Faculteit Ingenieurswetenschappen
Vakgroep Elektronica en Informatiesystemen
Voorzitter: Prof. Dr. Ir. J. Van Campenhout

Gelaagde spraakherkenning met syllaben als intermediaire eenheden

door

Joachim De Zutter

Promotor: prof. dr. ir. Jean. Martens
Begeleiders: ir. Bert Réveil, ir. Catherine Middag

Masterproef ingediend tot het behalen van de academische graad van
Master in de ingenieurswetenschappen: computerwetenschappen

Academiejaar 2009–2010

Dankwoord

Ik zou graag mijn thesisbegeleider Bert willen bedanken voor alle tijd en energie die hij in mij en in dit werk heeft gestopt. Het maken van een gelaagde spraakherkenner bevat heel wat hindernissen die zonder de nodige ervaring in het domein en de werking van de gespecialiseerde software moeilijk oplosbaar zijn. Ik wens ook promotor Jean-Pierre Martens te bedanken voor zijn vertrouwen en de geboden kans om met de getalenteerde mensen van het spraaklabo te mogen samenwerken.

Ik zou ook Kris Demuynck van de K.U.Leuven willen bedanken voor de hulp en de waardevolle informatie bij de implementatie van de SPRAAK herkenner.

Ik wil bovenal mijn ouders bedanken. Zij hebben mij door de jaren heen gesteund in mijn keuzes. Zij hebben mij de mogelijkheden geboden om mij als persoon te ontplooien en te verdiepen in mijn eigen interesses. Zij hebben mij het meest vertrouwen gegeven en tegelijk ook de bescherming van een thuis waar ik mij goed kan voelen.

Ik wil ook mijn vriendin Doroteia Ivanova bedanken voor de onvoorwaardelijke liefde en steun die ik van haar krijg.

"So too we are dwarfs astride the shoulders of giants. We master their wisdom and move beyond it. Due to their wisdom we grow wise and are able to say all that we say, but not because we are greater than they." - Isaiah di Trani

Joachim De Zutter, juni 2010

Toelating tot bruikleen

Joachim De Zutter, juni 2010

Gelaagde spraakherkenning met syllaben als intermediaire eenheden

door

Joachim De Zutter

Scriptie ingediend tot het behalen van de academische graad van
Master in de Computerwetenschappen

Academiejaar 2009–2010

Promotor: Prof. Dr. Ir. J.-P. Martens
Scriptiebegeleiders: Ir. B. Réveil, Ir. C. Middag
Faculteit Ingenieurswetenschappen
Universiteit Gent

Vakgroep Informatietechnologie
Voorzitter: Prof. Dr. Ir. J. Van Campenhout

Samenvatting

In dit werk probeer ik de beperkte lexicongrootte van een traditionele spraakherkenner optimaal te benutten. De beperkte lexicongrootte kan een probleem vormen wanneer in de te herkennen spraak een woord voorkomt dat niet behoort tot de interne woordenschat van de herkenner. Door het gebruik van fonetische syllaben als intermediaire basiseenheden probeer ik dit probleem te omzeilen. Voor de implementatie maak ik gebruik van de SPRAAK herkenner van de K.U.Leuven. Door deze spraakherkenner aan te passen is het mogelijk om een lattice van syllaben te herscoren met een woordgebaseerd taalmodel.

Trefwoorden

continue spraakherkenning, lexicale modellering, taalmodellering, fonetische syllaben, Out of Vocabulary woorden

Two-layered speech recognition with phonetic syllables as intermediary units

Joachim De Zutter

Supervisor(s): Jean-Pierre Martens, Bert Réveil

Abstract—**The limited lexicon size of large vocabulary continuous speech recognition (LVCSR) systems often leads to so-called Out Of Vocabulary (OOV) errors when a word is uttered that is not present in the recognition lexicon. In this work I try to circumvent this problem by using a two-layered recognition system. In the first layer a phonetic syllable recognizer is used to decode the speech into sequences of intermediary sub-word syllables. In the second layer a traditional word based recognizer is employed to rescore output lattices from the syllable recognizer and convert the syllable sequences into a word sequence hypothesis. The Word Error Rate (WER) on the N-Best Flemish Broadcast News (BN) development test set for a word based baseline recognizer with an 80K lexicon was compared with the estimated WERs for a syllable based recognizer with a 10K intermediary syllable lexicon. Although my two-layered system was not fully operative in the end, I found evidence that phonetic syllables may well be useful intermediary units to deal with the OOV problems in LVCSR.**

Keywords— **continuous speech recognition, lexical modeling, language modeling, phonetic syllables, Out Of Vocabulary words**

I. Introduction

Due to the time complexity of the decoder algorithm, the lexicon of some continuous speech recognizers is limited in size. Moreover, increasing the lexicon size exponentially increases the amount of text data required for robust language modeling, so a limited lexicon size is sometimes imposed by a lack of language modeling training data. On the other hand, it is also important that the content of the recognition lexicon ensures a sufficient coverage of the test language vocabulary. Otherwise a problem arises when the recognizer is confronted with words that do not belong to the internal lexicon. These words are called Out Of Vocabulary (OOV) words and it is estimated that every OOV word invokes 1.6 to 2.2 transcription errors on average [1]

In literature the use of sub-word recognition units has often been suggested in order to deal with the OOV problem because these sub-word units are considered to make the language modeling process more robust and because they intrinsically reduce the amount of OOV words that can be expected [2], [3]. For the Dutch language decompounding and splitting in morph units have been employed to reduce the amount of OOV words, while it was also shown that the reduction in OOV rate can be turned into an equal reduction of the WER [4], [7].

In this work I propose to work with phonetic syllables as sub-word units because phonetic syllables it assumed they are generative. There's a small set of phonetic syllables as opposed to orthographic sub-word units.

In a series of recognition experiments I compared the performance of a traditional 80K lexicon word based recognizer to that of my two-layered system that uses phonetic syllables as intermediary recognition units. The test were performed on the Flemish part of the Broadcast News (BN) benchmark that was created during the N-Best project [6]. For all experiments I employed the SPRAAK [7] speech recognizer from the K.U. Leuven.

The outline of this paper is as follows. In the next section, I will describe the experimental set-up I created for my experiments. Section III deals with the language and lexical modeling that I performed during my research. In that part of this paper I also compare words with phonetic syllables as basic language units. In section IV the most important recognition results are presented and in the final section I list the major conclusions of this work and I briefly discuss some ideas for future work.

II. Experimental set-up

I needed two things to perform my experiments: a test set and a speech recognizers.

A. The N-Best Benchmark

The aim of the N-Best project [6] was to create a Dutch benchmark test set for Broadcast News and Telephone Speech to evaluate large vocabulary continuous speech recognition systems (LVCSR). Speech recognizers are evaluated by calculating the WER. The benchmark distinguishes between Northern and Southern Dutch (Flemish) since the pronunciation differs substantially. The evaluation of the speech recognition system was performed on the Southern Dutch Broadcast News (BN) and Conversational Telephone Speech (CTS) sub-tasks of the N-Best project. In this work, the development test set for Flemish Broadcast News was used.

B. The SPRAAK recognition toolkit

The main set-up of the SPRAAK speech recognizer is discussed in detail in [7]. The system is capable of doing speaker segmentation and clustering, although this yields only a minor improvement in the WER. The acoustic models used are triphone Hidden Markov Models (HMM) with state emissions modeled by Gaussian Mixture Models (GMM) with globally tied Guassians. The acoustic models were left unmodified.

III. Language and lexical modeling

In order to build our language models a few things were needed: a text corpus with words and phonetic syllables as lexical units and a toolkit to construct language models from these corpora.

A. Text corpus

A normalised and filtered text corpus for statistical language modeling was used. The Mediargus corpus contains newspaper articles written during a time period of 5 years (1999-2004) of 12 Flemish (Southern Dutch) newspapers such as: Het Laatste

Nieuws, De Morgen, De Tijd, Gazet Van Antwerpen, Het Belang Van Limburg, Het Volk, Het Nieuwsblad and De Standaard. Although some experiments were performed on the entire Mediargus corpus, the De Standaard corpus was used for creating the syllabic language models.

The corpora were converted into syllabic lexical units using the g2p converter found in the autonomata g2p toolset [5]. The g2p was run in DUB mode to obtain Flemish transcriptions.

B. The SRI-LM toolkit

The language models were created with the SRI-LM toolkit [8] with the *make-big-lm* command. The lexicon for the word based language model consisted of the 80K most frequent words of the De Standaard corpus. The ARPA trigam backoff language models were generated using Kneser-Ney smoothing formulas [9] in the modified form as suggested by Goodman and Chen [10] (-kndiscount) with interpolation (-interpolate). For the cut off values, 1 was used (-gtXmin). The OOV word is mapped on an unknown word token. Table I shows the number of N-grams in the 80K language model.

lexicon size	1-grams	2-grams	3-grams
80K lexicon	80272	11228771	42791560

TABLE I

SIZE OF THE WORD BASED LANGUAGE MODEL

C. Coverage with words and syllables

Table II gives the amount of syllables required to form the most frequent words in the Mediargus corpus. The 80K most frequent words can be formed with a set of 10K syllables, which corresponds to a word coverage of about 96.99 % in the Mediargus corpus. The coverage rate for words and syllables is given in graph 1. Since more than the 80K most frequent words can be formed with the 10K syllables that are used by them, the coverage rates for syllables are an underestimation.

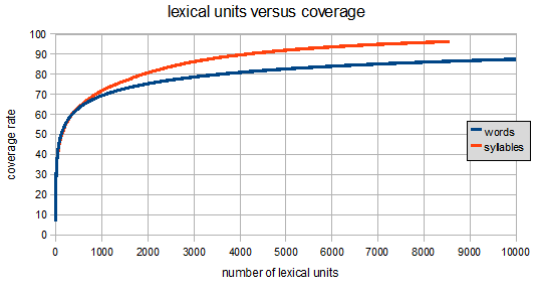


Fig. 1. Coverage for syllables and words

D. LM Perplexities

We selected the 10K syllables of the most frequent words in the De Standaard corpus to create a syllabic language model. An n-gram 10K syllable based language model was searched with a similar perplexity to the 3-gram 80K word based language model, given the same parts of the De Standaard text corpus. We took the first and last paragraphs of the corpus text of De Standaard to attain the results in III. We can conclude that a 4-gram syllable based language model has a similar perplexity to a 3-gram word based language model.

number of words	number of syllables	coverage
1319	1000	71.55%
3893	2000	80.5%
21938	5000	91.75%
40000	6762	94.4%
80000	9356	96.6%
94004	10000	96.99%

TABLE II

WORD COVERAGE OF THE SYLLABLES CORRESPONDING TO THE MOST FREQUENT WORDS

language model	perplexity (head)	perplexity (tail)
3-gram (80K words)	52.57	52.97
3-gram (10K syllables)	293.42	299.28
4-gram (10K syllables)	67.37	68.71
5-gram (10K syllables)	23.49	21.37

TABLE III

LANGUAGE MODEL PERPLEXITIES

IV. EXPERIMENTAL RECOGNITION RESULTS

A. Test set perplexities

In table IV, the test set perplexity of the 3-gram, 4-gram and 5-gram 10K syllabic language model was calculated for the N-Best development test set. We can see the 5-gram language model has a higher test set perplexity than the 4-gram language model due to overtraining. The best predictor for the test set is the 4-gram language model. The syllable based language models all have a higher perplexity than the word based language models.

language model	perplexity
3-gram 80K words	285.87
3-gram 10K syllables	601.08
4-gram 10K syllables	437.18
5-gram 10K syllables	449.89

TABLE IV

TEST SET PERPLEXITIES

B. Word based recognizers

The word based recognizer had an OOV-rate of 2.64 % (275 unrecognized words) and achieved a WER of 22.1 %.

C. Syllable based recognizers

The syllable based recognizers had a much lower OOV-rate (10 unrecognized syllables out of 18955) and a 3-gram and 4-gram syllable based recognizer achieved a syllable error rate of respectively 20.3 % and 21.7 %.

D. Two-layered recognizers

The beam search procedure found in continuous speech recognizers is a breadth-first search algorithm comparing the most likely hypotheses for the transcription of speech in parallel. This makes it possible to generate a word lattice, which is a type of directed graph consisting of nodes indicating the timeframe connected by arcs that describe the probability that the word hypothesis connecting the nodes is the right one. The breadth-first search algorithm has a high time complexity when alot of hypotheses have to be evaluated at once. This depends mostly on the lexicon size chosen when constructing the language model. A technique to solve this problem is called lattice rescoring: in a first phase, a lattice is created with a language model constructed from a small lexicon. This lattice may contain confusing lexical units, which are in a second phase rescored with a language model constructed from a larger lexicon. The functionality to rescore a lattice of phonemes has been introduced into the SPRAAK beam search procedure in April 2010. Although it was perfectly possible to generate a syllable lattice by using syllables as lexical units in a first phase recognizer, setting up the second phase recognizer to do the rescoring of the syllable lattice was not a trivial task. Although I was able to create the finite state automaton and give all the inputs to make the speech recognizer run, no end hypotheses were found after the second phase, due to an unknown cause.

V. Alternative results

A. Number of correctly recognized words as a syllable sequence

The number of fully recognizable words as a syllable sequence as given in the top-1 hypothesis of the syllable based recognizers was tested. The recognizer with a 4-gram language model could recognize 71.0 % of all words correctly. The recognizer with the 3-gram language model performed even better recognizing 77.1 % of all words correctly. The best result attained with a word based recognizer was 81.3 % of all words being correctly recognized. By rescoring the N-best lists of the syllable based recognizer with a word based language model we expect the syllable based recognizer to give much better recognition results than the word based recognizer.

