Modelling Context in Automatic Speech Recognition

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus, prof. dr. ir. J. T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
woensdag 4 juni 2008 om 10.00 uur
door

Pascal WIGGERS
Informatica Ingenieur
geboren te Den Helder.

Dit proefschrift is goedgekeurd door de promotoren:

Prof. dr. H. Koppelaar

Toegevoegd promotor:

Dr. drs. L. J. M. Rothkrantz

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof. dr. H. Koppelaar,	Technische Universiteit Delft, promotor
Dr. drs. L. J. M. Rothkrantz,	Technische Universiteit Delft, toegevoegd promotor
Prof. dr. C. M. Jonker,	Technische Universiteit Delft
Prof. dr. ir. J. L. G. Dietz,	Technische Universiteit Delft
Prof. dr. ir. F. C. A. Groen,	Universiteit van Amsterdam
Prof. Ing. Václav Matoušek, CSc.,	University of West Bohemia, Plzeň
PD Dr.-Ing. habil. E. Nöth,	Friedrich-Alexander-Universität Erlangen-Nürnberg

Preface

This thesis is the result of five years of research (and two more years of 'finishing the thesis') but the ground work was already laid in the year 2000 when I took a master level course in speech recognition taught by Leon Rothkrantz. The combination of Leon's infectious enthusiasm and the surprising fact that probability theory is the driving force behind speech recognition fuelled my fascination for the subject.

I would like to take this opportunity to thank Leon, not only for introducing me to the world of speech recognition, but even more so for giving me the opportunity and the freedom to do my PhD-research on the vague subject of 'context in speech recognition' and for all the support he has given me ever since!

I would also like to thank my promotor Henk Koppelaar for investing time in me and for all the suggestions that helped improve earlier versions of this thesis.

Many other people contributed to this thesis as well. I am grateful to Elmar Nöth, who, despite a busy schedule, took the time to read my manuscript with German Gründlichkeit. His comments were enormously helpful. I expressively thank Rogier van Dalen for his invaluable comments on the ideas and concepts expressed in this text as well as on the text itself. I enjoyed the discussions we had about speech, language, the universe and everything else, in which along the way he taught me a thing or two about c++ and LaTeX.

Many thanks go to the present and former members of the Man-Machine Interaction section for the great atmosphere they created. In particular, I would like to mention Jacek Wojdel, who pushed me to write my first paper; Zhenke Yang, who found some errors in this text that I and all other reviewers had missed and who provided some good suggestions to improve readability; and Martijn de Jongh, who had the ungrateful task of deciphering my sometimes enigmatic code. I want to thank Catholijn Jonker for giving me a chance to continue working in this group and giving me the time to finish my thesis. Bart, Ruud and Toos, without you behind the scenes nothing would work.

Last but not least, I would like to thank Ilse Vegt, sharing our troubles of being a PhD-student always made them lighter, and Eline Hoorweg for her support during the vacation days and late nights that I spend on my research and wrestling with this text.

Contents

> Whatever else people do when they come together – whether they play, fight, make love, or make automobiles – they talk. We live in a world of language. We talk to our friends, our associates, our wives and husbands, our lovers, our teachers, our parents and in-laws. We talk to bus drivers and total strangers. We talk face to face and over the telephone, and everyone responds with more talk. Television and radio further swell this torrent of words. Hardly a moment of our waking lives is free from words, and even in our dreams we talk and are talked to. We also talk when there is no one to answer. Some of us talk aloud in our sleep. We talk to our pets and sometimes to ourselves.
>
> *An introduction to Language*, Victoria Fromkin and Robert Rodman

Chapter 1

Introduction

In which the need for automatic speech and language processing is motivated. The state of the art in speech recognition is discussed. It is argued that the required levels of performance are not yet reached, but that the use of context may help to overcome the current limitations. Based on this hypothesis research questions are formulated.

Imagine that at this very moment someone nearby would start talking in a loud voice. Not only would it be hard to ignore the sound and concentrate on this text, but it would be hard indeed not to understand what this person is saying.

Speech recognition and understanding comes so naturally to us that we do not have to think about it at all. The cognitive processes are very rapid and almost completely subconscious. It seems hard if not impossible to control them. That is why all of us at some point in our childhood resorted to putting our fingers in our ears and started shouting or singing in order not to hear our parents telling us dos and don'ts.

As illustrated so well in the excerpt of Fromkin and Rodman (1993) at the start of this chapter language and in particular speech fills our lives; it is at the heart of human communication. Despite novel communication channels, such as e-mail and instant messaging, speech is still the primary means of communication, as is made clear by the popularity of mobile phones once again.

It is only logical that machine interface designers in their quest for a natural man-machine interface have turned to automatic speech recognition and production. A speech interface has many advantages and applications. In addition to being natural, speech is efficient. Most of us can speak faster than we can type. Speech is rich in expressions and can be very economical. A single sentence such as 'Play the video called "2001: a space Odyssey" with Dutch subtitles' can accomplish what would otherwise have required navigating through a menu or at least pushing several buttons.

Speech recognition is particularly well-suited to so-called hands-busy, eyes-busy tasks in which one needs to control several things at the same time. For example, surgeons were among the first to adopt speech recognition, as it allows them to dictate their actions to a computer while operating, which removes the burden of having to write a complete report after the operation. Along the same lines, think of the possibility of voice-dialling or controlling a navigation system while driving a car or of interfaces to telephone services and to mobile devices that are too small to manipulate with a keyboard; a market that still continues to expand rapidly (Phillips 2006).

When it comes to access to data collections and computerised services the gap between the haves and have-nots is rapidly increasing in our society. Speech recognition can have a major impact for groups of people that cannot operate a computer. The first group that comes to mind are disabled people. Being able to dictate a text to a computer or to operate machines by voice would make them less dependent on others. But there is an even larger group that can benefit from speech recognition: those that cannot read or write (Reddy 2006), a group that, according to the United Nations, contains 20% of the worlds population. For other users it may provide an additional channel to enter data. The ability to choose an input modality one is comfortable with can improve user experience (Carlson *et al.* 2006).

Many other applications exist, including automatic captioning of television programs, which is obligatory in Japan for example, and information extraction from audio, such as transcribing and summarising meetings.

Speech recognition is clearly useful, but researchers from different areas such as psychology, linguistics, electrical engineering and computer science have worked on the subject for the last 30 years for other reasons as well. First of all there is the engineering challenge of letting computers recognise speech, a glove thrown by Hollywood and numerous science fiction authors over and over again. From a scientific point of view, developing computation models of speech recognition may help provide us with insights in human speech recognition.

1.1 State-of-the-art in speech recognition

Speech recognition is still seen as a futuristic novelty. Popular belief has it that 'speech recognition does not work'. Those that have tried it report spending more time correcting errors than dictating. Indeed, speech recognition is something for

Table 1.1 – *Typical word error rates for various types of speech (speaker independent) as reported in 2007 (Interspeech 2007).*

type of speech	lexicon size	word error rate	human error rate
Digit recognition	10	0.5%	0.009%
Read newspaper speech	20000	3%	
Read newspaper speech	64000	5%	1%
Broadcast news	64000	10% – 20%	
Conversational speech	64000	20% – 40%	4%

the persistent. Present-day systems have to go through an extensive training phase to adapt to the speaking habits of a user to become useful. However, when compared to the earliest attempts at automatic speech recognition in the 1950s steady progress has been made.

These early systems could only recognise individual speech sounds or digits spoken in isolation by a particular person. In the 1980's the vocabulary size went up to several thousands of words and it became possible to recognise connected sequences of words such as bank account numbers. In the 1990s considerable progress was made. Recognising texts read out loud became feasible. Vocabulary sizes went up in the ten thousands and systems could recognise speech of arbitrary speakers. By the start of the 21st century speech recognition has matured to the point that some of the applications sketched above are feasible. In fact, a major software company now includes a speech recogniser for dictation and program control in their popular office suite. The focus of speech recognition research has shifted to recognising real-life spontaneous speech such as telephone conversations.

Table 1.1 illustrates the state-of-the-art in speech recognition. It lists the performance ranges, in terms of words recognised incorrectly, of speech recognition systems for various types of speech. These numbers are indications rather than precise scores as results can vary greatly between individual systems and data sets. It is clear that speech recognition is far from perfect. Note that the recognition rate of 95% reported for read speech, although an impressive number by itself, means that on average one in twenty words is recognised incorrectly. For a dictation task that would not be an acceptable figure. Moreover, given that such error rates are typically obtained on clear audio recorded with high quality equipment in a silent laboratory this number should be interpreted as an upper bound on performance rather than as an average. In real-life situations audio quality is often poor, and worse yet, not constant. Additionally, in spontaneous conversations people do not pronounce as well as when they are reading a text and conversational speech contains many hesitations, restarts, background noises and words that are unknown to the system. Under such circumstances the recognition rates of speech recognisers quickly drop, as illustrated by the results for broadcast news and conversational speech reported in table 1.1.

If the error rates on read speech should be interpreted as an upper bound on performance of present-day speech recognisers, the error rates on conversational speech

should be seen as a lower bound, as perfect recognition of conversational speech is not always necessary. For example, in an information extraction or summarisation task missing a few words will not be much of a problem.

100% recognition is a very ambitious goal indeed. The last column of table 1.1 shows that humans do not recognise every word that is spoken either and that also for humans, conversational speech is much harder to recognise than carefully pronounced speech.

1.1.1 Speech recognition paradigms

Part of the progress in speech recognition should be attributed to Moore's law, but the great leaps forward were caused by paradigm shifts. From a knowledge-based approach that worked with a complex interplay of heuristics based on linguistic theories in the early systems over a knowledge poor, deterministic template-matching approach in the 1980's to a robust statistical data-driven approach in the early 1990's that quickly became and still is the dominating paradigm (Jelinek 1976; Young 1995; Juang and Rabiner 2005; Furui 2005).

In the statistical approach recognising speech is formulated as finding among all possible word sequences the most likely word sequence given a speech signal. The amount of linguistic knowledge in the model is limited. A typical recogniser consists of two parts: an acoustic model and a language model. The acoustic model maps acoustic features obtained through signal processing techniques to words using a lexicon that specifies how words are pronounced in terms of phonemes, the basic units of speech. The language model defines how words can be put together to sentences. Rather than making a hard decision on the correctness of a sentence it usually assigns a probability to the hypotheses generated by the acoustic model. The probability that a word is part of the sentence is usually estimated based on the immediately preceding words.

The power of this approach is its robustness and the ability to generalise. It can easily deal with small distortions in the speech signal, with voices it has never heard before and new, possibly ungrammatical, word sequences. The approach does not rely on linguistic theories but can learn model parameters automatically from sample data using algorithms that are build on a firm mathematical ground. It emphasises similarities in speech and language, using broad statistical patterns that, despite the immense flexibility of language and all the variation in speech, can be found and hiding the complexity of language in probabilities.

The strength of the statistical approach is also its weakness. By relying on statistical averages it can find patterns that hold most of the time, but it misses the exceptions to the rules. This is clearly illustrated by the fact that after more than 15 years of research on the same corpora of read newspaper speech the error rates are still an order of magnitude larger than human recognition on the same type of speech.

Statistical models not only have difficulty with the subtle effects in language, but despite their ability to generalise, they cannot deal with patterns that greatly

differ from the — by definition limited set of — patterns they are trained on. A recogniser developed on a corpus of read financial news will obtain very poor results on conversational speech. Its performance will even drop considerably when tested on read speech of a different nature such as children's stories. Much of the progress in recent years has come from carefully constructing specialised systems for particular speech corpora. For example in broadcast news recognition it is not uncommon to separate recognisers for the anchorman, business news, interviews and so on (Nguyen *et al.* 2002).

Given these limitations, recognising spontaneous speech may be a bridge too far for the statistical approach. There are so little restrictions in conversational speech, the speech is pronounced less careful and the vocabulary is much larger, which means that there are many more similar words to choose from, that the exceptions seem to be the rule.

1.2 The quest for the holy grail

Some have argued that problems of the statistical approach to speech recognition sketched above can simply be solved, following one of the motto's of the speech recognition community: 'there is no data like more data' (Moore 2003). Training models on a data set that would contain sufficient examples of all the peculiarities of conversational speech would do the trick. However, given the complexity of conversational speech this would have to be an enormous amount of data indeed and as argued above the same argument does not even seem to hold for the much simpler task of recognising read speech. Given that the distribution of language approximates a Zipfian distribution (Manning and Schütze 1999), i.e. there are a few high frequency words and an never-ending number of low frequency words, we abandon all hope of ever finding enough data.

A better approach to overcome the difficulties posed by spontaneous speech and lift speech recognition to a level of human-like performance or beyond might be to further investigate and correct the weaknesses of the models used. A good start in this direction is to look at erroneous output of a speech recogniser. Below the transcriptions of two speech recordings generated by the speech recogniser for Dutch described in (Wiggers 2001; Wiggers *et al.* 2002*a*) are shown (a) together with the correct transcriptions (b).

(1) a. *en de botssimulator neemt en op de plaats van de autobestuurder en.*
and the crash simulator takes and on the place of the car driver and.

b. *in de botssimulator neemt een pop de plaats van de autobestuurder in.*
in the crash simulator takes a dummy the place of the car driver.
'in the crash simulator a dummy takes the place of the car driver.'

(2) a. *het aantal personen dat aan de kleur kan deelnemen is gebonden aan de maximum.*

the number persons that in the colour can take part is bound to the maximum.

b. *het aantal personen dat aan de cursus kan deelnemen is gebonden aan een maximum.*
the number of persons that in the course can take part is bound to a maximum.
'the number of persons that can take part in the course is bound to a maximum.'

From the first sentence it is clear that the recogniser confuses the small function words *en* and *in*. Inspection of other transcriptions showed that this is a reappearing error. In fact, high-frequency short function words pose a problem for most recognisers. The standard inelegant solution is to provide special acoustic models for these words. The language model of the recogniser, that uses the two preceding words to estimate the probability that a word is part of the sentence, is not capable of solving these errors as the phrases *en de botssimulator* and *de autobestuurder en* are by themselves correct and plausible phrases. Actually, as will be discussed in chapter 3 it is unlikely that the model contains statistics for phrases including rare words such as *botssimulator* and *autobestuur*. In practice even simpler statistics, such as the word frequencies, will be used. However, from the syntax of the whole sentence it is easy to see that the recognition result is incorrect. Grammatical knowledge would also help the recogniser to identify the word sequence *en op* as an error, but by itself it would not help much in correcting the error. Using the semantic knowledge that dummies are often used in crash tests it is clear what the correct words should have been.

The transcription produced by the recogniser in example (2) is grammatically correct but semantically implausible. Based on semantic knowledge and the fact that the correct word should sound like /klø[…]/ we can take a pretty good guess at the correct word though. In this case a well-trained language model might have been able to do the job by recognising the phase *cursus kan deelnemen*. But this is more of a patch than a real solution, as this particular trigram would be of no use if the sentence would have contained the phrase:

(3) *. . . aan de cursus, die maandag wordt gegeven, kan deelnemen . . .*
. . . in the course, given on monday, take part . . .

In conclusion, what both of these examples suggest is that we can solve quite a few of the errors made by a speech recogniser by introducing syntactic and semantic knowledge into the model.

Note that even with this knowledge the transcriptions found by the recogniser cannot be rejected as completely impossible. In a proper context both phrases may be valid. In general, we may need that broader context in our decision making process. Especially in spontaneous speech short sentences containing pronouns that refer to persons and object mentioned before are often used. In example (4) the

semantic information that can help to recognise the word *lion* is the word *Africa* two sentences earlier.

(4) *I've been to Africa last month.*
The weather was beautiful.
I saw a lion there.

