lines, while more distant varieties are connected by lighter lines. Even where no lines can be seen, very faint (often invisible) lines implicitly connect varieties which are very distant.

When inspecting the image, we note that the lines in Belgium are quite dark compared to the lighter lines in the Netherlands. This suggests that the varieties in Belgium are more strongly connected than those in (the north of) the Netherlands. Considering that the northern varieties in the Netherlands were found to have stronger connections than the southern varieties in the RND (Heeringa, 2004: 235), this result is opposite to what was expected.

We already indicated that the data of varieties in Belgium hardly contained any word boundaries (see Section 2.2.1), while this was not true for varieties in the Netherlands. Although unimportant for our subset containing only single word items, this could be a clue to the existence of structural differences in transcription method between Belgium and the Netherlands.

We conducted a thorough analysis of the dialect data, which showed large national differences in the number of phonetic symbols used to transcribe the items. Table 2.2 indicates the number of unused phonetic symbols in both countries, four neighbouring provinces and two neighbouring cities. For completeness, the number of unused tokens for all 1876 items for both countries is also included. Figure 2.3 gives an overview of the phonetic tokens which are not used in Belgium (for the complete GTRP set of 1876 items).

Table 2.3 illustrates some transcription differences between two neighbouring places near the border of Belgium and the Netherlands (see Figure 2.4). For this example, note that the phonetic symbols unused in Belgium include ɒ, ɪ, ʁ, ʊ and ʍ.

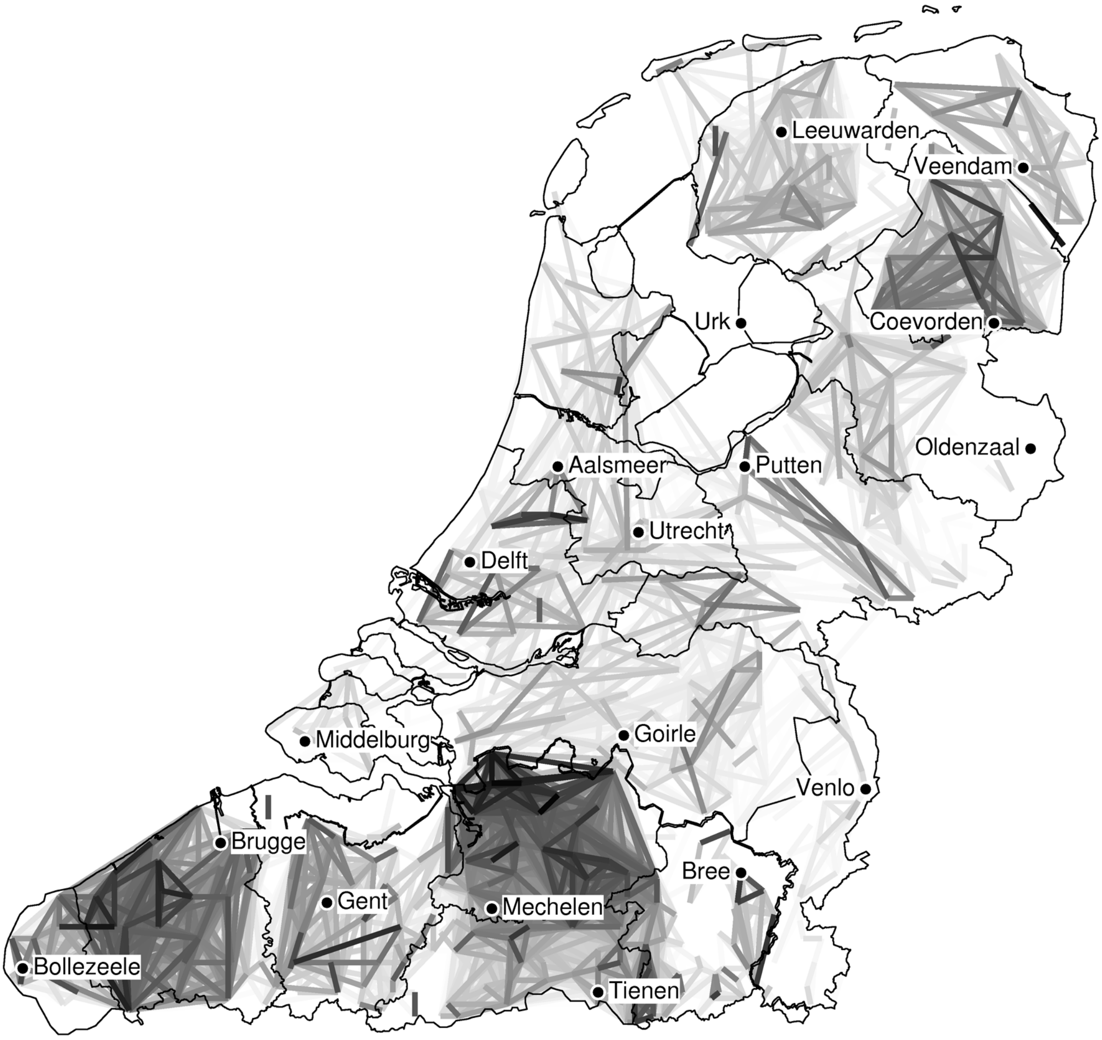

Figure 2.2. Average Levenshtein distance among 613 GTRP varieties. Darker lines connect close varieties, lighter lines more distant ones. We suggest that this view is confounded by differences in transcription practice. See the text for discussion, and see Figure 2.5 (below) for the view we defend.

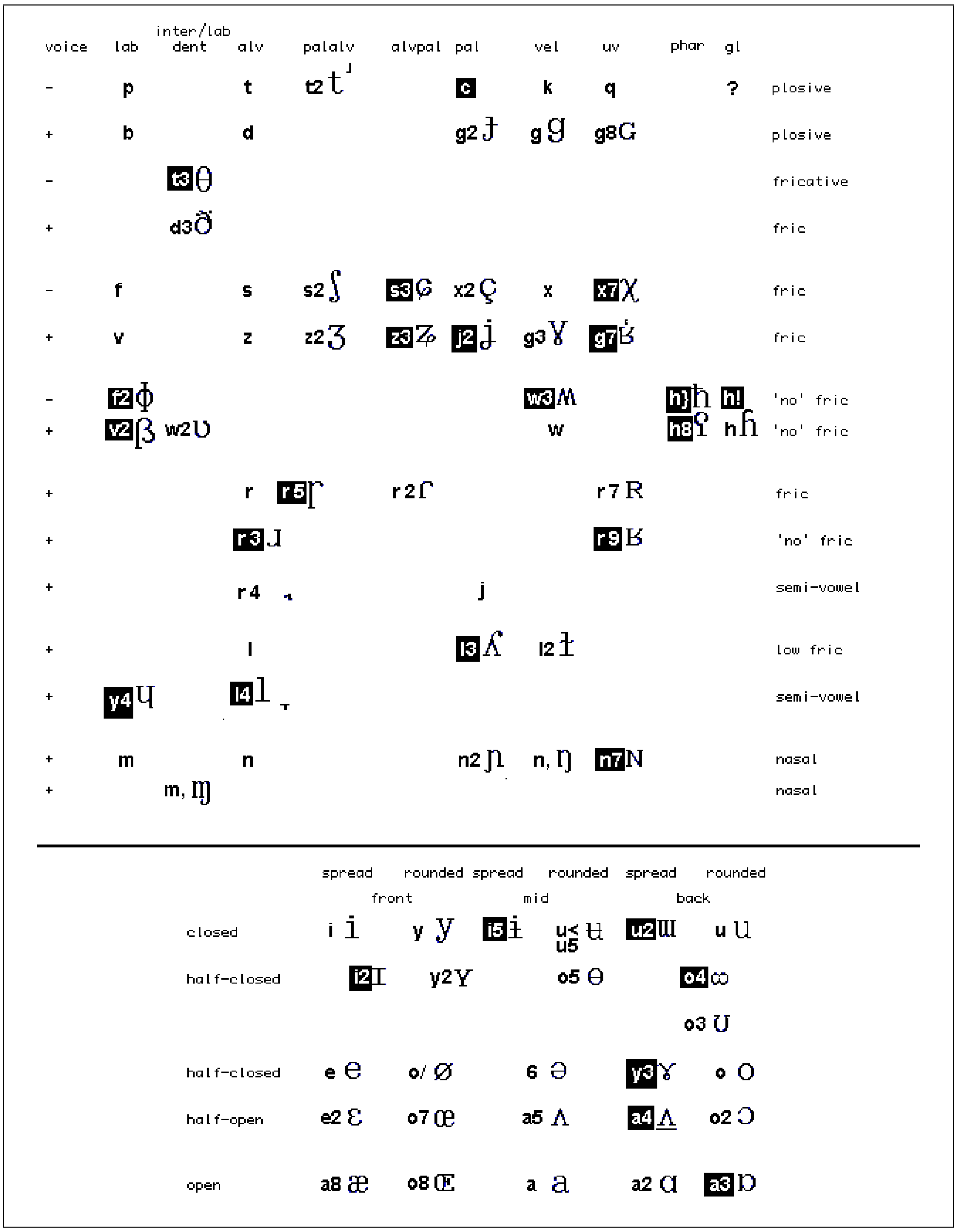
voice lab inter/lab dent alv palalv alvpal pal vel uv phar gl
- p t t2 ʈ c k q ? plosive
+ b d g2 ɟ g ɡ g8 ɢ plosive
- t3 θ fricative
+ d3 ð fric
- f s s2 ʃ s3 ɕ x2 ç x x7 χ fric
+ v z z2 ʒ z3 ʑ j2 ʝ g3 ɣ g7 ʁ fric
- f2 ɸ w3 ʍ h} ħ h! 'no' fric
+ v2 β w2 ʋ w h8 ʕ h ɦ 'no' fric
+ r r5 ɾ r2 ɽ r7 R fric
+ r3 ɹ r9 ʁ 'no' fric
+ r4 j semi-vowel
+ l l3 ʎ l2 ɫ low fric
+ y4 ɥ l4 l semi-vowel
+ m n n2 ɲ n, ŋ n7 N nasal
+ m, ɱ nasal
spread rounded spread rounded spread rounded
front mid back
closed i i y y i5 ɨ u< u5 ʉ u2 ɯ u u
half-closed i2 ɪ y2 ʏ o5 ɵ o4 ɷ
o3 ʊ
half-closed e e o/ ø 6 ə y3 ɤ o o
half-open e2 ɛ o7 œ a5 ʌ a4 ʌ o2 ɔ
open a8 æ o8 ɶ a a a2 ɑ a3 ɒ

Figure 2.3. All 83 Keyboard-IPA symbols used in the GTRP data (without diacritics). Symbols on a black background are not used in Belgian transcriptions. Original image: Goeman, Van Reenen and Van den Berg (Meertens Instituut).

Dutch	English	Goirle (NL)	Poppel (BEL)
baarden	beards	bɒrdə	bɔrdə
bij (vz)	at	bɛi	bɛi
blond	blonde	bɫɔnt	blɔnt
broeken	pants	brʊkə	brukən
donker	dark	dɔŋkər	doŋkər
hard	hard	hɑʀt	hɑrt
haver	oats	hɒvəʀ	hɔvər
kamers	rooms	kɒməʁs	kɔmərs
kinderen	children	kɛndəʀ	kɛndər
kloppen	knock	kɫɔpə	klɔpə
luisteren	listen	lʌstəʀə	lœstərə
missen	miss	mɪsə	mise
simpel	simple	sɪmpɔɫ	sempəl
sneeuw	snow	snouʍ	sneəw
tralies	bars	tʀɒlis	trɑlis
twaalf	twelve	twaləf	twɔləf
vogels	birds	voɣəɫs	vouɣəls
vriezen	freeze	vrizə	vrizən
woensdag	Wednesday	wunsdɑx	wunzdɑx
zeggen	say	zɛɣə	zɛɣən

Table 2.3. Phonetic transcriptions of Goirle (NL) and Poppel (BEL) including Dutch and English translations. Even though phonetic transcriptions are of comparable length and complexity, the Dutch sites vary consistently use a much wider range of phonetic symbols, confounding measurement of pronunciation distance.

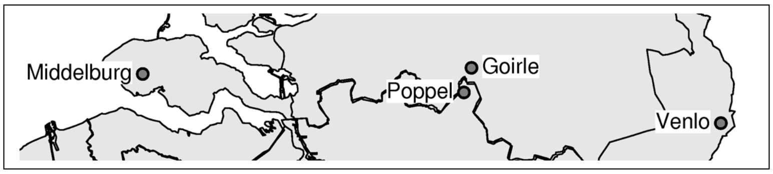

Figure 2.4. Relative locations of Poppel (Belgium) and Goirle (the Netherlands).

Transcriptions using fewer phonetic symbols are likely to be measured as more similar due to a lower degree of possible variation. Figure 2.2 shows exactly this result. Because of these substantial transcriptional differences between the two countries (see also Van Oostendorp, 2007; Hinskens and Van Oostendorp, 2006) it is inappropriate to compare the pronunciations of the two countries directly. Therefore, in what follows, we analyse the transcriptions of the two countries separately, and also discuss their pronunciation differences separately.

2.4.2 GTRP data, the Netherlands and Belgium separately

The data was highly consistent even when regarding the countries individually. Cronbach's α was 0.990 for dialect distances in the Netherlands and 0.994 for dialect distances in Belgium.

In Figure 2.5, the strong connections among the Frisian varieties and among the Groningen and Drenthe (Low Saxon) varieties are clearly shown. The dialect of Gelderland and western Overijssel can also be identified below the dialect of Drenthe. South of this group a clear boundary can be identified, known as the boundary between Low Saxon (northeastern dialects) and Low Franconian (western, southwestern and southern dialects). The rest of the map shows other less closely unified groups, for example, in Zuid-Holland and Noord-Brabant as well as less cohesive groups in Limburg and Zeeland.

Just as was evident in Figure 2.2, Belgian varieties are tightly connected in both the varieties of Antwerp as well as in those of West Flanders (see Figure 2.5). A lot of white lines are present in Belgian Limburg however, indicating more dissimilar varieties in that region. Note the weak lines connecting to the Ghent variety (indicating it to be very different from the neighbouring varieties); they appear to be masked by lines of closer varieties in the surrounding area.

By using multidimensional scaling (MDS; see Heeringa, 2004: 156 – 163) varieties can be positioned in a three-dimensional space. The more similar two varieties are, the closer they will be placed together. The location in the three-dimensional space (in x-, y- and z-coordinates) can be converted to a distinct colour using red, green and blue colour components. By assigning each collection site its own colour in the geographical map, an overview is obtained of the distances between the varieties. Similar sites have the same colour, while colour differs for more linguistically distant varieties. This method is superior to a cluster map (e.g., Heeringa, 2004: 231) because MDS coordinates are assigned to individual collection sites, which means that deviant sites become obvious, while clustering reduces each site to one of a fixed number of groups. Hence, clustering risks covering up problems.[1]

Because we are reducing the number of dimensions in the data (i.e. the dialect differences) to three by using the MDS technique, it is likely that some detail will be lost. To

[1] We discuss apparently exceptional sites at the end of this section, and we note here that these exceptions are indeed obvious in clustering as well.

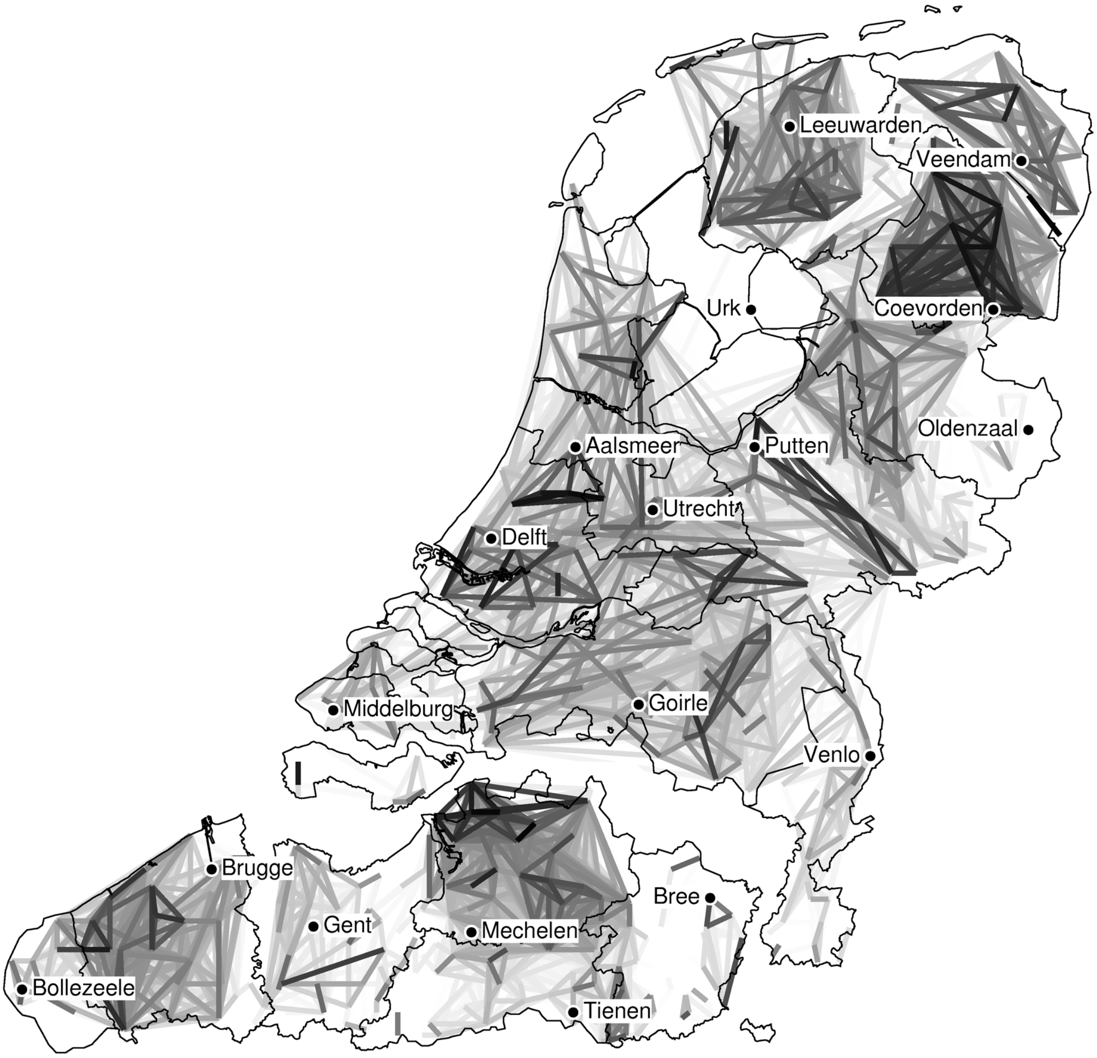

Figure 2.5. Average Levenshtein distance between 613 GTRP varieties. Darker lines connect close varieties, lighter lines more distant ones. The maps of the Netherlands (top) and Belgium (bottom) must be considered independently.

get an indication of the loss of detail, we calculate how much variance of the original data is explained by the three-dimensional MDS output. For the Netherlands, the MDS output explains 87.5% of the variance of the original dialect differences. For Belgium a comparable value is obtained: 88.1%. We therefore conclude that our MDS output gives a representative overview of the original dialect differences in both countries.

In Figure 2.6 and 2.7 the MDS colour maps of the Netherlands and Belgium are shown. The colour of intermediate points is determined by interpolation using Inverse Distance Weighting (see Heeringa, 2004: 156 – 163). Because the dialect data for Belgium and the Netherlands was separated, the maps should be considered independently. Varieties with a certain colour in Belgium are not in any way related to varieties in the Netherlands having the same colour. Different colours only identify distant varieties within a country.

To help interpret the colour maps, we calculated all dialect distances on the basis of the pronunciations of every single word in our GTRP subset. By correlating these distances with the distances of every MDS dimension, we were able to identify the words which correlated most strongly with the distances of the separate MDS dimensions.

For the Netherlands we found that the dialect distances on the basis of the first MDS dimension (separating Low Saxon from the rest of the Netherlands) correlated most strongly ($r = 0.66$) with distances obtained on the basis of the pronunciation of the word *moeten* 'must'. For the second MDS dimension (separating the north of the Netherlands, most notably Friesland, from the rest of the Netherlands) the word *donderdag* 'Thursday' showed the highest correlation ($r = 0.59$). The word *schepen* 'ships' correlated most strongly ($r = 0.49$) with the third MDS dimension (primarily separating Limburg from the rest of the Netherlands). For Belgium we found that the dialect distances obtained on the basis of the pronunciation of the word *wol* 'wool' correlated most strongly ($r = 0.82$) with the first MDS dimension (separating eastern and western Belgium). The word *schrijven* 'write' correlated most strongly ($r = 0.63$) with the second MDS dimension (separating the middle part of Belgium from the outer eastern and western part), while the word *vrijdag* 'Friday' showed the highest correlation ($r = 0.50$) with the third MDS dimension (primarily separating Ghent and the outer eastern Belgium part from the rest). Figure 2.6 and 2.7 also display these words and corresponding pronunciations in every region.