B. Comparison of OOV-zones word and syllable recognizers

In the OOV-zones of the word based recognizer, the syllable based recognizer produces positive results. The word based recognizer would recognize Proven, Vlamertinge and Roesbrugge-Haringe as grove, Flamend Inge and Rous Brugge - haar enge. The best hypothesis proposed by the syllable based recognizer was pro-v@, vla-m@r-tIN-@ and u-rus-brY-G@-ha-rIN-@. These results indicate that the rescoring of the syllable lattice with a bigger language model might lead to a much better recognition accuracy for syllable based recognizers.

VI. Influence of the lexical units on the execution speed

The influence of the choice of sub-word units on the size of the language model and the execution speed of the speech recognizer was calculated and is illustrated in table V. The syllable based recognizer requires a bigger language model and the execution speed is lower.

language model	number of units	size	execution speed
3-gram	80K words	221.1 MB	2:31:46
3-gram	10K syllables	97.5 MB	3:21:34
4-gram	10K syllables	374.6 MB	3:13:04

TABLE V
INFLUENCE ON THE SIZE OF THE LANGUAGE MODEL AND EXECUTION SPEED

VII. Conclusions and future work

We can conclude that although syllable based speech recognizers have a higher test set perplexity than word based recognizers, the best hypothesis found by syllable based recognizers in the OOV-zones of the word based recognizer is often a good approximation. Although a two-layered system where a syllable based recognition phase would be followed by lattice rescoring with a bigger word language model would have a slower execution speed, we expect the system to give better recognition results.

VIII. Acknowledgements

I would like to thank Kris Demuynck of the K.U.Leuven for his help with the software and Bert Réveil for helping to construct the language models.

References

[1] Frederik Stouten, Jacques Duchateau, Jean-Pierre Martens, Patrick Wambacq *Coping with disfluencies in Spontaneous Speech Recognition: acoustic detection and linguistic context manipulation* , Speech Communication, vol. 48: special issue on robust speech recognition, pp. 1590-1606

[2] Jan Kneissler, Dierich Klakow *Speech recognition for huge vocabularies by using optimized sub-word units* , Proc. of EUROSPEECH 2001, Aalborg, Denmark, 2001, pp. 69-73

[3] Martha Larson *Sub-Word-Based Language Models for Speech Recognition: Implications for Spoken Document Retrieval*

[4] Bert Réveil, Jean-Pierre Martens *Reducing speech recognition time and memory use by means of compound (de-)composition* , ProRISC conference, 2008

[5] Qian Yang, Jean-Pierre Martens, Nanneke Konings, Henk van den Heuvel *Development of a phoneme-to-phoneme (p2p) converter to improve the grapheme-to-phoneme (g2p) conversion of names* , LREC Conference, 2006

[6] David Van Leeuwen, Judith Kessens, *Evaluation plan for the North- and South-Dutch Benchmark Evaluation of Speech recognition Technology (N-Best 2008)*

[7] Kris Demuynck, Antti Puurula, Dirk Van Compernolle, Patrick Wambacq *The ESAT 2008 System for N-Best Dutch Speech Recognition Benchmark* , Automatic Speech Recognition & Understanding, 2009. ASRU 2009.

[8] Andreas Stolcke, *Srilm - An extensible language modeling toolkit* Proc. of ICSLP 2002, Denver, vol.2 pp.901-904

[9] Reinhard Kneser, Hermann Ney. *Improved backing-off for m-gram language modeling.* , Proc. of ICASSP-95, vol. 1, 181–184. 1995

[10] Stanley F. Chen and Joshua Goodman *An empirical study of smoothing techniques for language modeling* , Computer Speech and Language, vol. 13, no. 4, pp. 359.393, 1999.

Inhoudsopgave

Gebruikte afkortingen

ARPA	Advanced Research Projects Agency
BN	Broadcast News
CGN	Corpus Gesproken Nederlands
CSR	Continuous Speech Recognition
CTS	Conversational Telephone Speech
DUB	Dutch (België)
G2P	grapheme-to-phoneme
GPS	Global Positioning System
HMM	Hidden Markov Model
IPA	International Phonetic Association (Alphabet)
LVCSR	Large Vocabulary Continuous Speech Recognition
MFCC	Mel Frequency-scale Cepstral Coefficients
OOV	Out of Vocabulary
SPRAAK	Speech Processing, Recognition & Automatic Annotation Kit
WER	Word Error Rate
YAPA	Yet Another Phonetic Alphabet

Hoofdstuk 1

Inleiding

Automatische spraakherkenning vindt steeds vaker zijn ingang in concrete toepassingen. Een aantal voorbeelden hiervan zijn: gepersonaliseerde dicteerapplicaties die mensen in staat stellen om automatisch teksten te genereren, spraakgestuurde navigatiesystemen in wagens (GPS) of geautomatiseerde ondertitelingsprogramma's waarmee TV-uitzendingen van ondertitels kunnen worden voorzien. In dit werk richten we onze aandacht op de laatst genoemde toepassing. Immers, zelfs al gebruikt de openbare omroep VRT een dergelijke applicatie als hulpmiddel om hun programma's te ondertitelen, volledig foutloos gebeurt het proces nog niet. Eén van de hoofdredenen waarom de herkenning soms mis loopt, ligt bij de zogenaamde Out of Vocabulary (OOV) woorden.

Zoals ik in hoofdstuk 2 nader zal toelichten, werkt een klassieke spraakherkenner met een fonetisch woordenboek. Dit is een lijst van lexicale eenheden (doorgaans woorden) en hun verwachte uitspraken (strings van fonemen). De uitkomst van een spraakherkenner is steeds een sequentie van eenheden uit dat lexicon, ook als er in de verwerkte spraak woorden voorkwamen die niet tot dat lexicon behoren. Men noemt dergelijke woorden OOV-woorden. Uit onderzoek blijkt dat de herkenner gemiddeld 1.6 tot 2.2 fouten maakt per OOV-woord dat in de spraak voorkomt [1]. Het is dus belangrijk om de kans op OOV-woorden beperkt te houden zonder het lexicon al te groot te moeten maken. Een groter lexicon vergt namelijk meer geheugencapaciteit en verhoogt doorgaans de rekentijd. Toch

spreekt het voor zich dat men voor de ondertiteling van bijvoorbeeld nieuwsprogramma's een erg groot woordenboek zal nodig hebben als men alle woorden die kunnen opduiken wil opvangen (denk maar aan de vele verschillende eigennamen die kunnen worden genoemd).

De doelstelling van deze thesis is daarom het OOV-probleem te omzeilen, door een spraakherkenner te ontwikkelen met een gelaagde structuur. Deze herkenner zoekt in een eerste fase naar alle schijnbaar valabele sequenties van fonetische lettergrepen (syllaben) die de spraak kunnen verklaren. In een tweede fase worden de gemaakte syllabehypothesen dan opnieuw gescoord op basis van een traditioneel woordtaalmodel en een lexicon waarin de uitspraken van woorden als sequenties van syllaben zijn opgeslagen. De idee achter deze aanpak is tweeledig: enerzijds hopen we dat de set van fonetische lettergrepen in de meeste talen generatief is zodat een heel groot percentage van de bestaande woorden kan worden gevormd als een sequentie van één of meerdere syllaben uit een beperkte set. Als dat zo is, dan kunnen in theorie alle woorden nog correct worden herkend na de eerste fase van het beoogde herkenningsproces. Anderzijds zou een beperkte set fonetische syllaben de eerste herkenningsfase een stuk sneller (want vermoedelijk eenvoudiger) kunnen maken, waardoor we voor de tweede fase voldoende tijd creëren om de syllabesequenties om te zetten naar woordhypothesen. In deze fase zou de herkenner de ruimte kunnen krijgen om "onduidelijke" stukken van een syllabische sequentie als OOV-token te markeren. Deze OOV-tokens worden dan in een ultieme stap nogmaals herscoord, maar met een lexicon dat veel groter is dan datgene dat in fase 2 zal worden gebruikt.

De geconstrueerde herkenner zal uiteindelijk worden aangewend voor de transcriptie van nieuwsuitzendingen, en zijn prestaties zullen worden vergeleken met die van een traditionele herkenner.

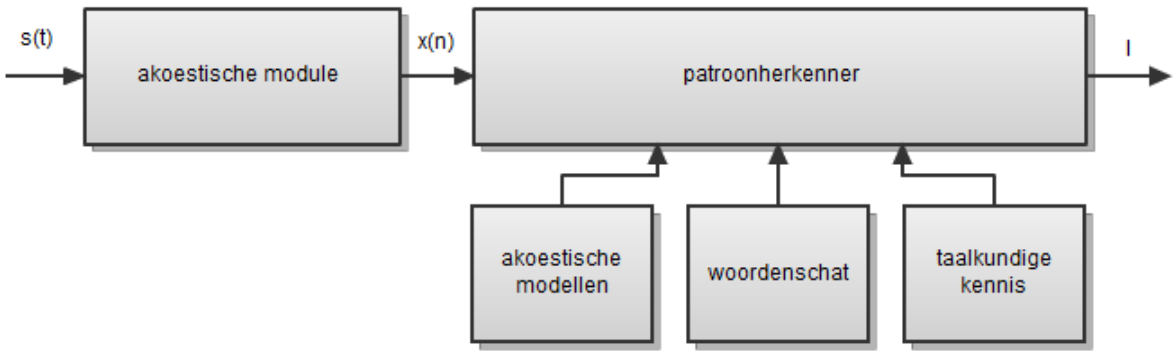
Hoofdstuk 2

De spraakherkenner

In dit hoofdstuk wordt eerst kort de algemene werking van een klassieke continue spraakherkenner behandeld. Daarbij wordt ook dieper ingegaan op de specifieke veranderingen aan de traditionele architectuur, die in deze thesis vereist waren.
Het tweede gedeelte van dit hoofdstuk bespreekt een aantal metrieken die kunnen worden gebruikt om de herkenner of delen van de herkenner te evalueren.

2.1 Algemene architectuur spraakherkenner

Wanneer men het in de literatuur heeft over continue spraakherkenners, dan slaat dat doorgaans op foneemgebaseerde Hidden Markov Model (HMM) systemen. Deze systemen - waarvan de architectuur schematisch is weergegeven in figuur 2.1 - steunen op twee grote basisblokken: een akoestische voorverwerkingsmodule en een centrale patroonherkenner. De patroonherkenner maakt op zijn beurt gebruik van drie interne kennisbronnen, zijnde een set van akoestische foneemmodellen (HMM-modellen), een uitspraakwoordenboek en een taalmodel. In de volgende subsecties gaan we dieper in op elk van de voorgenoemde onderdelen.



Figuur 2.1: De algemene architectuur van een spraakherkenner

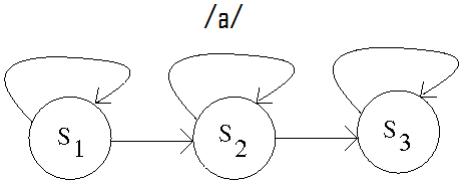
2.1.1 De akoestische voorverwerkingsmodule

De akoestische voorverwerkingsmodule zet het spraaksignaal s(t) (continue tijdsignaal) om in een discrete sequentie van parametervectoren x(n). Het proces wordt kenmerkextractie genoemd: opeenvolgende stukjes uit het geobserveerde spraaksignaal worden in dimensionaliteit gereduceerd tot een akoestische kenmerkvector die zo weinig mogelijk redundante informatie bevat. Elk van de parametervectoren karakteriseert typisch een segment van 30ms. Deze segmentlengte wordt zo gekozen omdat bij spraak 30ms als het kleinste betekenisvolle deel wordt beschouwd. Het segmentvenster verschuift over de spraak per 10ms. Deze spatiëring per 10ms wordt toegepast om de overgangsverschijnselen in het spraaksignaal aan de randen van het segmentvenster niet te verliezen tijdens de modellering.
De parameters die worden berekend zijn meestal Mel Frequency-scale Cepstral Coefficients (MFCCs) en hun eerste en tweede orde tijdsafgeleiden (39 parameters). Voor een uitgebreide beschrijving van het kenmerkextractieproces met MFCC-vectoren wordt verwezen naar [2].

2.1.2 De akoestische HMM-modellen

De eerste kennisbron van de patroonherkenner bestaat uit een collectie akoestische modellen. Akoestische modellen worden gebruikt om de kleinste herkenbare eenheden van spraak te modelleren: fonemen (of klanken). Elke taal beschikt over zijn eigen set van fonemen.