1.2.1 What makes speech recognition difficult?

The difficulty of speech recognition is one of ambiguity. There are many sources of variation and confusion. For example, when recognising conversational speech, the recogniser is confronted with different speakers, each of them with a unique physiology that determines voice quality, and each with his own dialect and speaking habits that determine pronunciation and choice of words. The pitch, loudness and speaking rate, among other things, may change with the moods and emotions of these speakers and if the speakers are aware of the speech recogniser they may start to overarticulate.

In continuous speech, individual phonemes are not pronounced separately, but overlap. A sound is altered by its neighbouring sounds and even by sounds farther away. Sounds may even completely disappear. Complete words may run together. These co-articulation effects are stronger if the speech is less formal.

A microphone will pick up other sounds than the voice of the speaker, such as background noises or background speech and reverberations. Some of these sounds can easily be identified as non-speech sounds and be filtered out, but others overlap with the speech signal. Different rooms have different reverberation characteristics and typically noise is not constant but changes over time, making the situation even more difficult. The distance to the microphone matters as does the quality of the microphone itself, as it may introduce distortion in the speech signal due to electrical noise, directional characteristics, echoes and dropouts.

On the language side the conversations to be recognised can be about anything, resulting in a very large set of possible words, containing many acoustically confusable words. In addition, spoken language is not always grammatically correct. It contains fragmented and unfinished sentences, hesitations, restarts and corrections as well as many filler words, such as *er* and *erm*.

As mentioned before, despite all these difficulties, successful applications of speech recognition already exist. The strategy employed by all of these applications is to restrict the variation that can occur. For example, as recognising a particular speaker is much easier than recognising speech from any speaker that can walk up to the system, such systems are typically tuned to the voice of one particular speaker. In addition, these systems often have a small task-specific vocabulary that is designed to make the acoustic confusability of the words as small as possible. Most of these systems will require a silent environment and recognise words in isolation or deal with read, or dictated speech that is much more articulated than spontaneous speech. For some applications not all words in the signal have to be transcribed.

Rather, particular content words have to be spotted; this makes the system much more robust.

But how to restrict variation in the case of conversational speech? First of all, we might note that while conversational speech can be about anything it is far from random. A conversation will be about a particular topic or at least show some coherence. Not all words in the vocabulary are equally likely at every point in time. Nevertheless, in a typical speech recogniser the vocabulary is a static entity, at all times all the words in it can be recognised, and the probabilities that a word or word combination occurs is always the same.

Furthermore, conversations take place in a particular context that influences the style of the conversation. For example, most of us speak in a different style when speaking on the phone to a relative than when engaging in a debate or when at a job interview.

Everyone is unique, but there are some characteristics that are very similar for members of particular groups. Such characteristics include dialect, gender and age. For example, the voice of a teenage girl will differ quite a lot from the voice of an elderly man. Because of the age difference it is likely that the vocabulary used by the two will also differ. And if we know that the girl grew up in a big city in the west of the Netherlands while the old man spend all of his life in a rural village in the south of the country, their pronunciation will also be different. These characteristics are so strong that a human listener would only need a few words from either speaker to accurately recognise them.

1.2.2 Human speech processing

The previous section reviewed strategies employed by successful speech recognisers to get some idea on how to deal with the general case of recognising spontaneous speech. We can turn to the best speech recogniser we know, human speech recognition, for more inspiration. Unfortunately, human speech recognition is far from being fully understood.

It is known that humans also have more difficulty processing speech when the size of the active vocabulary increases. Miller *et al.* (1951) found that as the number of words to choose from increases, the loudness of the speech signal has to increase relative to the noise for correct identification. In addition, it is easier to recognise words against background noise if they fit in the context (Bruce 1958) and it takes about twice as long to recognise a word out of context than in would within the context of a sentence (Lieberman 1963). So, it seems that context plays an important role in human speech recognition. Several findings in psycholinguistics support this hypothesis.

Probably the best-known result of psycholinguistics is the *phoneme-restoration effect* (Warren 1970), which shows that we do not have to hear all sounds in a word to identify the word. In an experiment participants listed to sentences in which a phoneme in one of the words is replaced by a cough or a beep. It turned out that participants do not notice the missing sound. Even when they were told a sound was

missing they could not correctly locate the cough in the speech, but rather would place it incorrectly at a word or phrase boundary (Harley 2001). For example, participants were presented with the set of sentences in (5) to (8) in which the last word was added to the same recording of the other words to make sure that there was nothing in the initial sentence that gave away the missing phoneme.

(5) *It was found that the *eel was on the orange.*

(6) *It was found that the *eel was on the axle.*

(7) *It was found that the *eel was on the shoe.*

(8) *It was found that the *eel was on the table.*

People would restore a phoneme that fitted the context given by the last word to obtain *peel*, *wheel*, *heel* and *meal* respectively (Warren and Warren 1970). Initially, these results were taken as evidence that the perception of speech is constrained by higher level information. Later research has weakened this claim and suggested that contextual information does not directly influence sound perception, but is used in a later stage.

In the *shadowing task* (Marslen-Wilson and Welsh 1978) participants are asked to repeat continuous speech they listen to as quickly as possible. The speech contains mispronunciations that the participant are not told about. About half of the time participants repeat the words that should have been in the speech rather than the mispronunciations. It was found that restorations are more likely to occur when the distortion is slight and the word is predictable from context. In a very constraining context participants would even restore mistakes if the distortions were very prominent.

On the other hand, people are very good at recognising clearly articulated speech that is improbable in the context. Apparently contextual information can be overridden by perceptual information. Shadowing experiments also showed that syntactic and semantic analysis in the human mind is incremental and starts before a clause has been heard completely (Harley 2001). In the *gating task*, participants are presented with more and more sounds of a word. After every increment, participants have to guess the word (Harley 2001; Grosjean 1980). This experiment allows to find the point in a word where it is identified by a listener. This isolation point is usually before the end of the word and often even before the point at which enough sound of the word have been heard to identify it uniquely (the uniqueness point). The gating task once again shows the importance of context in human speech processing. The participant in the experiment of Grosjean (1980) needed 333 ms on average to identify a word in isolation, but only 199 ms on average to identify a word in context. However, the experiment also showed that initially candidate words are generated that are not compatible with the context but do match the speech signal.

Studies of electrical activity in the brain have shown that for all words there is a peak of electrical activity about 400 ms after the onset of the stimulus. Van Petten *et al.* (1999) showed that this peak, called the *N400*, has a larger amplitude when a

word does not fit the context even when the the isolation point of the word has not been reached.

Based on studies as those cited above, several competing models of human speech recognition have been developed. The most important of which are based on the concept of activation, and all use both perceptual bottom-up information and top-down context information. The points of debate between proponents of different models is on how much context information is used and at which stage bottom-up and top-down information are combined. As these models can serve as an inspiration for models of automatic speech recognition we will now briefly discuss the concept of spreading activation and the two most important competing models of human speech recognition.

The assumption behind both models is that the human mind contains a lexicon of all words that we know. All knowledge we have about a word, e.g. the pronunciation of the word, it's meaning, related words and syntactic features are stored in the lexicon. With each word an activation level is associated. Initially, all activation levels are low or neutral.

When the initial sounds of a word are recognised the activation level of all words in the lexicon that start with these sounds will increase. If subsequent sounds in the speech signal also match a word, the activation level of the word will increase further. Eventually, the word with the highest activation level is picked as the correct word. Different versions of the activation process are used in different models. In some models information that does not match the word will lower the activation level in a word, in others the activation level of incorrect words is lowered when one of the words wins the competition. It is usually assumed that activation levels also decay with time and that words activate related words. The latter explains a process called priming. If we, for example, hear the word 'PhD thesis' it will take less time to recognise the word 'defence' than it would otherwise. The attractiveness of activation-based models is that they explain how we as humans can deal with speech signals despite the limitation of our auditory memory, that can only save acoustic details for a time span of about 200 ms (Rietveld and van Heuven 1997).

The TRACE model (McClelland and Elman 1986) is an interactive model based on neural networks. It consists of several levels. Starting with feature detector neurons at the lowest level that are connected to phoneme neurons on the next level that are in turn connected to words. The neurons at a level are interconnected, activation of a neuron can inhibit the activation of other neurons at that level. The information flow is bidirectional, it can flow from lower levels to higher levels and from higher levels to lower levels at all times.

The *cohort* model (Marslen-Wilson 1984; 1987) assumes that in the initial (prelexical) phase of word recognition all words are activated of which the initial sounds match the speech signal to some extend. After about 200 ms, the second phase, lexical selection starts, all activated words (the cohort) check whether the incoming signal still matches, if not they are removed from the cohort. Eventually, one word will be left, which will then be recognised. Typically, this will be the case when the uniqueness point of the word has been reached. All knowledge associated with the

word is now available. In the final stage, the postlexical verification and integration stage, it is checked whether the remaining speech sounds still match the predicted word and whether the word fits within the context of the sentence. If not, another candidate word has to be reactivated again. In the cohort model, only bottom-up information is used to activate the words that will form the initial cohort. Top-down information is only used in the second phase of the process to eliminate unlikely words. This prevents that listeners will hear what they expect to hear.

1.2.3 Context

The previous sections suggest that including more linguistic knowledge and more contextual knowledge into models of speech recognition might improve recognition. Excluding such information in models of speech recognition has been a deliberate choice; imposing only a limited set of soft restrictions on speech is what gives these models robustness. But by ignoring many of the regularities of language the task is made harder than it really is.

Of course the idea to use context is not new. Many speech processing systems that include a speech recogniser as a front-end do use linguistic information to correct errors in the transcriptions output by the speech recogniser (e.g. Seneff 1992; Gallwitz *et al.* 1998; Wahlster *et al.* 2001; Filisko and Seneff 2003). Such a serial structure is efficient but when it misses the correct solution in an early stage, it may not be capable of reconstructing it in a later stage.

In fact, context has always been included in speech recognisers, be it implicitly, as systems have always been developed for particular well-defined domains, in which the type of speech was known and the number of topics and the corresponding vocabulary limited.

Most commercial recognisers use separate models for male and female speakers. Some also include models for different age groups and speakers of dialects. Such information is obtained by having the user select the most suitable system by hand. But most of the time speakers of a dialect, children and elderly are dealt with by simply ignoring them. Corpora that are used to develop speech recognisers with usually focus on the speakers of the standard language between the age of 20 and 55.

The practice of developing speech recognisers for specific domains may be effective, but it is very labour-intensive and more importantly wastes resources, as for each of these domains new data sets have to be collected and annotated. Rather than building separate systems for say read speech and spontaneous speech using different data sets, it would be much better to use both sets in the construction of a single more general model that can benefit from the similarities in those set and which considers read speech as a more restricted version of conversational speech to deal with the differences. In the same way, using separate models for male and female voices loses information. These two groups share many features, for example coarticulation effects do not differ all that much between genders.

Including context in speech recognition might be the only way to achieve human-

like speech recognition. For example, a model that knows only about frequent word sequences will not be able to deal with phenomena such as irony and humour that often escape the standard patterns of language.

However, some care should be taken not to include too many constraints on language. The power of the current approach to speech recognition is that it does not need explicit models for all kinds of effects. As mentioned above, this gives the approach robustness in the face of grammatically incorrect sentence and noisy recordings. Including too many constraints can do more harm than good. Also note that many of the subtle effects in language are implicitly encoded in the statistics used in speech recognisers. That is why many attempts to improve speech recognition by including additional information about language have had little success. Rather than abandoning the statistical approach altogether, it is important to find out what context information to use and when to rely on statistics instead.

This is not a simple question and therefore we should not expect to find the answer right away. In fact, much of the research in speech recognition has focused on using additional knowledge of some sort in speech recognition. Work in this domain typically proceeds along the following lines: a feature that might be useful in speech recognition is identified, a probabilistic model that includes this feature is formulated and then most of the effort goes into deriving and implementing the algorithms needed to train the model and do inference with it. Although all of these models can be classified as probabilistic speech recognisers their complex algorithms are highly specialised and therefore difficult to integrate. To move speech recognition to the next level a new computation paradigm must be found that allows the integration of all sorts of contextual information and the interactions between those information sources to be modelled more naturally than the current framework. It should allow researchers to focus on model development and experimentation rather than on algorithm design. The need for such a framework is even larger as there is still a great gap between research communities that have sufficient knowledge of language to adequately identify the information that will be of use in speech recognition and those that have the knowledge of mathematics and computer science needed to design and implement new algorithms that can incorporate this knowledge.

1.3 Research questions

To be able to improve speech recognition one has to understand the current technology, in particular the boundaries of this technology.

1. *What is the state of the art in speech recognition?*

What exactly do we mean by context? In previous sections a sketchy overview of contextual knowledge that can be of use in speech recognition was given. To be able to choose a computational paradigm that can incorporate context, we need a more and more detailed description of possible knowledge sources.

2. *Which types of information might be of use in speech recognition?*

Once the types of knowledge that can be of use from a theoretical point of view have been identified, one should not rush into the development of model, but first investigate whether contextual information can be of value in practice.

3. *Does contextual knowledge have the potential to improve speech recognition?*

Using the knowledge of existing models and the types of information that an ideal model should be able to incorporate a new computation paradigm for speech and language processing can be chosen.

4. *Which computational paradigm is powerful enough to include the contextual knowledge?*

Existing models for speech recognition have many desirable features. Can these models be incorporated in the proposed paradigm?

5. *How do existing models fit in the new paradigm?*

As argued above a generic framework based on the computational paradigm should be available that allows for rapid model construction and experimentation. Such a framework should deal with the peculiarities of speech and language processing.

6. *How to construct a general purpose computational framework that allows one to experiment with knowledge-rich models for speech and language processing?*

To quote Herbert Simon: 'In the computer field, the moment of truth is a running program; all else is prophecy'. The same holds when proposing models. The feasibility of an approach is best shown by a successful experiment. The final aim of this research is to design and test new, context-rich models for speech recognition.

7. *Is the performance of language models that included contextual knowledge better than that of conventional models?*

1.4 Scope of the thesis

This thesis is about speech recognition, with a focus on language modelling, as this is the part of the speech recogniser in which contextual knowledge can be of particular use. However, many of the concepts discussed and ideas expressed in this thesis apply to other techniques in the broader domain of natural language processing as well. The framework and the language models developed in this work can directly be applied to handwriting recognition, machine translation and spelling correction. Other subfields of natural language processing such as parsing can benefit from context as much as speech recognition.

1.5 Structure of the thesis

The remainder of this thesis answers the questions formulated above. Chapters 2 and 3 review the state of the art in speech recognition. Chapter 2 gives an overview of the statistical approach to speech recognition. In chapter 3 the strengths and weaknesses of approaches to language modelling are discussed in detail. In particular, an overview of previous attempts to include additional context and linguistic knowledge in language models is given.

Which types of knowledge can be of use in speech recognition from a theoretical point of view is investigated in chapter 4. In this chapter we present our definition of context. Chapter 5 presents the results of a data analysis on a large corpus of spoken language that explores the influence of some of the contextual factors identified in chapter 4 and the relations between those factors.

In chapter 6 two speech recognition systems that we developed as part of this thesis work are presented. Both systems include contextual knowledge: the first at the acoustic level in the form of lip-reading information, the second includes domain knowledge in the language model. For both systems we experimented with several configurations that integrate contextual knowledge at different stages in the recognition process. The experiments prove that contextual knowledge can improve speech recognition.

In chapter 7 we argue that a new computational paradigm for speech recognition with contextual information is needed. Based on the results of chapters 4 and 5 the requirements for a computation framework for speech recognition are formulated in chapter 7. Several computational techniques are considered. It is claimed that dynamic Bayesian networks form a good starting point for a computational framework.