On the map of the Netherlands, varieties of the Frisian language can clearly be distinguished by the blue colour. The town Frisian varieties are purpler than the rest of the Frisian varieties. This can be seen clearly in the circle representing the Leeuwarden variety. The Low Saxon area can be identified by a greenish colour. Note that the dialect of Twente (near Oldenzaal) is distinguished from the rest of Overijssel by a less bluish green colour. The Low Franconian dialects of the Netherlands can be identified by their reddish tints. Due to its bright red colour, the dialect of Limburg can be identified within the Low Franconian dialects of the Netherlands.

For the Belgian varieties, the dialects of West Flanders (green) and Brabant (blue) can be clearly distinguished. In between, the dialects of East Flanders (light blue) and Limburg (red) can also be identified. Finally, the distinction between Ghent (pink) and its surrounding varieties (greenish) can be seen clearly.

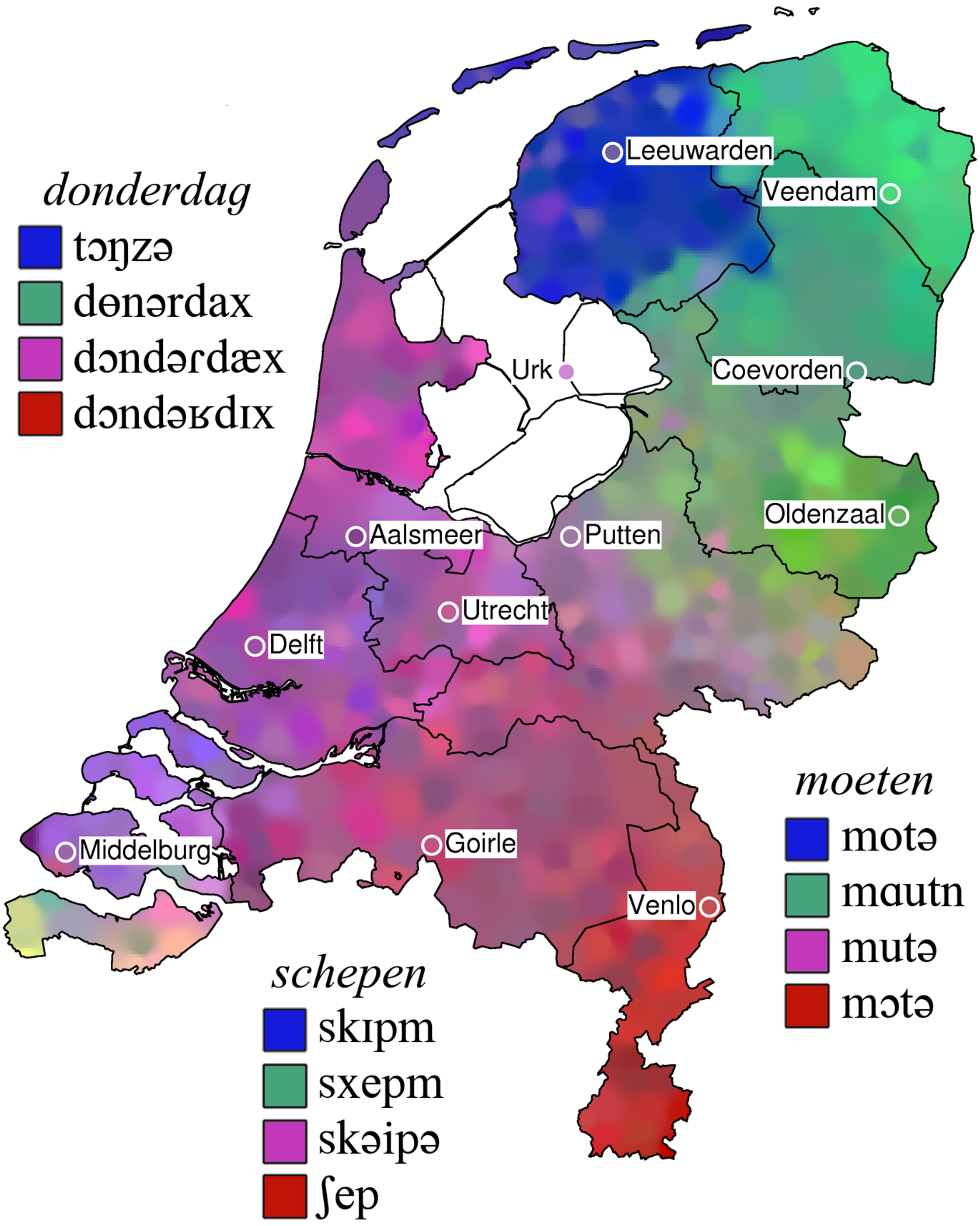

Figure 2.6. The GTRP data of the Netherlands reduced to its three most important dimensions via MDS (accounting for roughly 88% of dialect variation). Pronunciations of the word *moeten* 'must', *donderdag* 'Thursday', and *schepen* 'ships' correlate most strongly with the first, second and third MDS dimension respectively.

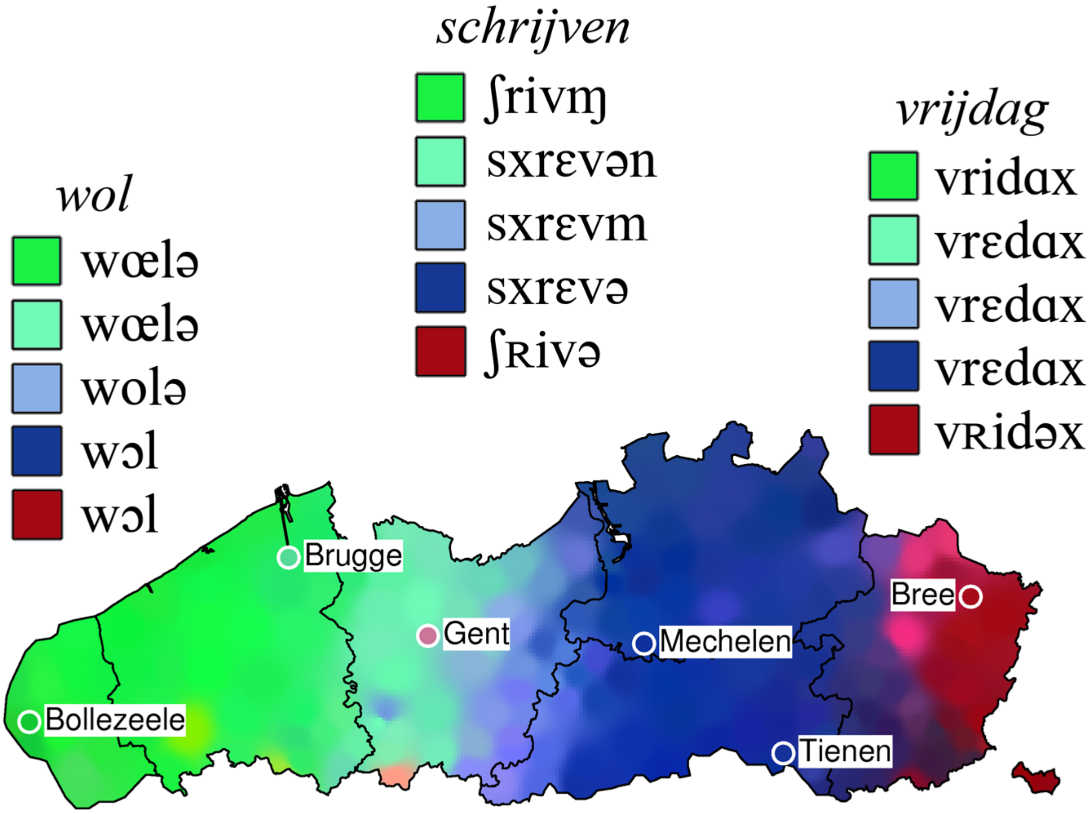

Figure 2.7. The GTRP data of Belgium reduced to its three most important dimensions via MDS (accounting for roughly 88% of dialect variation). Pronunciations of the word *wol* 'wool', *schrijven* 'write' and *vrijdag* 'Friday' correlate most strongly with the first, second and third MDS dimension respectively.

Apparent dialect islands

A careful examination of Figure 2.6 reveals a few sites whose MDS dimensions (and therefore colours) deviate a great deal from their surroundings. For example, there are two bright points around Twente (above the Oldenzaal label) which might appear to be dialect islands. Upon inspection it turns out that these points both used transcriptions by the same fieldworker, who, moreover, contributed almost only those (four) sets of transcriptions to the entire database. We therefore strongly suspect that the apparent islands in Twente are "transcriber isoglosses". Also Hinskens and Van Oostendorp (2006) reported the existence of transcriber effects in the GTRP data.

But the points in Twente are not the only apparent dialect islands. What can we do about this? Unfortunately, there are no general and automated means of correcting deviant transcriptions or correcting analyses based on them. At very abstract levels we can correct mathematically for differences in a very small number of transcribers (or fieldworkers), but we know of no techniques that would apply in general to the GTRP data. It is possible to compare analyses which exclude suspect data to analyses which include it, but we should prefer not to identify suspect data only via its deviance with respect to its neighbours.

2.4.3 GTRP compared to RND

Our purpose in the present section is to examine the GTRP against the background of the RND in order to detect whether there have been changes in the Dutch dialect landscape. We employ a regression analysis (below) to detect areas of relative convergence and divergence. The regression analysis identifies an overall tendency between the RND and GTRP distances, against which convergence and divergence may be identified: divergent sites are those for which the actual difference between the RND and GTRP distances exceeds the general tendency, and convergent sites are those with distances less than the tendency.

We are not analysing the rate of the changes we detect. Given the large time span over which the RND was collected, it would be illegitimate to interpret the results of this section as indicative of the rate of pronunciation change. This should be clear when one reflects first, that we are comparing both the RND and the GTRP data at the times at which they were recorded, and second, that the RND data was recorded over a period of fifty years. One could analyse the rate of change if one included the time of recording in the analysis, but we have not done that.

We verify first that the regression analysis may be applying, starting with the issue of whether there is ample material for comparison.

In Section 2.2.2 we mentioned that the comparison between the RND and GTRP in this chapter are based only on the 224 common varieties and the 59 common words. Although one might find this number of words quite small, we still obtained consistent results.

When we use the RND data, Cronbach's α is 0.95 for the data from the Netherlands and 0.91 for the data from Belgium. For the GTRP data we found Cronbach's α values of 0.91 and 0.95 respectively.

We correlated the RND distances with the GTRP distances and found a correlation of $r = 0.83$ for the Netherlandic distances, and a correlation of $r = 0.82$ for the Belgian distances. These correlations are significant ($p < 0.001$) according to the Mantel test, a test which takes into account the fact that the distances within a distance matrix are not fully independent (see Heeringa, 2004: 74 – 75 for a brief explanation of this test). The correlations indicate a strong, but not perfect relationship between the old RND dialect distances and the newer GTRP dialect distances. In the sections below we will examine these differences.

Comparison of transcriptions and distances

In Section 2.3 we described how we have measured pronunciation distances. The RND and the GTRP distances are measured in the same way, but the measurements are based on different kinds of transcriptions. As shown in Section 2.4.1, these differences may be reflected in the number of different phonetic symbols used in the transcriptions. Therefore we counted the number of different speech segments in the set of common varieties and common words for both the RND and the GTRP. Ignoring suprasegmentals and diacritics we found the following results:

	RND original	**RND modified**	**GTRP**
Netherlands	43	40	73
Belgium	42	40	44

In the column 'RND original' counts are given on the basis of the original, unchanged transcriptions. When calculating Levenshtein distances, we used a modified version of the transcriptions in which some of the different notations used by different transcribers, are normalised (see Heeringa, 2004: 217 – 226). Counts on the basis of these modified transcriptions are given in the column 'RND modified'.

If we wished to compare pronunciation directly between the RND and the GTRP, it would be important to verify that same scale. The table above shows that the number of different segments is about the same in all cases, except for the Netherlandic GTRP data which has a much higher number of different tokens (73). We now examine whether the number of different tokens influences our Levenshtein distance measurements. For both countries within each data source we calculated the mean and the standard deviation of the average Levenshtein distances of all pairs of varieties. Remember that each dialect distance represents the average number of operations needed to transform one pronunciation into another. We found the following results:

	RND mean	**GTRP mean**	**RND st. dev.**	**GTRP st. dev.**
Netherlands	1.58	2.03	0.51	0.44
Belgium	1.47	1.64	0.36	0.52

When comparing the means with the corresponding number of different tokens in the table above, we find the expected tendency that a lower number of distinctive tokens corresponds to lower distances. We do not find a clear relationship between the standard deviations and the number of different tokens.

We compared the RND dialect distances to corresponding GTRP dialect distances by means of a matched-pairs t-test. It turns out that GTRP distances are higher than the RND distances ($p < 0.001$ for both the Netherlands and Belgium). We emphasise that we do not interpret this as evidence that the Dutch dialects are diverging from one another in general for reasons we turn to immediately.

The differences in the number of different tokens on the one hand, and the differences in distances on the other hand, show that the results of the GTRP cannot be directly compared to the results of the RND. We will therefore use regression analysis to compare the results of the two different data sources.

Linear regression analysis

The idea behind regression analysis is that a dependent variable can be predicted by an independent variable. A linear regression procedure finds a formula which defines a linear relationship between the independent variable and the dependent variable. Because the relationship will usually not be perfectly linear, the values predicted by the formula on the basis of the independent variable will differ from the values of the dependent variable. The differences between the predicted values and the real observations of the dependent variable are called residues.

Since the past may influence the present but not vice versa, we regard the RND distances as the independent variable, and the GTRP distances as dependent. With regression analysis we obtain differences between the predicted GTRP distances and the real GTRP distances, i.e. the residues. These residues can be either positive or negative. A positive residue means that the real GTRP distance is larger than the GTRP distance predicted on the basis of the corresponding RND distance. A negative residue means the real GTRP distance is lower than expected on the basis of the corresponding RND distance.

As mentioned above, we cannot directly compare GTRP distances with RND distances. This means that we cannot determine whether varieties have converged or diverged in absolute terms. But residues tell us whether and to what degree some varieties have become relatively closer, and others relatively more distant. ‘Relatively’ means: in relation to distances of the other dialect pairs. We will refer to this as ‘relative convergence’ and ‘relative divergence’.

For instance, assume that variety A converged to variety B, variety C converged to variety D, and variety E converged to variety F. The varieties A and B converged more strongly than varieties C and D, and varieties E and F converged less strongly than varieties C and D. We are not able to detect the overall pattern of convergence, but we are able to detect that the relationships among the dialect pairs have changed with respect to their relative distances. Ignoring the overall pattern, we would find that varieties A and B have relatively converged, and varieties E and F have relatively diverged.

Figure 2.8 shows the residues. Varieties which have relatively converged are connected by blue lines, and varieties which have relatively diverged are connected by red lines. When we consider the Netherlands, we find that the Frisian dialects in the northwest, and the dialects in the eastern part of the province of Noord-Brabant (south of Goirle) and those in the province of Limburg (north and south of Venlo) have relatively converged.

The Frisian dialects are known to be very homogeneous. Therefore it is striking that the dialects became—relatively—even more similar to each other. The Frisian dialects have not converged toward the surrounding dialects, for example toward the Noord-Holland dialects, which are relatively close to standard Dutch. The internal convergence could be the result of the influence of standard Frisian in which case these dialects have become more standardised, i.e. closer to standard Frisian.

In contrast, the Limburg dialects are known to be very heterogeneous and relatively distant from standard Dutch. The strong relative convergence of Limburg and eastern Noord-Brabant dialects may be explained by convergence towards standard Dutch, which makes them more similar to each other and to some surrounding dialects which are relatively similar to standard Dutch. This idea is supported by a slight relative convergence toward dialects north of Brabant, in the south of the province of Gelderland.

Strong relative divergence is found among the Twente varieties, the area including and west of Oldenzaal. We have no good dialectological explanation for this. However, there were a large number of transcribers (6) in this small region and it could be that the divergence is caused by transcriber problems (e.g., see Section 2.4.2).

When examining Flanders in Figure 2.8, we find relative convergence in most provinces, probably again as the result of convergence towards standard Dutch. One clear exception is the variety of Ghent. Phonologically the variety of Ghent differs strongly from the surrounding varieties. For instance, all vowels in the variety of Ghent are longer than in the surrounding varieties. Since this development concerns a single variety, we would wish to verify that the Ghent data has been collected and transcribed in the same manner as the other data of other varieties. Assuming that the material is reliable and comparable, we would conjecture that the variety of Ghent has been influenced much less by standard (Flemish) Dutch, making the contrast to the surrounding dialects larger.

Principal component analysis

Principal component analysis (PCA) is a technique used for reducing multiple dimensions of a dataset to a lower number of dimensions. Dimensions which show similar patterns across the items, thus having high correlations, are fused to a single component. The PCA procedure transforms the data to a lower number of components so that the greatest variance is placed on the first principal component, the second greatest variance

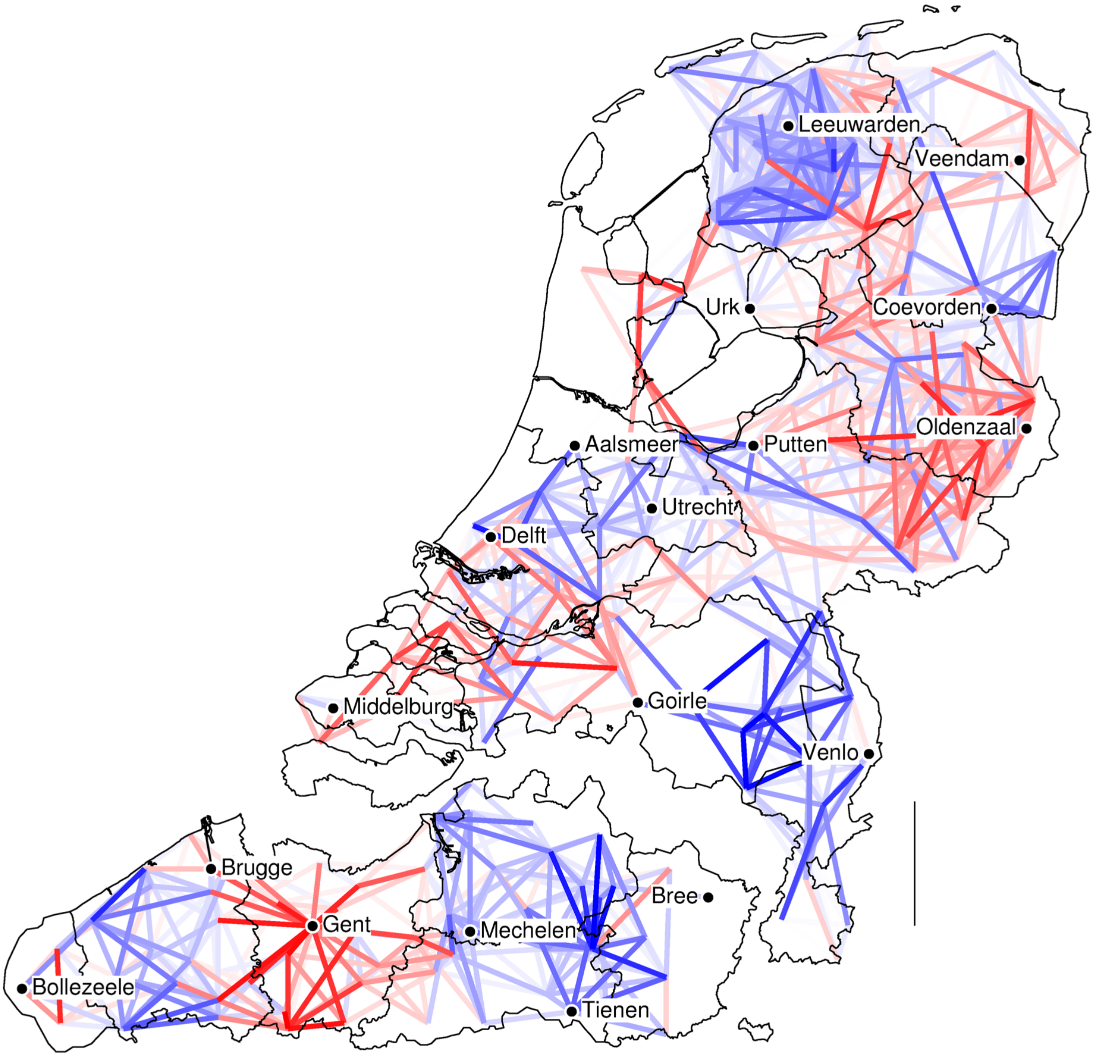

Figure 2.8. Relative convergence and divergence among dialects. Relative convergence means that dialects have become closer in relation to distances of the other dialect pairs and relative divergence means that dialects have become more distant in relation to distances of the other dialect pairs. The intensity of blue (red) represents the degree of relative convergence (divergence), and lines longer than the black vertical line in the lower right corner are not shown.

on the second component, and so on. The number of dimensions is reduced so that characteristics of the dataset that contribute most to its variance are retained (Tabachnik and Fidell, 2001: Chapter 13).