Fonemen kunnen worden opgesomd in een fonetisch alfabet zoals bv. IPA (International Phonetic Association - vaakst gebruikte standaard) [3]) of CGN (Corpus Gesproken Nederlands [4]), YAPA (Yet Another Phonetic Alphabet) en LH+ (Lernout & Hauspie) (de 3 meest gangbare alfabetten voor het Nederlands). In bijlage vindt men een inventaris van de Nederlandse foneemset. De gebruikte alfabetten zijn CGN en YAPA (zie bijlage A)
Om fonemen te modelleren gebruikt men doorgaans Hidden Markov modellen met 3 foneemtoestanden (figuur 2.2). Een Hidden Markov model is een eindige toestandsautomaat waarin de toestandsovergangen bepaald worden door probabiliteiten. Op elk tijdstip zijn er 2 keuzes: ofwel gaat men naar de volgende toestand (volgens een transitiekans), ofwel blijft men in dezelfde toestand. De transitiekansen bepalen dus mee hoe lang men in elke toestand van het foneem blijft bij het doorlopen van het model. Met andere woorden: de transitiekansen beïnvloeden mee of we een snelle dan wel een trage realisatie van een klank modelleren. De akoestische modellering van een klank gebeurt in elk van de



Figuur 2.2: Een Hidden Markov Model met 3 toestanden modelleert het foneem voor de /a/-klank

toestanden zelf, door middel van Gaussiaanse distributiemengsels (met diagonale covariantiematrices). Men gaat ervan uit dat bij de uitspraak van een bepaald foneem steeds gelijkaardige akoestische observaties zullen worden waargenomen. Teruggrijpend naar de uitleg over de akoestische voorverwerkingsmodule, houdt dat in dat men verwacht dat de parametervectoren die overeenstemmen met de realisatie van bijvoorbeeld een /a/-klank steeds gelijkaardige parameterwaarden zullen vertonen. De Gaussiaanse mengsels in de toestanden van het HMM voor de /a/-klank proberen de kansen van het optreden van die "typische" waarden dan te maximaliseren.
Als D de dimensionaliteit van de parametervectoren is, dan wordt de emissiekans voor

parametervector x in toestand s gegeven door middel van de volgende emissiedistributie.

$$b_s(x) = \sum_{k=1}^{K} g_{sk} \cdot N_{sk}(x), \sum_{k=1}^{K} g_{sk} = 1, g_{sk} > 0$$

$$N_{sk}(x) = \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi V_{skd}}} e^{-\frac{(x_d - \mu_{skd})^2}{2V skd}}$$

In deze formules staan de g_{sk}'s voor de gewichten van de K (k: 1..K) Gaussianen in de mengsels horend bij de S (s: 1..S) toestanden van het HMM. μ_{skd} en V_{skd} zijn het gemiddelde en de variantie van de d^e parameterwaarde uit x (d: 1..D) voor Gaussiaan k horend bij toestand s.
Om akoestische modellen te trainen wordt een grote trainingsdatabase gebruikt. Alle voorkomens van een bepaalde klank worden verwerkt door de akoestische voorverwerkingsmodule en de resulterende parametervectoren worden aangewend om de parameters van de Gaussiaanse mengseldistributies te schatten, zodanig dat ze het optreden van de trainingsvoorbeelden maximaliseren.

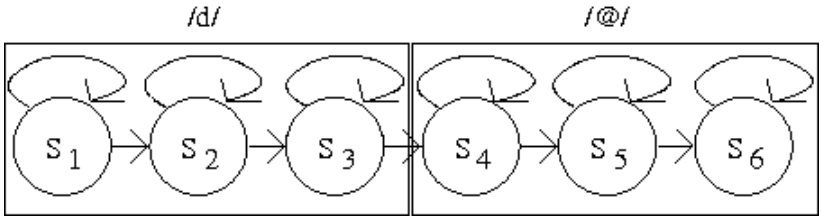
2.1.3 Het uitspraakwoordenboek of lexicon

Een woord is te schrijven als een sequentie van grafemen, zijn uitspraak door een sequentie van fonemen. Het uitspraaklexicon bevat voor ieder woord (orthografische transcriptie) de verwachte uitspra(a)k(en) (fonetische transcriptie). Deze fonetische transcripties kunnen verkregen worden door ze manueel op te stellen, maar omdat dit nogal tijdsrovend is voor grote lexica, wordt er meestal voor geopteerd om met een zogenaamde grafeem-naar-foneemomzetter (g2p-omzetter) te werken. Dit is een tool die, gegeven een orthografie, automatisch een plausibele fonetische transcriptie probeert te genereren. Doorgaans maakt deze tool gebruik van aangeleerde taalkennis die beschrijft welke klanken corresponderen met welke letters of lettersequenties [6]. Een voorbeeld van een lexicon met woorden als lexicale eenheden kan men zien in tabel 2.1.

orthografische transcriptie	fonetische transcriptie (CGN)
Antwerpen	'An-twEr-p@n
twaalf	'twalf
zit	'zIt
krijgt	'krE+xt
geld	'GElt

Tabel 2.1: Voorbeeld van een woordgebaseerd lexicon.

In combinatie met de akoestische modellen maakt het lexicon het mogelijk om woorden (lexicale eenheden) te herkennen. Door middel van het lexicon kunnen immers de akoestische modellen van de fonemen aaneengeschakeld worden, zodat bepaald kan worden welke observatievectoren men met welke probabiliteit bij de uitspraak van een woord mag verwachten. In figuur 2.3 bijvoorbeeld stellen S_1 tot S_6 de toestanden voor die moeten worden doorlopen om het woord 'de' te herkennen door middel van een Hidden Markov keten.



Figuur 2.3: Een Hidden Markov Model voor het woord 'de'

Traditioneel bevat het lexicon woorden als lexicale eenheden, maar in deze thesis gaan we soms ook werken met fonetische syllaben in het lexicon. Voor elke fonetische syllabe wordt dan analoog aangegeven uit welke fonemen ze is opgebouwd.

Bemerk dat ik in deze scriptie wel degelijk met fonetische syllaben als lexicale eenheden werk in mijn eerste herkenner, en niet met orthografische syllaben. Nochtans kan deze laatste optie een logischere keuze lijken, aangezien het uiteindelijk orthografische woordsequenties zijn die we willen herkennen. Het woord *peperkoek* bijvoorbeeld zou ook kunnen gezien worden als de orthografische sequentie *pe - per - koek* in plaats van de fonetische

sequentie /'pe - p@r - kuk/. Na herkenning van de orthografische syllaben moet men dan enkel de juiste delen weer aan elkaar plaatsen. Toch werk ik hier anders omwille van twee redenen:

1. Het is niet eenvoudig om woorden automatisch in correcte orthografische lettergrepen te splitsen, terwijl er voor fonetische syllaben wel een vrij eenvoudige werkwijze bestaat (zie verder)

2. Het is niet zeker dat de set van orthografische syllaben even beperkt is als de set van fonetische syllaben. Bijvoorbeeld: de woorden *Dewaele* en *waken* bevatten beide dezelfde fonetische syllabe /wa/, maar die wordt wel gerealiseerd in twee verschillende orthografische syllaben *wae* en *wa*. Ondanks het feit dat van deze laatste reden ook tegenvoorbeelden te bedenken zijn (bijvoorbeeld *ver* kan zowel als /v@r/ als als /vEr/ gelezen worden), lijkt het eerder dat de eerste groep voorbeelden in de meerderheid is.

2.1.4 Het taalmodel

Een taalmodel is een stochastisch model dat probabiliteiten toekent aan het optreden van sequenties van lexicale eenheden. Bij een traditionele herkenner, waar het lexicon woorden bevat, bepaalt het taalmodel dus de kans van optreden van een bepaalde woordsequentie. Het spreekt voor zich dat in de Nederlandse taal bijvoorbeeld sommige woordsequenties waarschijnlijker zijn dan andere. Als de herkenner op een gegeven moment de keuze heeft tussen de akoestisch gelijkaardige woordsequenties *één van de* en *heen fan te*, dan is het het taalmodel dat zal aangeven dat de eerste sequentie waarschijnlijk de juiste is.
Meer precies bevat het taalmodel de kansen van optreden van een volgend woord w_n, gegeven de voorgaande woorden:

$$P(w_n|w_1..w_{n-1})$$

Omdat er vaak vanuit gegaan wordt dat de verre context nog weinig invloed heeft op het volgende woord worden bovenstaande kansen vaak beperkt tot:

$$P(w_n|w_{n-N+1}..w_{n-1})$$

Algemeen spreekt men van N-gramkansen, specifieker van unigram (N=1), bigram (N=2), trigram (N=3), enz. kansen.

Om de N-gramkansen te schatten maakt men doorgaans gebruik van een grote trainingstekst. Typisch is dat een corpus dat handelt over het domein waarin men spraakherkenning wil gaan doen. Aan de hand van frequentietellingen van ieder woord, ieder woordenpaar, ieder woordtriplet, enz. doet men dan een poging om het relatieve voorkomen van de verschillende woordsequenties te schatten. Bepaalde N-grammen zullen echter niet in het corpus voorkomen. Toch wil dit niet zeggen dat deze N-grammen onmogelijk zijn. Men lost dit probleem daarom op door middel van uitsmeringstechnieken. Een naïeve uitsmeringstechniek bestaat erin bij de kansberekening van iedere N-gramteller 1 op te tellen. De bigramkans $P(w_2|w_1)$ dat w_2 opvolger is van w_1 aan de hand van de bigramtellers $c(w_i, w_j)$ $\forall w_i, w_j \in W^2$ (waarbij W de verzameling van woorden in het lexicon is), wordt dan bijvoorbeeld gegeven door:

$$P(w_2|w_1) = \frac{c(w_1,w_2)+1}{\sum_w c(w_1,w)+1}$$

Deze formule heeft het nadeel dat er te veel kans wordt toegekend aan alle N-grammen die in het corpus niet voorkomen. Daarom bestaan er ook meer geavanceerde uitsmeringstechnieken. Eén daarvan is de Kneser-Ney-uitsmering die ik later in dit werk zal gebruiken bij het trainen van mijn taalmodellen. Voor meer informatie over deze uitsmeringstechniek verwijs ik naar [10, 12].

Een ander nadeel van het woordgebaseerd N-grammodel is de niet-robuustheid van de N-gram schattingen. Om goede schattingen van alle N-grammen te krijgen heeft men immers een enorm groot corpus nodig. Stel dat men bijvoorbeeld een lexicon zou hebben van L woorden, dan heeft men L^3 mogelijke trigrammen. Niet al deze trigrammen zullen voldoende vaak voorkomen om een "correcte" N-gramschatting te kunnen maken. Men lost dit op door het berekenen van backoff factoren. Hierbij valt men terug op de kans van het (N-1)-gram bij het schatten van de kans van het N-gram. Dit wordt gerealiseerd door een discountwaarde D in te voeren die aangeeft hoeveel keer een woordsequentie minstens in het corpus moet voorkomen alvorens men er een N-gramkans zal van schatten. Door middel van deze aanpassingen worden de N-gramkansen berekend aan de hand van de frequentietellingen uit het corpus een nauwkeurigere benadering voor de invoer die de spraakherkenner verwacht.

De backoff factoren voor bigramkansen worden als volgt berekend:

De bigramkansen worden berekend door:

$$P(w_2, w_1) = \frac{c(w_1,w_2)-D}{\sum_{w,c(w_1,w)>k} c(w_1,w)} \text{ als } c(w_1, w_2) > k \geq D$$

Waarbij D de discount is die niet groter is dan k. Hierbij geeft k aan hoeveel keer een woordsequentie maximaal mag voorkomen om als niet representatief te worden beschouwd.

De backoff factoren worden gegeven door de formule:

$$P(w_2, w_1) = \gamma_{uni}(w_1)P(w_2) \text{ als } c(w_1, w_2) \leq k$$

Zodat de bigram backoff factor $\gamma_{uni}(w_1)$ wordt gegeven door:

$$\gamma_{uni}(w_1) = \frac{1-\sum_{w;c(w_1,w)>k} P(w|w_1)}{1-\sum_{w;c(w_1,w)>k} P(w)} \text{ als } c(w_1, w_2) \leq k$$

De trigramkansen worden berekend door:

$$P(w_3|w_1,w_2) = \frac{c(w_1,w_2,w_3)-D}{\sum_{w,c(w_1,w_2,w)>0} c(w_1,w_2,w)} \text{ als } c(w_1,w_2,w_3) > k \geq D$$

Waarbij de backoff factoren worden gegeven door de formule:

$$P(w_3|w_1,w_2) = \gamma_{big}(w_1,w_2)P(w_1,w_2) \text{ als } c(w_1,w_2,w_3) \leq k$$

Zodat de trigram backoff-factor $\gamma_{big}(w_1,w_2)$ wordt gegeven door:

$$\gamma_{big}(w_1,w_2) = \frac{1-\sum_{w;c(w_1,w_2,w)>k}}{1-\sum_{w;c(w_1,w_2,w)>k} P(w|w_2)} \text{ als } c(w_1,w_2,w_3) \leq k$$

In mijn werk zal ik niet alleen woordgebaseerde taalmodellen trainen, maar ook syllabegebaseerde taalmodellen. Deze laatste zullen aangeven wat de kans op optreden van een volgende fonetische syllabe is, gegeven de N voorgaande.

2.1.5 Het herkenningsproces

Samengevat kan men het herkenningsproces als volgt beschrijven: (1) het spraaksignaal wordt verwerkt tot een sequentie van compacte kenmerkvectoren, (2) de akoestische modellen proberen te achterhalen welke klanken overeenstemmen met de aangevoerde parametervectoren, (3) het lexicon probeert aan te geven welke woorden zouden kunnen overeenstemmen met de herkende klanken, en (4) het taalmodel geeft aan welke van de woordsequenties die zo ontstaan waarschijnlijk zijn en welke niet. De meest waarschijnlijke hypothese die de herkenner bekomt, zal hij naar voor schuiven.

Technisch gezien komt het erop neer dat de akoestische HMM-modellen voor de fonemen door middel van het uitspraaklexicon kunnen aaneengeschakeled worden tot HMM-modellen voor woorden, en door middel van het taalmodel verder aangeschakeld tot een HMM-model voor zinnen. Door gebruik te maken van het Viterbi algoritme wordt deze

samengestelde HMM doorlopen op zoek naar de beste hypothese, gegeven de aangevoerde sequentie van parametervectoren. Als de herkenner wordt bedreven in N-best mode, dan zal hij op zoek gaan naar de N beste (= meest waarschijnlijke) hypothesen horend bij de geobserveerde akoestische data. Een lattice is een gerichte graaf die de meest valabele (N-best) hypothesen van de herkenner bevat. In de lattice kan bijkomende informatie zijn opgeslagen, zoals bijvoorbeeld de akoestische scores en de taalmodelscores van elke hypothese. In mijn werk is het de bedoeling een lattice te genereren met de N beste syllabehypothesen in, en die lattice vervolgens te herscoren door een woordlexicon en een woordtaalmodel in rekening te brengen.