In chapter 8 we reformulate the language models of chapter 3 in terms of dynamic Bayesian networks and propose some novel, more advanced models that can incorporate the context information presented in chapters 4 and 5. Speech recognition is a computationally expensive task. The standard data structures and algorithms for inference in dynamic Bayesian networks are not very-well suited for the large state spaces that appear in speech recognition. To make speech recognition with Bayesian networks tractable we developed a number of data structures and algorithms that exploit the properties of models of speech and language. These are detailed in chapter 9. In chapter 10 we propose a novel Bayesian network language model that includes topic information. An unsupervised learning procedure for the model is introduced and experiments with the model are presented. Chapter 11 concludes this thesis.

Life's most important questions are, for the most part, nothing but probability problems.

Pierre-Simon Laplace (1749–1827)

Chapter 2

Speech Recognition

In which an overview of hidden Markov model based speech recognition is given. The advantages and limitations of this paradigm are discussed.

Speech recognition finds the most likely word sequence given an input signal. Let $\mathbf{O}$, which we will call the observation sequence, represent the speech signal, and let $\mathbf{W}$ be a sequence of words. Then, the most likely word sequence $\hat{\mathbf{W}}$ follows from:

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} \mathsf{P}(\mathbf{W}|\mathbf{O}). \tag{2.1}$$

Unfortunately, the conditional probability distribution in (2.1) is hard to compute directly[1] as, due to variation in speaker characteristics and environmental noise, almost every observation sequence will be unique. To make life easier the equation can be rewritten using Bayes' rule (Bayes 1763):

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} \frac{\mathsf{P}(\mathbf{O}|\mathbf{W})\mathsf{P}(\mathbf{W})}{\mathsf{P}(\mathbf{O})}. \tag{2.2}$$

On first sight, little progress is made, since we now have to estimate three probability distributions, each over an infinite number of states. In particular, the probability of

[1] Although neural networks for speech recognition do.

an observation sequence is still required. But note that as the observation sequence is a given, it will be the same for all word sequences, i.e. $\mathsf{P}(\mathbf{O})$ acts as a normalising constant in (2.2). Because we are only looking for the most likely sequence hypothesis and do not need the exact probabilities, it can be ignored. To find a solution to (2.1) we thus need to solve

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} \mathsf{P}(\mathbf{O}|\mathbf{W})\mathsf{P}(\mathbf{W}), \tag{2.3}$$

where $\mathsf{P}(\mathbf{W})$, the probability of a sequence of words, is called the language model and $\mathsf{P}(\mathbf{O}|\mathbf{W})$, the probability of the observation sequence given a particular sequence of words is the acoustic model. Language modelling is the topic of the next chapter. The remainder of this chapter will discuss acoustic modelling.

2.1 Signal processing

The input to a speech recogniser is an audio waveform picked up by a microphone. This signal contains much non-speech information, such as noise introduced by the environment and the recording hardware. Therefore, a preprocessing step is used to extract relevant features from the signal. In addition to removing noise, it also reduces some of the person and environment specific variations.

Many feature extraction methods have been developed, some based on acoustic concepts, others on knowledge of human speech production and perception. The most important techniques are Linear Predictive Coding and Mel Frequency Cepstral Analysis. As this thesis is not about signal processing, only a brief impression of these two techniques is given below. For more details see Ladefoged (1996) or Furui (2001).

Both methods convert the speech waveform into a sequence of real-valued feature vectors. A single vector is obtained by applying the signal processing technique of choice to a small segment of the signal, e.g. a 25 ms segment. This is done every 10 ms. The segments of sound of successive vectors overlap to make up for the discontinuity introduced by the discrete sampling of the signal.

Linear predictive coding Linear predictive coding (LPC) takes a mechanical view of human speech production, assuming that the speech signal is produced by a buzzer at the end of a tube. The glottis, the space between the vocal chords, produces the buzz, which is characterised by its intensity, which roughly corresponds to loudness, and frequency, which determines the pitch of the sound. The sound produced by the glottis will resonate in the tube formed by the vocal tract, i.e. the combination of throat, mouth and nose. As the wavelengths of the resonances are proportional to the length of the tube, they will change when the shape of the vocal tract is changed for example by movement of the tongue. These resonances, called formant frequencies in the case of speech, are important indications of the identity of a sound. Vowels can be characterised by the relative distance between the first and second formant frequency.

LPC estimates a filter that models the vocal tract, removes its effect from the speech signal and estimates the intensity and frequency of the remaining buzz, called the residue. The filter is a difference equation, called a linear predictor, which expresses each sample of the signal as a linear combination of previous samples. The coefficients of the linear predictor are estimated by minimising the mean-square error between the predicted signal and the actual signal. These coefficients characterise the filter and therefore the shape of the vocal tract. In speech recognition the first 12 coefficients are taken as a feature vector.

Mel-frequency cepstral coefficients One of the more common techniques of studying a speech signal is via the power spectrum. The power spectrum of a speech signal describes the frequency content of the signal over time. Typically, the peaks in a spectrum relate to the formant frequencies of a sound. The first step towards computing the power spectrum of the speech signal is to perform a Discrete Fourier Transform (DFT).

Psychophysical studies have shown that human perception of the sound frequencies does not follow a linear scale. This has led to the definition of the Mel scale that gives the subjective pitch of pure tones. As a reference point, the pitch of a 1 kHz tone, 40 dB above the perceptual hearing threshold, is defined as 1000 Mels. Other subjective pitch values are obtained by adjusting the frequency of a tone such that it is half or twice the perceived pitch of a reference tone with a known Mel frequency.

Feature extraction based on Mel frequency cepstral coefficients (MFCC) applies triangular windows at increasing distances according to the Mel scale to the power spectrum. The cepstral coefficients are computed by transforming the logarithm of the energy in each window to the cepstral domain using an inverse Discrete Fourier Transform. Cepstral coefficients can be seen as a parametric representation of the envelope of the spectrum, as such they correlate with the formant frequencies. Usually, first and second derivatives of the cepstral coefficients are taken and added to the speech vector to account for the continuous nature of the signal.

2.2 Acoustic modelling

The acoustic model gives the probability that an observation sequence of feature vectors corresponds to a sequence of words. As the number of different observation sequences is infinite, as is the number of different word sequences in realistic situations, a simple look up is not possible. Rather, a model that computes these probabilities on the fly is needed. The hidden Markov model does this.

2.2.1 Hidden Markov models

The *hidden Markov model* (HMM) is a powerful mathematical tool for modelling time series. It automatically performs dynamic time warping for signals that are locally squashed or stretched and can deal with small distortions in a signal. It

provides efficient algorithms for parameter estimation from data and has the ability to generalise to cases not in the training examples.

Hidden Markov models are a generalisation of the well-known Markov chains of probability theory that model a sequence of events in time. A Markov model consists of a set of states that correspond to events or observations. The model starts in one of the states and at every point in time makes a transition to another state according to a probability distribution. In first-order Markov models, the probability of moving from one state to another only depends on the current state:

$$P(X_{t+1} = q_i | X_1, X_2, \ldots, X_t) = P(X_{t+1} = q_i | X_t), \tag{2.4}$$

where X_t denotes a state at time t. In general, in k-th-order Markov models the probability of moving from one state to the next depends on the previous k states. In addition, transitions are independent of time:

$$P(X_{t+1} = q_i | X_t) = P(X_2 = q_i | X_1). \tag{2.5}$$

Figure 2.1 shows a first-order Markov chain that models different pronunciations for the word tomato. Transitions with non-zero probability are shown by arrows connecting two states. Different paths through the model correspond to the British and American pronunciations of the word. The self-loops allow the model to deal with the variation in duration of speech sounds. The transition from /t/ to /m/ allows the /ə/ sound to be skipped completely.

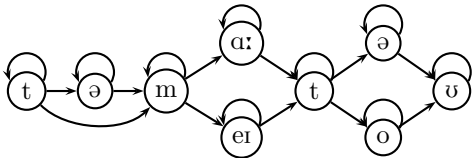


Figure 2.1 – *A pronunciation Markov model for the word tomato.*

In figure 2.1 a word is modelled as a sequence of phonemes, but the input signal in speech recognition is a sequence of real-valued vectors, several of which may correspond with a particular phoneme. The Markov model, in which each state corresponds to an observable event does not cover continuous observations.

The hidden Markov model extends the model by decoupling the observation sequence and the state sequence. For every state a probability distribution is defined that specifies how likely every observation symbol is to be generated in that particular state. Each state can in principle generate every observation symbol. Which state sequence generated an observation sequence becomes indistinguishable. The states are hidden. Formally, a hidden Markov model can be defined by the following parameters:

- The number of distinct observation symbols M.

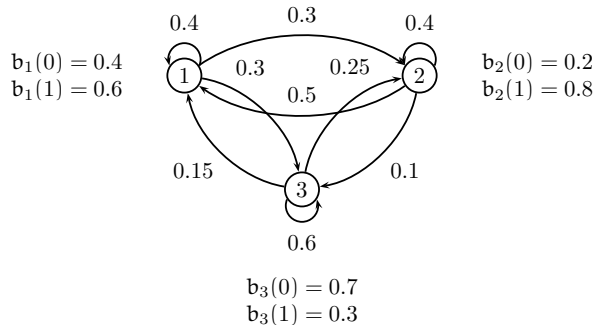


Figure 2.2 – *A 3 state hidden Markov model that generates sequences of 0's and 1's.*

- An output alphabet $L = \{l^1, l^2 \dots l^M\}$.
- The number of states N.
- A state space $Q = \{1, 2 \dots N\}$. States will usually be indicated by i, j. A state that the model is in at a particular point in time t will be indicated by q_t. Thus, $q_t = i$ means that the model is in state i at time t.
- A probability distribution of transitions between states, represented by the transition matrix $A = \{a_{ij}\}$, where $a_{ij} = P(q_{t+1} = j | q_t = i)$.
- An observation symbol probability distribution $B = \{b_j(k)\}$ in which $b_j(k) = P(o_t = l^k | q_t = j)$, where o_t is the observation at time t.
- The initial state distribution $\pi = \{\pi_i\}$ where $\pi_i = P(q_0 = i)$.

To indicate the complete parameter set of a model the notation $\lambda = (A, B, \pi)$ is used. Figure 2.2 shows a three state HMM with discrete probability distributions attached to each state.

2.2.2 HMM topology for speech recognition

Hidden Markov models can be used to model speech at various linguistic levels. Words are probably the most natural units to model. After all, we are interested in recognising words and as will be discussed in chapter 3 the language model also uses words as the basic unit. An additional advantage of word-level models is that they capture within-word coarticulation effects rather well. Actually, it is shown that because of these effects, the larger the unit, the better the recognition will be (Rabiner and Juang 1993). However, the parameters, i.e. the transition probabilities and output probabilities of HMMs are automatically set using a training corpus. To obtain reliable parameters a large number of examples for every unit is needed, e.g. words pronounced by men and women with different voice qualities at different speaking

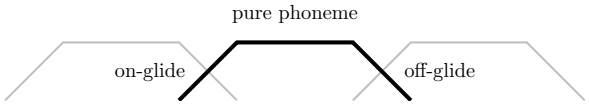


Figure 2.3 – *Three phases in the pronunciation of a phoneme.*

rates, within different sentences. Therefore, for large vocabulary speech recognition word units are not an option, but for small well-defined vocabularies, for example a set of commands, they are well suited. Usually, left-to-right model topologies are used in which the number of states depends on the number of phonemes in the word. One state per phoneme is a good rule of thumb.

If sub-word units, such as phoneme level models or syllable level models are used, data can be shared among words. As long as the sub-word units occur sufficiently often in different contexts in the training data, not all words in the dictionary have to be in the data. The vocabulary of a speech recogniser can easily be extended by specifying how a new word is expressed in terms of the sub-word units.

Phoneme models are used most often. As there are only 40 to 50 phonemes in languages like Dutch and English, HMM phoneme models can be adequately trained. Most topologies used in speech recognition are based on the assumption that there are three phases in the pronunciation of a phoneme. In the first phase the articulators are moving in position to pronounce the phoneme, this is called the on-glide of the phone. In this phase there may be some overlap with the preceding phone. In the second phase the sound of the phone is assumed to be pure and in the third, off-glide, phase the sound is released and the articulators start to move to the positions for the next phoneme. The process is schematically shown in figure 2.3. This suggests that at least three states should be used in a phoneme HMM. Adding more states means introducing more parameters and thus more degrees of freedom. Variations in a phoneme can be modelled more accurately but this also introduces a need for more training data to avoid undertraining. In addition, the paths through the model should not be too long. A five state model does not work for phonemes with a duration of only three time frames. There should always be a short-cut that can handle the shortest example in the training data. Figure 2.4 shows three model topologies that have successfully been used in various speech recognisers. The first model (2.4a) is a simple three state left-right model with state dependent output probabilities. The first and last smaller circles in the figure represent entry and exit null-states. They do not generate observations and are only used to concatenate models. The second model (2.4b) has five states, but provides transitions that skip the succeeding state. Therefore it is possible to pass through the model in only three steps. This model also has state dependent output probabilities.

The last model (2.4c), used by IBM (Jelinek 1999), has seven states and twelve transitions with transition dependent output probabilities. Only three different output distributions, corresponding with the three phases in a phoneme, are used. In

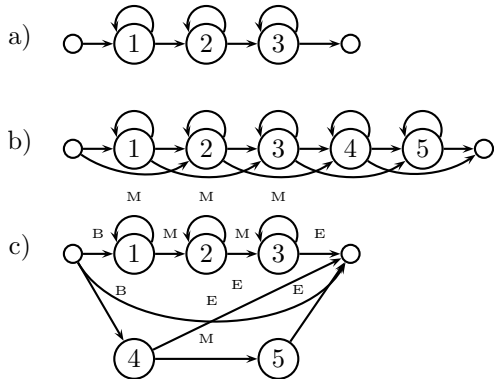


Figure 2.4 – *Model topologies for phoneme units.*

the figure the begin phase (on-glide) is marked with B the middle phase with M and the end phase (off-glide) with E.

Word models are obtained by concatenating the models corresponding to the phonemes that make up the word according to a pronunciation dictionary. As illustrated by figure 2.3 phonemes overlap in continuous speech. Depending on the position in the word and on the surrounding phonemes the sound of a phoneme may change completely. For example in American English a /t/ at the start of a word is aspirated but can be reduced to a tap between vowels (Jurafsky and Martin 2000). By treating phonemes as separate units that are inserted in a word based on phonetic transcriptions such coarticulation information is lost. Linguistics deals with such phenomena by differentiating between the abstract phonetic level and the phonemic level that specifies the exact pronunciation in context. In speech recognition a more pragmatic approach is taken. The simple phoneme models are replaced by context-dependent models, i.e. separate models are created for instances of a phoneme that occur in different contexts. The de facto standard in speech recognition is the triphone model, that includes the identities of the left and right neighbours of a phoneme. A disadvantage of triphone models is that they bring back the data-sparsity problem again. Western languages typically have more than ten thousand triphones. For some of these models there may be no examples or only a very few in the training data. Clustering of similar triphones offers a solution. This can either be done in a knowledge-based fashion using linguistic features of phonemes or in a data-driven manner. The models in a cluster share parameters. Often, individual triphone states rather than complete models are clustered.

2.2.3 Observation distributions

Because of differences between voices, prosodic variation occurring in speech and coarticulation effects, the acoustic realisation of a phoneme can vary. Therefore the

state-dependent output distributions are complex and different per phoneme. This is usually modelled using a mixture of normal distributions, named Gaussians in speech recognition lingo, defined by

$$b_j(o_t) = \sum_{m=1}^{M} c_{jm} \mathcal{N}(o_t, \mu_{jm}, \Sigma_{jm}), \tag{2.6}$$

where μ_{jm} and Σ_{jm} are the mean vector and covariance matrix of component m respectively and c_{jm} are positive valued mixture weights, the sum of which is 1. Gaussian mixtures can approximate any continuous probability density function in the sense of minimising the error between two density functions. The advantage of Gaussian mixtures is that the parameters can be learned from data using the standard HMM training algorithm.