With linear regression analysis we obtained a residue for each pair of varieties. When we have n_d varieties, each variety has a residue to each of the other $n_d - 1$ varieties and to itself (which is always 0). In this way a variety is defined as a range of n_d values, i.e. there are n_d dimensions. Each dimension shows a pattern of relative convergence and divergence among the varieties.

Because we have 164 varieties in the Netherlands, they are represented by 164 dimensions. The SPSS PCA procedure reduces the number of dimensions to 14 components. The first component represents 34.9% of the variance in the original data, the second component represents 13.6%, the third component 11.5%, etc. The 60 Belgian varieties represent 60 dimensions. With PCA the number of dimensions is reduced to 7 components. The first component represents 41.8% of the variance in the residues, the second one represents 22.5%, the third one 8%, etc. As we mentioned above, the greatest variance is placed on the first component. In Figure 2.9 the values of the first principal component are geographically visualised. Higher values are represented by lighter grey tones.

Considering the Netherlands, we find a sharp distinction between the Frisian area which is nearly white, and the rest of the Netherlands which is coloured more darkly. White colours signify dialects which behave similar to Frisian, and in this case, this is only Frisian. White thus means that varieties have a strong relative convergence towards Frisian. Black represents varieties without any pattern which converge or diverge to all other varieties to the same degree. So the main finding is that Frisian dialects converged with respect to each other, but not with respect to other dialects. Especially striking is the dark area found between Oldenzaal and Putten. This area is geographically close to the border between the northeastern Low Saxon dialects and the southern Low Franconian dialects. They do not converge or diverge more strongly with respect to some dialects as compared to others. Although this dark area could also be caused by the possible transcriber effect discussed in Section 2.4.2.

When looking at Flanders, we see a clear east-west division. The east is coloured nearly white, especially the province of Antwerp (north of Mechelen). The western part is coloured more darkly. White means that varieties have a strong relative convergence to dialects in the east (Brabant, Antwerp, and Limburg). Dark represents varieties that strongly converged toward dialects in the west (French Flanders and West Flanders). So the main pattern is that western varieties and eastern varieties both converge internally, even while they do not converge toward each other.

2.5 Discussion

In this chapter we have provided an aggregate analysis of the pronunciation in contemporary Dutch dialects as these are sampled in the GTRP. The sheer scale of the GTRP

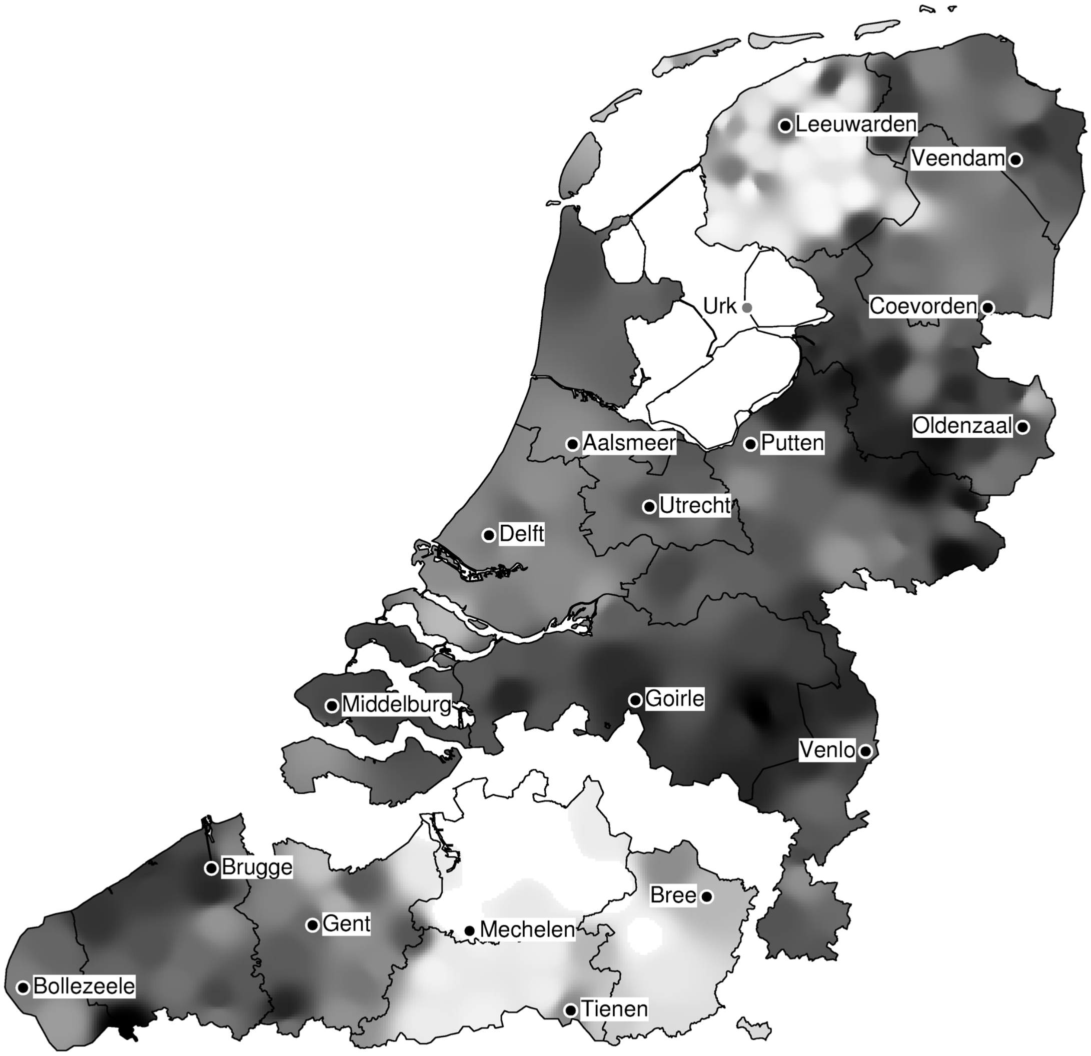

Figure 2.9. Grey tones represent values of the first component obtained with principal component analysis applied to the residues shown in Figure 2.8. Varieties which have the same pattern of relative convergence and divergence with respect to other varieties show similar grey tones. Thus Friesland and East Flanders house groups of dialects which have developed similarly within the two countries, and in fact, convergently. The maps of the Netherlands and Belgium should be interpreted independently from each other.

guarantees the basis for a reliable analysis, which in turn demonstrates that the Dutch-speaking landscape is still richly contoured with Friesland, Limburg and Low Saxony as the most distinct areas.

In order to protect our analysis from potential, perhaps subtle differences in measurement scale due to transcription differences between the RND and the GTRP, we used the residues of a regression analysis in order to identify the most dynamic areas of convergence and divergence. The comparison between the situation in roughly the mid-twentieth century as documented in the RND and the current situation (as documented by the GTRP) revealed that Friesland, Flemish Brabant, West Flanders, and Limburg are areas of dynamic convergence, while Ghent and the southeastern part of Low Saxony are areas of divergence. We also qualified this general characterisation, noting that the RND was collected over a fifty year period, which prevents us from drawing conclusions with respect to the rate of pronunciation change.

We extracted the first principal component from the residues of the regression analysis, which revealed that Friesland and eastern Flanders are behaving coherently. We would like to emphasise that the use of regression analysis, including the application of PCA to its residues, is an innovation in dialectometric technique.

In addition, we examined an apparent discrepancy in the degree of phonetic discrimination provided by GTRP transcriptions for the Netherlands as opposed to that provided for transcriptions for Belgium. After further examination, we concluded that the discrepancy is genuine, and that care is required in analyses involving subsamples of the GTRP involving sites in both countries. An aggregate analysis such as ours is certainly prone to confounding due to discrepancies in data sampling, recording and transcription, but let us hasten to add that single variable analyses are by no means immune to these problems.

This line of work suggests several further investigations. First, it would be interesting to attempt to interpret the second and third principal components of the relative changes, an undertaking which would require more attention to phonetic detail than we have exercised here. Second, we are interested in potential means of correcting for the sorts of transcription differences noted. Are there automatic means of "reducing" the more detailed transcriptions to less detailed ones? Or must we settle for purely numeric corrections, which would mean that we have little to no opportunity to interpret the "corrections" linguistically? A project which would interest us, but which could only be undertaken in collaboration with the "stewards" of the GTRP would be to map the more complex Dutch transcription system onto the simpler Flemish one. This could, of course, turn out to involve too many decisions about individual sounds to be feasible, but it could also turn out to be straightforward.

Third, in discussing the Netherlandic part of the GTRP we noted clues that fieldworker and transcription practices may be confounding analyses to some degree (see also Hinskens and Van Oostendorp, 2006). This potential confound is bothersome, and it would be rewarding to eliminate it. The most rewarding, but the most difficult strategy would be to try to analyse pronunciation difference purely acoustically, eliminating the need for transcriptions. Perhaps more realistic would be to develop strategies to identify clues that transcriptions are being produced differently and also to quantify the degree to which different transcription might distort measurements. But even in the absence of general techniques, it would be useful to know where transcriber differences may exist in the GTRP.

Finally, a very exciting, and also promising opportunity suggests itself in the rich sample of morphological variation represented in the GTRP, which, after all, is the basis of the *Morfologische Atlas van de Nederlandse Dialecten* (MAND; De Schutter et al., 2005). Although Seguy (1973) and Goebl (1984) include morphological variables in their dialectometric work, the morphological material is analysed at a categorical level, i.e. in which only "same" and "different" are distinguished. The development of a measure of morphological distance reflecting not only the potentially differing exponence of common morphological categories (which after all are already reflected in pronunciation difference), but also reflecting the effect of non-coincidental categories (such as the second Frisian infinitive), would be a rewarding challenge.

3 Dialect pronunciation comparison using Pair Hidden Markov Models

Abstract[*]

Pair Hidden Markov Models (PairHMMs) are trained to align the pronunciation transcriptions of a large contemporary collection of Dutch dialect material, the Goeman-Taeldeman-Van Reenen-Project (GTRP, collected 1980 – 1995). We focus on the question of how to incorporate information about sound segment distances to improve sequence distance measures for use in dialect comparison. PairHMMs induce segment distances via expectation maximisation (EM). Our analysis uses a phonologically comparable subset of 562 items for 424 localities in the Netherlands and 189 localities in Belgium. We evaluate the work first via comparison to analyses obtained using the Levenshtein distance on the same datasets and second, by comparing the quality of the induced vowel distances to acoustic differences.

3.1 Introduction

Dialectology catalogues the geographic distribution of the linguistic variation that is a necessary condition for language change (Wolfram and Schilling-Estes, 2003), and is sometimes successful in identifying geographic correlates of historical developments (Labov, 2001). Computational methods for studying dialect pronunciation variation have been successful using various edit distance and related string distance measures, but unsuccessful in using segment differences to improve these (Heeringa, 2004). The most successful techniques distinguish consonants and vowels, but treat e.g. all the vowel differences as the same. Ignoring the special treatment of vowels vs. consonants, the techniques regard segments in a binary fashion—as alike or different—in spite of the overwhelming consensus that some sounds are much more alike than others. There have been many attempts to incorporate more sensitive segment differences, which do not necessarily perform worse in validation, but they fail to show significant improvement (Heeringa, 2004).

Instead of using segment distances as these are (incompletely) suggested by phonetic or phonological theory, we can also attempt to acquire these automatically.

[*] An adapted version of this text, excluding Belgian results, was accepted at the workshop *Computing and Historical Phonology: 9th ACL Special Interest Group for Morphology and Phonology* (2007) as: M. Wieling, T. Leinonen, and J. Nerbonne. Inducing Sound Segment Differences using Pair Hidden Markov Models.

Mackay and Kondrak (2005) introduce Pair Hidden Markov Models (PairHMMs) to language studies, applying them to the problem of recognising "cognates" in the sense of machine translation, i.e. pairs of words in different languages that are similar enough in sound and meaning to serve as translation equivalents. Such words may be cognate in the sense of historical linguistics, but they may also be borrowings from a third language. We apply PairHMMs to dialect data for the first time in this paper. Like Mackay and Kondrak (2005) we evaluate the results both on a specific task, in our case, dialect classification, and also via examination of the segment substitution probabilities induced by the PairHMM training procedures. We suggest using the acoustic distances between vowels as a probe to explore the segment substitution probabilities induced by the PairHMMs.

Naturally, this validation procedure only makes sense if dialects are using acoustically more similar sounds in their variation, rather than, for example, randomly varied sounds. But why should linguistic and geographic proximity be mirrored by frequency of correspondence? Historical linguistics suggests that sound changes propagate geographically, which means that nearby localities should on average share the most changes. In addition some changes are convergent to local varieties, increasing the tendency toward local similarity. The overall effect in both cases strengthens the similarity of nearby varieties. Correspondences among more distant varieties are more easily disturbed by intervening changes and decreasing strength of propagation.

In the next section the dialectal material used in this chapter is discussed. Section 3.3 explains the Pair Hidden Markov Model, while the results are presented in Section 3.4. Finally, Section 3.5 will provide a discussion of these results.

3.2 Material

In this chapter the same data is used as introduced in Section 2.2.1. Because the GTRP transcriptions of Belgian varieties are fundamentally different from transcriptions of Netherlandic varieties (see Section 2.4.1), we will use and analyse the data of the 424 varieties in the Netherlands separately from the data of the 189 Belgian varieties. The geographic distribution of the varieties for both Belgium and the Netherlands can be seen in Figure 2.1.

3.3 The Pair Hidden Markov Model

In this chapter we will use a Pair Hidden Markov Model (PairHMM), which is essentially a Hidden Markov Model (HMM) adapted to assign similarity scores to word pairs and to use these similarity scores to compute string distances. For completeness we will start with a short introduction of the Hidden Markov Model. For a more detailed introduction, please refer to Rabiner and Juang (1986) or Rabiner (1989).

The Hidden Markov Model μ is defined by the following parameters (see Mackay, 2004):

- Set of states: $S = \{s_1, \cdots, s_N\}$
- Output alphabet: $\Sigma = \{\sigma_1, \cdots, \sigma_M\}$
- Initial state probabilities: $\Pi = \{\pi_i \,|\, i \in S\}$
- State transition probabilities: $A = \{a_{i,j} \,|\, i, j \in S\}$
- State emission probabilities: $E = \{e_k(b) \,|\, k \in S, b \in \Sigma\}$
- State sequence: $\mathbf{X} = (X_1, \cdots, X_T)$, $X_t \in S$
- Output sequence: $\mathbf{O} = (o_1, \cdots, o_T)$, $o_t \in \Sigma$

An HMM generates an output sequence (i.e. observation sequence) based on the parameters above. The HMM starts in a state i with probability π_i, goes from a state i to a new state j with probability a_{ij}, and emits a certain output symbol b in every state k with probability $e_k(b)$. Formally the HMM μ is defined as a triplet: $\mu = (A, E, \Pi)$.

Unlike a regular Markov Model, the output sequence of a Hidden Markov Model does not uniquely determine the state sequence. In a regular Markov Model every state emits a determined output symbol, but in an HMM a state can output one of several output symbols based on the emission probabilities. Hence, the term "hidden" refers to the hidden state sequence (Eddy, 2004). The probability of an observation sequence given an HMM can be calculated by using well known HMM algorithms such as the Forward algorithm and the Viterbi algorithms (e.g., see Rabiner, 1989).

A Pair Hidden Markov Model (PairHMM) is an adapted Hidden Markov Model in the sense that it generates two (aligned) observation streams in parallel instead of one. The PairHMM was originally proposed by Durbin et al. (1998) and has successfully been used for aligning biological sequences. Mackay and Kondrak (2005) adapted the algorithm to calculate similarity scores for word pairs in orthographic form, focusing on identifying translation equivalents in bilingual corpora.

Their modified PairHMM has three emitting states representing the basic edit operations: a substitution state (M), a deletion state (X) and an insertion state (Y). In the substitution state two symbols x_i and y_j are emitted with probability $p_{x_i y_j}$. In the deletion state symbol x_i is emitted against a gap with probability q_{x_i} and in the insertion state a gap is emitted against symbol y_j with probability q_{y_j}. The model is shown in Figure 3.1. The four transition parameters are specified by λ, δ, ε and τ. There is no explicit start state; the probability of starting in one of the three states is equal to the probability of going from the substitution state to that state. The only non-emitting state in the PairHMM is the end state which is always the final state in a state sequence.

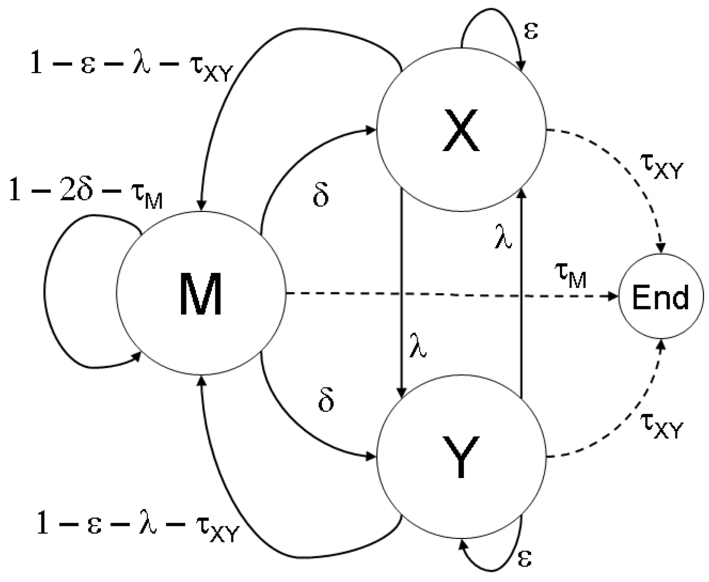

Figure 3.1. Pair Hidden Markov Model. Image courtesy of Mackay and Kondrak (2005).

In this chapter we use the PairHMM of Mackay and Kondrak (2005) to align phonetically transcribed words. For instance, a possible alignment (including the state sequence) for two Dutch dialectal variants of the word 'milk', [mɔəlkə] and [mɛlək], is given by:

m	ɔ	ə	l		k	ə
m	ɛ		l	ə	k	
M	M	X	M	Y	M	X

We will use the Pair Hidden Markov Model to obtain a similarity score for every word pair. For that purpose in the following section several PairHMM algorithms are introduced and explained. However, before the Pair Hidden Markov Model can be used to calculate the similarity scores, the parameters of the PairHMM have to be determined. Section 3.3.2 introduces the Baum-Welch algorithm which can be used to estimate these parameters. All PairHMM algorithms discussed in these sections have been described by Durbin et al. (1998) and adapted to the word alignment task by Mackay and Kondrak. (2005; however more detailed descriptions can be found in Mackay, 2004).

3.3.1 Calculating word pair similarity scores using the PairHMM

There are several ways to calculate the similarity score for a given word pair when the transition and emission probabilities of the PairHMM are known. First, we can use the Forward (or Backward) algorithm, which takes all possible alignments into account, to calculate the probability of the observation sequence given the PairHMM and use this probability as a similarity score (corrected for length). Second, we can use the Viterbi algorithm to calculate the probability of the best alignment and use this probability as a similarity score (also corrected for length). Third, we can adapt the Viterbi and Forward algorithms to take into account how likely it is that a pair of words occur together while they have no underlying relationship (i.e. the log-odds algorithm). These algorithms will be explained in detail in the following sections. To help understand the specific PairHMM algorithms, in every section the general HMM algorithm is explained first.

The Forward algorithm

In general, a naive method to calculate the probability of an observation sequence of length T using a Hidden Markov Model is to calculate the probability for every state sequence (each consisting of $\mathcal{O}(T)$ operations) and sum them. For N states there are N^T possible state sequences thus yielding an exponential time complexity of $\mathcal{O}(TN^T)$ (see Rabiner, 1989).