In de praktijk kan de zoekruimte bij het decoderen enorm groot worden, afhankelijk van de lexicongrootte. Beam search is een breedte-eerst zoekmethode waarbij meerdere hypothesen parallel worden onderzocht. De uitvoeringstijd van de beam search wordt op elk tijdstip beperkt door die hypothesen uit te sluiten met een probabiliteit die kleiner is dan de probabiliteit van de huidige tophypothese verminderd met een bepaalde drempelwaarde. Dit proces wordt snoeien ('pruning') genoemd en dit gebeurt afhankelijk van de zoekbreedte (de 'beam width'). Een lage zoekbreedte kan ervoor zorgen dat de beam search niet termineert of dat er slechts een quasi-optimale oplossing wordt gevonden.
Eén van de hypothesen van deze thesis is dat slechts een beperkt aantal fonetische syllaben zal nodig zijn om het merendeel van de Nederlandse woorden te kunnen vormen. Als dit zo is, dan is een volgende hypothese dat dit beperkte aantal syllaben de zoekruimte bij het herkennen drastisch zal beperken, waardoor de herkenning in de eerste fase sneller zal kunnen verlopen. De gewonnen tijd kan dan gebruikt worden om de syllabelattice (ook al een beperkte zoekruimte) te herwerken tot een woordhypothese.

2.2 Evaluatiemetrieken

In dit gedeelte bespreek ik een aantal evaluatiematen voor (delen van) de herkenner.

2.2.1 De WER van de spraakherkenner

Continue spraakherkenners worden in de literatuur over het algemeen vergeleken door middel van de Word Error Rate (WER) die zij behalen op een bepaalde herkenningstest. De WER is gedefineerd als het percentage bewerkingen dat nodig is om hypothese van de herkenner om te zetten in de te herkennen referentie.

$$WER = \frac{\#S+\#I+\#D}{N},$$

met $\#S$ het aantal substituties, $\#I$ het aantal inserties en $\#D$ het aantal deleties.

De formule voor WER is afgeleid van de formule voor Levenshtein afstand. De Levenshtein afstand [19] (ook wel de bewerkingsafstand genoemd) tussen twee tekenreeksen werd door Vladimir Levenshtein gedefinieerd als het minimum aantal aanpassingen nodig om de ene tekenreeks in de andere om te zetten.

Er bestaan andere variaties op de WER-formule waarbij men andere gewichten toekent aan de bewerkingen die men meer of minder verkiest uit te voeren [20].

2.2.2 De OOV-rate van het lexicon/taalmodel

Het lexicon wordt meestal in ruimte beperkt: hoewel de geheugencomplexiteit linear is, zorgt een groter lexicon er immers voor dat er bij het zogenaamde beam search algoritme van de spraakherkenner meer hypothesen zullen worden onderzocht. De patroonherkenner onderzoekt namelijk alle hypothesen die op elk moment waarschijnlijk genoeg zijn, om zo de beste hypothese te vinden. Dit maakt de herkenner doorgaans trager, wat niet altijd gewenst is.

Wanneer echter een te herkennen woord niet in het lexicon voorkomt, dan is dit een zogenaamd Out Of Vocabulary (OOV) woord. Uiteraard kan dit woord nooit correct herkend worden. De OOV-rate van het lexicon/taalmodel is daarom een belangrijk designcriterium. Voor elke herkenner is het de betrachting om de OOV-rate op de te verwachten woordenschat zoveel mogelijk te beperken, zodat het aantal fouten bij de uiteindelijke herkenning zo

laag mogelijk kan gehouden worden. Doorgaans vormt men zich een idee van de OOV-rate van het lexicon/taalmodel door de dekkingsgraad te berekenen op een grote hoeveelheid respresentatief tekstmateriaal.

2.2.3 De perplexiteit van het taalmodel

De perplexiteit van het taalmodel vertegenwoordigt de a priori informatie in het taalmodel. Perplexiteit is een ander woord voor verbijstering. Het geeft weer hoe zeer we nog verbijsterd zijn over een taal, gegeven de N-gramkansen in het taalmodel.
Perplexiteitswaarden kunnen berekend worden op een gedeelte van de trainingsdata van het taalmodel, om een idee te krijgen van hoe onzeker de identiteit van het volgende woord nog is, gegeven de informatie in het taalmodel. De algemene perplexiteitsformule wordt gegeven door:

$$PPL = e^H,$$

waarbij H de entropie is van de trainingstekst. H wordt berekend als:

$$H = -\sum_{w_1 \ldots w_n} P(w_1 \ldots w_n) \log P(w_n | w_1 \ldots w_{n-1})$$

Hoe kleiner de perplexiteitswaarde, hoe meer informatie het taalmodel vertegenwoordigt.

Naast perplexiteit van het taalmodel bestaat er ook zoiets als perplexiteit van de testset. Dit is gedefinieerd als het inverse van de gemiddelde a priori kans van optreden van de correcte woorden in de testset, gegeven de voorafgaande context en het taalmodel. Wanneer de testset bestaat uit de woorden $w_1, \ldots w_N$, dan wordt de perplexiteit van de testset voor een N-gram taalmodel gegeven door:

$$P_{testset} = \prod_{k=1}^{N} P(w_k | w_{k-n-1} \ldots w_{k-1})$$

Dit kan ook worden geschreven als:

$$P_{testset} = e^{\sum_{k=1}^{N} \log P(w_k|w_{k-n-1}...w_{k-1})}$$

Zodat de perplexiteitsformule voor de gemiddelde a priori kans per woord dus gegeven wordt door:

$$PPL_{testset} = e^{-\frac{1}{N} \sum_{k=1}^{N} \log P(w_k|w_{k-n-1}...w_{k-1})}$$

In mijn werk heb ik perplexiteiten berekend met de SRI LanguageModeling Toolkit (zie sectie 3.1.2).

Historisch is de perplexiteitsberekening gebaseerd op een formule uit de informatietheorie, genaamd de entropieformule. Entropie werd door grondlegger van de informatietheorie Claude Shannon gedefineerd als de maat voor informatiedichtheid in een reeks gebeurtenissen. Wanneer een gebeurtenis plaatsvindt waarvan vooraf onzeker was of deze daadwerkelijk zou gebeuren, ontstaat informatie. Shannon liet zich door Boltzmann en Gibbs inspireren om de entropieformule in de informatietheorie te introduceren.

Hoofdstuk 3

Taalanalyse en bouw van de taalmodellen

Een belangrijke taak in deze thesisopdracht was het trainen van zowel een syllabisch als een woordgebaseerd taalmodel. Om dat te bewerkstelligen, waren een aantal zaken vereist: een groot tekstcorpus, een toolkit om taalmodellen mee te trainen en een manier om woorden automatisch om te zetten naar fonetische syllaben. In het eerste deel van dit hoofdstuk bespreek ik elk van de voorgaande vereisten en leg ik nader uit hoe ik uiteindelijk mijn taalmodellen heb getraind. Daarna volgt een gedeelte over taalanalyse, waarin woorden en fonetische syllaben met elkaar vergeleken worden als basisbouwstenen voor de Nederlandse taal. De dekkingsgraad die met beide bouwstenen kan worden behaald wordt vergeleken, alsook de perplexiteiten van de taalmodellen die kunnen worden getraind.

3.1 Het trainen van een taalmodel

Om de nodige taalmodellen te kunnen trainen heb ik gebruik gemaakt van het Mediargus tekstcorpus, de SRI Language Modeling toolkit en de Autonomata grafeem-naar-foneem toolbox.

3.1.1 Het Mediargus tekstcorpus

Het Mediargus corpus is een verzameling van artikels geschreven gedurende een tijdspanne van 5 jaar (1999-2004), afkomstig uit 12 verschillende Vlaamse kranten en tijdschriften. De kranten die tot het corpus behoren zijn De Morgen, De Standaard, De Tijd, Gazet Van Antwerpen, Het Belang Van Limburg, Het Laatste Nieuws, Het Nieuwsblad en Het Volk. Het spreekt voor zich dat voor een spraakherkenner die is bedoeld om nieuwsuitzendingen automatisch te transcriberen een dergelijk corpus van kranten goed geschikt als leercorpus voor het opstellen van een taalmodel. Toch heb ik niet alle krantenteksten gebruikt tijdens mijn onderzoek. Omwille van geheugenbeperkingen was ik niet in staat N-gram taalmodellen te trainen op de volledige 8-koppige krantenset voor $N > 3$. Omdat dergelijke grootteordes wel nodig waren bij het onderzoek rond syllabische taalmodellen, heb ik besloten om mij voor het trainen van de taalmodellen te beperken tot de krantenartikels uit De Standaard. Desalniettemin zijn sommige resultaten die ik presenteer in mijn taalanalyse (zie sectie 3.2) gebaseerd op de data van het volledige krantenbestand van Mediargus, omdat dat een beter beeld geeft van de Nederlandse taal (want berekend op meer data).
In totaal duiken in de 8 verzamelingen krantenartikels $1.22 \cdot 10^9$ woorden op, waarvan er $4.42 \cdot 10^6$ uniek zijn. In de subset van De Standaard zitten $1.12 \cdot 10^8$ woorden en daarvan zijn er $1.14 \cdot 10^6$ uniek. Om een idee te geven van de opslagruimte nodig voor het De Standaard corpus geef ik nog mee dat de verzameling artikels uit De Standaard een tekstbestand is van 739 MB.

3.1.2 De SRI Language Modeling toolkit

De SRI Language Modeling toolkit werd ontwikkeld door Andreas Stolcke [9] en is een gebruiksvrije toolkit voor het opbouwen en toepassen van statistische taalmodellen. De toolkit wordt hoofdzakelijk gebruikt om taalmodellen te trainen voor spraakherkenning maar kan naar verluidt ook nuttig zijn voor machinevertaling. Het pakket was voor het eerst beschikbaar in 1995 en wordt sindsdien geregeld vernieuwd. Het hoofdtype van taalmodellen dat door SRI-LM wordt ondersteund zijn de klassieke N-gram taalmodellen.

Ze worden altijd geschreven in ARPA-formaat. Figuur 3.1 geeft de hoofdlijnen weer van een voorbeeld ARPA 3-gram taalmodel

```
\data\
ngram 1=n1
ngram 2=n2
...
ngram N=nN

\1-grams:
p          w          b

\2-grams:
p          w1 w2      b2

\N-grams:
p          w1 ... w(n)

\end\
```

Figuur 3.1: Een ARPA 3-gram taalmodel

De eerste regels van het ARPA language model bestand bevatten het aantal unigrammen, bigrammen, ... N-grammen. Daaronder staan dan de unigrammen zelf met de probabiliteiten, eventueel gevolgd door de backoff factoren.

Het is met de SRI-toolbox dat ik mijn N-gram taalmodellen heb getraind.

Naast het bouwen van taalmodellen kan de SRI toolbox ook aangewend worden om gewoon frequentietellingen van N-grammen te bepalen. Zo heb ik het script *make-batch-counts* gebruikt om de unigram voorkomens van de woorden in de 8 krantencorpussen samen te tellen door middel van het volgende commando:

make-batch-counts ./mediargus_kranten.datalijst 2 /bin/cat ./mediargus-count -order 1

De output van dit commando is een woordenlijst van unieke woorden in de folder ./mediargus-count (-order 1 wil zeggen: unigrammen), met daarbij de woordfrequentie, ingepakt in gzip

formaat. Met een zelf geschreven Perl-script (*sort-ngrams.pl*) heb ik deze unigramtellingen vervolgens gesorteerd volgens frequentie, en de 13 meest frequente woorden in het Mediargus corpus (die meer dan 10^7 keer voorkomen) zijn hier weergegeven in tabel 3.1.

woord	frequentie
de	79906370
van	34906406
het	33960119
een	31398191
in	27297024
en	26190916
op	15121165
dat	14339591
is	12851113
voor	11909447
te	11642448
met	11330539
zijn	10829745

Tabel 3.1: De 13 meest frequente woorden van het Mediargus corpus

Een analoog experiment op het De Standaard corpus leverde quasi dezelfde lijst van meest frequente woorden op (ook “die” in de lijst).

3.1.3 Fonetische syllabische tekstcorpora

Om een woordtaalmodel te trainen waren de nodige data meteen voorhanden (Mediargus - De Standaard). Het bouwen van syllabisch taalmodel daarentegen vergt de beschikbaarheid van een gesyllabifieerd trainingscorpus. Om zo’n dergelijk corpus te construeren, heb ik gebruik gemaakt van de DUB-versie (Vlaamse versie) van de Nederlandse grafeem-naar-foneemomzetter (g2p-omzetter) uit de Autonomata G2P toolkit [6].