2.2.4 The probability of an observation sequence

To perform recognition with a hidden Markov model, the probability of the observation sequence has to be calculated for every model as specified by (2.3). Every state in an HMM can generate every observation symbol, so there will be many paths through a model that correspond to an observation sequence. To find the total probability of the observation given the model the sum of the probabilities of individual paths has to be taken:

$$\begin{aligned} P(O|\lambda) &= \sum_{\forall q_1, q_2, \ldots, q_T} P(O, q_1, q_2, \ldots q_T|\lambda) \\ &= \sum_{\forall q_1, q_2, \ldots, q_T} \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) \ldots a_{q_{T-1} q_T} b_{q_T}(o_T). \end{aligned} \tag{2.7}$$

For an observation sequence of length T for each state sequence q about $2T$ calculations have to be performed. At each time step there are N different states that can generate a given observation. Therefore, there are N^T different state sequences that can generate the observation sequence. It follows that the time complexity of equation (2.7) is $O(2TN^T)$. For realistic values of N and T this quickly becomes infeasible. Therefore a more efficient procedure is needed. Fortunately, such a procedure exists; it is called the forward algorithm. The algorithm belongs to the class of dynamic programming algorithms. Instead of considering each state sequence in turn it calculates the values for all sub-sequences up to some time step in parallel, using the results from the previous time step. This is highly efficient as many paths share the same sub-paths. Essentially, the algorithm uses the Markov assumption underlying the model to 'shift sums under multiplications'. For example, for 3 time

steps:

$$\sum_{q_1,q_2,q_3} \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) a_{q_2 q_3} b_{q_3}(o_3) = \sum_{q_3} \left(\sum_{q_2} \left(\sum_{q_1} \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} \right) b_{q_2}(o_2) a_{q_2 q_3} \right) b_{q_3}(o_3). \quad (2.8)$$

This is expressed by a recursive relation. Let $\alpha_t(i)$ be the probability of being in state i at time t and having observed the partial observation sequence $o_1 o_2 \dots o_t$ so far, given the model λ:

$$\alpha_t(i) = P(o_1 o_2 \dots o_t, q_t = i|\lambda). \quad (2.9)$$

$\alpha_t(i)$ can be computed inductively, as follows:

Initialisation

$$\alpha_1(i) = \pi_i b_i(o_1), \quad (2.10)$$

Induction

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(o_{t+1}), \quad (2.11)$$

Termination

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \quad (2.12)$$

Figure 2.5 schematically shows the idea behind the forward algorithm. This figure is called a trellis. At each time step all model states are considered. All paths that end up in a particular state at a particular time are combined. As there are only N nodes at each time step all possible state sequences will remerge into these N nodes no matter how long the observation sequence. At time t each calculation only involves the N previous values of $\alpha_{t-1}(i)$, because each of the N grid points can be reached from only the N grid points at the previous time slot. So this procedure only requires on the order of TN^2 calculations, rather than $2TN^T$ as required by the direct calculation.

2.2.5 The most likely state sequence

The forward algorithm gives the probability that a sequence of observations is generated by a model λ. However, the individual states of a Markov model may have some meaning, for example the phonemes in a word, and we might be interested in the sequence of states that is most likely to have generated the observation sequence $O = o_1 o_2 ... o_T$. This boils down to maximising $P(\mathbf{q}|O, \lambda)$. As as the probability of

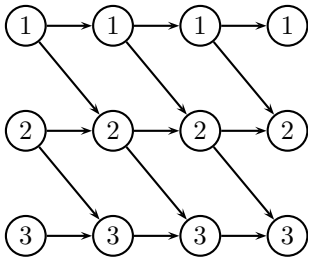


Figure 2.5 – *The forward algorithm visualised by a trellis for a 3 state left-to-right* HMM *with self loops.*

the observation sequence can be seen as a constant this is equivalent to maximising $P(q, O|\lambda)$:

$$\hat{\mathbf{q}} = \arg\max_{\mathbf{q}} P(\mathbf{q}|O, \lambda) = \max_{\mathbf{q}} \frac{P(\mathbf{q}, O|\lambda)}{P(O)} = \max_{\mathbf{q}} P(\mathbf{q}, O|\lambda), \tag{2.13}$$

where $\hat{q}$ is the most likely state sequence. Let $\delta_t(i)$ be the probability of the most likely path from the start into some state i at time t:

$$\delta_t(i) = \max_{q_1 q_2 \cdots q_{t-1}} P(q_1 q_2 \dots q_{t-1}, q_t = i, o_1 o_2 \dots o_t|\lambda), \tag{2.14}$$

then

$$\delta_T(i) = \max_{\mathbf{q}=q_1 q_2 \cdots q_T} P(\mathbf{q}, O|\lambda) \tag{2.15}$$

is the probability of the most likely state sequence for the observation sequence. $\delta_t(i)$ can be calculated using the same recursive procedure as in the forward algorithm, using maximisation over previous states instead of summation. This algorithm is known as the Viterbi algorithm. Formally, it is defined as:

Initialisation

$$\delta_1(i) = \pi_i b_i(o_1), \tag{2.16}$$

$$\psi_1(i) = 0, \tag{2.17}$$

Recursion

$$\delta_t(j) = \max_{1 \leqslant i \leqslant N} [\delta_{t-1}(i) a_{ij}]\, b_j(o_t), \tag{2.18}$$

$$\psi_t(j) = \arg\max_{1 \leqslant i \leqslant N} [\delta_{t-1}(i) a_{ij}]\,, \tag{2.19}$$

Termination

$$\hat{P} = \max_{1 \leqslant i \leqslant N} [\delta_T(i)], \tag{2.20}$$

$$\hat{q}_T = \arg \max_{1 \leqslant i \leqslant N} [\delta_T(i)], \tag{2.21}$$

Path backtracking

$$\hat{q}_t = \psi_{t+1}(\hat{q}_{t+1}), \tag{2.22}$$

where $\psi_t(i)$ keeps track of the state at time $t-1$ that the most likely path to state i at time t passes through. By backtracing those values we can find the most likely state sequence. The algorithm relies heavily on the Markov property of the underlying model. At each time step t it assumes that the most likely path into the current state will be part of the most likely path over the entire model through this state.

The Viterbi algorithm plays an important role in speech recognition. As mentioned before, word models are constructed by concatenating phoneme models. These word models are in turn connected as specified by a language model. The result is one large composite HMM with which sequences of words can be recognised. This is done by the Viterbi algorithm. The path found by the algorithm will lead through a sequence of words that specify the recognised word string. Actually, this method is not guaranteed to find the most likely word sequence, as the probability of a word sequence given an observation sequence should include all paths through that word sequence rather than just one. However, this is computationally expensive. The use of the cheaper Viterbi algorithm is justified by the fact that in practice it is very rare that the word string corresponding to the most likely path is not the word string corresponding to the most probable set of paths.

n-best Search The Viterbi algorithm finds the most likely single path through an HMM. As mentioned before this is not guaranteed to be the optimal solution, and even if it were it still does not have to be the right solution. Therefore, it is often desirable to find the N most likely word sequences given the observed acoustic data O, which can then be used as inputs to more refined models. The required n-best search differs from the Viterbi search in one aspect only: Instead of retaining only one path leading into each trellis state, each trellis state is split into n sub-states, one for each of the n most likely paths leading into the unsplit state from the split state of the trellis' previous stage. The n most likely paths may share structure. It is not uncommon for these paths to differ in a single word or only in timing aspects. A lattice is a more efficient representation that stores hypotheses in the form of a directed graph of words. Usually, acoustic and language model scores as well as the number of frames spend in a word or phoneme are added to the lattice.

Beam search For realistic vocabulary sizes the composite model and hence the search space will still be huge. Therefore, unlikely paths are typically pruned during

the search process. This is done by a beam search. Hypotheses that are below a certain probability level are pruned. At each trellis stage i the maximum probability of the states at stage $i-1$ defined by $P_{i-1}^{M} = \max \delta_{i-1}(q)$ serves as a basis for a dynamic threshold:

$$\tau_{i-1} = \frac{P_{i-1}^{m}}{K}, \tag{2.23}$$

where K is a suitable chosen constant. All states q' that satisfy $\delta_{i-1}(q') < \tau_{i-1}$ are removed from the search, i.e. $\delta_{i-1}(q')$ is set to zero.

2.2.6 Learning model parameters

In the discussion so far the parameters of the models, that is the transition probabilities and the observation probabilities, have been taken as a given. In practice, these values are unknown and we therefore would like to have a method that can automatically determine these parameters $\lambda = (A, B, \pi)$ in such a way that the model best matches the data it represents. Mathematically this means we want a method that has the following maximum likelihood property:

$$\hat{\lambda} = \arg\max_{\lambda} P(O|\lambda). \tag{2.24}$$

There is no known analytical solution for this equation, but an iterative procedure known as the Baum-Welch algorithm or Forward-Backward algorithm exists that chooses $\lambda = (A, B, \pi)$ such that its likelihood $P(O|\lambda)$ is locally maximised using a sample of the type of data the model is supposed to generate. Basically, the algorithm counts the average number of times a state is visited and the number of times a transition from a state i to another state j is made as well as the average number of times that a specific observation symbol is generated in a state. From these counts the transition and observation probabilities are derived. To calculate these counts we define a backward probability $\beta_t(i)$ to complement the forward probability $\alpha_t(i)$ from section 2.2.4:

$$\beta_t(i) = P(o_{t+1} o_{t+2} \dots o_T | q_t = i, \lambda). \tag{2.25}$$

Like the forward probability $\beta_t(i)$ can be computed recursively using a backwards pass over the trellis. The probability of being in state i at time t given the entire observation sequence and the model can be defined as:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)}. \tag{2.26}$$

Intuitively, this formula states that the probability of being in a state can be obtained by taking the probability of all paths that pass through this state. We find all paths through a state by combining all ingoing paths with every outgoing path, hence the multiplication of the forward and backward probabilities. The nominator in this formula, the probability of the observation sequence, can be interpreted as the

probability of all possible paths through the model for this observation sequence and follows from the forward algorithm. It ensures that the γ_t sum to one. Similarly, we can also define the probability of being in state i at time t and state j at time $t+1$, given the model and the observation sequence by:

$$\xi(i,j) = \frac{P(q_t = i, q_{t+1} = j, O|\lambda)}{P(O|\lambda)} = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{P(O|\lambda)}. \tag{2.27}$$

If $\gamma_t(i)$ is summed over the time index t, the expected number of times that state i is visited is obtained or equivalently the number of transitions made from state i. Similarly, summation of $\xi_t(i,j)$ over t can be interpreted as the expected number of transitions from state i to state j. Now, using the concept of counting event occurrences, we can estimate a_{ij} as the expected number of transitions from state i to state j normalised by the expected number of transitions from state i:

$$\tilde{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}. \tag{2.28}$$

Similarly, $b_j(k)$ can be estimated by dividing the expected number of times in state j at which symbol l^k was observed by the expected number of times the system is in state j:

$$\tilde{b}_j(k) = \frac{\sum_{t=1, o_t = l^k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}. \tag{2.29}$$

The initial state distribution for state j is equal to the expected frequency with which state j is visited:

$$\tilde{\pi} = \gamma_1(j). \tag{2.30}$$

As the left hand side of these equations also appears on the right hand side an iterative procedure has to be used to improve the model parameters. $\hat{\lambda}$ is used in place of λ in each iteration until the values stop changing within certain limits or until the performance on an evaluation data set stabilises. Training with multiple observation sequences is straightforward. The counts ξ and γ are calculated for each observation sequence. The model parameters can now be estimated by averaging over these counts, i.e.:

$$\tilde{a}_{ij} = \frac{\sum_{w=1}^{W} \sum_{t=1}^{T^w - 1} \xi_t^w(i,j)}{\sum_{w=1}^{W} \sum_{t=1}^{T^w - 1} \gamma_t^w(i)}, \tag{2.31}$$

where W is the number of observation sequences. The formulas for b and π change accordingly. Up to this point the algorithms were explained in terms of a finite alphabet of observation symbols and thus discrete probability functions could be used to generate these observations. In case of continuous Gaussian mixtures $b_j(o_t) = N(o_t, \mu_j, U_j)$ with mean vector μ_j and covariance matrix U_j for state j the re-estimation formula becomes:

$$\mu_j = \frac{\sum_{t=1}^{T} \gamma_t(j) o_t}{\sum_{t=1}^{T} \gamma_t(j)}, \tag{2.32}$$

$$\Sigma_j = \frac{\sum_{t=1}^{T} \gamma_t(j)(o_t - \mu_j)(o_t - \mu_j)'}{\sum_{t=1}^{T} \gamma_t(j)}. \tag{2.33}$$

In speech recognition, acoustic model training is usually done using a corpus of speech annotated at the sentence level. For every sentence all phonemes of the words in the sentence are strung together in a composite HMM on which Baum-Welch re-estimation is performed using the corresponding audio sequence. This embedded training procedure effectively performs audio segmentation and parameter learning.

2.2.7 Adaptation

Although the training and recognition techniques described in this chapter can produce high performance recognition systems when trained on a large corpus of speech data, these systems can be improved by customising the HMMs to the characteristics of a particular speaker. By collecting data from a speaker and training a model set on this speaker's data alone, the speaker's characteristics can be modelled more accurately. Such systems are known as speaker dependent systems, and on a typical word recognition task, may have half the errors of a speaker independent system. The drawback of speaker dependent systems is that a large amount of data (typically hours) must be collected in order to obtain sufficient model accuracy.

Rather than training speaker dependent models, adaptation techniques can be applied. In this case, by using only a small amount of data from a new speaker, a good speaker independent model set can be adapted to better fit the characteristics of this new speaker. This can be done using maximum likelihood linear regression (MLLR) (Gales 1998), a technique that computes a set of linear transformations by solving a maximisation problem using the EM technique (that is discussed in 7.6.2). These transformations will reduce the mismatch between an initial model set and the adaptation data by shifting the means and alter the variances in the initial system so that each state is more likely to generate the adaptation data.

Model adaptation can also be done using a Bayesian approach, in which the speaker independent model parameters and the feature vectors in the adaptation data are combined to estimate a new model set. To know how much the model parameters should be changed by the data an occupation likelihood for the model state is needed. This can be obtained by running the Viterbi algorithm. By integrating the Bayesian adaptation approach in the Viterbi algorithm, the model set can be adapted during recognition. Each time an utterance is recognised the system is adapted a little more, so the system incrementally gets tailored to the speaker's voice.

2.3 Variations

Many variations on HMMs have been proposed in literature. For example, a quick scan of the proceedings of Interspeech 2006 (Interspeech 2006) resulted in a wealth of new HMM-based models: Phone Vector DHMMs (Kim *et al.* 2006), multi-path HMMs

(Hämäläinen *et al.* 2006) to create syllable-level models. Hybrid HMM/Bayesian-network models (Markov and Nakamura 2006) that allow the coding of speaker characteristics in the output distributions (Sakti *et al.* 2006). Factorial HMMs (Virtanen 2006), Trajectory HMMs (Zen *et al.* 2006), Local Transformation Models (Miguel *et al.* 2006), a multiple regression hidden semi-Markov model (Tachibana *et al.* 2006) and a 'state-dependent phonetic tied-mixture model with head-body-tail structured HMM' (Park and Ko 2006).

Such models typically try to change the mathematical properties of HMMs. For example Semi-HMMs (Russell and Moore 1985), Inhomogeneous HMMs (Ramesh and Wilpon 1992) as well as the models of Sitaram and Sreenivas (1997) and Wang (1997) all try to add state duration modelling to HMMs (for an overview see Van Dalen (2005)). Other variations make it possible to include additional information in the model. For example, multi-stream HMMs (Bourlard *et al.* 1996) have been used to combine speech recognition and lipreading (Wiggers *et al.* 2002*b*).