To find a more efficient solution, we can make use of the dynamic programming approach introduced in Section 2.3.1. The main idea is that the complexity can be reduced by keeping track of optimal solutions to sub-problems (observation sequences of shorter length). This algorithm is called the Forward algorithm and uses the Forward variable $\alpha_i(t)$ to store the probability of being in state i at time t (i.e. $X_t = i$) and having seen the partial observation sequence $o_1 \cdots o_t$. The Forward variable is formally defined as:

$$\alpha_i(t) = P(o_1 \cdots o_t, X_t = i|\mu). \tag{3.1}$$

Table 3.1 shows how the Forward variables are calculated recursively for every time step t and are combined to give the probability of the observation sequence **O** for the model μ. The time complexity of the Forward algorithm is $\mathcal{O}(TN^2)$ and therefore much more efficient than the naive approach (Rabiner, 1989).

Table 3.2 shows the Forward algorithm adapted for the PairHMM described by Mackay (2004; see also Figure 3.1). In this table $^\bullet$ represents an action performed for all states: M (substitution state), X (deletion state) and Y (insertion state). The value $p_{x_i y_j}$ is the probability of matching a character at position i in observation stream x (i.e. string x of length n) with a character at position j in string y of length m (a substitution). Analogously, q_{x_i} is the probability of matching a character at position i in string x with a gap in string y (a deletion), while q_{y_j} is the probability of matching a gap in string x with a character at position j in string y (an insertion).

The conversion is straightforward; the most notable change is that there are two observation streams instead of one. The positions in these observation streams are indicated by i and j respectively.

In calculating the observation probability, the PairHMM Forward algorithm considers all possible alignments of the two observation streams. For instance, when calculating

1. Initialisation

$$\alpha_i(1) = \pi_i e_i(o_1), \quad 1 \leq i \leq N.$$

2. Induction

$$\alpha_j(t) = e_j(o_t) \sum_{i=1}^{N} \alpha_i(t-1) a_{ij}, \quad 1 < t \leq T, 1 \leq j \leq N.$$

3. Termination

$$P(\mathbf{O}|\mu) = \sum_{i=1}^{N} \alpha_i(T).$$

Table 3.1. Forward algorithm for Hidden Markov Models. This formulation assumes an observation sequence $o_1 \cdots o_n$ where o_j is the observation at time j. The value $\alpha_i(t)$ is the chance of being in state i at time t and having seen the partial observation sequence $o_1 \cdots o_t$. Note that it sums over all the paths through the HMM.

the probability of the observation sequences [a b] and [b a], the probabilities of the following alignments are calculated (with – indicating a gap):

a b b a	a b – b – a	a b – – b a	a b – – – – b a	a – b b a –
a – b – b a	a – b – – b – a	a – – b – b a –	– a b b a –	– a b b – a
– a b – b – – a	– a – b b – a –	– – a b a b – –		

The probability calculated by the Forward algorithm can be used to determine the similarity between word pairs. However, due to the multiplication of probabilities, longer observation sequences will yield a relative lower probability than shorter observation sequences. To correct for this effect in determining similarity, a correction depending on the length q of the longest observation sequence is applied to yield the final Forward similarity score. The constant value C was determined by experimentation (in our case $C = 0.08$).

$$\frac{P(\mathbf{O}|\mu)}{C^q}. \tag{3.2}$$

1. Initialisation

$$f^M(0,0) = 1 - 2\delta - \tau_M, f^X(0,0) = f^Y(0,0) = \delta.$$

$$\forall i,j : f^\bullet(i,-1), f^\bullet(-1,j) = 0.$$

2. Induction: for $0 \leq i \leq n, 0 \leq j \leq m,$ except $(0,0)$

$$\begin{aligned} f^M(i,j) &= p_{x_iy_j}[(1 - 2\delta - \tau_M)f^M(i-1,j-1) \\ &\quad + (1 - \epsilon - \lambda - \tau_{XY})(f^X(i-1,j-1) + f^Y(i-1,j-1))]. \\ f^X(i,j) &= q_{x_i}[\delta f^M(i-1,j) + \epsilon f^X(i-1,j) + \lambda f^Y(i-1,j)]. \\ f^Y(i,j) &= q_{y_j}[\delta f^M(i,j-1) + \epsilon f^Y(i,j-1) + \lambda f^X(i,j-1)]. \end{aligned}$$

3. Termination

$$P(\mathbf{O}|\mu) = \tau_M f^M(n,m) + \tau_{XY}(f^X(n,m) + f^Y(n,m)).$$

Table 3.2. Forward algorithm for Pair Hidden Markov Models

The Backward algorithm

The Forward algorithm calculates the observation probability $P(\mathbf{O}|\mu)$ by starting at the beginning of the observation sequence. Correspondingly, it is also possible calculate $P(\mathbf{O}|\mu)$ by starting at the end of the observation sequence. This approach is followed in the Backward algorithm which uses the Backward variable $\beta_i(t)$ to store the probability of being in state i at time t and seeing the partial observation sequence $o_{t+1} \cdots o_T$. It is formally defined as:

$$\beta_i(t) = P(o_{t+1} \cdots o_T | X_t = i, \mu). \tag{3.3}$$

The Backward algorithm is highly similar to the Forward algorithm and is shown in Table 3.3 for the regular Hidden Markov Model and in Table 3.4 for the PairHMM (Mackay, 2004). The time complexity of the Backward algorithm is equal to the time complexity of the Forward algorithm, $\mathcal{O}(TN^2)$ (Rabiner, 1989).

As in the Forward case, we correct for the length of the longest observation sequence q with equation (3.2) to obtain the word similarity score.

1. Initialisation

$$\beta_i(T) = 1, \quad 1 \leq i \leq N.$$

2. Induction

$$\beta_i(t) = \sum_{j=1}^{N} \beta_j(t+1) a_{ij} e_j(o_{t+1}), \quad T > t \geq 1,\ 1 \leq i \leq N.$$

3. Termination

$$P(\mathbf{O}|\mu) = \sum_{i=1}^{N} \pi_i \beta_i(1).$$

Table 3.3. Backward algorithm for Hidden Markov Models

1. Initialisation

$$b^M(n,m) = \tau_M, b^X(n,m) = b^Y(n,m) = \tau_{XY}.$$

$$\forall i,j : b^\bullet(i,m+1), b^\bullet(n+1,j) = 0.$$

2. Induction: for $n \geq i \geq 0, m \geq j \geq 0$, except (n,m)

$$\begin{aligned}
b^M(i,j) &= (1 - 2\delta - \tau_M) p_{x_{i+1}y_{j+1}} b^M(i+1,j+1) \\
&\quad + \delta(q_{x_{i+1}} b^X(i+1,j) + q_{y_{j+1}} b^Y(i,j+1)). \\
b^X(i,j) &= (1 - \epsilon - \lambda - \tau_{XY}) p_{x_{i+1}y_{j+1}} b^M(i+1,j+1) \\
&\quad + \epsilon q_{x_{i+1}} b^X(i+1,j) + \lambda q_{y_{j+1}} b^Y(i,j+1)). \\
b^Y(i,j) &= (1 - \epsilon - \lambda - \tau_{XY}) p_{x_{i+1}y_{j+1}} b^M(i+1,j+1) \\
&\quad + \epsilon q_{y_{j+1}} b^Y(i,j+1) + \lambda q_{x_{i+1}} b^X(i+1,j).
\end{aligned}$$

3. Termination

$$P(\mathbf{O}|\mu) = (1 - 2\delta - \tau_M) b^M(0,0) + \delta(b^X(0,0) + b^Y(0,0)).$$

Table 3.4. Backward algorithm for Pair Hidden Markov Models

The Viterbi algorithm

Besides calculating a similarity score for the word pairs based on all alignments, it is also possible to calculate a similarity score for the word pairs based on the best alignment, i.e. the single alignment having the highest probability (Rabiner, 1989).

The Viterbi algorithm is a dynamic programming algorithm similar to the Forward algorithm and calculates the probability of the best path. In its original form the Viterbi algorithm also stores the state sequence of the best path. Because we are only interested in obtaining a similarity score for each word pair and not in the exact alignment, we do not need to store the state sequence. In this case, the Viterbi algorithm only differs from the Forward algorithm by calculating the maximum instead of the sum.

The Viterbi algorithm uses the variable $\delta_i(t)$ to store the highest probability of the single best state path through a Hidden Markov Model μ consisting of t observations and ending in state i. This variable is formally defined as:

$$\delta_i(t) = \max_{X_1 \cdots X_{t-1}} P(X_1 \cdots X_{t-1}, o_1 \cdots o_t, X_t = i|\mu). \tag{3.4}$$

The Viterbi algorithm for the general Hidden Markov Model (without storing the best state sequence X^*) is shown in Table 3.5. The time complexity of the Viterbi algorithm is $\mathcal{O}(TN^2)$ and therefore equal to the time complexity of the Forward algorithm.

Table 3.6 shows the Viterbi algorithm adapted for the PairHMM described by Mackay (2004, see also Figure 3.1). The conversion is analogous to the conversion of the Forward algorithm (see Table 3.1 and 3.2).

1. Initialisation

$$\delta_i(1) = \pi_i e_i(o_1), \quad 1 \leq i \leq N.$$

2. Induction

$$\delta_j(t) = e_j(o_t) \max_{1 \leq i \leq N} \delta_i(t-1) a_{ij}, \quad 1 < t \leq T, 1 \leq j \leq N.$$

3. Termination

$$P(X^*) = \max_{1 \leq i \leq N} \delta_i(T).$$

Table 3.5. Viterbi algorithm for Hidden Markov Models

1. Initialisation

$$v^M(0,0) = 1 - 2\delta - \tau_M, v^X(0,0) = v^Y(0,0) = \delta.$$

$$\forall i, j : v^\bullet(i,-1), v^\bullet(-1,j) = 0.$$

2. Induction: for $0 \leq i \leq n, 0 \leq j \leq m,$ except $(0,0)$

$$\begin{aligned}
v^M(i,j) &= p_{x_i y_j}\text{max}\left\{\begin{array}{l} (1-2\delta-\tau_M)v^M(i-1,j-1) \\ (1-\epsilon-\lambda-\tau_{XY})v^X(i-1,j-1) \\ (1-\epsilon-\lambda-\tau_{XY})v^Y(i-1,j-1) \end{array}\right\}. \\
v^X(i,j) &= q_{x_i}\text{max}\left\{\begin{array}{l} \delta v^M(i-1,j) \\ \epsilon v^X(i-1,j) \\ \lambda v^Y(i-1,j) \end{array}\right\}. \\
v^Y(i,j) &= q_{y_j}\text{max}\left\{\begin{array}{l} \delta v^M(i,j-1) \\ \epsilon v^Y(i,j-1) \\ \lambda v^X(i,j-1) \end{array}\right\}.
\end{aligned}$$

3. Termination

$$P(X^*) = \text{max}(\tau_M v^M(n,m), \tau_{XY} v^X(n,m), \tau_{XY} v^Y(n,m)).$$

Table 3.6. Viterbi algorithm for Pair Hidden Markov Models

The probability calculated by the Viterbi algorithm can also be used to determine the similarity between word pairs. However as in the Forward case, longer observation sequences will yield a relative lower probability than shorter observation sequences. To correct for this effect in determining the Viterbi similarity score, the same correction is applied as in (3.2) (with q being the length of the longest string and C a constant value set to 0.08):

$$\frac{P(X^*)}{C^q}. \tag{3.5}$$

The log-odds algorithm

Another method to calculate the similarity score is using the log-odds algorithm (Durbin et al., 1998). The log-odds algorithm uses a random model to represent how likely it is that a pair of words occur together while they have no underlying relationship. Because we are looking at word alignments, this means an alignment consisting of no substitutions but only insertions and deletions. Mackay and Kondrak (2005) propose a random

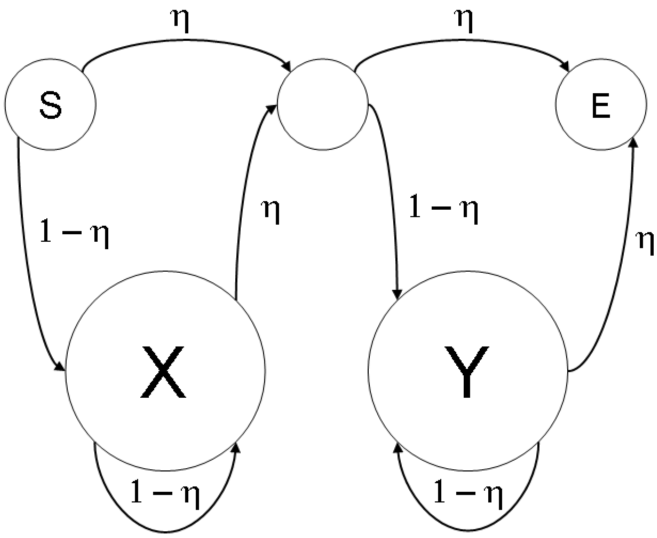

Figure 3.2. Random PairHMM. Image courtesy of Mackay and Kondrak (2005).

model which only has an insertion state and a deletion state and generates one word completely before the other (see Figure 3.2). For instance, according to the random model the alignment (including the state sequence) for two Dutch dialectal variants of the word 'milk', [mɔəlkə] and [mɛlək], is given by:

m	ɔ	ə	l	k	ə					
						m	ɛ	l	ə	k
X	X	X	X	X	X	Y	Y	Y	Y	Y

The model is described by the transition probability η. The emission probabilities can be either set equal to the insertion and deletion probabilities of the word similarity model (Durbin et al., 1998) or can be specified separately based on the token frequencies in the dataset (Mackay and Kondrak, 2005).

The final log-odds similarity score of a word pair is calculated by dividing the Viterbi or Forward probability by the probability generated by the random model, and subsequently taking the logarithm of this value. When using the Viterbi algorithm the regular log-odds score is obtained, while using the Forward algorithm yields the Forward log-odds score (Mackay, 2004). Note that there is no need for additional normalisation; by dividing two models we are already implicitly normalising. Obviously, the time complexity of the log-odds algorithm is similar to the time complexity of the Forward and Viterbi algorithms, $\mathcal{O}(TN^2)$.

3.3.2 Finding the best model parameters

Unfortunately, there is no method known that can find the model parameters that will in general maximise $P(\mathbf{O}|\mu)$. However, the best model parameters can be estimated by applying an Expectation Maximisation technique, known as the Baum-Welch algorithm (Baum et al., 1970) which uses the Forward and Backward algorithms introduced earlier. The Baum-Welch algorithm iteratively reestimates the transition and emission probabilities until a local optimum is found and has time complexity $\mathcal{O}(TN^2)$, where N is the number of states and T is the length of the observation sequence.

The Baum-Welch algorithm

Intuitively, the transition probability a_{ij} can be calculated by dividing the number of transitions from state i to state j by the total number of transitions from state i for a series of observations $\mathbf{O}$ in a model μ. Unfortunately this probability cannot be determined directly because the state path is unknown in a Hidden Markov Model. However, we can estimate this probability by dividing the *expected* number of transitions from state i to state j by the *expected* number of transitions from state i (Jurafsky and Martin, 2000). We will first define the probability ξ as the probability of being in state i at time t and state j at time $t+1$. More formally:

$$\xi_t(i,j) = P(X_t = i, X_{t+1} = j|\mathbf{O}, \mu). \tag{3.6}$$

The following holds according to the laws of probability:

$$P(X|\mathbf{O}, \mu) = \frac{P(X, \mathbf{O}|\mu)}{P(\mathbf{O}|\mu)}. \tag{3.7}$$

And thus

$$\xi_t(i,j) = \frac{P(X_t = i, X_{t+1} = j, \mathbf{O}|\mu)}{P(\mathbf{O}|\mu)}. \tag{3.8}$$

Recall that the probability of being in state i at time t after having seen the partial observation sequence $o_1 \cdots o_t$ is given by the Forward variable $\alpha_i(t)$ in (3.1). Furthermore, the probability of going from state i to state j can be calculated by multiplying the transition probability a_{ij} with the emission probability $e_j(o_{t+1})$. Finally, the probability of seeing the partial observation sequence $o_{t+1} \cdots o_T$ given that the state is j is represented by the Backward variable $\beta_j(t)$ in (3.3). Hence, we can rewrite (3.8) to:

$$\xi_t(i,j) = \frac{\alpha_i(t) a_{ij} e_j(o_{t+1}) \beta_j(t+1)}{P(\mathbf{O}|\mu)}. \tag{3.9}$$

Because $P(\mathbf{O}|\mu)$ is equal to the Forward probability or Backward probability (Table 3.1 and 3.3), we can also calculate $P(\mathbf{O}|\mu)$ by combining both probabilities. Note that we have to sum $\alpha_k(t)\beta_k(t)$ over all states k to take all state paths into account:

$$\xi_t(i,j) = \frac{\alpha_i(t)a_{ij}e_j(o_{t+1})\beta_j(t+1)}{\displaystyle\sum_{k=1}^{N} \alpha_k(t)\beta_k(t)}. \tag{3.10}$$

Summing ξ_t over all t gives the expected number of transitions from state i to state j. Dividing this value by the expected number of transitions from state i yields the estimated transition probability $\hat{a}_{ij}$. More formally:

$$\hat{a}_{ij} = \frac{\displaystyle\sum_{t=1}^{T-1} \xi_t(i,j)}{\displaystyle\sum_{t=1}^{T-1}\sum_{j=1}^{N} \xi_t(i,j)}. \tag{3.11}$$

Similarly, an estimation of the emission probability $e_i(k)$ can be found by dividing the expected number of times in state i symbol k is observed (i.e. pair of symbols in the PairHMM) by the expected number of times state i is visited.

The probability $\gamma_i(t)$ of being in state i at time t can be defined as:

$$\gamma_i(t) = P(X_t = i|\mathbf{O}, \mu). \tag{3.12}$$

Due to (3.7) this equals:

$$\gamma_i(t) = \frac{P(X_t = i, \mathbf{O}|\mu)}{P(\mathbf{O}|\mu)}. \tag{3.13}$$

Using a similar conversion as in (3.9) and (3.10), we can rewrite this to:

$$\gamma_i(t) = \frac{\alpha_i(t)\beta_i(t)}{\displaystyle\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)}. \tag{3.14}$$

Summing γ_i over all t gives the expected number of times state i is visited. This can be used to estimate the emission probability $\hat{e}_i(k)$ as shown in (3.15). Note that the numerator in (3.15) indicates how often symbol k is observed in state i.

$$\hat{e}_i(k) = \frac{\sum_{\substack{t=1 \\ o_t=k}}^{T} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}. \tag{3.15}$$

Finally, the initial state probability π_i can be estimated by the number of times state i is visited at time $t = 1$. Due to (3.12) the estimation of the initial state probability $\hat{\pi}_i$ is given by:

$$\hat{\pi}_i = \gamma_i(1). \tag{3.16}$$

The PairHMM Baum-Welch algorithm

The transformation of these formulas to the PairHMM case is straightforward; the main difference is that instead of summing over t in (3.11) and (3.15), we have to sum over the positions u and v in the two observation streams x and y (of length U and V respectively) using the PairHMM Forward and Backward algorithms (see Table 3.2 and 3.4).

To estimate the transition probabilities of the PairHMM, we first transform (3.10) into:

$$\xi_{u,v}(i,j) = \frac{f_i(u,v)a_{ij}e_j(\diamond_1)b_j(\diamond_2,\diamond_3)}{\sum_{k=1}^{N} f_k(u,v)b_k(u,v)}. \tag{3.17}$$

The value of the transition probability a_{ij} depends on states i and j and can be found in Figure 3.1. For example, if state i is M and state j is X, the value of a_{ij} equals δ. The emission probability e_j depends on state j and on the symbol or symbol pair $\diamond_1$:

- when state j equals the substitution state M: $e_j(x_{u+1}, y_{v+1}) = p_{x_{u+1}y_{v+1}}$
- when state j equals the deletion state X: $e_j(x_{u+1}) = q_{x_{u+1}}$
- when state j equals the insertion state Y: $e_j(y_{v+1}) = q_{y_{v+1}}$

The indices $(\diamond_2, \diamond_3)$ also depend on state j:

- when state j equals M: $(\diamond_2, \diamond_3) = (u+1, v+1)$
- when state j equals X: $(\diamond_2, \diamond_3) = (u+1, v)$
- when state j equals Y: $(\diamond_2, \diamond_3) = (u, v+1)$.

Using (3.17) and the fact that all word pairs (referred to by index w) are used in estimating the transition probabilities, we can transform (3.11) into:

$$\hat{a}_{ij} = \frac{\displaystyle\sum_{w}\sum_{u=0}^{\tilde{U}}\sum_{v=0}^{\tilde{V}}\xi^{w}_{u,v}(i,j)}{\displaystyle\sum_{w}\sum_{u=0}^{\tilde{U}}\sum_{v=0}^{\tilde{V}}\sum_{j=1}^{N}\xi^{w}_{u,v}(i,j)}. \tag{3.18}$$

The value of $\tilde{U}$ and $\tilde{V}$ depends on the final emitting state:

- when the final emitting state is M: $\tilde{U} = U - 1$ and $\tilde{V} = V - 1$
- when the final emitting state is X: $\tilde{U} = U - 1$ and $\tilde{V} = V$
- when the final emitting state is Y: $\tilde{U} = U$ and $\tilde{V} = V - 1$.