In een eerste stap van de corpusconstructie heb ik met de grafeem-naar-foneemomzetter de $4.42 \cdot 10^6$ unieke woorden uit de oorspronkelijke krantenteksten fonetisch getranscribeerd. De gegenereerde fonetische transcripties bevatten naast informatie over de uitspraak van de woorden ook informatie over de syllabegrenzen. In het CGN-alfabet bijvoorbeeld wordt het symbool /-/ gebruikt om een syllabegrens binnen een woord aan te geven, terwijl een spatie in de fonetische transcriptie een woordgrens - en dus ook een syllabegrens - aanduidt. In figuur 3.2 wordt een voorbeeld van een dergelijke CGN-transcriptie gegeven voor het

orthografie	g2p-transcriptie	resultaat
VRT-programmma	ve-Er-'te pro-'GrA-ma	ve Er te pro GrA ma

Tabel 3.2: Een voorbeeld van een orthografische transcriptie door de g2p-omzetter

woord het woord "VRT-programma" (het symbool /'/ wordt gebruikt voor het aangeven van een klemtoon in de uitspraak). Door de woorden te transcriberen had ik dus een lijst gegenereerd waarin voor elk woord werd weergegeven uit welke syllaben het is opgebouwd. In een tweede stap van het proces heb ik dan de syllabische versies van de krantencorpora gebouwd door in elk van de originele tekstbestanden de woorden te vervangen door de overeenkomstige fonetische syllaben. Omdat de akoestische modellen van de herkenner waar ik mee zou werken (zie hoofdstuk 4) waren getraind voor de YAPA-foneemset, ging deze omzetting gepaard met een scriptfunctie die de niet-identieke CGN-foneemsymbolen omzette naar YAPA-formaat. Hierbij was er even onduidelijkheid of de 'oeu' in "oeuvre" naar /@:/ of /Y:/ moest worden omgezet. In een paper omtrent fonemische transcriptie [5] vond ik omtrent dit leenvocaal:

"De /oe/ van .freule. en .oeuvre. wordt soms weergegeven door een verlenging van de /Y/ of van de /@/. Aangezien beide dicht bij elkaar aanleunen is de keuze voor de meer gangbare /Y:/ te verkiezen (maar in sé ongemotiveerd). Het idiosyncratisch symbool /9:/ is te vermijden."

Er werd uiteindelijk besloten om zowel /Y:/ als /Y+/ om te zetten in /@:/ zodat de finale symboolconversies worden weergegeven in tabel 3.3.

Na omzetting van de woorden in syllaben wordt het volgende stuk tekst uit De Morgen:

CGN	YAPA
E+	Eˆ
Y:	@:
Y+	@:
A+	Oˆ
2	&

Tabel 3.3: De gehanteerde conventie voor het omzetten van CGN naar YAPA

```
<s> in de digitale animatiefilm Monsters Inc worden we
geintroduceerd in Monstropolis een soort parallelle wereld
die door alle mogelijke en onmogelijke creaturen bevolkt wordt </s>
<s> het zijn die monsters die 's nachts onder de bedden
en in de kasten van kinderkamers opduiken en zo het jonge
volkje aan het schrikken brengen </s>
```

vervangen door

```
<s> In d@ di Gi ta l@ a ni ma si fIlm mOn st@rs INk wOr d@n w@
G@ In tro dy sert In mOn stro po lIs @n sort pa rA lE l@ we r@lt
di dor A l@ mo G@ l@ k@ En On mo G@ l@ k@ kre a ty r@n b@ vOlkt wOrt </s>
<s> h@t zE$\wedge$n di mOn st@rs di s nAxs On d@r d@ bE d@n
En In d@ kAs t@n vAn kIn d@r ka m@rs Ob d@: k@n En zo h@t jON @
vOl kj@ an h@t sxrI k@n brEN @n </s>
```

Zoals is te zien doet het script zowel de splitsing in syllaben als de omzetting van CGN naar YAPA. De syllabische versie van het corpus van De Standaard werd verder gebruikt voor het maken van de syllabische taalmodellen.

3.1.4 Taalmodellen trainen met de SRI LM toolkit

Het trainen van een taalmodel met SRI LM gebeurt door de volgende functieaanroep na het tellen van de N-grammen met *make-batch-counts*:

make-big-lm -lm filename.ngram.lm -name filename.ngram.lm -read counts.ngrams/merge-iter-1.ngrams.gz -order N -vocab vocabulary.words -unk -map-unk "<UNK>´´ -gt1min X -gt2min Y ... -kndiscount -interpolate*

Met de volgende parameters maakte ik de N-gram taalmodellen:

1. De -lm parameter geeft aan hoe het taalmodel zal worden genoemd. De naam verwijst naar de gebruikte corpustekst, de orde van het taalmodel, gevolgd door de grootte van het lexicon, indien syllabisch taalmodel betrof voegde ik "syl´´ toe.

2. De -read parameter geeft waar het bestand met de N-gram frequentietellers kan gevonden worden, gemaakt met bijvoorbeeld *make-batch-counts*.

3. De -order parameter bepaalt de orde van het resulterende taalmodel (bijvoorbeeld 3).

4. De -vocab parameter bepaalt de woordenschat (het lexicon).

5. De -unk parameter geeft aan dat het taalmodel ieder woord dat niet in het lexicon voorkomt mapt op het onbekende woord teken (een "open woordenschat´´ taalmodel).

6. De -map-unk parameter geeft aan hoe het onbekende woord teken wordt genoemd.

7. De -gtXmin parameters geven de Good-Turing cut-off waarden aan voor orde 1 ... N. Deze Good-Turing discounting methode wordt beschreven in [11]. De discounting methode wordt aangewend voor het herdistribueren van de waarschijnlijkheidsmassa van de meer frequent geobserveerde gebeurtenissen naar de minder frequente en ongeziene gebeurtenissen en heeft een invloed op de perplexiteit van het taalmodel. Volgens de paper van Katz is een waarde van 5 of zo een goede keuze. Voor het maken van de syllabische taalmodellen werd echter meestal de waarde 1 gebruikt.

8. Wanneer we -kndiscount als parameter geven dan wordt in plaats van de discounting methode van Katz gebruik gemaakt van deze beschreven door Kneser en Ney in de

aangepaste vorm door Goodman en Chen [12].

9. De -interpolate parameter geeft aan dat er interpolatie moet worden gebruikt waarbij de N-gram waarschijnlijkheidsdistributies geïnterpoleerd worden met lagere orde schattingen. Het resultaat is een standaard backoff taalmodel [12] dat soms betere resultaten geeft.

3.2 Woorden versus syllaben in het Nederlands

3.2.1 De dekkingsgraad met woorden... Én met syllaben

aantal woorden	aantal syllaben	dekkingsgraad
1000	822	69.16%
1319	1000	71.55%
2000	1339	75.08%
3893	2000	80.5%
5000	2306	82.44%
10000	3327	87.29%
20000	4763	91.28%
21938	5000	91.75%
40000	6762	94.4%
80000	9356	96.6%
94004	10000	96.99%
160000	12515	98%

Tabel 3.4: Dekkingsgraad voor de syllaben behorend bij de meest frequente woorden

De beperkte lexicongrootte werd in verschillende papers [14] [15] reeds aangehaald als motivatie om subwoord eenheden te gebruiken om meer woorden te herkennen. Fonetische syllaben bevinden zich tussen fonemen en woorden, waarbij er in theorie een oneindig

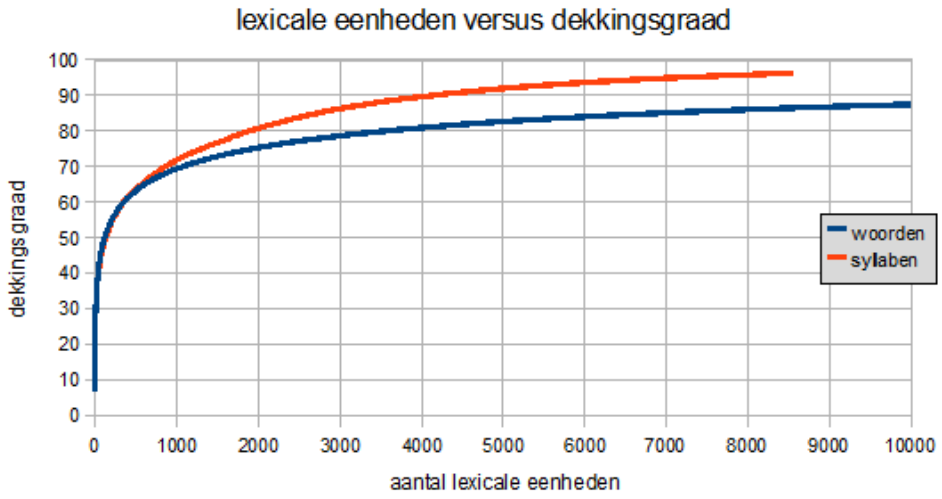
aantal woorden	aantal syllaben	dekkingsgraad
1000	45	32.88%
2000	54	35.63%
5000	79	40.14%
10000	103	43.78%
20000	154	51.8%
40000	246	60.3%
80000	439	70.51%
160000	857	81.86%
185200	1000	84.33%
313500	2000	93.32%
437900	5000	98.26%
481200	9996	99.07%
499400	20016	99.17%

Tabel 3.5: Dekkingsgraad voor de meest frequente syllaben

groot aantal woorden kan worden gevormd, die met een vijftigtal fonemen kunnen worden uitgesproken. Het bleek reeds in de literatuur dat samengestelde woorden opsplitsen in deelwoorden in het Nederlands ervoor zorgt dat men een lagere OOV-rate en een lagere WER bekomt [13]. Ook wanneer men woorden opsplitst in morfemen bekomt men betere resultaten [8]. In een poging om het OOV-probleem deels te omzeilen willen we in deze scriptie een spraakherkenner met een gelaagde structuur ontwikkelen. Deze herkenner zoekt in een eerste fase naar alle schijnbaar valabele sequenties van fonetische lettergrepen (syllaben) die de spraak kunnen verklaren. In een tweede fase herscoort hij de gemaakte hypothesen op basis van een traditioneel taalmodel en een lexicon waarin de uitspraken van woorden als sequenties van syllaben zijn opgeslagen.

Een eerste vraag die ik in deze thesis wilde beantwoorden, houdt verband met één van de hypothesen die in het thesisvoorstel wordt gemaakt: zijn de fonetische syllaben in het Nederlands generatief? Om dit na te gaan heb ik in de eerste plaats gekeken naar de

dekkingsgraden die kunnen worden bereikt met syllabelexicons van verschillende grootte. De dekkingsgraad is het percentage van woorden in het corpus dat kan worden gevormd met de lexicale eenheden. We rekenden hierbij de dekkingsgraad uit van woorden die kunnen gevormd worden met een beperkt aantal syllaben behorend bij de meest frequente woorden (tabel 3.4, figuur 3.4). We rekenden ook uit hoeveel woorden we konden vormen met een selectie van de meest frequente syllaben (tabel 3.5, figuur 3.6). Zoals we zien in de tabellen kunnen we met een beperkte verzameling van 10000 syllaben zo goed als alle woorden vormen, syllaben zijn dus generatief. We zien dat we met de meest frequente syllaben een hogere dekkingsgraad bekomen dan met de syllaben van de meest frequente woorden. Het is echter meest zinvol dat de spraakherkenner frequent gebruikte woorden en de bijhorende syllaben herkent, dan frequente syllaben die overeenkomen met veel woorden die minder frequent voorkomen. In figuur 3.2 vinden we de dekkingsgraad voor de meest frequente woorden en de bijhorende syllaben.



Figuur 3.2: Dekkingsgraad voor woorden en syllaben

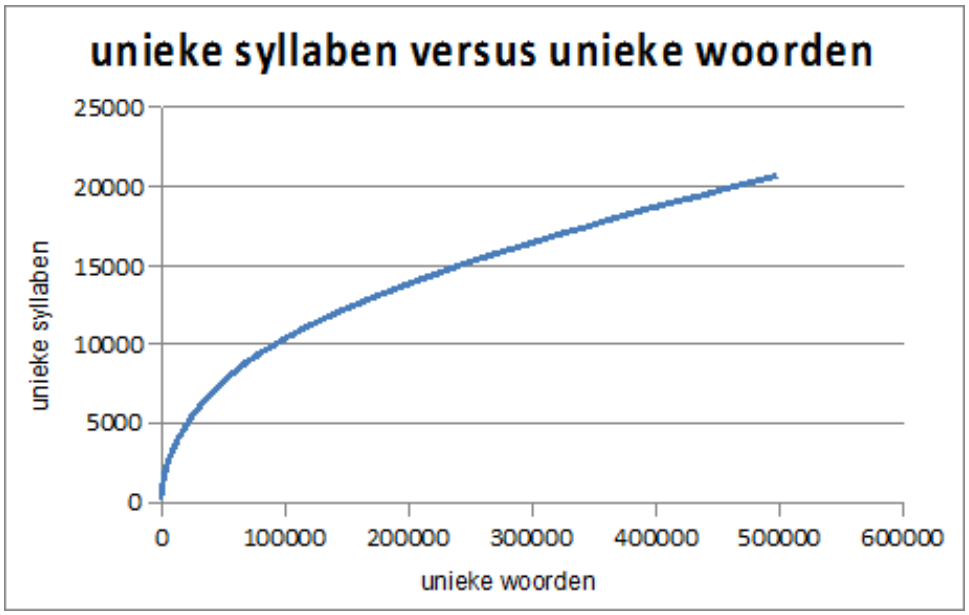
$$\text{dekkingsgraad} = L/N$$

met L = het aantal woorden uit het lexicon in het corpus en N = het totaal aantal woorden in het corpus.