More than anything else, the incredible number of variations on HMMs signals that the formalism has been stretched to its boundaries.

You know a word by the company it keeps.

J. R. Firth

Chapter 3

Language Modelling

In which an overview of the state of the art in language modelling is given. The focus is on models that incorporate some context. Strengths and weaknesses of different methods are discussed.

A recurring subtask in natural language processing is to judge whether a sequence of words constitutes a well-formed sentence in a given language or similarly which of a number of sentence hypotheses is syntactically and semantically most plausible.

As shown in the previous chapter a speech recogniser needs this information to choose among hypotheses based on acoustic evidence. Optical character recognition and handwriting recognition hypothesise sentences based on visual information (Hu *et al.* 1996; Plamondon and Srihari 2000). In spelling correction one wants to identify misspelled words (Mays *et al.* 1991) and in statistical machine translation the fluency of sentences in the target language is rated (Brown *et al.* 1990).

Statistical language models fulfil this task by assigning a probability to every word sequence in a language. The idea is that some word sequences are much more likely than others because of syntactic, semantic and pragmatic constraints.

There are multiple ways to define the probability of a sentence, but commonly the chain-rule of probability theory is applied to rephrase the task as assigning a conditional probability to every word in a sentence given the words preceding it:

$$\mathsf{P}(w_{1,\mathrm{t}}) = \mathsf{P}(w_1)\mathsf{P}(w_2|w_1)\mathsf{P}(w_3|w_1w_2)\ldots\mathsf{P}(w_\mathrm{n}|w_{1,\mathrm{t}-1}), \tag{3.1}$$

where w_i is the i-th word in the sentence and $w_{1,T} = w_1 w_2 \dots w_T$[1]. Therefore, language modelling is often described as predicting the next word in a sentence given the history of words. As this formulation was first used by Shannon (1951) to estimate the entropy of English, the task is sometimes referred to as Shannon game. All the applications mentioned above face the difficult problem of reconstructing a sentence from noisy input. Speech contains coarticulation effects and is distorted by acoustic noise, while handwriting recognition has to deal with many different handwritings. In spelling correction and machine translation noise comes in the form of spelling errors and improper translations respectively. In each case, the word history may provide valuable information as to how a hypothesis may continue.

To make this approach work one needs to find accurate probabilities. Based on the assumption that past behaviour can be used to predict the future, i.e. the model is stationary, the parameters of the model are usually estimated from a large set of example sentences. As pointed out in section 1.2, no matter how large a data set is, one cannot derive reliable probabilities for all possible word histories. As a solution, independence assumptions are made that have the effect of grouping histories into equivalence classes. As most word sequences never occur, it is reasonable to assume that simplifications will not considerably weaken the model.

The art of language modelling thus lies in finding good equivalence classes. On one hand the number of classes should be large enough to get sufficiently precise classification on the other hand it should be small enough to ensure reliable parameter estimation, and to keep the amount of storage room used and time taken to retrieve information within reason.

As will be shown in subsequent sections, independence assumptions in language modelling are often quite strong. As the task of a language model is not exact analysis of a sentence to judge its grammaticality, as in parsing, but rather to rank sentences in order of likeliness, one can rely on rather simple models of language. These models have the advantage that they can be trained from large amounts of data and can implicitly capture linguistic knowledge.

The next section discusses methods that judge the performance of a language model. The remainder of this chapter will give an overview of the most important language models that have been proposed in literature, starting with the de facto standard n-gram models and moving on to models that contain semantic or syntactic information, which constitute the state-of-the art in language modelling.

3.1 Quality measures

To evaluate the quality of language models and to be able to compare competing models several measures have been defined. Cross-entropy, perplexity and word error rate are most commonly used.

[1] Equation (3.1) may suggest that the probabilities of sentences of a particular length sum to one, but rather the sum of the probabilities of all sentences in the language should be one. We can accomplish this by introducing an end-of-sentence symbol.

3.1.1 Cross-entropy

Given that we can think of a language model as a device that predicts the words in a text we want it to assign a high probability to a test text. Therefore, we can compare models by looking at the probability they assign to a common test text. However, these probabilities, being the result of repeated multiplication, may get very small, so that often the average log probability is used instead:

$$\mathsf{LP}(w_{1,t}) = -\frac{1}{t}\log \mathsf{P}(w_{1,t}). \tag{3.2}$$

This quantity is actually the cross-entropy from information theory (see e.g. Cover and Thomas 1991), which measures how close a probability model q comes to the real distribution p of some random variable X. It is defined as:

$$\mathsf{H}(p,q) = -\sum_{x\in X} p(x)\log q(x). \tag{3.3}$$

Unfortunately, for most phenomena p is unknown, but for a stationary and ergodic language source[2] and a large enough sample $w_{1,t}$, the sample mean of the negative log probability of a model will converge to its cross-entropy with the true model, that is

$$\mathsf{H}(p,q) = -\lim_{t\to\infty}\frac{1}{t}\log q(w_{1,t}). \tag{3.4}$$

We can approximate this by applying (3.2) to a large data set. Cross-entropy can be thought of as as the amount of surprise in seeing the text given our model. The smaller the amount of surprise the better the model. Entropy is measured in bits[3]. One of the nice properties of entropy reductions is that they are additive, which allows for figures that are easy to interpret.

3.1.2 Perplexity

Rather than cross entropy language modelling literature often reports the related perplexity measure:

$$\mathsf{PP}(w_{1,t}) = 2^{\mathsf{LP}(w_{1,t})}. \tag{3.5}$$

This is the geometric average of the inverse probability of the words measured on the test data:

$$\mathsf{PP} = \sqrt[t]{\prod_{i=1}^{t}\frac{1}{\mathsf{P}(w_i|w_{1,i-1})}}, \tag{3.6}$$

and can intuitively be thought of as the average number of choices a model has to make. A uniform distribution over a vocabulary of $|V|$ words would have perplexity $|V|$.

[2] An ergodic process does not get stuck in some substate. As a consequence one can draw conclusions about its statistical properties from a sufficiently large sample.

[3] To do so, we should use the logarithm with base 2 in the entropy formulas.

Perplexity and cross-entropy are both measures of the model and a data set. It is thus essential that the test data is representative for the application in which the model is to be used.

3.1.3 Word error rate

For speech recognition the quality of a language model can also be measured in terms of word error rate (WER) either by including the model directly into a recogniser or by rescoring n-best lists or lattices produced by a recogniser. Word error rate is defined as the number of insertions, deletions and substitutions per 100 words when the output transcription is aligned, according to a minimum edit distance (Wagner and Fischer 1974), with the correct transcription. In literature the word accuracy is often quoted. The word accuracy percentage is defined as 100% - WER.

As speech recognition experiments are computationally much more costly than calculating perplexities one would hope that there is a strong correlation between perplexity and word error rate, but as many find (Iyer *et al.* 1997; Chen *et al.* 1998; Clarkson and Robinson 1999), this is not the case. One of the problems is that perplexity does not take into account other factors such as acoustic confusability. A high perplexity language model that is good at discriminating between acoustically similar words might be better for speech recognition than a language model with lower perplexity. The other problem with perplexity is that it does not penalise unlikely hypotheses. It only measures how well a model does at predicting positive examples. Despite these drawbacks perplexity remains the most commonly used quality measure for language modelling.

3.2 n-grams

n-grams are the most common type of language model. They put all histories with the same last $(n-1)$ words in one equivalence class. Put differently the model equals an $(n-1)$th order Markov model. The choice of n depends on the amount of training data available, but things quickly get out of hand as the number of potential n-grams scales exponentially with n. Therefore, the most common choice of n is three. It turns out that, despite its simplicity, the resulting trigram model is surprisingly powerful and hard to beat indeed. It is easy and efficient to train and easy to include in a speech recogniser because of the sequential decomposition. This allows for a minimal delay in response from the recogniser to the user. In addition, because of the strong independence assumptions, trigrams are relatively robust in the face of recognition errors.

However, the simplicity of trigrams, and of n-grams in general, is also their greatest disadvantage: they are unable to model long-range dependencies between words. As a consequence probability mass is incorrectly distributed over the space of possible sentences. One can easily make up nonsense sentences that are plausible in terms of n-grams, such as (1).

(1) The dog is sitting behind a keyboard and mouse is chased by the morning sun is bright.

On the other hand correct sentences may get a lower probability than they should. For example, in (2) the trigram probability of 'lifted' following 'and white' is likely to be low.

(2) The plane, painted blue and white, lifted off.

The problem is of course that n-grams only use local information. As words may not depend on the previous $n-1$ words at all n-grams can unnecessarily fragment training data. More advanced models try to address these shortcomings.

Before we move on to those models we will first discuss how to obtain the parameters of an n-gram. A simple solution is to count relative frequencies in a training data set:

$$P_{\text{MLE}}(w_i|w_{i-n+1,i-1}) = \frac{C(w_{i-n+1,i-1}w_i)}{C(w_{i-n+1,i-1})}. \tag{3.7}$$

This is the maximum likelihood estimate (MLE) of the n-gram probability. It maximises the likelihood of the training corpus and does not waste any probability mass on things that are not in the data. In general maximum likelihood estimates as such are not very suited for language modelling. The problem is that there is a small number of n-grams that occur very frequently and a large number of n-grams that may eventually occur but that will not show up in the training set. Experiments conducted at IBM (Bahl *et al.* 1983) showed that 23% of the trigrams in a test set never occurred in the 1.5 million word training set. The combined probability mass of those rare n-grams is substantial and cannot be ignored. This is especially true as according to equation (3.1) a single zero probability will make the probability of an entire sentence zero.

Even for very large data sets the problem of unseen events is unavoidable, because the number of parameters typically exceeds the potential number of unique trigrams in the data set by several orders of magnitude. For example, large data sets currently contain about a hundred million words, whereas a small vocabulary of 1000 words would already require the estimation of a billion parameters. Thus rather than relying on relative frequencies directly one should use estimators that allow for unseen events. Those will be discussed in the next section.

3.3 Smoothing

To overcome the problem of unseen events and unreliable counts from sparse data a number of techniques have been developed that correct the bias of maximum likelihood estimates by intelligently shifting around some probability mass. Such techniques are often called smoothing techniques as they make distributions more uniform. *Discounting techniques* do so by removing probability mass of observed events and redistributing it over unseen events, while *interpolation* and *backing-off* approaches combine several models of different granularity. When probabilities

are estimated from sparse data, smoothing techniques can significantly improve the accuracy of the model. Chen and Goodman (1996) find that the relative performance of techniques depends greatly on training data size and n-gram order.

3.3.1 Discounting

Discounting techniques remove some of the probability mass from the original counts and redistribute the remaining probability mass over unseen events.

Additive smoothing

The simplest discounting method is to assume that every n-gram occurs δ times more than it does, where $0 < \delta \leqslant 1$.

$$P_{\text{ADD}}(w_i|w_{i-n+1,i-1}) = \frac{C(w_{i-n+1,i-1}w_i) + \delta}{C(w_{i-n+1,i-1}) + \delta|V|}, \tag{3.8}$$

where $|V|$ is the size of the vocabulary. Often $\delta = 1$ is used (Lidstone 1920; Jeffreys 1948). This comes down to constructing a Bayesian estimator that assumes a uniform prior on events (Manning and Schütze 1999). This specific instance is sometimes called Laplace smoothing. Because there are so many unseen events, additive smoothing typically assigns too much probability mass to unseen events and the resulting estimates are linear in the MLE frequencies that are typically incorrect for low counts.

Held-out estimation

Rather than setting the probability mass of all unseen events to some constant, held-out estimation (Jelinek and Mercer 1985) uses an empirical approach to find how much of the probability mass should be assigned to unseen events. It counts how often n-grams that appear r times in the training data appear in a held-out data set. Let N_r be the number of n-grams that occur r times:

$$N_r = |\{w_{i-n+1,i} : C(w_{i-n+1,i}) = r\}|. \tag{3.9}$$

Then the held-out estimate of r is:

$$r_h = \frac{1}{N_r} \sum_{\{w_{i-n+1,i}|C(w_{i-n+1,i})=r\}} C_h(w_{i-n+1,i}). \tag{3.10}$$

where the $C_h(w_{i-n+1,i})$ are n-gram counts on the held-out data set.

Cross-validation is a related approach in which the training data is partitioned in k subsets each of which is subsequently used as held-out set. The estimated frequency is then found by taking the average of the held-out estimates.

Good-Turing discounting

Good-Turing discounting (Good 1953), the original idea of which is attributed to Turing, does not need a held-out data set but instead states that to find the probability of events with zero or low counts we must look at the number of events with higher counts. Let N_r as in (3.9) be the number of n-grams that appear r times. The Good-Turing estimate of r (for $r \geqslant 1$) is:

$$r_{gt} = (r+1)\frac{E[N_{r+1}]}{E[N_r]}. \tag{3.11}$$

This quantity is obtained by calculating the expected value of the true unknown probabilities of the n-grams given the counts from the training data. See Church and Gale (1991) for a derivation. The model is based on the assumption that the events have a binomial distribution. This is generally not true for n-grams, nevertheless the method works well for large data sets with a large vocabulary.

For sufficiently large counts $E[N_r]$ can be replaced by N_r, but for large r this is typically not the case (and large counts are assumed to be reliable anyway), so in practice Good-Turing is mainly used for n-grams with small counts. One of the consequences of the model is that the estimate the count of n-grams we have never seen is based on counts of n-grams we have seen once. The left-over probability mass reserved for unseen events is $\frac{N_1}{N}$ (Gale and Sampson 1995).

Good-Turing estimates cannot deal with $N_r = 0$. Therefore, the N_r values typically have to be smoothed to make sure that they are all above zero. However, Good-Turing estimates are hardly ever used by themselves for n-gram smoothing (except in textbooks) but do appear as a component in several more sophisticated smoothing techniques.

Absolute discounting

Absolute discounting uses the simple strategy of subtracting a small constant from all non-zero counts and redistributing the accumulated mass over unseen events:

$$r_{abs} = \begin{cases} r - D & \text{if } r > 0, \\ \frac{D}{N_0}\sum_r N_r & \text{if } r = 0. \end{cases} \tag{3.12}$$

The constant D can be estimated from held out data. Ney *et al.* (1994) suggest $D = \frac{N_1}{N_1+2N_2}$. The intuition behind this method is that the estimates of high frequency words are rather good. Removing a small amount of mass from those counts will not change much. It can be shown that in practice absolute discounts are close to Good-Turing discounts. Absolute discounting forms the basis of a number of successful smoothing strategies.

3.3.2 Interpolation

Discounting methods typically assign the same probability to all unseen events, but imagine that neither of the bigrams *who are* and *who art* occur in the training data.

Unless we are modelling the works of Shakespeare one would expect the probability the first bigram to be higher than that of the second. We can get the desired effect by taking the fact that *are* is a common word into account, i.e. by looking at its unigram frequency. This is what deleted interpolation (Jelinek and Mercer 1980) does. It is based upon the idea that lower order n-grams can give valuable information when estimating probabilities from sparse data. This information is included through linear interpolation. For example instead of a trigram a weighted sum of trigram, bigram and unigram probabilities is used:

$$\hat{P}(w_i|w_{i-1}w_{i-2}) = \lambda_1 P(w_i|w_{i-1}w_{i-2}) + \lambda_2 P(w_i|w_{i-1}) + \lambda_3 P(w_i). \tag{3.13}$$

If not all words in the vocabulary occur in the training set one can also include a uniform distribution over the vocabulary in the summation. Often a more general scheme is used in which the weights themselves are a function of the history:

$$P_{di}(w_i|w_{i-n+1,i-1}) = \lambda_{i-n+1,i-1} P(w_i|w_{i-n+1,i-1}) + (1 - \lambda_{i-n+1,i-1}) P_{di}(w_i|w_{i-n+2,i-1}). \tag{3.14}$$

Different sets of weights are used for different contexts. A common approach is to use the frequencies of frequencies to partition the data in a number of buckets. A trigram with many samples should have a higher weight than a trigram with only a few samples. Linear interpolation is not only used to combine n-grams, but is often used to combine different language models. Such models are also referred to as mixture models.