To estimate the emission probabilities of the PairHMM we can transform equation (3.14) into:

$$\gamma_i(u,v) = \frac{f_i(u,v)b_i(u,v)}{\displaystyle\sum_{j=1}^{N} f_j(u,v)b_j(u,v)}. \tag{3.19}$$

Using (3.19) and the fact that all word pairs (referred to by index w) are used in estimating the emission probabilities, we can transform (3.15) into:

$$\hat{e}_i(k) = \frac{\displaystyle\sum_{w}\sum_{\substack{u=0\\ \diamond_1}}^{U}\sum_{\substack{v=0\\ \diamond_2}}^{V}\gamma^{w}_{i}(u,v)}{\displaystyle\sum_{w}\sum_{u=0}^{U}\sum_{v=0}^{V}\gamma^{w}_{i}(u,v)}. \tag{3.20}$$

Where $\diamond_1$ is defined as $x_u = k_x$ when i equals the substitution or deletion state (in the insertion state a gap is present in observation stream x therefore k_x is not present) and $\diamond_2$ is defined as $y_v = k_y$ when i equals the substitution or insertion state (in the deletion state k_y is not present).

In our PairHMM, the initial state probabilities are set equal to the probability of going from the substitution state M to that state (see Figure 3.1). For completeness note that it is also possible to estimate the initial state probability π_i by estimating the number of times state i is visited at the start of both strings (for all words w), effectively transforming (3.16) into:

$$\hat{\pi}_i = \sum_w \gamma_i^w(0,0). \tag{3.21}$$

After starting with the initial probabilities, Π, A, E of the model μ we can estimate new parameters $\hat{\Pi}$, $\hat{A}$, $\hat{E}$ of the new model $\hat{\mu}$ by using the calculations explained above. Baum et al. (1970) proved that

$$P(\mathbf{O}|\hat{\mu}) \geq P(\mathbf{O}|\mu). \tag{3.22}$$

Because the probabilities will only be equal at a critical point (e.g., a local maximum), we can start with a random set of probabilities and run the procedure above to increase the quality of the model with respect to the training data. This procedure can be repeated until $P(\mathbf{O}|\hat{\mu}) = P(\mathbf{O}|\mu)$ and the best model is obtained for the specific set of starting probabilities. Because the Baum-Welch algorithm applies the Forward and Backward algorithms, all possible alignments of two words are considered in each training iteration (e.g., see the alignments for the observation sequences [a b] and [b a] at page 40).

Calculating dialect distances

When the parameters of the complete model have been determined, the model can be used to calculate the alignment probability for every word pair. As in Mackay and Kondrak (2005) and explained earlier, we use the Forward and Viterbi algorithms in both their regular (normalised for length) and log-odds form to calculate similarity scores for every word pair. Subsequently, the distance between two dialectal varieties can be obtained by calculating all word pair scores and averaging them.

3.3.3 A parallel implementation of the PairHMM

The original PairHMM software, created by Wesley Mackay (2004) and kindly made available by Grzegorz Kondrak, was not implemented for use on a parallel computer. Because we had two very large datasets, consisting of approximately 20 million and 100 million word pairs for Belgium and the Netherlands respectively, we decided to implement a parallel version of the PairHMM software.[1] A parallel implementation makes it possible to speed up the computation time needed by dividing the work among multiple processors. In our case, the speedup of the training and scoring algorithms by using a parallel implementation was approximately linear (e.g., using 2 processors reduced processing time by 50%). In the following sections the conversion of the training and scoring algorithms is explained in detail.

[1] In addition we also found and corrected some errors in the original source code.

Conversion of the training algorithm

The original version of the training software expected a data file consisting of all word pairs (every word pair on a single line) and thus read the input data in the following way (for one iteration):

```
ARRAY params := read_model_parameters()

ARRAY results := initial_values

FOR every_line_in_the_inputfile DO
  word1 := read_first_word(line)
  word2 := read_second_word(line)
  results := results + train_PHMM(word1, word2, params)
END

generate_new_model_parameters(results)
```

Because of the size of our dataset, this meant using an input file of approximately 10 gigabytes. Obviously, this size imposed a large bottleneck to the system. Therefore, we adapted the program to generate all word pairs on the fly from the available dialect files (containing all dialectal variants for a single meaning):

```
ARRAY params := read_model_parameters()

ARRAY results := initial_values

FOR file := 1 TO all_dialect_files DO
  FOR line1 := 1 TO last_line DO
    FOR line2 := (line1 + 1)  TO last_line DO
      word1 := get_word_from_line(line1)
      word2 := get_word_from_line(line2)
      results := results +
                 train_PHMM(word1, word2, params)
    END
  END
END

generate_new_model_parameters(results)
```

This implementation reduced the memory requirements by a factor of 3500 to about 3 megabytes. To prevent order effects in training, every word pair was considered twice (e.g., $w_a - w_b$ and $w_b - w_a$). Note that for simplicity this is not shown in the code. As can be observed, the final result is obtained by calculating the results of the `train_PHMM` method for every word pair in the dataset and adding them together (corresponding to the sum over all w in (3.18), (3.20) and (3.21)).

Because the input data is separated into 562 separate files instead of a single file, and the parameters remain the same during one iteration, a parallel implementation of the PairHMM training is straightforward (the first `IF` line shows the division of work among all processors by using the modulo operation):

```
ARRAY params := read_model_parameters()

ARRAY results := initial_values

FOR file := 1 TO all_dialect_files DO
  IF (file MOD total_nr_of_processors) == processor_id THEN
    FOR line1 := 1 TO last_line DO
      FOR line2 := (line1 + 1)  TO last_line DO
        word1 := get_word_from_line(line1)
        word2 := get_word_from_line(line2)
        results := results +
                   train_PHMM(word1, word2, params)
       END
     END
  END
END

IF all_procs_finished AND proc_id == output_processor THEN
  add_results_of_all_processors_together()
  generate_new_model_parameters(results)
END
```

Thus, every processor obtains an equal share of the input files to train with, calculates the sum of the `train_PHMM` method for all word pairs in the designated files and sends the results to the processor generating the output. This processor then adds all values together and generates the new parameters of the model, which are used in the next iteration.

Conversion of the scoring algorithms

As well as for the training software, the original version of the scoring software also expected a data file consisting of all word pairs as input, again introducing a large memory bottleneck.

In the scoring phase a similarity score has to be calculated for every dialect pair based on the word pair alignments. Therefore the input was arranged in files consisting of all words for one specific dialectal variety. The line numbers in every file correspond with a single meaning (e.g., line 2 in every file contains a dialectal variant of the word 'ape').

Also in this case parallelisation is straightforward (again, the first `IF` line in the code below shows the division of work among all processors). Every processor can calculate the similarity score for a single dialect pair independently by adding all alignment scores for the word pairs. Because not all words have to be present in the input files, the scores are averaged afterwards (this is not shown in the code below):

```
ARRAY params := read_model_parameters()

ARRAY result = 0

FOR place1 := 1 TO all_place_files DO
  IF (place1 MOD total_nr_of_processors) == processor_id THEN
    FOR place2 := (place1 + 1) TO all_place_files DO
      FOR line := 1 TO last_line DO
        word1 := get_word_from_line_of_file(place1, line)
        word2 := get_word_from_line_of_file(place2, line)
        result[place1,place2] := result[place1,place2] +
                      test_PHMM(word1, word2, params)
      END
    END
  END
END

IF all_proc_finished AND proc_id == output_processor THEN
  gather_all_results_and_output()
END
```

Thus, every processor obtains an equal share of the input files to test. Because every processor calculates a distinct set of dialect distances, no processor will write to an already filled part of the `result` array. Finally note that `test_PHMM` can be one of all available PairHMM scoring algorithms, i.e. the Viterbi or Forward algorithms in their regular or log-odds form.

3.4 Results

To obtain the best model probabilities, we trained the PairHMMs with all data available. For the Netherlands a PairHMM was trained with all data from the 424 Netherlandic localities, while for Belgium a PairHMM was trained with all data from the 189 Belgian localities. For every dialectal variety there were on average 540 words with an average length of 5 tokens. As mentioned earlier, to prevent order effects in training every word pair was considered twice (e.g., $w_a - w_b$ and $w_b - w_a$). Therefore, the total number of word pairs considered in one training iteration was almost 20 million for Belgium and almost 100 million for the Netherlands.

After starting with more than 6700 uniform initial substitution probabilities, 82 insertion and deletion probabilities and 5 transition probabilities, convergence was reached for the data of the Netherlands after nearly 1500 iterations, taking 10 parallel processors each more than 10 hours of computation time. For Belgium, more than 1250 iterations were needed to reach convergence on 2500 substitution probabilities, 50 insertion and deletion probabilities and 5 transition probabilities (taking 10 parallel processors each more than 3 hours of computation time).

In the following paragraphs we will discuss the quality of the trained substitution probabilities as well as comment on the dialectological results obtained with the trained models.

3.4.1 Trained substitution probabilities

We are interested both in how well the overall sequence distances assigned by the trained PairHMMs reveal the dialectological landscape of the Netherlands and Belgium, and also in how well segment distances induced by the Baum-Welch training reflect linguistic reality. A first inspection of the latter is a simple check on how well standard classifications are respected by the segment distances induced.

Intuitively, the probabilities of substituting a vowel with a vowel or a consonant with a consonant (i.e. same-type substitution) should be higher than the probabilities of substituting a vowel with a consonant or vice versa (i.e. different-type substitution). Also the probability of substituting a phonetic symbol with itself (i.e. identity substitution) should be higher than the probability of a substitution with any other phonetic symbol. To test this assumption, we compared the means of the above three substitution groups for vowels, consonants and both types together.

In line with our intuition, we found a higher probability for an identity substitution as opposed to same-type and different-type non-identity substitutions, as well as a higher probability for a same-type substitution as compared to a different-type substitution. For both PairHMMs this result was highly significant in all cases: vowels (all $p's < 0.05$), consonants (all $p's < 0.001$) and both types together (all $p's < 0.001$).

3.4.2 Vowel substitution scores compared to acoustic distances

PairHMMs assign high probabilities (and scores) to the emission of segment pairs that are more likely to be found in training data. Thus we expect frequent dialect correspondences to acquire high scores. Since phonetic similarity effects alignment and segment correspondences, we hypothesise that phonetically similar segment correspondences will be more common than phonetically remote ones, more specifically that there should be a negative correlation between PairHMM-induced segment substitution probabilities presented above and phonetic distances.

We focus on segment distances among vowels, because it is straightforward to suggest a measure of distance for these (but not for consonants). Phoneticians and dialectologists use the two first formants (the resonant frequencies created by different forms of the vocal cavity during pronunciation) as the defining physical characteristics of vowel quality. The first two formants correspond to the articulatory vowel features height and advancement. We follow variationist practice in ignoring third and higher formants. Using formant frequencies we can calculate the acoustic distances between vowels.

Because the occurrence frequency of the phonetic symbols influences substitution probability, we do not compare substitution probabilities directly to acoustic distances. To obtain comparable scores, the substitution probabilities are divided by the product of the relative frequencies of the two phonetic symbols used in the substitution. Because substitutions involving similar infrequent segments now get a much higher score than

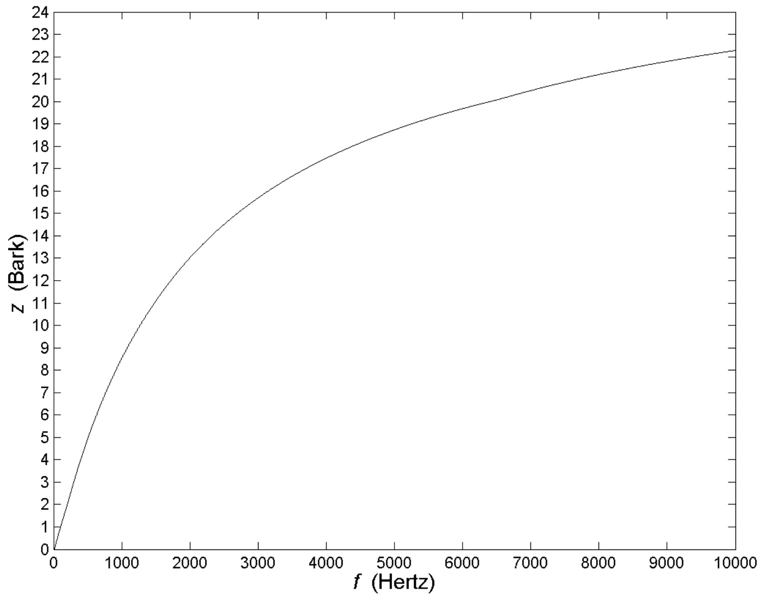

Figure 3.3. Conversion of frequency to Bark scale (Traunmüller, 1990)

.

substitutions involving similar, but frequent segments, the logarithm of the score is used to bring the respective scores into a comparable scale.

In the program PRAAT we find Hertz values of the first three formants for Dutch vowels pronounced by 50 male (Pols et al., 1973) and 25 female (Van Nierop et al., 1973) speakers of standard Dutch. The vowels were pronounced in a /hVt/ context, and the quality of the phonemes for which we have formant information should be close to the vowel quality used in the GTRP transcriptions. By averaging over 75 speakers we reduce the effect of personal variation. For comparison we chose only vowels that are pronounced as monophthongs in standard Dutch, in order to exclude interference of changing diphthong vowel quality with the results. Nine vowels were used: /i, ɪ, y, ʏ, ɛ, a, ɑ, ɔ, u/.

We calculated the acoustic distances between all vowel pairs as a Euclidean distance of the formant values. Since our perception of frequency is non-linear, using Hertz values of the formants when calculating the Euclidean distances would not weigh $F1$ enough. We therefore transform frequencies to Bark scale (see Figure 3.3) in better keeping with human perception. We used the formula of Traunmüller (1990) shown in (3.23) for this transformation, since it is most suitable for speech analysis (see Carter, 2002: Appendix 1).

$$z = \frac{26.81f}{1960 + f} - 0.53. \tag{3.23}$$

Low frequency correction: *if* $z < 2, z' = z + 0.15(2 - z)$.
High frequency correction: *if* $z > 20.1, z' = z + 0.22(z - 20.1)$.

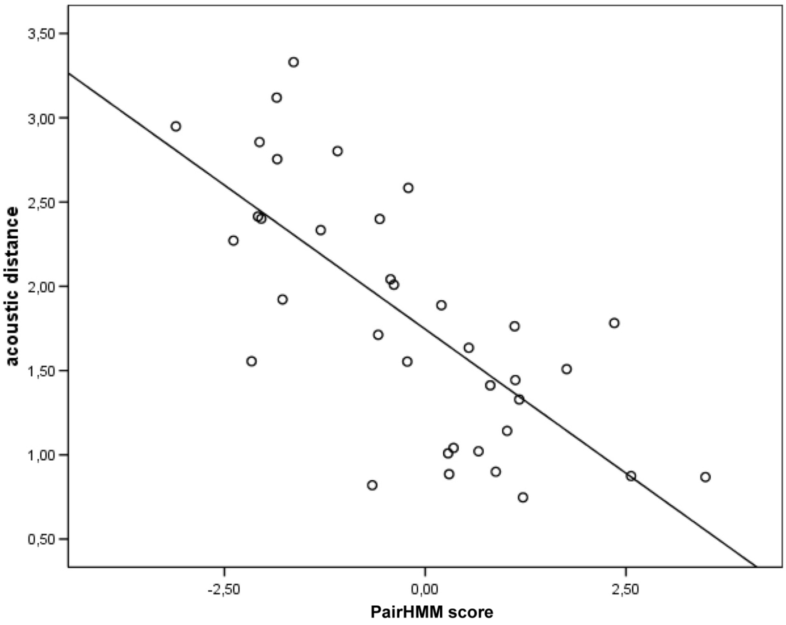

Figure 3.4. Predicting acoustic distances based on PairHMM scores of the Netherlands. Acoustic vowel distances are calculated via Euclidean distance based on the first two formants measured in Hertz, normalised for speaker. $r = -0.72$

For the Netherlands, the correlation between the acoustic vowel distances based on two formants in Bark and the logarithmical and frequency corrected PairHMM substitution scores is $r = -0.65$ ($p < 0.01$). But Lobanov (1971) and Adank (2003) suggested using standardised z-scores, where the normalisation is applied over the entire vowel set produced by a given speaker (one normalisation per speaker). This helps in smoothing the voice differences between men and women. Normalising frequencies in this way resulted in a correlation of $r = -0.72$ ($p < 0.001$) with the PairHMM substitution scores of the Netherlands. Figure 3.4 visualises this result. Both Bark scale and z-values gave somewhat lower correlations when the third formant was included in the measures.

The Belgian data did not include the vowels /ɪ/ and /ʏ/, but also in this case we found (slightly weaker) significant correlations between the PairHMM substitution scores and the acoustic vowel distances based on two formants (Bark scale: $r = -0.61, p < 0.01$; standardised z-scores: $r = -0.59, p < 0.01$).

The strong correlations demonstrate that the PairHMM scores reflect phonetic (dis)similarity. The higher the probability that vowels are aligned in PairHMM training, the smaller the acoustic distance between two segments. We conclude therefore that the PairHMMs indeed align linguistically corresponding segments in accord with phonetic similarity. This likewise confirms that dialect differences tend to be acoustically slight rather than large, and suggests that PairHMMs are attuned to the slight differences which accumulate locally during language change. Also we can be more optimistic about combining segment distances and sequence distance techniques, in spite of Heeringa(2004; Chapter 4) who combined formant track segment distances with Levenshtein distances without obtaining improved results.

3.4.3 Dialectological results

To see how well the PairHMM results reveal the dialectological landscape of the Netherlands and Belgium, we calculated the dialect distances for both countries separately with the Viterbi and Forward algorithms (in both their regular and log-odds versions) using the trained model parameters.

To assess the quality of the dialectal results, we use the LOCAL INCOHERENCE measurement which measures the degree to which geographically close varieties also represent linguistically similar varieties (Nerbonne and Kleiweg, 2007).

The LOCAL INCOHERENCE measurement is formally defined as:

$$\mathrm{I}_l = \frac{1}{n} \sum_{i=1}^{n} \frac{D_i^L - D_i^G}{D_i^G}. \tag{3.24}$$

Where

$$D_i^L = \sum_{j=1}^{k} d_{i,j}^L \cdot 2^{-0.5j} \tag{3.25}$$

$$D_i^G = \sum_{j=1}^{k} d_{i,j}^G \cdot 2^{-0.5j} \tag{3.26}$$

$d_{i,j}^L, d_{i,j}^G$: geographical distance between locations i and j

$d_{i,1\cdots n-1}^L$: geographical distance sorted by increasing *linguistic* difference

$d_{i,1\cdots n-1}^G$: geographical distance sorted by increasing *geographical* distance.

In short, for a given location i all other locations are sorted in two ways: based on their increasing geographic distance and based on their increasing linguistic distance. The value of D_i^G will depend on the geographical distance from location i to the k geographically nearest locations and the value of D_i^L will depend on the geographical distance to the k linguistically nearest locations. In this thesis, we fix k to the value 8, analogously to Nerbonne and Kleiweg (2007). Due to the exponential function in (3.25) and (3.26) more weight is given to (geographically or linguistically) closer locations. Because linguistic distance should preferably increase with geographic distance, the ideal situation would be that $D_i^L = D_i^G$ for all locations i. In this case I_l would equal 0, which is also the lowest value. Higher values of I_l indicate poorer dialectal results. However note that LOCAL INCOHERENCE cannot be used as a "gold standard" for quality in dialectological measurements, but lacking a better evaluation method, rather only as an indicative heuristic.

Because the exact value of I_l (in a non-ideal situation) depends on the specific geographic distances, comparing the value of I_l for different measurements only makes sense for the same set of locations. Therefore, the LOCAL INCOHERENCE measurement cannot be used to compare the effectiveness of a certain linguistic measurement for the Netherlands as opposed to Belgium.

Just as Mackay and Kondrak (2005), we found the overall best performance (lowest LOCAL INCOHERENCE) was obtained using the log-odds version of the Viterbi algorithm, with distinct frequency-based insertion and deletion probabilities for the random model. The Forward log-odds algorithm (with distinct probabilities for the random model) was a good runner-up with only slightly lower performance. For both log-odds algorithms, setting the insertion and deletion probabilities of the random model equal to those of the word similarity model had a negative impact on performance. Finally, applying the regular Forward and Viterbi algorithms to obtain the dialect distances gave the worst performance.