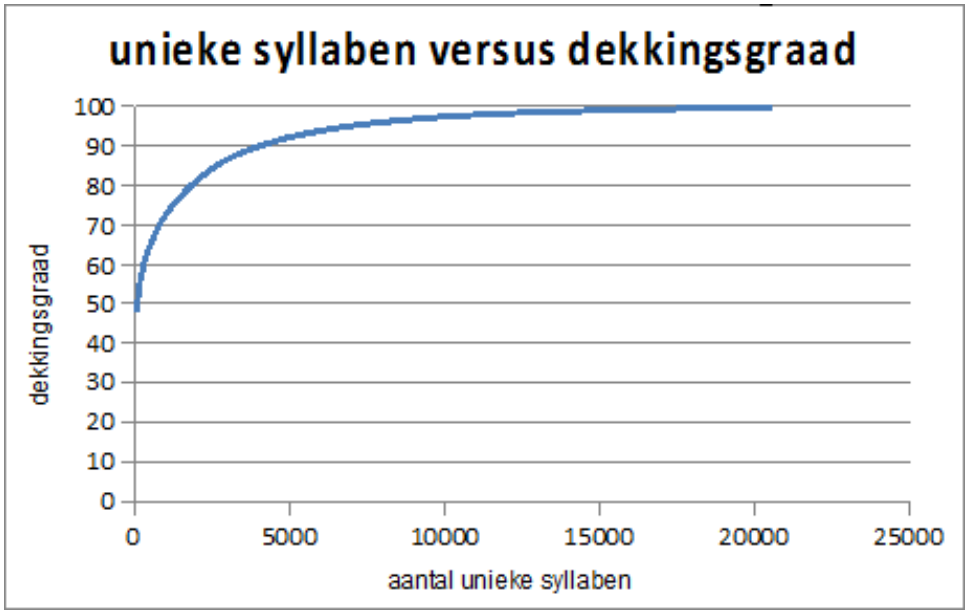
Gegeven een woordenlijst gesorteerd volgens frequentie kunnen we het gemiddeld aantal syllaben per woord berekenen voor het De Standaard (tabel 3.6) corpus.

aantal woorden	aantal syllaben	gemiddeld aantal syllaben per woord
12005	5000	2.401
25584	10000	2.5584
54152	20000	2.7076
114653	40000	2.8663
242800	80000	3.035
515316	160000	3.2207
1099716	320000	3.4366

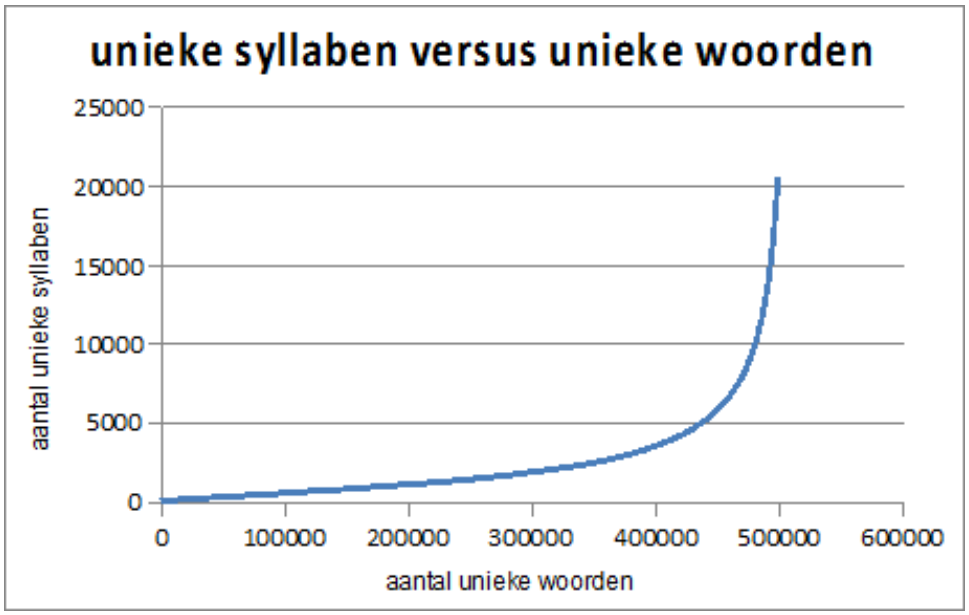
Tabel 3.6: Gemiddeld aantal syllaben per woord in De Standaard



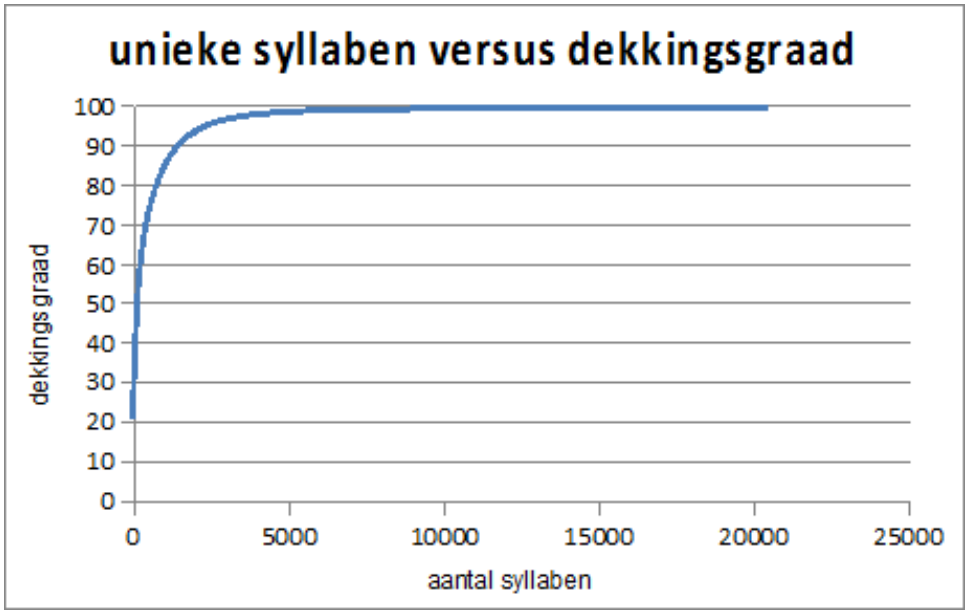
Figuur 3.3: Unieke syllaben versus unieke woorden (syllaben gesorteerd volgens woordfrequentie)



Figuur 3.4: Dekkingsgraad versus unieke syllaben (syllaben gesorteerd volgens woordfrequentie)



Figuur 3.5: Unieke syllaben versus unieke woorden (syllaben gesorteerd volgens syllabefrequentie)



Figuur 3.6: Dekkingsgraad versus unieke syllaben (syllaben gesorteerd volgens syllabefrequentie)

3.2.2 Perplexiteiten van woord- en syllabetaalmodellen

Een voorbeeld van een 3-gram van woorden ziet er als volgt uit:

... de helmboswuivende Hektor ...

De syllabische versie ziet er echter als volgt uit:

... de hElm bOs w@: v@n d@ hEk tOr ...

Door de verminderde context bij syllaben is een nieuwe vraag die rijst of we in het syllabisch taalmodel even veel a priori informatie in het taalmodel kunnen bekomen als bij het woordgebaseerd taalmodel. Een logische vraag hierbij is hoe groot N moet zijn bij het N-gram syllabisch taalmodel om een gelijkaardige perplexiteit van het taalmodel te bekomen. We wensen dat de onzekerheid hierbij gelijkaardig zou zijn. We kunnen gemakkelijk inzien dat door de kortere lexicale eenheden die men probeert te voorspellen, de syllabische herkenner zal benadeeld zijn en een hogere perplexiteit zal bezitten. We maakten een lijst van ongeveer 10K syllaben behorend bij de 80K meest frequente woorden. We kunnen met deze 10K syllaben uiteraard meer dan 80K woorden vormen. We weten reeds uit tabel 3.4 dat 80K woorden overeenkomen met een dekkingsgraad van 96.6 % in het corpus. Omwille van de kortere lexicale eenheden bij een 3-gram syllabisch model is er minder contextinformatie dan bij een 3-gram woordgebaseerd taalmodel. Een n-gram syllabisch taalmodel werd gezocht met een perplexiteit die ongeveer gelijk is aan dat van een 3-gram woordgebaseerd taalmodel. Wanneer men er de literatuur op naslaat vindt men dat trigram taalmodellen met woorden als lexicale eenheden die gebruik maken van de aangepaste Kneser-Ney uitsmeringsformules [10] van Chen en Goodman [12] vaak de beste keuze zijn. Soms geven de taalmodellen die interpolatie gebruiken een iets beter resultaat zodat ook hiervoor werd gekozen.

Voor het tellen van het aantal trigrammen van woorden in het De Standaard corpus werd gebruik gemaakt van *make-batch-counts*:

make-batch-counts files.lst 2 cat ./cnts.3gram/ -order 3

Met make-big-lm kan dan een trigram backoff taalmodel met een lexicon van 80K woorden dat gebruikmaakt van aangepaste Kneser-Ney uitsmering en interpol atie worden aangemaakt als volgt:

make-big-lm -read ./cnts.3gram/.gz -name DeStandaard.3gram.80Kwords.lm -order 3 -vocab ../NewTextDeStandaard/voc80K.wlist -text ../NewTextDeSta ndaard/DeStandaard.txt -lm DeStandaard.3gram.80Kwords.biglm.lm -gt1min 1 -gt2min 1 -gt3min 1 -kndiscount -interpolate*

Op dezelfde manier werden de syllabische taalmodellen aangemaakt, maar dan met fonetische syllaben als lexicon. We namen de eerste en de laatste 100 zinnen van de corpustekst van het De Standaard (head en tail) corpus. Deze zinnen bevatten respectievelijk 1243 en 1474 woorden. De perplexiteiten vinden we in tabel 3.7. Bij het woordgebaseerde taalmodel traden bij het head gedeelte 40 OOVs op, en bij het tail gedeelte 50. Bij het syllabische taalmodel trad bij het head gedeelte 1 OOV op, bij het tail gedeelte traden 3 OOVs op. Vermits de perplexiteit van het 5-gram taalmodel te laag ligt, gaan we er van uit dat het 4-gram 10K syllabisch taalmodel qua perplexiteit het best overeenkomt met het 3-gram 80K woordgebaseerd taalmodel. In tabel 3.8 staan de perplexiteiten die werden bekomen voor de syllabische taalmodellen. De formule die we gebruikten voor het berekenen van de perplexiteit zoals gebruikt door de SRI-LM toolkit is als volgt:

$$ppl = 10^{\frac{-\sum_{w_i \in T} logprob(w_i)}{\#woorden - \#OOVs + \#zinnen}}$$

taalmodel	deel	som der logprobabiliteiten	perplexiteit
3-gram (80K woorden)	head	-2242.13	52.57
3-gram (80K woorden)	tail	-2627.46	52.97

Tabel 3.7: Taalmodel perplexiteiten voor een 3-gram woordgebaseerd taalmodel

taalmodel	deel	som der logprobabiliteiten	perplexiteit
3-gram	head	-3311.38	293.42
3-gram	tail	-3889.92	299.28
4-gram	head	-2453.83	67.37
4-gram	tail	-2885.95	68.71
5-gram	head	-1839.67	23.49
5-gram	tail	-2089.18	21.37

Tabel 3.8: Taalmodel perplexiteiten voor een 3-gram, 4-gram en 5-gram syllabisch taalmodel gemaakt met make-big-lm

3.3 Grootte taalmodellen

Het aantal N-grammen in een 80K woordgebaseerd en een 10K syllabisch taalmodel wordt gegeven in 3.9, de absolute grootte van de taalmodellen wordt gegeven in 3.10. Een 3-gram syllabische taalmodel is dus kleiner dan een woordgebaseerd taalmodel van dezelfde orde.

lexicon	unigrammen	bigrammen	trigrammen	viergrammen	vijfgrammen
80K woorden	80272	$1.12 \cdot 10^7$	$4.28 \cdot 10^7$		
10K syllaben	9619	$2.75 \cdot 10^6$	$2.27 \cdot 10^7$	$6.21 \cdot 10^7$	$1.00 \cdot 10^8$

Tabel 3.9: Groottes van de 3,4,5-gram 10K syllabische taalmodellen

taalmodel	aantal eenheden	absolute grootte taalmodel
3-gram	80K woorden	221.1 MB
3-gram	10K syllaben	97.5 MB
4-gram	10K syllaben	374.6 MB

Tabel 3.10: Absolute grootte van de taalmodellen

Hoofdstuk 4

Spraakherkenning met behulp van fonetische syllaben

In dit hoofdstuk zal ik de technische implementatie bespreken van de spraakherkenner die gebruikmaakt van syllaben als intermediaire eenheden. Hierbij baseer ik mij op de spraakherkenner van de K.U.Leuven die de Speech Processing, Recognition Automatic Annotation Kit (SPRAAK) wordt genoemd.

4.1 Experimentele set-up

In de volgende twee subsecties gaan ik in op twee aspecten van de gebruikte set-up: de testdata, en de herkenner zelf.