The weights of the model are found automatically using a held out data set. One can do so using a version of the EM algorithm, but for simple cases such as (3.13) an analytical solution exists (Jelinek 1999).

3.3.3 Back-off models

Given enough evidence relative frequencies give very good probability estimates. Back-off models implement this idea by using a set of component models. If there is enough evidence the most detailed model will be used, otherwise a less specific model is used instead.

Katz backing-off

Back-off models are typically defined by a recursive function. Katz backing-off, which is commonly used in speech recognition systems, is the canonical form of backing-off:

$$P_K(w_i|w_{i-n+1,i-1}) = \begin{cases} \frac{d_{w_{i-n+1,i-1}}(C(w_{i-n+1,i}))}{C(w_{i-n+1,i-1})} & \text{if } C(w_{i-n+1,i}) > 0, \\ \alpha(w_{i-n+1,i-1}) P_K(w_i|w_{i-n+2,i-1}) & \text{otherwise,} \end{cases} \tag{3.15}$$

where d is a discounting ratio. Katz (1987) uses $d = 1$ for counts that are larger than some constant k (e.g. k=5) assuming that the MLE estimate for those counts

will be accurate. For counts in $0 < r \leqslant k$, d is based on the Good-Turing estimate. α is a normalising factor such that only the discounted probability mass is distributed over n-grams estimated by backing-off and the total probability mass equals 1. Backing-off models can be seen as a special case of the general formulation of linear interpolation, that choose exactly one of the (discounted and normalized) component distributions, i.e. one of the weights is 1 while the others are zero.

Witten-Bell smoothing

Witten-Bell smoothing uses the idea that one can think of an unseen n-gram as one that simply has not yet occurred. To estimate the probability that a previously unseen word occurs after a history $w_{i-n+1,i-1}$ one can count the number of times a new word occurred after this history in the training data:

$$T = |\{w_j : C(w_{j-n+1,j}) > 0\}|, \tag{3.16}$$

which is simply the number of unique n-gram types starting with that history. The Witten-Bell discount, which can be seen as an approximation of the probability that a previously unseen word occurs after the history $w_{i-n+1,i-1}$, is derived from this count:

$$(1 - \lambda_{w_{i-n+1,i-1}}) = \frac{T}{T + \sum_{w_i} C(w_{i-n+1,i})}, \tag{3.17}$$

which is then used as an interpolation weight:

$$P_{WB}(w_i|w_{i-n+1,i-1}) = \lambda_{w_{i-n+1,i-1}} P_{MLE}(w_i|w_{i-n+1,i-1}) + (1 - \lambda_{w_{i-n+1,i-1}}) P_{WB}(w_i|w_{i-n+2,i-1}). \tag{3.18}$$

One can interpret this as saying that if the probability that a unseen word will occur after the current history is high, one should use the lower count, otherwise the MLE count will be reasonably accurate. Chen and Goodman (1996) find that this estimator does not do as well as other smoothing techniques.

Kneser-Ney smoothing

Kneser-Ney smoothing (Kneser and Ney 1995) is an extension of absolute discounting that estimates the parameters of the lower order distributions in such a way that the marginals of the smoothed distribution match the marginals of the training data. The intuition behind the technique is that the influence of lower-order distributions is only important if few or no counts are present in the higher-order distributions. Therefore, the lower order counts should be optimised for those situations.

Kneser-Ney smoothing looks at the number of contexts a word appears in rather than at the number of times a word appears to create a backed-off distribution. This can be motivated by looking at common words that only occur after a single word. Chen *et al.* (1998) use *San Francisco* as an example. Most smoothing techniques will assign a relatively high probability to bigrams ending with *Francisco* based on

its high unigram count. But in fact the original bigram counts were much more accurate.

The original model was formulated as a back-off model, but Chen *et al.* (1998) show that the interpolated version of the model, defined by

$$P_{KN}(w_i|w_{i-n+1,i-1}) = \frac{\max(C(w_{i-n+1,i}) - D, 0)}{C(w_{i-n+1,i-1})} + \lambda_{w_{i-n+1,i-1}} P_{KN}(w_i|w_{i-n+2,i-1}), \tag{3.19}$$

where

$$\lambda_{w_{i-n+1,i-1}} = \frac{D}{C(w_{i-n+1,i-1})} |\{w_i : C(w_{i-n+1,i}) > 0\}|, \tag{3.20}$$

generally yields better performance. Arguing that 'the ideal average discount for n-grams with one or two counts is substantially different from the ideal average discount for n-grams with higher counts' they also introduce a variation called Modified Kneser-Ney smoothing that uses three different discount parameters. They find that Interpolated and Modified Kneser-Ney are superior to other smoothing techniques across training data size, corpus types, n-gram order and across clustering techniques (Goodman 2000).

3.4 Distant n-grams

If one uses longer histories to predict a word, the probability of seeing the exact same context gets smaller, but on the other hand the probability of having seen a similar context becomes larger. Distant n-grams predict the probability of the next word based on $n-1$ words that are some distance back in the history. For example in the sentence *the dog jumped and barked* the word *dog* is much more predictive for the word *barked* than *and*. By themselves distant n-grams are less powerful than n-grams but the combination of those models can realise better perplexity and word accuracy than either of the approaches alone (Rosenfeld 1994; Siu and Ostendorf 2000).

By combining low-order distant n-grams one can get a performance similar to a higher order n-gram. This can be a useful strategy for small amounts of data. Goodman (2000) systematically explores many different combinations of distant n-grams and finds that with a combination of distant bigrams up to $n = 5$ one can approximate a trigram. Like n-grams, distant n-grams are position-bound. In addition, they put training instances that contain exactly the same words but have different distances in different equivalence classes. They thus unnecessarily fragment training data.

3.5 Class-based language models

Class-based models map the words in the vocabulary on a smaller number of classes and then calculate n-grams over those classes rather than over the words themselves.

There are several ways in which the probabilities can be defined, but the most commonly used scheme predicts the current class based on the preceding classes and the current word based on the current class:

$$\mathsf{P}\left(w_i|w_{i-1}w_{i-2}\right) = \mathsf{P}\left(w_i|\mathsf{K}(w_i)\right)\mathsf{P}\left(\mathsf{K}(w_i)|\mathsf{K}(w_{i-1})\mathsf{K}(w_{i-2})\right), \tag{3.21}$$

where $\mathsf{K}(w_i)$ is the class of word w_i. Different types of classes have been used, including part-of-speech tags and manually constructed classes of words that are semantically or syntactically similar, such as days of the week or numbers, or application specific classes. The best performance is typically obtained with classes that are automatically generated using some form of clustering (Brown *et al.* 1992; Kneser and Ney 1993; Ueberla 1995; Niesler *et al.* 1998; Gao and Goodman 2002). Some models allow words to belong to more than one class. The class then becomes a hidden variable.

As the number of possible histories is greatly reduced, class-based models have fewer parameters than their word-based counterparts[4]. As a result the parameter estimates are much more reliable and the models are smaller and faster. The biggest advantage of class-based models is that they can generalise over histories: they can predict the probability of unseen events by assuming that they are similar to observed events.

The drawback is that the models are less specific than word models. Compared with n-grams they lose information. This problem can be overcome by combining the class-based model with a word-based model to get the best of both worlds. It has been shown that those mixed models can decrease both perplexity and word error rate especially in situations where limited amounts of training data are available (Jelinek 1990; Niesler and Woodland 1996; Kneser and Steinbiss 1993; Heeman 1999; Samuelsson and Reichl 1999; Goodman 2000). Like n-grams class-based models cannot model long-distance dependencies.

3.6 Cache-based language models

The language models described so far all have fixed parameters based on training data. However, data analysis shows that if a word is used it is more likely to occur again than a standard n-gram would predict (Rosenfeld 1994). This notion that words occur in bursts is captured by cache-based models.

Caches are typically implemented by keeping a list of the previous k words. Probabilities are computed as the relative frequency of the words within the cache. The resulting distribution is then interpolated with a standard n-gram. Cache-based models were first described by Kuhn and de Mori (1990). They added caches to a class-based model. For every part-of-speech they maintained a separate 200 word cache. They realised a 14% decrease in perplexity over their baseline class-based model.

[4]Unless, of course, $n = 1$ or the number of clusters equals the vocabulary size.

Others have added caches to word based models (Iyer and Ostendorf 1996) and used probabilities of recently occurring bigrams and trigrams (Jelinek *et al.* 1991). However, the performance gain compared with unigram caches has been little, which is likely because the estimates for the resulting probabilities are not reliable.

It has also been noted that the probability of word reappearance again decreases with increasing distance. This has been implemented with decaying caches (Clarkson and Robinson 1997).

Although cache-based models have been shown to achieve large perplexity reductions this does not translate in word error rate reductions. The problem is that the cache contains previously recognised words. If those are incorrect then the probability that the word will be recognised incorrectly again the next time it is heard increases.

3.7 Triggers

Cache-based models do not capture dependencies within a sentence. For example, one would expect the occurrence of a word not only to increase the probability of itself reoccurring, but also of semantically related words. Trigger-based models (Rosenfeld 1994) extend the idea of cache-based models to deal with such dependencies. The model defines the concept of a trigger pair: if the first word of the trigger pair occurs in the word history then the probability of the second word is increased. As the potential number of trigger pairs is huge (the square of the vocabulary size), a mutual information criterion is used to select trigger pairs. It turns out that the largest performance gain is obtained from self-triggers and same-root triggers, so there is little improvement over a cache-based approach. Rosenfeld reports that for two thirds of the words in the training set the highest mutual information trigger was the word itself and for 90% of the words the self-trigger was among the top 6 triggers. One might expect that word-stem triggers work better than word triggers, but Rosenfeld found that this gives little improvement.

A maximum entropy approach is typically used to combine triggers with n-grams and distant n-grams to capture both local and long distance information. Some of the largest perplexity reductions reported in literature have been realised with such models (Rosenfeld 2000).

3.8 Latent semantic analysis

Multi-span language models (Bellegarda 1998) make use of techniques developed in information retrieval to capture long-distance semantic relationships, in particular a paradigm called latent semantic analysis (LSA) (Baeza-Yates and Ribeiro-Neto 1999). LSA represents documents as well as the words in these documents as vectors in a low-dimensional space.

An important property of this space is that words whose vector representations are close tend to appear in the same kind of documents and documents that appear

close tend to contain semantically similar content. As a consequence words and documents that are semantically linked also are close in this space. In language modelling training data is used to assign a vector to every word. When calculating the probability of a sentence, the word history is interpreted as a document for which a vector is constructed as well that is used to calculate the distance to all words in the vocabulary. A probability distribution is then found by normalising distances in such a way that the total probability mass is one. Alternatively, the cosine distances are raised to a power (optimised on held-out data) before normalising, as normalised cosines typically have a much smaller dynamic range than n-grams (Coccaro and Jurafsky 1998). Obviously, this approach requires that the training data is partitioned in semantically homogeneous documents (where documents can for example be sentences or articles).

Coccaro and Jurafsky (1998) use a slightly different model. They only consider word vectors and combine the vectors associated with every word in the history to obtain a centroid in vector space.

The multi-span model is similar to the trigger-based model in that it models long-distance semantic dependencies between words, but unlike the trigger-based model it does not require explicit trigger-pair selection. Furthermore, the model can also find indirect relations between words that never actually occur in the same documents in the training data, but that do occur in similar contexts.

The model does not use any syntactic or position information at all. Therefore it is a poor language model as such. But the information contained in it is largely complementary to the information in a standard n-gram model. Bellegarda (1998) proposes a Bayesian combination of LSA models (as a global prior) and n-grams that performs better than either of the models alone. The best models, that also employ clustering in vector space as a means of smoothing, realise a relative decrease of 30% in perplexity. Coccaro and Jurafsky (1998) combine the model with an n-gram using a geometric mean rather than a linear combination, which gives a high probability if both models agree and a low probability if either one of the components assigns a low probability to a word. The contributions of the component models are weighted by the entropy of the frequency of a word in the training corpus. It makes sure that for common words, which will have a high entropy, the n-gram will provide most of the probability mass.

The LSA model is a bag of words model, all words in the history have an equal amount of influence. As argued above the influence of words decreases with distance. Zhang and Rudnicky (2002) introduce distance weighting in the model. They use three LSA models that take different amounts of history into account, respectively at the document, paragraph and sentence level. The models are combined together with a standard trigram using a softmax neural network. The model does better in terms of perplexity than a trigram model and than a standard LSA model. However, the perplexity did not increase much if document and sentence level models were left out of the combination. So, one might wonder whether it is not just a case of finding an optimal length for the part of the history being used.

3.9 Mixture models

Language use may vary greatly in terms of style and topic. This information is lost in standard language models that calculate global statistics over a heterogeneous data set. Mixture models try to recover this information by identifying subsets in the data for which specific models are build. Modelling thus starts with partitioning the data for example using a manually tagged data set or some form of automatic clustering (Clarkson and Robinson 1997; Gotoh and Renals 1999). Soft-clustering may be used, where a document may belong to more than one cluster.

Then for each subset a n-gram model is trained. Those models are then combined using linear interpolation, where weights are trained on held-out data (Kneser and Steinbiss 1993):

$$P(w_i|w_{i-n+1,i-1}) = \sum_{k=1}^{m} \lambda_k P_k(w_i|w_{i-n+1,i-1}). \tag{3.22}$$

The number of components is once again a balance between specificity and reliability. If too many components are used, individual models will be trained on a sparse data set and memory and computational requirements will be increased. The model may run in to trouble if the topics in the test data differ from those in the training data. Inclusion of the global model as a component in the mixture may partially overcome this problem.

Sentence level mixtures (Iyer and Ostendorf 1996) combine m component models at the sentence level rather than at the n-gram level. Each component contains the n-gram statistics of a specific topic or a broad class of sentences. Probability of a word sequence is given by:

$$P(w_{1,T}) = \sum_{k=1}^{m} \lambda_k \left[\prod_{i=1}^{T} P_k(w_i|w_{i-n+1,i-1}) \right]. \tag{3.23}$$

They use agglomerative clustering, with a similarity measure based on the combination of inverse document frequencies, at the article level. The initial n-gram parameter estimates are based on this partitioning and are subsequently iteratively re-estimated using the expectation-maximisation algorithm (see section 7.6.2) using all data for all topics to obtain a soft clustering. To avoid zero entries Witten-Bell smoothing is incorporated in the re-estimation. To further deal with fragmentation of the training data (components may suffer from data sparsity) they interpolate the topic specific models with a general model at the n-gram level and to account for non-topic sentences a global model is included in the mixture in addition to the m components:

$$P(w_{1,T}) = \sum_{k=1}^{m,G} \lambda_k \left[\prod_{i=1,T} \theta_k P_k(w_i|w_{i-1,i-2}) + (1-\theta_k) P_G(w_i|w_{i-1,i-2}) \right]. \tag{3.24}$$

The weights λ_k and θ_k are re-estimated using held-out data. Furthermore a separate content word cache for all component models is used. The unadapted mixture model is reported to reduce perplexity by 22% over the unadapted trigram language model. The dynamic cache resulted in an additional 14,5% perplexity reduction. The mixture models gave a 3% – 4% reduction in WER. Adding caches gave little to no additional improvement.

In those initial experiments only a small number of sentence mixtures was used but Goodman (2000) found that a large number of mixtures (up to 64 for a data set of 284,000,000 words) can work even better. The approach works especially well for larger training sets.