Following Mackay and Kondrak (2005), we also experimented with a modified PairHMM obtained by setting non-substitution parameters constant. Rather than using the transition, insertion and deletion parameters (see Figure 3.1) of the trained model, we set these to a constant value as we are most interested in the effects of the substitution parameters. We indeed found slightly increased performance (in terms of LOCAL INCOHERENCE) for the simplified model with constant transition parameters using the Viterbi log-odds algorithm (with distinct probabilities for the random model). However, since there was a very high correlation (Netherlands: $r = 0.98$; Belgium $r = 0.97$) between the results of the full and the simplified model and the resulting clustering was also highly similar, we will use the Viterbi log-odds algorithm using all trained parameters to represent the results obtained with the PairHMM method.

3.4.4 A comparison between PairHMM and Levenshtein results

In this comparison we use a slightly modified version of the regular Levenshtein distance algorithm, which enforces a linguistic syllabicity constraint: only vowels may match with vowels, and consonants with consonants. The specific details are described in Section 2.3.1.

The PairHMMs yielded dialectological results quite similar to those of Levenshtein distance. The LOCAL INCOHERENCE of the two methods was similar, and the dialect distance matrices obtained from the two techniques correlated highly (Netherlands: $r = 0.89$; Belgium: $r = 0.94$). Given that the Levenshtein distance has been shown to yield results that are consistent (Cronbach's $\alpha = 0.99$) and valid when compared to dialect speakers judgements of similarity ($r \approx 0.7$), this means in particular that the PairHMMs are detecting dialectal variation quite well.

Figure 3.5 shows the dialectal maps for the results obtained using the Levenshtein algorithm (top) and the PairHMM algorithm (bottom) for the Netherlands. The maps on the left show a clustering in ten groups based on UPGMA (Unweighted Pair Group Method with Arithmetic mean; see Heeringa, 2004 for a detailed explanation). In these maps phonetically close dialectal varieties are marked with the same symbol. However note that the symbols can only be compared within a map, not between the two maps (e.g., a dialectal variety indicated by a square in the top map does not need to have a relationship with a dialectal variety indicated by a square in the bottom map). Because clustering is unstable, in that small differences in input data can lead to large differences in the classifications derived, we repeatedly added random small amounts of noise to the data and iteratively generated the cluster borders based on the noisy input data. Only borders which showed up during most of the 100 iterations are shown in the map. The maps in the middle show the most robust cluster borders; darker lines indicate more robust borders. The maps on the right show a vector at each locality pointing in the direction of the region it is phonetically most similar to.

A number of observations can be made on the basis of these maps. The most important observation is that the maps show very similar results. For instance, in both methods a clear distinction can be seen between the Frisian varieties (north) and their surroundings as well as the Limburg varieties (south) and their surroundings. Some differences can also be observed. For instance, at first glance the Frisian cities among the Frisian varieties are separate clusters in the PairHMM method, while this is not the case for the Levenshtein method. Since the Frisian cities differ from their surroundings a great deal, this point favours the PairHMM. However, when looking at the deviating vectors for the Frisian cities in the two vector maps, it is clear that the techniques again yield similar results.

Figure 3.6 shows the dialectal maps for the results obtained using the Levenshtein algorithm (left) and the PairHMM algorithm for Belgium (right). Despite that there are slight differences between the maps (e.g., the border strength in the middle maps), again the main observation is that both methods yield very similar results. A more detailed description of the results using the Levenshtein distance on the GTRP data of both the Netherlands and Belgium can be found in Chapter 2.

Although the PairHMM method is much more sophisticated than the Levenshtein method, it yields very similar results. This may be due to the fact that the datasets are large enough to compensate for the lack of sensitivity in the Levenshtein technique, and the fact that we are evaluating the techniques at a high level of aggregation (average differences in 540-word samples).

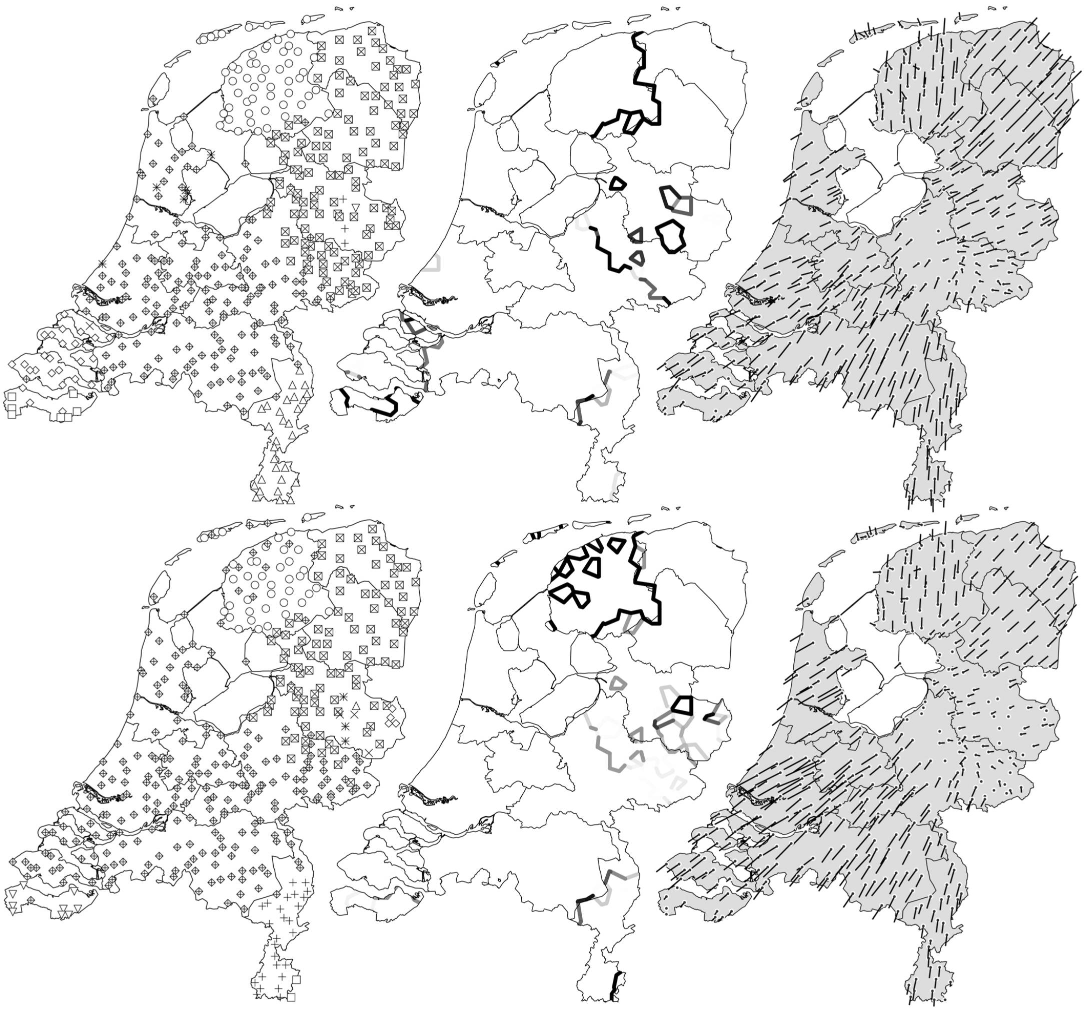

Figure 3.5. Netherlandic dialect distances for Levenshtein method (top) and PairHMM method (bottom). The maps on the left show the ten main clusters for both methods, indicated by distinct symbols. Note that the shape of these symbols can only be compared within a map, not between the top and bottom maps. The maps in the middle show robust cluster borders (darker lines indicate more robust cluster borders) obtained by repeated clustering using random small amounts of noise. The maps on the right show for each locality a vector towards the region which is phonetically most similar. See section 3.4.4 for further explanation.

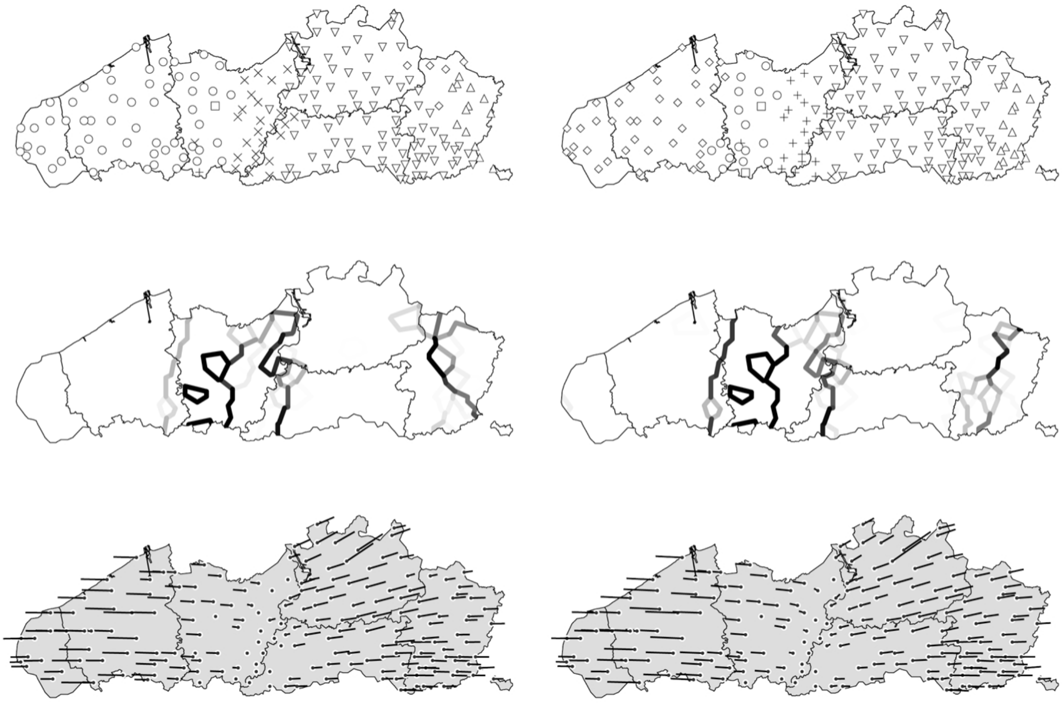

Figure 3.6. Belgian dialect distances for Levenshtein method (left) and PairHMM method (right). The maps at the top show the six main clusters for both methods, indicated by distinct symbols. Note that the shape of these symbols can only be compared within a map, not between the left and right maps. The maps in the middle show robust cluster borders (darker lines indicate more robust cluster borders) obtained by repeated clustering using random small amounts of noise. The maps at the bottom show for each locality a vector towards the region which is phonetically most similar. See section 3.4.4 for further explanation.

3.4.5 Combining PairHMM probabilities and the Levenshtein distance

Besides using the algorithms of the PairHMM explained above, it is also possible to use the trained emission probabilities in assigning costs to the substitution, insertion and deletion operations in the regular Levenshtein algorithm (see Section 2.3.1). First, all probabilities are converted to comparable scores by correcting for their frequency and taking the logarithm of the resulting value (see Section 3.4.2). Subsequently these scores are converted to costs by subtracting the score from 0, mapping a high score (i.e. a likely substitution) to a low cost and vice versa. The new Levenshtein distance can now be calculated by taking into account the distinct costs for each substitution, insertion and deletion. We will refer to this version of the Levenshtein distance as LLW (i.e. Levenshtein distance with learnt weights).

Again the use of a more sophisticated approach did not yield distinct results. The correlation between the dialect distances calculated with the LLW and the other algorithms (PairHMM Viterbi log-odds and regular Levenshtein distance) was $r > 0.95$ for both the Netherlands and Belgium. Furthermore the LOCAL INCOHERENCE of the three methods did not differ significantly. The similarity between the PairHMM and the LLW approach is visualised with vector maps for the Netherlands and Belgium in Figure 3.7 and 3.8 respectively. In a vector map, a vector at each locality is pointing in the direction of the region it is phonetically most similar to. For completeness note that the vector maps obtained using the regular Levenshtein algorithm are shown in Figure 3.5 and 3.6.

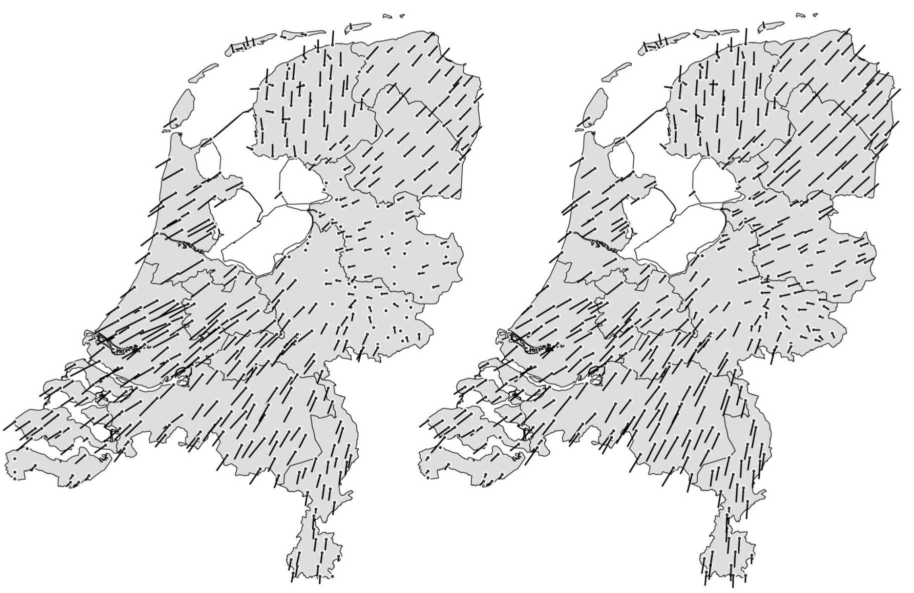

Figure 3.7. Vector map for PairHMM (left) and Levenshtein with learnt weights approach (right). The maps show for each (Netherlandic) locality a vector towards the region which is phonetically most similar.

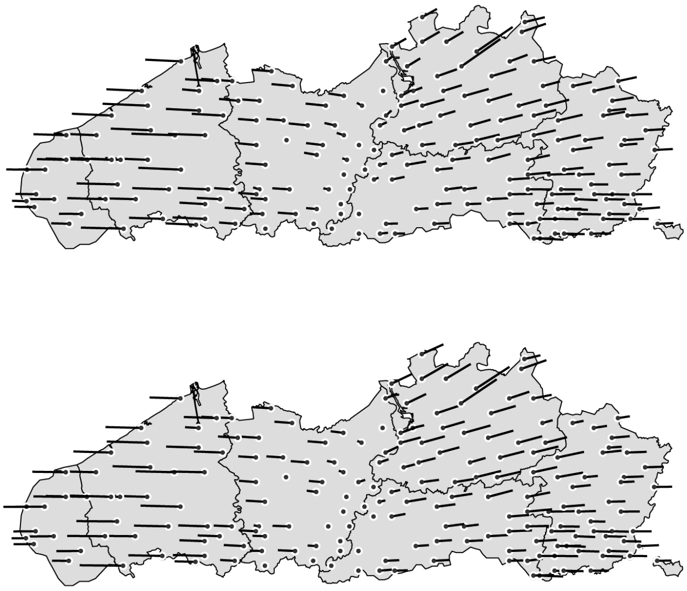

Figure 3.8. Vector map for PairHMM (top) and Levenshtein with learnt weights approach (bottom). The maps show for each (Belgian) locality a vector towards the region which is phonetically most similar.

3.5 Discussion

The presented research confirms Mackay's (2004) work showing that PairHMMs align linguistic material well and that they induce reasonable segment distances at the same time. We have extended that work by applying PairHMMs to dialectal data, and by evaluating the induced segment distances via their correlation with acoustic differences. We noted above that it is not clear whether the dialectological results improve on the simple Levenshtein measures, and that this may be due to the level of aggregation and the large sample sizes. But we would also like to test PairHMMs on a dataset for which more sensitive validation is possible, e.g. the Norwegian set for which dialect speakers judgements of proximity is available (Heeringa et al., 2006); this is clearly a point at which further work would be rewarding.

At a more abstract level, we emphasise that the correlation between acoustic distances on the one hand and the segment distances induced by the PairHMMs on the other confirm both that alignments created by the PairHMMs are linguistically responsible, and also that this linguistic structure influences the range of variation. The segment distances induced by the PairHMMs reflect the frequency with which such segments need to be aligned in Baum-Welch training. It would be conceivable that dialect speakers used all sorts of correspondences to signal their linguistic provenance, but they do not. Instead, they tend to use variants which are linguistically close at the segment level.

Finally, we note that the view of diachronic change as on the one hand the accumulation of changes propagating geographically, and on the other hand as the result of a tendency toward local convergence suggests that we should find linguistically similar varieties nearby rather than further away. The segment correspondences PairHMMs induce correspond to those found closer geographically.

We have assumed a dialectological perspective here, focusing on local variation (Dutch), and using similarity of pronunciation as the organising variationist principle. For the analysis of relations among languages that are further away from each other—temporally and spatially—there is substantial consensus that one needs to go beyond similarity as a basis for postulating grouping. Thus phylogenetic techniques often use a model of relatedness aimed not at similarity-based grouping, but rather at creating a minimal genealogical tree. Nonetheless similarity *is* a satisfying basis of comparison at more local levels.

4 Dialect pronunciation comparison and spoken word recognition

Abstract[*]

Two adaptations of the regular Levenshtein distance algorithm are proposed based on psycholinguistic work on spoken word recognition. The first adaptation is inspired by the Cohort Model which assumes that the word-initial part is more important for word recognition than the word-final part. The second adaptation is based on the notion that stressed syllables contain more information and are more important for word recognition than unstressed syllables. The adapted algorithms are evaluated on the GTRP data of the Netherlands and Belgium and a relatively small Norwegian dataset for which dialect speakers judgements of proximity is available.

4.1 Introduction

The Levenshtein distance algorithm is a popular sequence-based method used to measure the perceptual distances between dialects (Heeringa, 2004). In the Levenshtein algorithm every edit operation is assigned a certain cost (in our case all operations have the same cost, 1; see also 2.3.1). The location of the edit operations is not relevant in determining the cost; a substitution at the first position of both strings has the same cost as a substitution at the final position of both strings. While this is a sensible notion, there are some theories of spoken word recognition which suggest another approach.

Although it is natural to examine psycholinguistic theories of word recognition as a source of ideas about which parts of words might be most important to dialect perception, we should also be aware that word recognition and dialect perception are different. The task of spoken word recognition is to determine which word was said, while the purpose of dialect variation is to signal the speaker's provenance. Thus aspects of the speech signal that support word recognition may not support inferences about the speaker's (geographic) identity. This is related to the semiotic division between the relation of signs to denotations (or meanings) on the one hand and the relation of signs to senders or interpreters on the other (Bühler, 1934). From the point of view of communication (or word recognition), dialect variation only adds noise to a signal. So we shall not pretend to

[*] This text was submitted in a slightly different form to the *Workshop on Computational Phonology*, Borovets, Bulgaria (2007) as: M. Wieling, and J. Nerbonne. Dialect Pronunciation Comparison and Spoken Word Recognition.

criticise theories of word recognition, even in case it turns out that they contribute little to dialect perception. But it is equally plausible that the mechanisms that make some parts of the speech signal more important for recognition and perception would also be important dialectologically.

The Cohort Model was the first very influential theory on spoken word recognition. The Cohort theory (Marslen-Wilson and Welsh, 1978; Marslen-Wilson, 1987) proposes that word recognition occurs by activating words in memory based on the sequential (left-to-right) processing of the input sound. The first phoneme of a word activates all words which start with that sound, the *word-initial cohort*. Additional phonemes narrow the cohort by ruling out members which do not match the heard sound sequence. For instance after hearing the first phoneme of the word 'elephant', the words 'elephant', 'elevator' and 'enemy' are activated. After hearing the second phoneme the cohort is reduced to the words 'elephant' and 'elevator'. Subsequent phonemes will reduce the number of items in the cohort until only the word 'elephant' remains and is recognised. Hence, the start of the word is more important than later parts of the word (Marslen-Wilson and Zwitserlood, 1989). Even though the Cohort Model has a number of drawbacks (e.g., correct recognition of a word is not possible when the start of a word is misheard) and other theories of word recognition have been proposed which do not rely on left-to-right activation (see Jusczyk and Luce, 2002 and Luce and McLennan, 2005), the start of a word is nevertheless important in word recognition (Walley, 1988).

There is also evidence for the importance of stressed syllables in word recognition. First, stressed syllables are more easily identified out of their original context than unstressed syllables (Cutler et al., 1997). And second, stressed syllables have been found to be more informative than unstressed syllables (Altman and Carter, 1989).

The Levenshtein algorithm can be easily extended to incorporate these theories. Cohort theory can be modelled by weighting differences in the beginning of both strings more heavily than differences at the end. The importance of stressed syllables can be modelled by weighting differences in stressed syllables more strongly than differences in unstressed syllables.