4.1.1 De N-Best testdata

Het doel van het N-best project [21] was een infrastructuur op te zetten voor benchmark evaluatie van continue spraakherkenners voor de Nederlandse taal, en onderzoekers te laten participeren in het evalueren van nieuwe spraakherkenners. Spraakherkenners kan men hiermee evalueren op nieuwsuitzendingen (Broadcast News, afgekort tot BN) en telefoonconversaties (Conversational Telephone Speech, afgekort tot CTS). De benchmark maakt

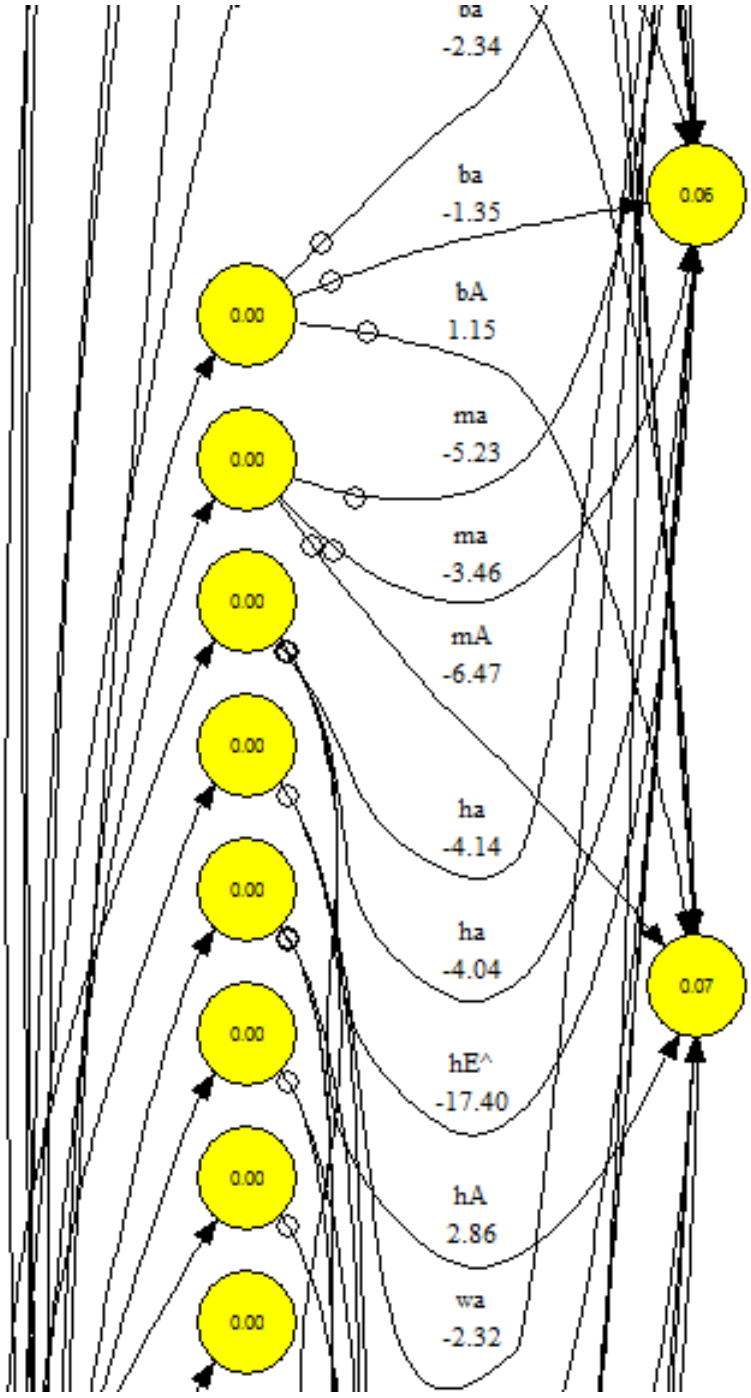
ook onderscheid tussen Nederlands (Northern Dutch) en Vlaams (Southern Dutch), vermits de uitspraak in Nederland verschillend is tegenover België. Voor de experimenten maakte ik gebruik van de Southern Dutch development test set voor Vlaamse nieuwsuitzendingen en telefoonconversaties.

4.1.2 SPRAAK herkenner met lattice herscoring

Van Kris Demuynck van de K.U.Leuven ontving ik een SPRAAK setup zoals beschreven in [8] met een aantal scripts waarmee het mogelijk is een lattice te genereren en lattice herscoring zoals beschreven in de literatuur [22] in de praktijk toe te passen. In een eerste fase wordt hierbij een lattice van lexicale eenheden gegenereerd met een beperkt lexicon waarbij het mogelijk is dat er onduidelijke stukken voorkomen, en daarna wordt deze lattice herscoord met een taalmodel met een groter lexicon, waarbij de onduidelijke stukken worden verbeterd.

Het spraakherkenningssysteem is in staat om aan sprekersegmentatie en clustering te doen, hoewel dit slechts een kleine verbetering geeft in de WER. Hierbij wordt bij de spraakherkenning onderscheid gemaakt of het een mannenstem, een vrouwenstem of een stilte betreft. De spraakherkenner maakt gebruik van trifoon Hidden Markov Modellen als akoestische modellen voor het herkennen van de YAPA foneemset. De akoestisch modellen bleven onaangepast en worden in meer detail besproken in [8].

Het aanmaken van de lattice gebeurde door middel van het script *GOdev*, het herscoren van de lattice gebeurt met het script *GOdev_wlat*. De lattice zelf wordt in een tekstbestand opgeslaan door middel van de letters O (Open Node), C (Close Node), A (Arc), D (Dummy Arc) gevolgd door getallen. SPRAAK bevat een script waarmee aan de hand van de lattice (*.LAT bestand) en het lexicon (*.DIC bestand) de lattice visueel kan worden voorgesteld met GraphViz genaamd *spr_wlat2dot.py*. Zo een bestand is gemakkelijker leesbaar doordat de indices die verwijzen naar het lexicon werden vervangen door de woorden zelf. In figuur 4.1 is een stuk van een lattice met syllaben als lexicale eenheden te zien.



Figuur 4.1: Een deel van een digraaf die een lattice met syllaben als lexicale eenheden voorstelt

4.2 Invloed van de lexicongrootte op de OOV-rate en de WER

In een eerste experiment werd de invloed van de lexicongrootte op de OOV-rate en de WER van een continue spraakherkenner onderzocht die gebruikt maakt van woordgebaseerde taalmodellen gebaseerd op een lexicon die bestaat uit de 20K, 40K en 80K meest frequente woorden van de corpustekst van De Standaard. Tabel 4.2 geeft het aantal woordcombinaties voor een 20K, 40K en 80K lexicon dat in het taalmodel gebaseerd op het corpus van De Standaard voorkomt. In tabel 4.1 is effectief te zien dat wanneer de OOV-rate verlaagt, de WER ook verlaagt. De OOV-rate speelt dus een belangrijke rol bij het verbeteren van spraakherkenners.

lexicongrootte	OOV-rate	WER (Substituties, Deleties, Inserties)
20K lexicon	6.37 % (726 / 10411)	29.4 % (19.8%, 3.7%, 5.9%)
40K lexicon	4.47 % (465 / 10411)	25.4 % (17.4%, 3.9%, 4.1%)
80K lexicon	2.64 % (275 / 10411)	22.1 % (15.2%, 3.5%, 3.4%)

Tabel 4.1: Invloed van de lexicongrootte op de word error rate

lexicongrootte	unigrammen	bigrammen	trigrammen
20K lexicon	20045	6574049	32342804
40K lexicon	40010	8937110	38264144
80K lexicon	80272	11228771	42791560

Tabel 4.2: Groottes van de 3-gram woordgebaseerde taalmodellen

4.3 Woordtaalmodellen versus syllabische taalmodellen: test set perplexiteit

Ik berekende de perplexiteit van een syllabisch taalmodel op basis van de 10K meest frequente syllaben van het De Standaard corpus op de N-best development (71 zinnen, 10411 woorden, 18955 syllaben, 1.81 syllaben per woord) testset. Het aantal syllabe OOVs is 10. De resultaten staan in tabel 4.3. Taalkundig gezien is de syllabische herkenner niet in het voordeel, de testsetperplexiteit ligt hoger: het is logisch dat een ruimere context nodig is om een volgende syllabe beter te kunnen voorspellen, maar dan nog zal een woordgebaseerd taalmodel met hogere waarschijnlijkheid een volledig woord correct kunnen voorspellen dan het voorspellen van slechts één enkele syllabe bij een syllabisch taalmodel. We zien bij de syllabische taalmodellen ook dat het 5-gram taalmodel wegens overtraining slechter presteert dan het 4-gram taalmodel.

taalmodel	perplexiteit
3-gram 80K words	285.87
3-gram 10K syllaben	601.08
4-gram 10K syllaben	437.18
5-gram 10K syllaben	449.89

Tabel 4.3: Perplexiteiten 10K syllabisch taalmodel

4.4 Gelaagde spraakherkenning met een syllabische tussenstap

In de volgende subsecties geef ik de resultaten van de experimenten die werden gedaan met syllabeherkenners.

4.4.1 Syllabeherkenning

Door het aantal deleties en inserties in balans te brengen door middel van de word insertion penalty, kan men de syllabe error rate verlagen. Op deze manier werd ook het 4-gram syllabisch taalmodel geoptimaliseerd 4.5. Deze optimalisatie werd ook doorgevoerd op het 3-gram woordgebaseerd taalmodel zodat de syllabeherkenner niet bevooroordeeld is 4.4. In het referentiebestand werden de woordgrenzen tussen de syllaben vastgelegd. We zien slechtere resultaten bij het 4-gram taalmodel doordat slechts gedeeltelijke oplossingen worden gevonden. Het beam search algoritme termineert ook niet altijd. Door de kleinere absolute grootte van het taalmodel en de betere syllabe error rate valt het 3-gram syllabisch taalmodel dus te verkiezen.

taalmodel	WIP 8kHz	WIP 16kHz	Correct	Substituties	Deleties	Inserties	WER
3-gram	-4	-3	78.9 %	16.2 %	4.8 %	0.5 %	21.5 %
3-gram	-2	-1	80.3 %	15.7 %	4.1 %	0.6 %	20.3 %

Tabel 4.4: Invloed van de word insertion penalty bij het 3-gram taalmodel

taalmodel	WIP 8kHz	WIP 16kHz	Correct	Substituties	Deleties	Inserties	WER
4-gram	-4	-3	73.3 %	12.3 %	14.4 %	0.4 %	27.1 %
4-gram	-2	-1	78.6 %	12.9 %	8.5 %	0.5 %	21.9 %
4-gram	-1	-0.5	78.9 %	12.8 %	8.3 %	0.6 %	21.7 %

Tabel 4.5: Invloed van de word insertion penalty bij het 4-gram taalmodel

4.4.2 Van syllabe- naar woordsequenties

Ik wou de lattice van syllaben herscoren met een groter woordgebaseerd taalmodel om de lattice van syllaben om te zetten in sequenties van woorden. Van Kris Demuynck van de K.U.Leuven ontving ik hiervoor een aantal scripts om aan de hand van de bekomen lattice van syllaben een lattice van woorden te bekomen, gebruikmakend van broncode die in april

2010 aan de SPRAAK v0.9.326 werd toegevoegd en gebaseerd is op FLaVoR. Het gebruik van de scripts zelf was niet triviaal. De auteur omschrijft de broncode als volgt:

> Beam search, i.e. quasi time synchronous search with pruning, through the word/phone lattice. The bsearch flavor variant allows an extra lexicon layer in between to convert phone sequences to words.

De scripts maakten gebruik van een woordenboek met fonemen, een woordenboek met woorden als sequenties van fonemen en een confusion matrix om aan de hand van een aangepaste beam search procedure van SPRAAK een sequentie van woorden te bekomen. Er is ook functionaliteit voorzien voor het evalueren van het resultaat (*.RES bestand) ten opzichte van een corpus bestand (*.COR), om zo de WER te berekenen. Aan de hand van het script *GO MAKE LEX FST* werd eerst een eindige toestandsautomaat gemaakt die dan door de SPRAAK herkenner werd gebruikt voor de omzetting van de lattice. Voor het omzetten van de lattice werd gebruik gemaakt van het MIT FST programma [23] die de automaat opslaat in *.FST formaat. Als invoer aan *GO MAKE LEX FST* gaf ik de 80K woordenlijst en de 10K syllabenlijst in UTF-8 formaat. Aan het eind van iedere syllabe werd een /:/ toegevoegd opdat de sequentie op een unieke manier zou te parsen zijn. De invoer moet in UTF-8 formaat staan omdat anders in het *.FST bestand bijvoorbeeld 'Comit' komt in plaats van 'Comité-P'.

Het begin van de woordenlijst ziet er als volgt uit:

```
.key
NENTRY  80270
###
[ ] nodes ROOT
[ ] add_words ROOT [] ROOT
</s>    </s>
's      s:
-index  In:dEks:
```

```
-jarig  ja:r@x:
-jarige ja:r@:G@:
-jarigen        ja:r@:G@n:
-plussers       plY:s@rs:
-tal    tAl:
-urige  y:r@:G@:
<UNK>   YNk:
A       a:
A'er    a:@r:
```

Het begin van de syllabenlijst ziet er zo uit:

```
.key
DICTIONARY      Syllable dictionary
NENTRY  9618
LANGUAGE        Dutch

SPELLING        SEGMENTS
########################
&:      &
&fp:    &fp
&ks:    &ks
&p:     &p
&s:     &s
&t:     &t
&y:     &y
@:      @
@l:     @l
@ls:    @ls
@lt:    @lt
```

```
@m:     @m
@n:     @n
```

Het aanmaken van de eindige toestandsautomaat gaf geen foutmeldingen. Het bekomen *.FST bestand werd via het programma iconv omgezet van UTF8 naar LATIN1 formaat. De aangepaste beam search procedure van SPRAAK maakte dan gebruik van FlaVoR [24]. De omzetting van de lattice van syllaben in een sequentie van woorden werd gestart met het script *GO BSEARCH FLAVOR SPRAAK* . Het systeem werd uitgetest zonder gebruik te maken van een confusion matrix. Ik kreeg de melding dat bij de FlaVoR variant van de word lattice beam search (wlbs) geen eindhypothese werd gevonden:

```
$ ./GO_BSEARCH_FLAVOR_SPRAAK
...
ERROR main_thread.spr_wlbs_flavor_process().0 : no end hypothesis found
...
```

In de source code van SPRAAK vond ik het C bronbestand die de foutmelding veroorzaakt terug onder src/lib/cwr/wlat/wlat_bsearch_.c. Ik besloot om een 32-bit debug versie van SPRAAK te compileren om uit te zoeken waar het misgaat. In het bestand config.py paste ik de optionele C compiler vlaggen aan en ik voegde voor iedere debug versie de switch -DFULL_DEBUG toe, waarna ik probeerde een 32-bit debug versie te compileren in een aparte directory onder de build directory met het commando *scons CONFIG=debug*. In het SPRAAK wlat bsearch *.C bronbestand moesten nog een aantal aanpassingen worden gemaakt. We zien nu inderdaad meer informatie en foutmeldingen van de vorm:

```
...
ERROR main_thread.wlbs_flavor_check().0 :
incorrect hi_score in nwt_list @ el 0x92d7ff0
(hi_score=-1.84467e+19, should be -7.3787e+19)
...
```

Het is echter onduidelijk hoe dit probleem kan worden opgelost.