3.10 Topic-based language models

Mixtures provide a way to encode topicality in a language model, but other models have been proposed. Mahajan *et al.* (1999) avoid predefined clustering of the data. Instead, they construct a topic specific language model on the fly by comparing the history with documents in a database using a TF-IDF-measure (see chapter 10) and selecting the k highest ranked documents to train a language model on which is then combined with other components such as a trigram and a cache-based model.

Gildea and Hofmann (1999) consider the topic to be a latent variable according to:

$$P(w_i|w_{i-n+1,i-1}) = \sum_t P(w_i|t)P(t|w_{i-n+1,i-1}), \tag{3.25}$$

where t is a topic. The number of values t can take is predetermined and the model is trained using the expectation maximisation algorithm. For testing the $P(t|d)$ that is the probability of a topic given the document are omitted and instead the weights $P(t|h)$ are estimated on the history so far using a single iteration of an online variation of the EM algorithm. The probabilities are combined with n-gram probabilities under the assumption that both models deliver marginal probabilities that should be preserved under the combined model:

$$P(w_i|h_i, w_{i-n+1,i-1}) \approx P(w_i|h_i)P(w_i|w_{i-n+1,i-1})P(w_i). \tag{3.26}$$

Although this assumption clearly does not hold (h_i and $w_{i-n+1,i-1}$ are not independent) the authors show that it yields better results than linear or log-linear interpolation of both models. The model has lower perplexity than LSA language models, but it does not achieve a significant word error rate reduction. However, a results analysis in terms of word frequencies suggests that 'the model may be able to help on content words'.

Khudanpur and Wu (1999) investigate the idea that many words and word n-grams do not really depend on a topic and one should not unnecessarily fragment training data. They include topic specific unigrams in a maximum entropy model along with topic independent n-gram constraints. Topics are found by automatic clustering. Words whose unigram frequency in a cluster significantly differs from its

frequency in the whole corpus are seen as topic related words. During testing the topic of a conversation is determined from the 10-best hypotheses found in a first recognition pass. The maximum entropy model is then used for rescoring.

3.11 Whole sentence models

Unlike all the other models discussed in this chapter these models do not use the chain rule to decompose the probability of a word sequence, but treat the sentence as a whole. This way sentence level features, such as the coherence or grammaticality of a sentence can be used as can external influences from preceding utterance or from pragmatic or dialogue levels. These features are incorporated in a maximum entropy framework (Rosenfeld 1994). Unfortunately, exact training of these models is intractable and methods such as Monte Carlo sampling have to be used.

3.12 Tree-based language models

Tree-based language models use binary decision trees to partition the history in equivalence classes. At every node in the tree a yes/no question about the word history is asked, the answer determines the path taken down the tree. At each leaf of the tree a probability distribution is defined over all the words in the vocabulary. The strength of the tree-based approach is that any question about the history can be asked, so an optimal equivalence classification can be encoded (Jelinek 1999). However, the space of possible questions is too large to be searched exhaustively. Therefore, in practice simplifications of the model are used (Bahl *et al.* 1989). Tree-building algorithms are usually greedy and not guaranteed to find the optimal tree. Furthermore, irrelevant questions or the wrong order of questions may unnecessarily fragment training data. The leaf distributions are typically sparse and have to be smoothed.

3.13 Grammar-based models

Grammar based models incorporate linguistic constraints in the language model in order to assign most of the probability mass to syntactically plausible sentences.

They use the syntactic structure of the sentence seen up to a certain point to extract meaningful information from the word history to predict the likelihood of the next word in the sentence. Unlike n-grams these models are not position bound, but can capture long distance relations between words, using grammar rules to identify the words in the history that are relevant in the prediction of a word.

Many grammar based language models and statistical parsers that can be used as such have been developed (Black *et al.* 1992; Charniak 1993; Magerman 1995; Stolcke 1995; Collins 1996; 1997; Charniak 1997; Chelba and Jelinek 1998; Roark and Johnson 1999; Collins 1999; Charniak 1999; 2001; Chelba 2000; Roark 2001;

Uytsel *et al.* 2001). Different parser types and grammar formalisms have been used, but most are based on or related to probabilistic context-free grammars that will be introduced next. Subsequently, the most representative and successful examples of grammar-based language models will be discussed.

3.13.1 Probabilistic context free grammars

In linguistics it is generally assumed that words group together in phrases, called constituents, that themselves can group in to bigger phrases, subsentences and sentences. Constituents display complex recursive interaction. To capture this grammars are used. Context free grammars for example consist of rules that specify how a constituent can be expanded into a sequence of smaller units.

For language modelling (and for statistical parsing) we need to assign probabilities to sentences. Probabilistic context free grammars (PCFGs) simply extend context free grammars by assigning probabilities to rewrite rules. Formally, a probabilistic context-free grammar consists of (Manning and Schütze 1999):

- A set of terminals (words) $w^k \in V$, where V is the vocabulary.
- A set of non-terminals $\{N^i\}$.
- A start symbol N^1.
- A set of rules $R = \{N^i \rightarrow \lambda^j\}$, where λ^j is a sequence of terminals and non-terminals .
- A set of probabilities corresponding to these rules. Each rule has a probability attached to it and the probabilities of all rules with the same non-terminal at the left-hand side sum to one, i.e. $\forall i \sum_j P(N^i \rightarrow \lambda^j) = 1$.

The probability of a rule can be interpreted as the probability that the production $N^i \rightarrow \lambda^j$ is chosen given that N^i will be expanded. The rule probabilities are thus conditional probabilities.

The set of all possible strings that can be generated by the grammar is called the language L. The grammar specifies how the start symbol can be expanded into a string from this language.

The probability of a parse can be found by calculating the joint probability of all the rewrite rules that have been used to build the tree. To find this probability a number of independence assumptions are made:

1. Place invariance: The probability of a constituent does not depend on where in the string the words it dominates are.

$$\forall k, l \quad P(N^j_{k,l} \rightarrow \lambda^j) = P(N^j \rightarrow \lambda^j). \tag{3.27}$$

2. Context-free: The probability of a constituent does not depend on words not dominated by the non-terminal.

$$P(N^i_{k,l} \rightarrow \lambda^j | \text{anything outside k through l}) = P(N^i_{k,l} \rightarrow \lambda^j). \tag{3.28}$$

3. Ancestor-free: The probability of a subtree does not depend on nodes in the derivation outside the subtree.

$$P(N^i_{k,l} \rightarrow \lambda^j | \text{any ancestor nodes outside } N^i_{k,l}) = P(N^i_{k,l} \rightarrow \lambda^j). \tag{3.29}$$

Typically, there are several ways in which a sentence can be parsed. The probability of a sentence (according to the grammar G) is given by the sum of all possible parses for the sentence:

$$P(w_{1,n}) = \sum_T P(w_{1,n}, T). \tag{3.30}$$

An efficient algorithm, called the inside algorithm, similar to the forward algorithm for HMMs, exists that finds the probabilities of all parses of a sentence.

As PCFGs provide a probability for each string in a given language and these probabilities sum to one, they can be used as language models for applications such as speech recognition and indeed have been (Jurafsky *et al.* 1995). However, as such they are very poor language models and are not capable to improve upon the simpler n-gram models. The problem here stems from the very nature of PCFGs, the context-freeness assumption. In a PCFG the probability of a word only depends on its pre-terminal, i.e. its POS-tag. Thus the probability of the word 'apple' only depends on the fact that it is a noun and as the probabilities of all words that can be a noun should sum to one it may have a relatively low probability. A bigram model on the other hand may assign a higher probability to the word 'apple' given that the previous word is 'green'. It thus includes some notion of context, however crude it may be.

Including lexical knowledge in the model may (partially) solve this problem. This can be done by associating a headword (and possibly a POS-tag) with each non-terminal in the parse tree. Intuitively, the headword of a constituent is the most important word in a constituent. A non-terminal inherits its lexical head from one of its children, while the other children are modifiers to the head child. The grammar rules become of the form:

$$N^i(n^h) \rightarrow N^j(n^j) \dots N^h(n^h) \dots N^k(n^k), \tag{3.31}$$

where n^h is the headword of the constituent. This leads to an explosion in the number of rules making direct estimation of rule probabilities infeasible due to data sparseness. Therefore some form of decomposition of the rule probability is usually needed.

In theory a PCFG can be learned from a sample of language using an incarnation of the EM algorithm called the inside-outside algorithm. The basic idea is to generate all possible production rules with some initial probabilities and then run the

training algorithm, which will hopefully assign zero probability to impossible rules, and relatively high probabilities to the rules of the 'correct grammar'.

However, this scheme does not work very well due to a number of reasons. For starters, there is no bound to the possible number of rules in the grammar, as one can indefinitely add non-terminals to the right hand side of a rule. Another problem is that the iterative inside-outside algorithm might get stuck in a local maximum during training. As a matter of fact it is highly unlikely that the optimal solution will be found. Charniak (1993) reports an experiment where each of 300 trials of grammar induction produced a different grammar, none of which resembled a grammar as a human would design it. Even if the initial grammar is restricted to certain rules training may completely change the meaning of non-terminals as the only hard constraint in the definition of PCFGs is that the top of a parse tree should be the start symbol N_1.

So, for practical applications usually examples are given of what parse trees should look like. This can for example be done by bracketing, that is indicating which part of a sentence should form a constituent, for example (the man (patted (the dog))). One can take this one step further by also including non-terminal annotation in the training data, showing complete parse trees. Collections of such trees are called treebanks. The Penn Treebank is most widely used for English.

When using a treebank training becomes very simple: one just has to count the frequencies of local subtrees and normalise those to get probabilities. The disadvantage of this approach is that the grammar can only handle rules seen in the training data; it cannot generalise.

3.13.2 The structured language model

The structured language model (SLM) of Chelba and Jelinek (1999) is based on a shift-reduce parser (Allen 1995) but has been specifically designed as a language model for speech recognition.

The syntactic structure is developed incrementally in the shape of bottom-up partial parses with headword annotation while traversing the sentence in a left-to-right manner. The model can thus be used for the decoding of word lattices. The model operates by means of three modules that ensure that all possible binary branching parses with all possible headwords and non-terminal label assignments for the word sequence $w_1 \dots w_k$ can be generated:

1. The WORD-PREDICTOR predicts the next word w_{k+1} given the partial parses and then passes control to the TAGGER.

2. The TAGGER predicts the POS-tag t_{n+1} of the next word given the left context and the newly predicted word and then passes control to the PARSER.

3. The PARSER grows the already existing binary subtrees by repeatedly shifting new nonterminals onto the stack or combines the top-most elements of the stack (reduce) into a new subtree that gets its headword either from the left or

from the right. The parser stops if no more operations are possible and then passes control back to the PREDICTOR.

Each of the components have a conditional probability attached to it. The left-context is the collection of those binary subtrees whose span is completely included in the word history. For the WORD-PREDICTOR the independence assumption is made that the word to be predicted only depends on the identities of the headwords of the two previous binary subtrees not yet included in a bigger constituent. Word prediction in an SLM is thus similar to a trigram. If the binary branching structure would always be right-branching and all POS-tags and non-terminal labels would be mapped to a single type, the model would be equivalent to a trigram. The parts of the left-context used by the tagger are the POS-tags of the headwords of the previous two subtrees.

As the number of possible parses for a given word prefix w_k grows exponentially with k, a multistack algorithm resembling a beam-search is used. Each stack contains the partial parses constructed by the same number of predictor and parser operations.

The language model probability of the next word w_k in the sentence is found by summing over all parses on all stacks with k predictions.

Parameter re-estimation cannot be done using dynamic programming. Instead a variation of the EM algorithm is used that uses the count of model actions based on the derivations of the n-best hypotheses to redistribute probability mass.

The model has been shown to outperform the trigram model in terms of perplexity on the UPenn treebank corpus. The data used in these tests was made more 'speech-like' by removing punctuation and capitalisation and replacing all but the 10000 most common words with <UNK> and using a single symbol for all numbers. A further improvement was achieved by interpolating the model with a trigram.

3.13.3 Probabilistic top-down language model

Roark (2001) notes that bottom-up parsers such as described in the previous section typically produce partial derivations that consist of unconnected tree fragments. Therefore, it might assign probability mass to tree fragments that cannot be part of a sentence generated by the grammar.

He proposes to use a top-down probabilistic parser that creates rooted trees built from left to right, thus allowing the calculation of probabilities for prefix-strings $w_1 w_2 \ldots w_i$ of all sentences in the grammar. Unlike the SLM this is a true probabilistic parser although it has been developed with language modelling in mind.

Roark points out that top-down guidance may improve the efficiency of the search as more and more conditioning events are extracted from the derivation for use in the probabilistic model. The rooted and fully connected partial derivations produced by the top-down parser ensure that all of the conditioning information that may be extracted from the top-down left-context is already specified. This gives an interesting analogy with human language understanding, as psychological evidence

suggests that humans incrementally build a syntactic and semantic interpretation while listening or reading.

The model starts with a left-factored PCFG, so that all productions are binary, this way predictions about what non-terminals are expected later in the string are delayed until more of the string has been seen. Normally in top-down parsing a parent node and the rule expanding it are announced before any of its children, the result of left-factoring is that the parent is identified before its children, but the rule expanding the parent is known only after all of its children are known.

To choose which events in the left context should be used to condition on, functions are used that take the partial tree structure as an argument and return a value, upon which the rule probability can be conditioned.

The functions chosen by Roark 'follow from the intuition (and experience) that what helps parsing is different depending on the constituent that is being expanded' (Roark 2001, p. 295) All functions return either parent or sibling node labels of some specific distance from the left-hand side or head information from c-commanding constituents[5].

When a production is conditioned upon the headword of the constituent, given the left-to-right orientation, this head has often not yet been encountered. In such a case, the head of the last child within the constituent is used as a proxy for the constituent head. All conditional probabilities are estimated using empirically observed relative frequencies and are smoothed using deleted interpolations.

The parsing algorithm is based on a beam search. As a consequence the prefix-probabilities found are not exact probabilities as they are based only on the parses that are found and the corresponding language model is thus not proper as the sum over the vocabulary is less than one. The running time of the parser is polynomial.

Tests showed that the accuracy of the parses is improved when conditioned on more information. It was found that the addition of c-commanding heads has only a moderate effect on the parser accuracy, but a very large effect on the perplexity. Perplexity tests showed that the model does slightly better when used as a language model than the model of Chelba and Jelinek (1998). More improvement is obtained when the model is interpolated with a trigram model, which suggests that the trigram model contains information orthogonal to that of the grammar model. When used as a language model interpolation with some other model is necessary anyway (in this case a unigram was used) since the parser may garden-path which results in a part of the string that receives no probability estimate. Rescoring test on speech recogniser output n-best list showed a 1.7% absolute reduction in word error rate. It is suggested that this modest result stems from the fact that the model relies heavily on grammatical context, which makes it difficult for the model to recover from recognition errors. This once again shows the need for robust language models that can handle partially ungrammatical input.

[5] A node A c-commands a node B if and only if A does not dominate B and the lowest branching node that dominates A also dominates B.

3.13.4 Immediate-head parsing language models

In (Charniak 2001) it is noted that all of the most accurate statistical parsers are lexicalised and are all of what Charniak calls the immediate-head type, i.e. the properties of the immediate descendants of a constituent are assigned probabilities that are conditioned on the lexical head of this constituent. Furthermore most of these parsers are generative, making them suitable in theory to be used as a language model for speech recognition.

However, parsing models that have been utilised as language models for speech recognition such as those described in the previous sections, are not of the immediate-head type, but parse left-to-right, which has the advantage that it can give guidance to the acoustic search process and it allows for interpolation with trigram models. Charniak argues that the latter reason is only valid as long as the interpolated model is better than a stand-alone grammar model and that left-to-right parsing with immediate-head information might be possible by revising probabilities once more knowledge is available. The use of the superior immediate-head parsers may then improve language model perplexity and speech recognition accuracy. Although the paper does not solve the second problem it shows how to extend a statistical parser to include trigram-like information.