4.2 Material

In this chapter we use the same data as introduced in Section 2.2.1, the GTRP data for both the Netherlands and Belgium (separately as suggested in Section 2.4.1). Furthermore we use a Norwegian dataset for which dialect speakers' judgments of proximity are available (Heeringa et al., 2006). The Norwegian dataset consists of 15 places for which 58 different words of the fable 'The North Wind and the Sun' were phonetically transcribed. The perceptual distances were obtained by similarity judgements of groups of high school pupils from all 15 places; the pupils judged all dialects on a scale from 1 (most similar to native dialect) to 10 (least similar to native dialect). Note that these perceptual distances are not necessarily symmetrical; an inhabitant from region A may rate dialect B more different than an inhabitant from region B rates dialect A. In all datasets stress was predominantly placed on the first syllable.

4.3 Adapted Levenshtein distance algorithms

It is straightforward to adapt the Levenshtein algorithm shown in Section 2.3.1 to allow for custom weighting based on the positions `i` and `j` in both strings. The more general algorithm is shown below. Note that the regular Levenshtein distance can be calculated by setting `COST_FUNCTION(i,j)` to 1 for every `i` and `j`.

```
LEVEN_TABLE(0,0) = 0

FOR i := 1 TO LENGTH(string1)
   LEVEN_TABLE(i,0) := LEVENTABLE(i-1, 0) + COST_FUNCTION(i,0)
END

FOR j := 0 TO LENGTH(string2)
   LEVEN_TABLE(0,j) := LEVENTABLE(0, j-1) + COST_FUNCTION(0,j)
END

FOR i := 1 TO LENGTH(string1) DO
   FOR j := 1 TO LENGTH(string2) DO
      LEVEN_TABLE(i,j) :=
         MIN(
              LEVEN_TABLE(i-1, j) + INS_COST * COST_FUNCTION(i,j),
              LEVEN_TABLE(i, j-1) + DEL_COST * COST_FUNCTION(i,j),
              IF finalchar1 = finalchar2 THEN
                 LEVEN_TABLE(i-1, j-1) // no cost
              ELSE
                 LEVEN_TABLE(i-1, j-1) + SUBST_COST * COST_FUNCTION(i,j)
              END
            )
   END
END

LEVEN_DIST := LEVEN_TABLE( LENGTH(string1), LENGTH(string2) )
```

As explained in Section 2.3.1, we use a slightly adapted version of the Levenshtein algorithm displayed above. The modified Levenshtein algorithm enforces a linguistic syllabicity constraint: only vowels may match with vowels, and consonants with consonants.

4.3.1 Cohort inspired Levenshtein algorithms

In the Cohort Model the importance of a word segment is maximal at the onset of a word and decreases from there until the end. This can be modelled by setting the cost of an edit operation highest at the start of both strings, while gradually decreasing the cost when traversing to the end of both strings.

We experimented with several weighting schemes to model the Cohort theory: a linear decay, an exponential decay, a square root decay and a (natural) logarithmic decay of the cost. The respective cost functions are specified in the pseudocode below. Note that the optimal parameters for the exponential and linear decay functions were defined by experimentation.

```
// Exponential decay cost function
COST_FUNCTION(i,j) := POW( 1.1, ( LENGTH(string1) - i +
                                  LENGTH(string2) - j ) )

// Linear decay cost function
COST_FUNCTION(i,j) := 0.2 * ( LENGTH(string1) - i +
                              LENGTH(string2) - j ) + 1

// Square root decay cost function
COST_FUNCTION(i,j) := SQRT( LENGTH(string1) - i +
                            LENGTH(string2) - j + 1 )

// Logarithmic decay cost function
COST_FUNCTION(i,j) := LOG( LENGTH(string1) - i +
                           LENGTH(string2) - j + EXP(1) )
```

Figure 4.1 visualises these cost functions for two strings which have an added length of 10 tokens. For every method the cost of an edit operation is highest at the start (left side of the graph) and lowest at the end of the strings (right side of the graph). The final edit operation in every cost function always has cost 1. The cost of earlier operations depends on the position in both strings. For example, the cost of a different-token substitution of the second character in string A with the first character in string B can be found by looking at value 3 on the x-axis.

In the following we will refer to the Cohort-inspired Levenshtein algorithms as LEVENCOHORT algorithms.

4.3.2 Stress based Levenshtein algorithm

To model the idea that stressed syllables are more important than unstressed syllables, the adapted Levenshtein algorithm ideally should assign a higher cost to edit operations which occur in stressed syllables rather than in unstressed syllables.

Because it was not possible to identify the stressed syllable in every dataset and stress was placed predominantly on the first syllable in all datasets[1], the stress-based method was approximated by assigning a larger cost to edit operations occurring within the first three positions of both words. The resulting cost function is shown in the pseudocode below.

[1] The Norwegian data at http://www.ling.hf.ntnu.no/nos appears to contain no non-initial stress.

```
// Stress based cost function
HIGH_COST := 2
LOW_COST := 1

IF (i <= 3) AND (j <= 3) THEN
   COST_FUNCTION(i,j) := HIGH_COST
ELSE
   COST_FUNCTION(i,j) := LOW_COST
END
```

We will refer to the adapted Levenshtein distance algorithm as the LEVEN-STRESS algorithm.

4.3.3 Length normalisation

It is obvious that pairs of longer strings will on average have a larger Levenshtein distance than pairs of shorter strings. This bias is even greater for the LEVEN-COHORT algorithms because the average costs for edit operations are higher for longer strings than for shorter strings (i.e. an initial edit operation will have a higher cost for longer strings than shorter strings, while the final edit operation always has cost 1).

Because it is likely that dialect perception is word-based (Heeringa, 2004), it makes sense to normalise the Levenshtein distance such that it is length independent. For the regular Levenshtein distance there are several normalisation methods. Heeringa et al. (2006)

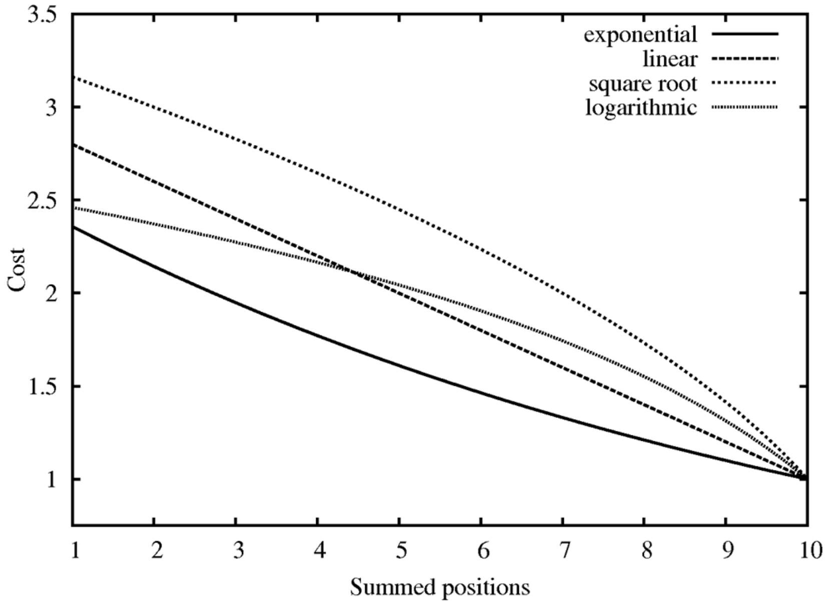

Figure 4.1. Cost functions for the adapted Levenshtein algorithms. For every method the cost of an edit operation is highest at the start (left side of the graph) and lowest at the end of the strings (right side of the graph). The graph shows the position-dependent costs for two strings which have an added length of 10 tokens. For all algorithms the final operation has a cost of 1. The cost of earlier operations depends on the position in both strings.

pointed out that normalisation by alignment length was the most natural procedure because then similarity and difference were each others' inverses (i.e. sum to 1). Other methods include normalising by the length of the longest string, normalising by the length of the shortest string and normalising by the average string length.

Unfortunately these methods are not suitable to normalise the distances obtained with the LEVEN-COHORT (or LEVEN-STRESS) algorithms. To see this, consider two different strings of length 1. The regular Levenshtein distance of these strings is exactly 1 (a single substitution). Because in this case both strings have the same length and the only edit-operation involved is the substitution, all normalisation methods mentioned above yield the same normalised value (in this case 1). It is easy to see that for two strings of length 2 which do not have a character in common, the raw Levenshtein distance is 2 (two substitutions), while the normalised Levenshtein distance equals 1. Thus, in both cases the normalised Levenshtein distance is the same. This makes sense because in both situations the two strings are maximally different.

When considering the LEVEN-COHORT algorithm for two different strings of length 1, the raw and normalised adapted Levenshtein distance again equal 1 (because the final edit operation in every LEVEN-COHORT algorithm has cost 1). However when we increase the string length for both strings with one character, the raw distance will increase with a value larger than 1 (see Figure 4.1) and thus normalising by string length will insufficiently counterbalance the positional weighting and yield a higher relative distance for longer strings. An example of this relative distance increase is shown in the table below. The first line of numbers shows the raw distances, while the bottom line shows the normalised distances for the linear LEVEN-COHORT algorithm.

`r`	`r o`	`r o o`	`r o o d`
`g`	`g e`	`g e e`	`g e e l`
1	2.4	4.2	6.4
1	**1.2**	**1.4**	**1.6**

Fortunately it is possible to construct a better normalisation method which can be applied to the LEVEN-STRESS and LEVEN-COHORT algorithms. In the following section we will adapt the method of normalisation by alignment length while also preserving the desired feature that similarity and difference are each others' inverses.

Normalisation by alignment cost

Instead of normalising by alignment length, we normalise by the cost of the alignment. The cost of a specific alignment can be found by assuming that all aligned identical symbols are replaced by (different-symbol) substitutions. The distance of the new alignment (with the same length as the original alignment) is used for normalisation. Note that this approach equals normalising by alignment length when the costs of all edit operation equal 1, because in that situation the alignment length is equal to the cost of the alignment.

To make this approach clear, consider a possible alignment (including the costs) using the regular Levenshtein algorithm for two Dutch dialectal variants of the word 'milk', [mɔəlkə] and [mɛlək]:

m	ɔ	ə	l		k	ə
m	ɛ		l	ə	k	
0	1	1	0	1	0	1

The total Levenshtein distance of these two words is 4. The cost of this alignment can be calculated by replacing all identical symbol pairs with different-symbol substitutions (additional costs are marked in boldface):

m	ɔ	ə	l		k	ə
m̂	ɛ		l̂	ə	k̂	
1	1	1	**1**	1	**1**	1

The cost of this alignment is 7 and so the normalised Levenshtein distance is $\frac{4}{7}$. The total similarity is equal to the additional costs (in boldface) introduced by replacing all identical symbol pairs with different-symbol substitutions, in this case 3. Because the normalised similarity is $\frac{3}{7}$, similarity and difference are each others' inverses. As pointed out earlier, these normalised values are equal to the values which are obtained by normalising by alignment length.

To see that this normalisation approach can also be used when position-dependent costs are used, consider the alignment (and corresponding edit operation costs) for the two dialectal variants of the word 'milk' using the linear LEVEN-COHORT algorithm.

m	ɔ	ə	l		k	ə
m	ɛ		l	ə	k	
0	2.4	2.2	0	1.6	0	1

The total LEVEN-COHORT distance of these two words is 7.2. The cost of this alignment equals 13 and is calculated as follows:

m	ɔ	ə	l		k	ə
m̂	ɛ		l̂	ə	k̂	
2.8	2.4	2.2	**1.8**	1.6	**1.2**	1

In this case the normalised LEVEN-COHORT distance equals $\frac{7.2}{13}$, while the normalised similarity equals $\frac{5.8}{13}$.

The normalisation method introduced above will always yield normalised distance and similarity values in the range $[0, 1]$. Because the cost of an alignment is equal to the sum of the similarity and the distance of that alignment, the normalised values will always sum to 1 and thus are each others' inverses when normalised. For instance, two identical aligned strings of any length will have a normalised similarity of 1 (and distance of 0), while two completely different strings of any length will have a normalised distance of 1 (and similarity of 0).

In the previous examples, the cost of a substitution did not depend on the symbols involved. For instance, substituting an /a/ with an /e/ did not differ from substituting an /a/ with an /o/ (if the positions were the same). However, when substitution costs vary, like in Chapter 3, it is not immediately clear which substitution cost should be used to calculate the similarity of two identical symbols (indicated in boldface in the examples above). In that case we suggest using the highest substitution cost involving that symbol.

Heeringa et al. (2006) reported that the results using the raw Levenshtein distances were a better approximation of dialect differences as perceived by dialect speakers than results based on normalised Levenshtein distances. Because our normalisation method is comparable to the method Heeringa et al. (2006) used, we will examine if this is also the case in this study.

4.4 Results

First we assessed the reliability of the distance measurements using Cronbach's α (see also Section 2.3.4). For the Norwegian distance measurements Cronbach's α ranged between 0.86 and 0.87, while it was equal to 0.99 for the Belgian and Netherlandic distance measurements. Because these values are much higher than the accepted threshold in social science (where $\alpha > 0.70$ is regarded as acceptable) we conclude that our distance measurements are highly consistent.

To evaluate the results, we used the LOCAL INCOHERENCE measurement described in Section 3.4.3. Although the LOCAL INCOHERENCE cannot be used as a "gold standard", it can be used as an indicative heuristic for quality in dialectological measurements. Additionally, the quality of the Norwegian distances was assessed by correlating them with the perceptual distances.

We calculated the LOCAL INCOHERENCE values of the dialect distances obtained using the LEVEN-STRESS and LEVEN-COHORT algorithms on the Norwegian data and the Netherlandic and Belgian GTRP data. Table 4.1 shows these values for both the normalised as well as the unnormalised data. In calculating the LOCAL INCOHERENCE values, the geographic distances for the Netherlands and Belgium were measured "as the crow flies", while we used travelling time for Norway due to its rugged landscape (Gooskens, 2005). Because the LOCAL INCOHERENCE values are based on geographical distance (or travel time), the values in Table 4.1 can only be compared within a single column (i.e. dataset), but not between the three separate columns.

For the Belgian and Netherlandic dialectal data we can observe slightly improved results (lower LOCAL INCOHERENCE) using the adapted algorithms as compared to the regular

Levenshtein algorithm. The LEVEN-STRESS algorithm yields the best performance, while the exponential LEVEN-COHORT algorithm performs worst. In contrast, for the Norwegian data the best performance is obtained using the regular Levenshtein algorithm.

When inspecting the correlations of the Norwegian dialect distances with the perceptual distances in Table 4.2 we observe a similar pattern. However, Heeringa et al. (2006) mentioned that, since dialect distances satisfy the triangle inequality (i.e. $\forall x, y, z : d(x, y) \leq d(x, z) + d(z, y)$), the involved dialect distances cannot be seen as independent observations. We analyse the relationship between the computed dialect distances and the perceptual distances by calculating the correlation coefficient, but its statistical significance cannot be assayed in the usual way, e.g., via a table in a statistics test or via a software package such as SPSS or R. To solve this problem the Mantel test (Bonnet and Van de Peer, 2002) can be used, which determines the significance of the correlation by repeatedly permuting the matrix rows and columns and recalculating the correlation coefficient. By using this method, Heeringa et al. (2006) found that the correlation coefficients needed to differ by more than 0.1 to indicate statistical significance. Hence, the different algorithms all yield similar performance on the Norwegian dataset.

As mentioned earlier, Heeringa et al. (2006) indicated that normalising the Norwegian dialect distances reduced performance. However as can be seen in Table 4.1 this is not the case for the Netherlandic and Belgian distances. Normalising the Netherlandic distances improves results slightly, while normalising the Belgian distances improves results more clearly. Furthermore, Table 4.2 also shows no reduced performance for the normalised Norwegian dialect distances when correlating them with the perceptual distances as compared to the unnormalised distances. When examining the Norwegian data more closely, we note that the average word length is only 3.5 tokens. This means that our position-sensitive weightings have relatively little opportunity to distinguish themselves. It is therefore not surprising that the LEVEN-COHORT and LEVEN-STRESS approaches perform roughly the same as the regular Levenshtein algorithm.

	NL	BEL	NOR
exponential	1.93 (1.92)	0.76 (0.73)	0.44 (0.48)
linear	1.92 (1.91)	0.76 (0.73)	0.44 (0.49)
square root	1.91 (1.90)	0.76 (0.73)	0.43 (0.50)
logarithmic	1.91 (1.90)	0.76 (0.73)	0.43 (0.49)
stress	1.89 (1.89)	0.75 (0.75)	0.45 (0.49)
regular	1.94 (1.94)	0.80 (0.79)	0.37 (0.45)

Table 4.1. LOCAL INCOHERENCE values of the calculated distances for the Norwegian data (NOR) and the Netherlandic (NL) and Belgian (BEL) GTRP data using the algorithms described in Section 4.3. The LOCAL INCOHERENCE of the regular Levenshtein algorithm is given in the bottom row. The values between parentheses are based on the normalised distances, while the other values are based on the unnormalised distances. Lower values within a column indicate better results.

	Correlation r
exponential	0.63 (0.64)
linear	0.63 (0.64)
square root	0.63 (0.64)
logarithmic	0.64 (0.64)
stress	0.66 (0.64)
regular	0.66 (0.66)

Table 4.2. Correlations of the calculated distances using the using the algorithms described in Section 4.3 with the Norwegian perceptual data. The correlation with the dialect distances calculated using the regular Levenshtein algorithm is given in the bottom row. The values between parentheses are based on the normalised distances, while the other values are based on the unnormalised distances.

Even though the algorithms introduced in Section 4.3 calculate dialect distances using different approaches, the results are very similar. The Norwegian dialect distances calculated with the adapted algorithms correlated highly with the regular Levenshtein distances ($r > 0.97$). This was also the case for Belgium ($r > 0.97$) and the Netherlands ($r > 0.95$). Because of these high correlations, the dialectal maps based on the adapted algorithms resemble the maps obtained using the regular Levenshtein distance as shown in Chapter 2 and 3 a great deal.

To give an example of the high level of similarity between the results of the regular and the adapted Levenshtein distance algorithms, Figure 4.2 shows the dialectal maps for the Netherlandic results obtained using the regular Levenshtein algorithm (top) and the logarithmic LEVEN-COHORT algorithm (bottom). The Belgian maps show comparable similarity and are therefore not displayed. A detailed description of the three map types can be found in Section 3.4.4.

4.5 Discussion

In this chapter we have developed a number of algorithms to calculate dialect distances based on theories of spoken word recognition. Unfortunately these algorithms did not show consistent results across all datasets. While improved results were found on the GTRP datasets using the adapted algorithms, this was not the case for the Norwegian dataset. We emphasise that our results do *not* reflect on the theories of word recognition we employed, as word recognition and the recognition of signals of geographical or social identity may be very different.