4.4.3 Alternatieve resultaten

Omdat het systeem met 2 herkenners niet werkte, ben ik op zoek gegaan naar een aantal resultaten die toch iets interessants zouden kunnen zeggen over het verschil tussen syllabegebaseerde en woordgebaseerde spraakherkenning. Aan de hand van een script genaamd *getPercentageCorrectWords* was het mogelijk het resultaat van de spraakherkenner te evalueren en het aantal volledig correct herkende woorden in de top-1 hypothese bij syllabische herkenning te bekomen. Dit gebeurt aan de hand van een referentiebestand met woordgrenzen. Dit geeft de resultaten in tabel 4.6. Door het aanpassen van de woordinsertiekost kan men de WER beinvloeden, als woordinsertiekost -3 en -4 en taalmodelfactor 3.5 en 3 voor nieuwsuitzendingen en telefoonconversaties respectievelijk bleek optimaal te zijn. Hoewel het beste systeem 77.1 % woorden volledig correct herkent in tegenstelling tot de 81.3 % correct herkende woorden bij het woordgebaseerd systeem, geeft dit resultaat wel aan dat het herscoren van een 3-gram syllabisch taalmodel, waarvan men weet dat het taalmodel in absolute grote kleiner is, tot zinvolle resultaten kan leiden. Het aanpassen van de zoekbreedte kan ervoor zorgen dat de eerste fase van het genereren van een lattice van syllaben snel kan gebeuren waarna het resultaat door herscoring kan worden verbeterd.

taalmodel	WIP BN (16kHz)	WIP CTS (8kHz)	Correct herkende woorden
3-gram	-3	-4	75.9 %
3-gram	-1	-2	77.1 %
4-gram	-3	-4	71.0 %

Tabel 4.6: Het aantal volledig correct herkende woorden

Ik vergeleek ook wat er gebeurt in de OOV-zones bij de N-Best development testset. Er treden vooral OOVs op bij eigennamen. De derde zin van de N-Best development testset bevat bijvoorbeeld de bedrijfsnaam Belgavia die niet in het lexicon voorkomt. De woordgebaseerde herkenner splitst Belgavia in Belga en via. De syllabische herkenner geeft als beste hypothese: b@l-Ga-vi-a. De vierenvijftigste zin bevat de plaatsnamen Proven, Vlamertinge en Roesbrugge-Haringe. De woordgebaseerde herkenner gaat duidelijk de mist

in en gaf als beste hypothesen grove, Flamend Inge, Rous Brugge - haar enge. Terwijl de syllabische herkenner geeft: pro-v@, vla-m@r-tIN-@, u-rus-brY-G@-ha-rIN-@.

We zien dat de syllabische herkenner beter OOV woorden kan herkennen dan de woordgebaseerde herkenner, doordat de syllaben waaruit de woorden bestaan foutloos worden herkend. De eerste letter van Proven wordt door de syllabische herkenner correct herkend als een p in plaats van een g. Wanneer een lattice wordt gegenereerd van syllaben waarin meerdere hypothesen worden onderzocht verwacht ik dat mits herscoren met een groter lexicon, de herkenningsresultaten beduidend beter zullen zijn dan van een woordgebaseerde herkenner.

4.4.4 Invloed van de lexicale eenheden op de uitvoeringssnelheid

De invloed van de keuze van sub-woord eenheden op de grootte van het taalmodel en de uitvoeringssnelheid van de herkenner werd berekend en is weergegeven in tabel 4.7. De syllabische herkenner is trager wegens de hogere perplexiteit van de syllabische taalmodellen en door de kortere lexicale eenheden is er ook meer kans op verwarring tijdens de spraakherkenning.

We gaan ervan uit dat de snelheid van de syllabische herkenner kan worden verbeterd door het aanpassen van de zoekbreedte van de beam search, waardoor de eerste fase van de syllabische herkenning sneller zou moeten kunnen verlopen, ten nadele van de nauwkeurigheid van de herkenning.

taalmodel	aantal eenheden	uitvoeringssnelheid
3-gram	80K woorden	2:31:46
3-gram	10K syllaben	3:21:34
4-gram	10K syllaben	3:13:04

Tabel 4.7: Invloed van de keuze van het taalmodel op de uitvoeringssnelheid

Hoofdstuk 5

Conclusies en verder onderzoek

De belangrijkste conclusies van dit thesiswerk zijn dat de taalmodellen die voor syllabische herkenning worden gebruikt een hogere test set perplexiteit vertonen. Dit heeft tot gevolg dat een syllabische herkenner minder snel werkt.

Wanneer een te herkennen woord zich niet in het lexicon bevindt zien we dat een woordgebaseerde herkenner een foutief woord zal kiezen uit het lexicon dat gelijkaardig is aan het gezochte woord. De syllabische herkenner is echter wel in staat om de woorden syllabisch gezien volledig correct te herkennen zonder dat het gezochte woord bekend hoefde te zijn bij het maken van het lexicon van syllaben.

Wanneer men dit combineert met herscoring met een groter taalmodel dan kan reeds een goede selectie van syllabehypothesen worden gemaakt vooraleer een groter taalmodel wordt aangewend waardoor men dan sneller kan zoeken.

De syllabische herkenner met herscoring met een woordtaalmodel zal in de eerste fase niet sneller zal zijn dan een woordgebaseerd taalmodel, tenzij men de zoekbreedte van de beam search procedure terugschroeft. De herkenningsexperimenten geven echter aan dat het systeem wel tot betere herkenning zou kunnen leiden.

Jammer genoeg is het niet gelukt de lattice herscoring van de lattice van fonetische syllaben door middel van een woordgebaseerd taalmodel in de praktijk werkend te krijgen. Verdere uitwerking van de tweede fase spraakherkenner bleek niet triviaal, waardoor het

nog niet mogelijk is geweest eindhypothesen na herscoring te bekomen.

Wanneer de tweede fase herkenner zou werken, dan zit deze zo in elkaar dat men er ook de verwarring tussen de syllaben in een matrix kan aan meegeven. De herkenning zou dus kunnen worden verbeterd door de matrix automatisch aan te passen naargelang de resultaten die men uit de tweede fase herkenner bekomt.

Bijlage A

De CGN- en YAPA-foneemset voor het Nederlands

In de volgende tabellen staan de CGN- en YAPA-foneemsymbolen voor de verschillende fonemen uit het Nederlands. Per foneem wordt een voorbeeldwoord gegeven waarin het foneem optreedt.

Naast de symbolische representatie van de echte fonemen, bestaan in de foneemalfabetten soms ook symbolen voor andere klanken. In het YAPA-alfabet wordt zo bijvoorbeeld een symbool voorzien voor stilte (#), voor lachen of kuchen (%) en voor onverstaanbare spraak (∗).

Voorbeeld	CGN	YAPA	Voorbeeld	CGN	YAPA	Voorbeeld	CGN	YAPA
*p*et	p	p	*f*link	f	f	*m*es	m	m
*b*al	b	b	*v*oelen	v	v	*n*acht	n	n
*t*est	t	t	*s*inds	s	s	tweeli*ng*	N	N
*d*us	d	d	*z*eel	z	z	plu*nj*e	J	J
*k*anarie	k	k	*sj*aal	S	S	*l*eeft	l	l
za*k*doek	g	g	gara*g*e	Z	Z	*r*ol	r	r
			a*ch*ter	x	x	*j*aagt	j	j
			a*ch*ter	x	x	*w*agen	w	w
			ze*g*en	G	G			
			*h*uren	h	h			

Tabel A.1: De medeklinkers: occlusieven, fricatieven en overige.

Voorbeeld	CGN	YAPA	Voorbeeld	CGN	YAPA	Voorbeeld	CGN	YAPA
b*ie*r	i	i	*aa*n	a	a	d*oe*n	u	u
h*i*k	I	I	b*uu*r	y	y	v*oo*r	o	o
v*ee*l	e	e	p*u*t	Y	Y	l*o*g	O	O
v*e*l	E	E	d*eu*r	2	&	l*a*t	A	A
			d*e*	@	@			

Tabel A.2: De klinkers: vooraan, in het midden en achteraan de mondholte gevormd.

Voorbeeld	CGN	YAPA
vacc*in*	E~	E~
c*on*ge	O~	O~
croiss*ant*	A~	A~
parf*um*	Y~	Y~
*ij*s	E+	E^
k*ou*s	A+	O^
h*ui*s	@^	@^
milit*air*	E:	E:
k*ou*s	O:	O:
h*ui*s	@:	@:
*oeu*vre	Y:	@: of Y:
r*oei*t	uj	uj
n*ooi*t	oj	oj
dr*aai*	aj	aj
d*u*w	yw	yw
n*ieuw*	iw	iw
eeuw	ew	ew

Tabel A.3: Diftongen en zeldzame fonemen.

Bibliografie

[1] F. Stouten, J. Duchateau, J.-P. Martens, P. Wambacq, "Coping with disfluencies in spontaneous speech recognition: acoustic detection and linguistic context manipulation", in *Speech Communication*, volume 48, 2006.

[2] J.-P. Martens, "Spraakverwerking", Universiteit Gent.

[3] International Phonetic Association, "Handbook of the international phonetic association, a guide to the use of the international phonetic alphabet", 1999.

[4] `http://lands.let.kun.nl/cgn/`.

[5] `http://lands.let.kun.nl/cgn/doc_Dutch/topics/version_1.0/annot/phonetics/fon_prot.pdf`.

[6] Q. Yang, J.-P. Martens, N. Konings, H. Van Den Heuvel, "Development of a phoneme-to-phoneme (P2P) converter to improve the grapheme-to-phoneme (G2P) conversion of names", in *Proceedings of LREC 2006*, p. 287-292, Italy, 2006.

[7] Center for Processing Speech and Images. `http://www.esat.kuleuven.be/psi/`.

[8] K. Demuynck, A. Puurula, D. Van Compernolle, P. Wambacq, "The ESAT 2008 system for N-Best Dutch speech recognition benchmark", in *Proceedings of the eleventh biannual IEEE workshop on Automatic Speech Recognition and Understanding (ASRU 2009), IEEE*, pp. 339 - 344, Italy, 2009.

[9] A. Stolcke, "SRILM - an extensible language modeling toolkit", in *Proceedings of the International Conference on Spoken Language Processing, Volume 2*, pp. 901-904, Denver, 2001.

[10] R. kneser, H. Ney, "Improved backing-off for m-gram language modeling", in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Volume 1*, pp. 181-184, 1995.

[11] S. M. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer", in *IEEE Transactions on Acoustics, Speech and Signal Processing*, pp. 400-401, March 1987.

[12] S. F. Chen, J. Goodman, "An empirical study of smoothing techniques for language modeling", in *Computer Speech & Language, Volume 13*, pp. 359-393, 1999.

[13] B. Réveil, J.-P. Martens, "Reducing speech recognition time and memory use by means of compound (de-)composition", in *Proceedings of the 19th Annual Workshop on Circuits, Systems and Signal Processing (ProRISC 2008)*, Netherlands, 2008.

[14] J. Kneissler, D. Klakow, "Speech recognition for huge vocabularies by using optimized sub-word units", in *Proceedings of EUROSPEECH 2001*, pp. 69-73, Aalborg, Denmark, 2001.

[15] M. Larson, "Sub-word-based language models for speech recognition: implications for spoken document retrieval", in *Proceedings Workshop on Language Modeling*, 2001.

[16] V. Siivola, T. Hirsimäki, M. Creutz, M. Kurimo,"Unlimited vocabulary speech recognition based on morphs discovered in an unsupervised manner", in *Proceedings of EUROSPEECH*, Geneva, Switzerland, September 2003.

[17] O.-W. Kwon, J. Park, "Korean large vocabulary continuous speech recognition with morpheme-based recognition units", in *Speech Communication, Volume 39*, 2003.

[18] T. Kazuya, O. Atsunori, I. Fumitada, "Estimating entropy of a language from optimal word insertion penalty", in *Lecture Notes in Computer Science, Volume 1214*, 1998.

[19] V.I. Levenshtein,"Binary codes capable of correcting deletions, insertions and reversals", in *Soviet Physics Doklady, Volume 10*, 1966.

[20] M.J. Hunt, "Figures of merit for assessing connected-word recognisers", in *SIOA, Volume 2*, pp. 127-131, Netherlands, 1989.

[21] D. Van Leeuwen, J. Kessens, "Evaluation plan for the North- and South-Dutch benchmark evaluation of speech recognition technology (N-Best 2008)", in *Proceedings of ISCA Interspeech*, pp. 1354-1457, 2008.

[22] E. Arsoy, M. Saraclar, "Lattice extension and rescoring based approaches for LVCSR of Turkish", in *IEEE Transactions on Audio, Speech, and Language Processing*, pp. 163-173, January 2009.

[23] L. Hetherington, "The MIT finite-state transducer toolkit for speech and language processing", in *Proceedings of the ICSLP*, Korea, 2004.

[24] K. Demuynck, T. Laureys, D. Van Compernolle, H. Van Hamme, "FLaVoR: a flexible architecture for LVCSR", in *Proceedings of EUROSPEECH*, pp. 1973-1976, Geneva, Switzerland, September 2003.

Lijst van figuren

Lijst van tabellen