There are also some differences between the Norwegian dataset and the GTRP datasets which are worth mentioning. First, the Norwegian dataset is very small (less than 1000 items in total) compared to the size of the GTRP datasets (both consist of more than 100,000 items). Due to the small size and the fact that dialect distances are not statistically independent, it is almost impossible to find significant differences between the

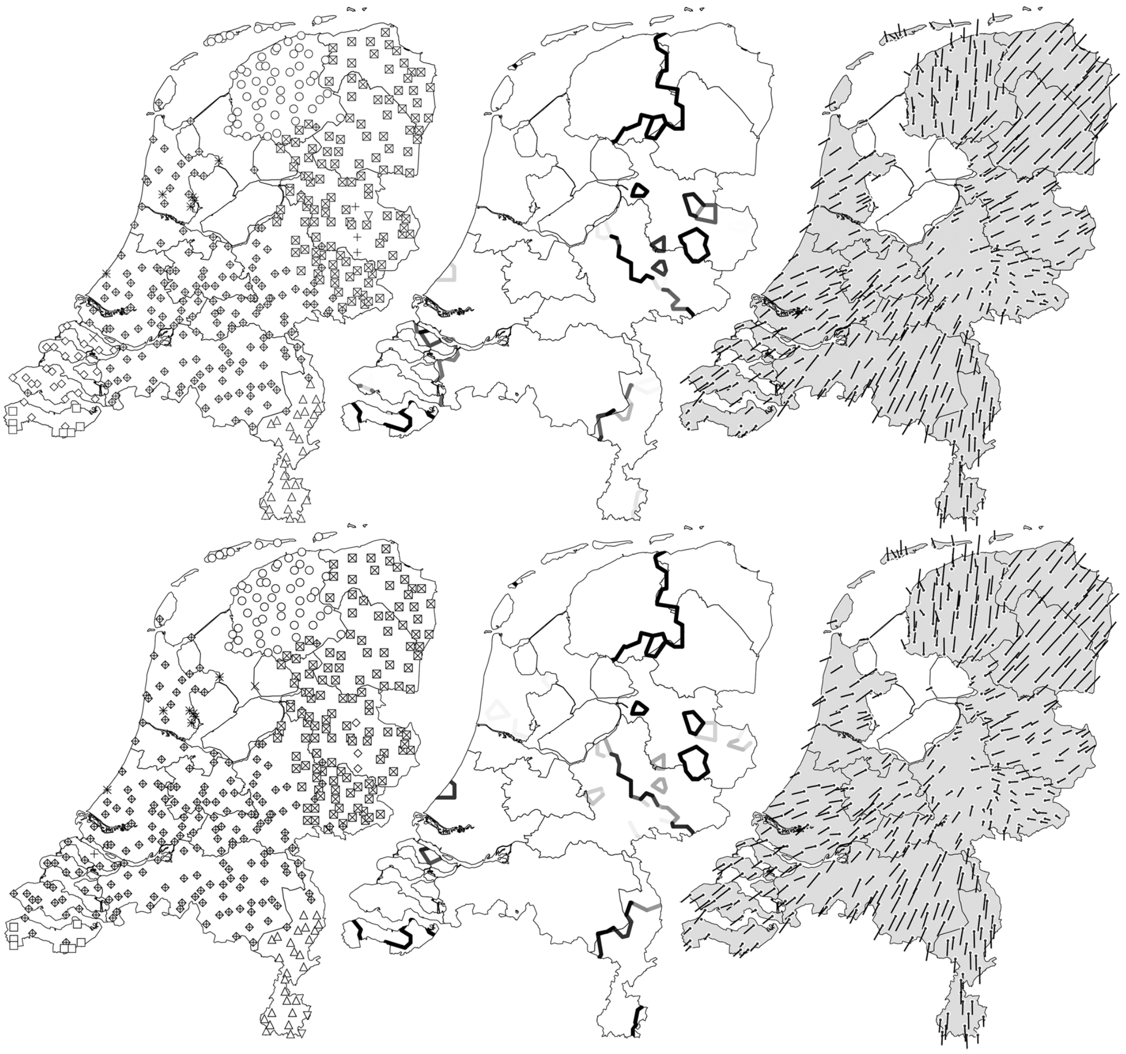

Figure 4.2. Dialect distances for regular Levenshtein method (top) and logarithmic LEVENCOHORT method (bottom). The maps on the left show the ten main clusters for both methods, indicated by distinct symbols. Note that the shape of these symbols can only be compared within a map, not between the top and bottom maps. The maps in the middle show robust cluster borders (darker lines indicate more robust cluster borders) obtained by repeated clustering using random small amounts of noise. The maps on the right show for each locality a vector towards the region which is phonetically most similar. A more detailed explanation about these map types can be found in Section 3.4.4.

results of the different algorithms using the Norwegian perceptual data (Heeringa et al., 2006). Second, there is a large difference between the average word length for the GTRP data and the Norwegian data. The average word length in the GTRP data is 5.5 tokens, while it is only 3.5 tokens for the Norwegian data. Because our algorithms employ a cost function based on position and word length, this likely influences the results. For example, consider the LEVEN-STRESS algorithm which weighs differences in the first three tokens more heavily. Because in the Norwegian dataset the average word consists of only slightly more than three tokens, the LEVEN-STRESS approach will almost be equal to the regular Levenshtein algorithm.

The LEVEN-STRESS algorithm described in Section 4.3.2 uses an approximation of the position and length of the stressed syllable. It would be interesting to evaluate the performance of this algorithm when the exact position and length of the stressed syllable can be used instead. Furthermore, it would be very appealing to compare the performance of the LEVEN-STRESS algorithm to the performance of the LEVEN-COHORT algorithm on a dataset where stress is predominantly placed on the final syllable. In that case the LEVEN-STRESS algorithm weighs differences at the end of the words more strongly, while the LEVEN-COHORT algorithm weighs differences at the start of the words more strongly.

Besides applying position-dependent weighting, another sensible approach could be to weight edit operations based on the type of the sound segments involved. For instance, there is evidence that consonants and vowels are not equally important in word recognition. Several studies found that correcting a non-word into an intelligible word is easier to do when there is a vowel mismatch than a consonant mismatch (Cutler et al., 2000; Marks et al., 2002; Van Ooijen, 1996), e.g. *teeble* → *table* versus *teeble* → *feeble*. It would be interesting to adapt the Levenshtein distance algorithm to incorporate this assumption, for instance by assigning lower costs to vowel-vowel substitutions than for consonant-consonant substitutions.

Together with the adapted Levenshtein algorithms, we also introduced a normalisation method for the new algorithms which enforces that similarity and distance are each others' inverses. In contrast to Heeringa et al. (2006) we do not find support for preferring unnormalised distances over normalised distances. However this does not contradict their results. In our algorithms a stronger bias towards longer words is present than in their study, hence normalisation is more important.

Even though there are differences in performance on the GTRP datasets and the Norwegian dataset, we found that the dialect distances calculated using the adapted algorithms for a single dataset were highly similar to the results obtained with the regular Levenshtein algorithm. A possible cause for this similarity is the aggregate level of analysis; we are looking at the language level instead of the word level. As a better indicator of the performance of the adapted Levenshtein algorithms, it would be very useful to examine the performance on the word level. For instance by evaluating the algorithms on the task of recognising cognates (Kondrak and Sherif, 2006).

5 Conclusions and Future Prospects

In the previous chapters we have provided a thorough aggregate analysis of pronunciation in the most recent Dutch dialect data source, the Goeman-Taeldeman-Van Reenen-Project data (GTRP). Besides comparing the dialectal situation it represents to the *Reeks Nederlands(ch)e Dialectatlassen* (RND; see Chapter 2), we have attempted several (novel) approaches to obtain dialect distances, consisting of using the regular Levenshtein distance (Chapter 2), Pair Hidden Markov Models (Chapter 3) and position-dependent Levenshtein algorithms (Chapter 4).

The first method we applied was the Levenshtein distance, because previous studies (Bolognesi and Heeringa, 2005; Heeringa, 2004; Kessler, 1995; Nerbonne et al., 1996; Nerbonne and Siedle, 2005) reported that it was a simple but effective method in dialect comparison. In accordance with these studies, we obtained sensitive dialectological results using the Levenshtein distance. Unfortunately we also discovered large genuine transcriptional differences between the Netherlandic GTRP data as compared to the Belgian GTRP data.[1] As mentioned earlier, it would be very useful to investigate the existence of potential means to correct these transcriptional differences. For instance by reducing the more detailed Netherlandic transcriptions to the less detailed Belgian transcriptions.

Our second method consisted of using Pair Hidden Markov Models to investigate the usefulness of incorporating automatically obtained segment distances in calculating dialect distances. Although the generated segment distances corresponded to perceptual (vowel) distances, the overall results were highly similar to the results obtained using the Levenshtein distance.

Finally, we attempted to modify the Levenshtein distance such that it allowed for position-dependent weighting. This approach was inspired by psycholinguistic work on spoken word recognition which states that the importance of a sound segment depends on the position within a word. Although sensible results were obtained using this approach, again they strongly resembled the results obtained using the regular Levenshtein distance.

To summarise, all three dialect comparison methods yielded sensible dialectal results, lending support to the validity of the methods. Surprisingly, however, no significant differences could be observed between the separate methods. As mentioned earlier, we believe the sheer size of the GTRP (with over 100,000 items per country) is a likely cause for this similarity. Even though the similarity of individual word pairs will be highly

[1] In tandem, we identified about 150 transcription errors in the data which were corrected by the Meertens Instituut in a revised version.

dependent on the specific algorithm, the aggregate result is obtained by averaging over all word pairs and thus differences will be smoothed out to a large extent.

On the dialect level we cannot conclude that the novel dialect comparison methods are an improvement over the regular Levenshtein distance. However, on the word level Mackay and Kondrak (2005) showed that word pair similarity ratings obtained with the Pair Hidden Markov Model were superior to those obtained with the Levenshtein distance. Also for the position-dependent Levenshtein algorithms it would be useful to assess the performance on the word level.

We indicated earlier that there is evidence that consonants and vowels are not equally important in word recognition (Cutler et al., 2000; Marks et al., 2002; Van Ooijen, 1996; and see also Kondrak, 2003). Hence, it would be worth investigating if an adapted Levenshtein algorithm which incorporates this assumption (e.g., by assigning lower costs to vowel-vowel substitutions than for consonant-consonant substitutions) would offer increased performance on either dialect or word level as compared to the regular Levenshtein algorithm.

All dialect comparison algorithms discussed previously operate by assigning a similarity score to individual words and averaging them to obtain the dialect distance. When normalising word pair similarity scores, this approach assumes that all word pair similarity scores contribute equally to the dialect distance.[2] But is it not strange that while we are investigating more sensitive and novel phonologically inspired methods to measure segment distances in determining word similarity, we regard every word as equally important in determining dialect similarity? Hence, an interesting avenue for further research would be to incorporate the relative importance of every word in calculating dialect distances. For instance, by weighting words based on their frequency, like in Nerbonne and Kleiweg (2007) who reported improved lexical comparison results using Goebl's (1984) inverse frequency weighting method which weights infrequent words more heavily than frequent words.

[2] Note that when we do not normalise word pair similarity scores, longer words will on average have a larger influence on dialect distances than shorter words.

Bibliography

Adank, P. (2003). *Vowel Normalization - a perceptual-acoustic study of Dutch vowels*. Ponsen & Looijen, Wageningen.

Altman, G. and Carter, D. (1989). Lexical stress and lexical discriminability: Stressed syllables are more informative, but why? *Computer Speech and Language*, 3:265–275.

Auer, P., Hinskens, F., and Kerswill, P. (2005). *Dialect Change: Convergence and Divergence in European Languages*. Cambridge University Press.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov Chains. *The Annals of Mathematical Statistics*, 41(1):164–171.

Blancquaert, E. and Peé, W., editors (1925 – 1982). *Reeks Nederlans(ch)e Dialectatlassen*. De Sikkel, Antwerpen.

Bolognesi, R. and Heeringa, W. (2005). De invloed van dominante talen op het lexicon en de fonologie van Sardische dialecten. *Gramma/TTT: tijdschrift voor taalwetenschap*, 9:45–84.

Bonnet, E. and Van de Peer, Y. (2002). zt: a software tool for simple and partial Mantel tests. *Journal of Statistical Software*, 7(10):1–12.

Bühler, K. (1934). *Sprachtheorie. Die Darstellungsfunktion der Sprache*. Gustav Fischer, Jena.

Carter, P. (2002). *Structured Variation in British English Liquids: The Role of Resonance*. PhD thesis, University of York.

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika*, 16:297–334.

Cutler, A., Dahan, D., and Van Donselaar, W. (1997). Prosody in the comprehension of spoken language: A literature review. *Language and Speech*, 40(2):141–201.

Cutler, A., Sebastian-Galles, N., Soler-Vilageliu, O., and Van Ooijen, B. (2000). Constraints of vowels and consonants on lexical selection: cross-linguistic comparisons. *Memory & Cognition*, 28(5):746–55.

Daan, J. and Blok, D. P. (1969). *Van Randstad tot Landrand; toelichting bij de kaart: Dialecten en Naamkunde, volume XXXVII of Bijdragen en mededelingen der Dialectencommissie van de Koninklijke Nederlandse Akademie van Wetenschappen te Amsterdam*. Noord-Hollandsche Uitgevers Maatschappij, Amsterdam.

De Schutter, G., Van den Berg, B., Goeman, T., and De Jong, T. (2005). *Morfologische Atlas van de Nederlandse Dialecten (MAND) Deel 1*. Amsterdam University Press, Meertens Instituut - KNAW, Koninklijke Academie voor Nederlandse Taal- en Letterkunde, Amsterdam.

De Wulf, C., Goossens, J., and Taeldeman, J. (2005). *Fonologische Atlas van de Nederlandse Dialecten (FAND) Deel IV*. Koninklijke Academie voor Nederlandse Taal- en Letterkunde, Gent.

Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, United Kingdom.

Eddy, S. (2004). What is a hidden Markov model? *Nature Biotechnology*, 22:1315–1316.

Filali, K. and Bilmes, J. (2005). A dynamic Bayesian framework to model context and memory in edit distance learning: an application to pronunciation classification. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 338–345, Michigan. ACL.

Goebl, H. (1984). *Dialectometrische Studien. Anhand italoromanischer, rätoromanischer und galloromaischer Sprachmaterialien aus AIS und ALF, volume 191–193 of Beihefte zur Zeitschrift für romanische Philologie*. Max Niemeyer Verlag, Tübingen.

Goeman, T. (1999). *T-deletie in Nederlandse Dialecten. Kwantitatieve analyse van structurele, ruimtelijke en temporele variatie*. Holland Academic Graphics/Thesus, The Hague.

Goeman, T. and Taeldeman, J. (1996). Fonologie en morfologie van de Nederlandse dialecten. een nieuwe materiaalverzameling en twee nieuwe atlasprojecten. *Taal en Tongval*, 48:38–59.

Gooskens, C. (2005). Traveling time as a predictor of linguistic distance. *Dialectologia et Geolinguistica*, 13:38–62.

Gooskens, C. and Heeringa, W. (2004). Perceptive evaluation of Levenshtein dialect distance measurements using Norwegian dialect data. *Language Variation and Change*, 16:189–207.

Goossens, J., Taeldeman, J., and Verleyen, G. (1998). *Fonologische Atlas van de Nederlandse Dialecten (FAND) Deel I*. Koninklijke Academie voor Nederlandse Taal- en Letterkunde, Gent.

Goossens, J., Taeldeman, J., and Verleyen, G. (2000). *Fonologische Atlas van de Nederlandse Dialecten (FAND) Deel II + III*. Koninklijke Academie voor Nederlandse Taal- en Letterkunde, Gent.

Heeringa, W. (2004). *Measuring Dialect Pronunciation Differences using Levenshtein Distance*. PhD thesis, Rijksuniversiteit Groningen.

Heeringa, W. (2001). De selectie en digitalisatie van dialecten en woorden uit de Reeks Nederlandse Dialectatlassen. *TABU, Bulletin voor Taalwetenschap*, 31:61–103.

Heeringa, W. and Braun, A. (2003). The use of the Almeida-Braun System in the measurement of Dutch dialect distances. *Computers and the Humanities*, 37(3):257–271.

Heeringa, W., Kleiweg, P., Gooskens, C., and Nerbonne, J. (2006). Evaluation of string distance algorithms for dialectology. In Nerbonne, J. and Hinrichs, E., editors, *Linguistic Distances*, pages 51–62, Shroudsburg, PA. ACL.

Hinskens, F. and Van Oostendorp, M. (2006). De palatalisering en velarisering van coronale nasaal-plosief clusters in GTR. Talige, dialectgeografische en onderzoekerseffecten. *Taal en Tongval*, 58:103–122.

Jurafsky, D. and Martin, J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, New Jersey.

Jusczyk, P. W. and Luce, P. A. (2002). Speech perception and spoken word recognition: Past and present. *Ear and Hearing*, 23(1):2–40.

Kerswill, P. (2006). Migration and language. In Mattheier, K., Ammon, U., and Trudgill, P., editors, *Sociolinguistics/Soziolinguistik. An international handbook of the science of language and society, 2nd edition*, volume 3, Berlin. De Gruyter.

Kessler, B. (1995). Computational dialectology in Irish Gaelic. In *Proceedings of the Seventh Conference of the European Chapter of the Association for Computational Linguistics*, pages 60–66, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Kondrak, G. (2003). Phonetic Alignment and Similarity. *Computers and the Humanities*, 37(3):273–291.

Kondrak, G. and Sherif, T. (2006). Evaluation of several phonetic similarity algorithms on the task of cognate identification. *Proceedings of the Workshop on Linguistic Distances*, pages 43–50.

Labov, W. (2001). *Principles of Linguistic Change. Vol.2: Social Factors*. Blackwell, Malden, Mass.

Lobanov, B. M. (1971). Classification of Russian vowels spoken by different speakers. *Journal of the Acoustical Society of America*, 49:606–608.

Luce, P. A. and McLennan, C. T. (2005). Spoken word recognition: The challenge of variation. In Pisoni, D. and Remez, R., editors, *The Handbook of Speech Perception*, pages 591–609, Oxford. Blackwell Publishing.

Mackay, W. (2004). Word Similarity using Pair Hidden Markov Models. Master's thesis, University of Alberta.

Mackay, W. and Kondrak, G. (2005). Computing word similarity and identifying cognates with Pair Hidden Markov Models. In *Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL)*, pages 40–47, Morristown, NJ, USA. Association for Computational Linguistics.

Marks, E. A., Moates, D. ., Bond, Z. S., and Stockmal, V. (2002). Word reconstruction and consonant features in English and Spanish. *Linguistics*, 40(2):421–438.

Marslen-Wilson, W. D. (1987). Functional parallelism in spoken word-recognition. *Cognition*, 25(1-2):71–102.

Marslen-Wilson, W. D. and Welsh, A. (1978). Processing interactions and lexical access during word recognition in continuous speech. *Cognitive Psychology*, 10:29–63.

Marslen-Wilson, W. D. and Zwitserlood, P. (1989). Accessing spoken words: The importance of word onsets. *Journal of Experimental Psychology. Human Perception and Performance*, 15(3):576–585.

Nerbonne, J., Heeringa, W., Van den Hout, E., Van der Kooi, P., Otten, S., and Van de Vis, W. (1996). Phonetic distance between Dutch dialects. In Durieux, G., Daelemans, W., and Gillis, S., editors, *CLIN VI: Proc. from the Sixth CLIN Meeting*, pages 185–202. Center for Dutch Language and Speech, University of Antwerpen (UIA), Antwerpen.

Nerbonne, J. and Kleiweg, P. (2007). Toward a dialectological yardstick. *Accepted for publication in Journal of Quantitative Linguistics*.

Nerbonne, J. and Siedle, C. (2005). Dialektklassifikation auf der Grundlage aggregierter Ausspracheunterschiede. *Zeitschrift für Dialektologie und Linguistik*, 72:129–147.

Nunnally, J. C. (1978). *Psychometric Theory*. McGraw-Hill, New York.

Pols, L. C. W., Tromp, H. R. C., and Plomp, R. (1973). Frequency analysis of Dutch vowels from 50 male speakers. *The Journal of the Acoustical Society of America*, 43:1093–1101.

Rabiner, L. R. (1989). A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Rabiner, L. R. and Juang, B. (1986). An introduction to Hidden Markov Models. *ASSP Magazine, IEEE*, 3(1):4–16.

Seguy, J. (1973). La dialectometrie dans l'Atlas linguistique de la Gascogne. *Revue de linguistique romane*, 37:1–24.

Tabachnik, B. and Fidell, L. (2001). *Using Multivariate Statistics*. Allyn & Bacon: 4th edition, Boston.

Taeldeman, J. and Verleyen, G. (1999). De FAND: een kind van zijn tijd. *Taal en Tongval*, 51:217–240.

Traunmüller, H. (1990). Analytical expressions for the tonotopic sensory scale. *The Journal of the Acoustical Society of America*, 88:97–100.

Van den Berg, B. L. (2003). *Phonology & Morphology of Dutch & Frisian Dialects in 1.1 million transcriptions*. Goeman-Taeldeman-Van Reenen project 1980-1995, Meertens Instituut Electronic Publications in Linguistics 3. Meertens Instituut (CD-ROM), Amsterdam.

Van Nierop, D. J. P. J., Pols, L. C. W., and Plomp, R. (1973). Frequency analysis of Dutch vowels from 25 female speakers. *Acoustica*, 29:110–118.

Van Ooijen, B. (1996). Vowel mutability and lexical selection in English: evidence from a word reconstruction task. *Memory & Cognition*, 24(5):573–83.

Van Oostendorp, M. (2007). Kenmerkeconomie in de gtrp-database. *To appear in Taal en Tongval*.

Walley, A. (1988). Spoken word recognition by young children and adults. *Cognitive Development*, 3:137–165.

Wieling, M., Heeringa, W., and Nerbonne, J. (2007a). An aggregate analysis of pronunciation in the Goeman-Taeldeman-Van Reenen-Project data. *Taal en Tongval*, 59(1).

Wieling, M., Leinonen, T., and Nerbonne, J. (2007b). Inducing sound segment differences using Pair Hidden Markov Models. In Nerbonne, J., Ellison, T. M., and Kondrak, G., editors, *Computing and Historical Phonology: 9th ACL Special Interest Group for Morphology and Phonology*, pages 48–56.

Wieling, M. and Nerbonne, J. (2007). Dialect pronunciation comparison and spoken word recognition. Submitted to the Workshop on Computational Phonology, Borovets, Bulgaria, 6/2007.

Wolfram, W. and Schilling-Estes, N. (2003). Dialectology and linguistic diffusion. In Joseph, B. D. and Janda, R. D., editors, *The Handbook of Historical Linguistics*, pages 713–735. Blackwell, Malden, Massachusetts.

List of Figures

List of Tables