A generative probabilistic model is used to calculate the probability of a sentence (Charniak 1999). The constituents in a parse tree are considered top-down head-first. First the POS-tag t (corresponding to the headword) of the constituent c is generated, followed by the lexical head h and the expansion e of the constituent:

$$P(S,T) = \prod_{c \in T} P(t|l,H)P(h|t,l,H)P(e|h,t,l,H). \tag{3.32}$$

Where l is the non-terminal label of the constituent and H captures the relevant history of c (may differ for each of the three components).

Unlike the previous model, that obtained the probabilities of the production rules directly from the treebank, this model adopts the scheme defined by Collins (1997) for expansion of a constituent. In general this scheme can be seen as a Markov grammar that assigns probabilities to any possible expansion. The expansion e of a constituent with label l looks like:

$$l \rightarrow L_{n+1}L_n \ldots L_1 M R_1 \ldots R_m R_{m+1}, \tag{3.33}$$

where L_{n+1} and R_{m+1} are stop symbols and M is the head-child. The probability of this production can now be decomposed in many ways, among which the use of k^{th} order Markov assumptions that would condition the generation of a modifier upon the k previously generated labels.

To test this hypothesis the parser described in the previous section was trained on similar data as the SLM of Chelba and Jelinek (1998; 1999), i.e. a 'speechified' version of the UPenn-treebank and its results were compared with a trigram and the left-to-right models of Chelba and Roark.

The model outperforms earlier models; after interpolation with a trigram it outperforms the grammar alone model and previous interpolated models. However interpolation is at the sentence level as interpolation at the word level is not possible.

Analysis of the results revealed that the trigram model does better on noun-phrases like 'Monday night football'. This can be incorporated in the grammar model by conditioning also on the grandparent node, which has the result of conditioning on two previous words, like the trigram does.

This immediate tri-head model gave a further improvement (14% over the previous best model). The performance is better than that of interpolated models, however interpolation results in further improvement. This is due to the fact that the cases where the trigram model does better are often those cases where the parser fails. Thus improvement of the underlying parser should improve the model's perplexity further.

The superior parsing performance of immediate-head parsers also carries over to speech recognition language models and much of the knowledge incorporated in trigram models can be thus captured within a grammar model. This makes such a model particularly suited for rescoring of word lattices in a multi-pass speech recogniser. However, for tight integration with the acoustic models a way should be found to enable forward parsing, i.e. to generate the words in the sentence from left-to-right.

"So this is our ranking system," said Chomsky. "As you can see, the highest rank is yellow."
"And the new ideas?"
"The green ones? Oh, the green ones don't get a colour until they've had some seasoning. These ones, anyway, are still too angry. Even when they're asleep, they're furious. We've had to kick them out of the dormitories — they're just unmanageable."
"So where are they?"
"Look," said Chomsky, and pointed out of the window. There below, on the lawn, the colourless green ideas slept, furiously.

Flash fiction, Hugh Cook

Chapter 4

Sources of Knowledge

In which types of knowledge that can be of use for speech and language processing are investigated from a theoretical point of view. A definition of what we mean by context is given.

The first time one is confronted with the solution to speech recognition outlined in the previous chapters, the reaction may be one of amazement. Surprise that all there is to it is collecting a bunch of statistics over acoustics and word combinations. The knowledge of language incorporated in speech recognisers remains limited to the lexicon that encodes how words are pronounced. In fact, this thesis grew out of that amazement.

A closer look at these statistical models reveals that they implicitly capture many of the phenomena of language, e.g. triphones capture coarticulation, the lexicon may contain information on pronunciation variation and n-grams capture many important syntactic and semantic collocations and idioms. If anything, the statistical approach to speech recognition is a tribute to the beauty and power of mathematics. Nevertheless, the feeling that there is much unused but potentially useful knowledge available remains. Not only theoretical knowledge of language, but also in the form of information in the environment of a running speech recogniser. Is there no way in which we can benefit from that information to get closer to the goal of human-like speech recognition?

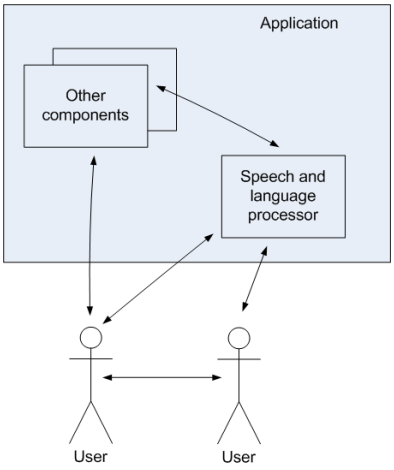


Figure 4.1 – *Context in speech and language processing.*

In this chapter we hypothesise that this is the case. That information provided by the context can be of use in speech and language processing. The chapter starts by defining what we mean by context. Subsequently, we borrow from the fields of linguistics, psychology and sociolinguistics to investigate which information can be of use in speech recognition.

4.1 Context

In chapter 2 speech recognition was introduced as the task of mapping an audio signal to a sequence of words. However, when a speech recogniser or for that matter any language processing system is used it never runs in isolation. Figure 4.1 shows a language processing system in context. Typically, the language processor functions as a front-end to some application. This can be a very specific application such as a dialogue system that provides access to a database or it can be as general as a word processor. The application will have several other components, some of which may interact with the speech recogniser. In particular, there may be other components that act as an interface to the user, such as keyboard and mouse, but also recognition of facial expressions or gestures.

Obviously, the context includes the user or a group of users. Users may interact directly with the speech recogniser, e.g. with a dictation application or the users may interact with each other and the speech recogniser records the interaction, as is the case when transcribing television broadcasts or business meetings. In either case we can talk about a conversation, be it a conversation among users or a conversation between a user and the system. The latter includes the situation where the user engages in a monologue and the computer only listens. Finally, the environment

surrounding the system and its users can be relevant for the speech recogniser as users may be referring to items in their immediate surroundings or for example talk about the news of that day. Based on this discussion we will subdivide context in three sources of knowledge:

1. User knowledge.
2. Conversational knowledge.
3. World knowledge.

We will now take a closer look at each of these knowledge sources and discuss if and how they can be of use for speech recognition. The focus will be on recognising spontaneous speech, as this is the most general instance of the problem and also the task that is most likely to benefit from contextual knowledge because it has little restrictions on its vocabulary and users are often not aware of the presence of a speech recogniser.

4.1.1 User knowledge

Speech characteristics differ per person. The sound of our voices is determined by the physiology of our vocal tracts and we all have our habits when picking the words to formulate a thought. This is why person-dependent speech recognition is so much easier than person-independent speech recognition and why so much effort in speech recognition research has gone in techniques for user-adaptation.

However, sociolinguistic research has shown that speech and language characteristics also have a social function (Boves and Gerritsen 1995). Language is (subconsciously) used to indicate group membership. Sometimes this is obvious: female voices are typically higher pitched than male voices and we are all aware of differences between dialects, but other group specific language use is less apparent.

Below we will discuss the most important sociolinguistic dimensions. Knowledge of these dimensions can be of great benefit to speech and language processing as it might help to adapt a system more quickly to a user than purely acoustic adaptation techniques would. In particular, it allows to better predict the speech characteristics of the user without the need for an extensive set of examples as traditional techniques do. In addition, knowledge of dimensions that separate groups of speakers can help to develop more accurate models of speech. As mentioned in chapter 1, many state-of-the-art speech recognisers already use separate sets of acoustic models for male and female speaker rather than lumping all data together (Woodland *et al.* 1994; Kawahara *et al.* 2000; Pellom 2003).

Dialects and languages Differences in language use by different groups is probably most obvious for dialects. A dialect is a variation of a language spoken in a particular region. Dialect reveals itself most clearly in pronunciation. For example in the region of Amsterdam in the Netherlands /z/ is often devoiced. In the south-eastern part of the Netherlands the velar fricatives /ɣ/ and /x/ are pronounced as

palatal fricatives /j/ and /ç/ respectively. But dialects also differ at other linguistic levels. Words may have a different meaning in different dialects — the word bill means banknote in American English but not in British English — or are used more often in a dialect than in the standard language, e.g. the personal pronoun 'gij' (you) is used mainly in the southern part of the Netherlands. Dialects often include words that are non-existent in the standard language. The related languages Dutch and Flemish differ on the morphological level, in Dutch the diminutive suffix '-je' is used, while in Flemish (and is some southern regions in the Netherlands) '-ke' is used. Although less obvious, dialects may also differ at the syntactic level. For example the order of the verbs in a relative clause is swapped in the Northern part of the Netherlands.

As the world is getting smaller because of increased mobility and modern communication tools such as telephone, internet and television, dialects are moving towards the standard language (Boves and Gerritsen 1995).

Gender Much has been written about differences between male and female speech. The clearest difference is in the pitch of male and female voices. The main cause is physiological, the male vocal chords are about 17–24 mm, while female vocal chords are in the range of 13–17 mm. The difference in length (and mass) leads to a different fundamental frequency. Pulses of the fundamental frequency also differ (Rietveld and van Heuven 1997).

However, pitch of the voice also has a cultural aspect. Van Bezooijen (1995) found that Japanese and American women between 20–30 years old on average had higher pitched voices than Dutch and Swedish women of the same age. The women were selected on length as well to rule out large physiological differences. In a similar experiment it was shown that women, who are successful in the business world often speak with a lower pitched voice than they would naturally.

As mentioned above the difference in the pitch of the voice presents itself right away, but there are other differences. Many have noted a difference in the choice of words (Lakoff 1975; Maltz and Borker 1982). Male speakers allegedly curse more and talk more businesslike about things while women talk more about people and use more adjectives and amplifying adverbs. Boves and Gerritsen (1995) remark that much of this early research is anecdotal and far from conclusive. On the other hand, more recent corpus-based research does reveal different pattern in language use by male and female speakers (Rayson *et al.* 1997; Kilgarriff 2001; Harnqvist *et al.* 2003).

Some of these differences can be traced back to two differences between male and female speakers that do consistently appear in sociolinguistic research: differences in interaction patterns and use of the standard language. In a conversation male speakers more often use minimal answers such as *'mmm'* while female speakers often use phrases showing that they are listening, e.g. *oh really? good for you* and they ask more questions (Brouwer 1991). Male speakers have a tendency to interrupt their (female) conversation partner more often.

Female speakers more often use forms of the standard language than male speak-

ers (Boves and Gerritsen 1995), who more often use dialect. Several explanations for this phenomena have been given. One hypothesis is that women want their children to learn the standard language. Brouwer (1989) found that women with children use less dialect than women without children and in fact men with children use less dialect than men without children (but still more than women). In addition, women often have jobs that require use of the standard language, such as secretary. While male jobs at the same social level, such as construction worker, do not. If these hypotheses are correct, the fact that social differences between men and women are more and more disappearing implies that differences in language use are disappearing.

Social group When hearing someone speak we can typically make a pretty accurate guess not only at the region someone is from but also at his or her social class or group. In general, lower social classes use more dialect than the higher social classes in which it is prestigious to speak the standard language. Social differences show most clearly in pronunciation. Some social groups develop their own language variation to show group membership. For example in the Netherlands one can easily recognise members of student societies. Immigrants often have traces of their native language in their language. Second or third generation immigrants still do, not because they can't help it, but rather as a display of group membership (Rietveld and van Heuven 1997).

Van Bezooijen (1985) found that social class does not only show at the phonetic level, but that even the sound quality of voices has a social aspect. Lower class voices have, on average, a more nasal quality while in higher class voices sound more creaky.

One might expect that higher social classes use longer and grammatically more complex sentences. Van den Broeck (1980) and Jansen (1981) have shown that this is not the case. There is little social variation in spoken language syntax. It is likely that the ability to construct a sentence has more to do with limitations of the short-term memory that affect all of us in an equal way than with social class or gender.

Age Speech of different age groups is very different. Because their speaking organs are not yet full-grown, small children speak with very high pitched voices. In puberty the length of the male vocal chords increases rapidly, lowering the voice by a whole octave. After the age of 60 the fundamental frequency of the voice starts to increase again (Rietveld and van Heuven 1997). The number of syllables per second also increases during childhood and decreases again for elderly people.

Language is constantly changing, new words appear and others disappear or get a different meaning. Language change is strongly related to age (Boves and Gerritsen 1995). Especially young people are notoriously known for the use of their own jargon. When they get older people stop adapting their language to new forms. Language change is a gradual effect, the change takes place on a word by word basis. This is called lexical diffusion.

The most salient changes are at the lexical level, but in fact language changes at all levels: phonological, morphological and syntactic. For example, in the Netherlands the tendency to pronounce long vowels such as /eː/ as diphthongs is gaining momentum (Jacobi *et al.* 2005; van Heuven *et al.* 2005), e.g. /beːt/ 'bit' becomes [bɛit]. The diphthong /ɛi/ in turn is lowered to [ai]: /tɛid/ 'time' becomes [taid]. Stroop (1990), who dubbed the phenomena *Polder Dutch*, found that this chance was initiated by higher educated middle class women.

There is an interplay between dialect and age (Boves and Gerritsen 1995). Younger and older people have a tendency to use more dialect than people between the ages of 25 and 55. An explanation for this effect, called age-grading, might be that people with small children and a job take more effort to speak the standard language. The effect has often been mistaken for language change, in particular as an indication that dialect is disappearing.

4.1.2 Conversational knowledge

Language variation not only depends on speakers, but also on the situation. The same person may use very different styles of speaking depending on the background and purpose of a conversation.

Speaking style Situations can be ordered on a continuous scale from informal to formal (Boves and Gerritsen 1995). In an informal conversation with family or friends for example speech is typically more sloppy, with more hesitations and restarts and fragments rather than full sentences and with more use of dialect. Labov (1972) hypothesises that the formality of a conversation can be determined by the amount of attention we give to the correctness of our language. Word use also depends on the speaking style. Boves and Gerritsen (1995) give the example of accusing someone of lying. In an informal situation (1) will do just fine, while in a formal setting one could phrase this as (2).

(1) you're lying!

(2) that seems a rather subjective description of the situation to me.

According to Bell (1984) not (only) the amount of effort we spend on the correctness of our speech determines the style of speech, but the audience we are talking to is of importance. For example, imagine someone asking for the time, if he would be asking a stranger, even in an informal setting, he would likely use a polite phrase such as (3). If the same person in the same situation would be asking a friend he might phrase the request as (4).

(3) I'm sorry, could you please tell me what time it is?

(4) Heh, what's the time?

Differences in language between social groups become smaller in formal settings than in informal settings. In fact, people who normally do not use the standard language (or prestige dialect) may display hypercorrect language use in formal setting. They use more formal terms than speakers of the standard language would in the same situation (Boves and Gerritsen 1995).

Speaking style can change within a single conversation. Imagine for example a student taking an oral exam. Before the exam the teacher might use informal language to make the student feel at ease. During the exam the language can become much more formal. After the exam the student and the teacher may engage in informal chit-chat again.

Interaction Related to speaking style is the type of interaction people are engaged in. People tend to change their language use to improve communication (Boves and Gerritsen 1995). For example a lower class servant in a restaurant may accommodate his language to that of his upper class clients. In a conversation people will search for common ground (Jurafsky and Martin 2000). Initially, they might use different terms to describe the same objects, but after a while they will converge to the same terms.

For speech recognition it greatly matters whether people know they are talking to a machine, as in dictation task, or not. People talking to a speech recogniser will often use more formal speech. They may start to hyper-articulate (Oviatt *et al.* 1996).

Topic All conversations revolve around certain topics. The vocabulary used clearly depends on the topic. This especially holds for content words and phrases such as 'war on terrorism'. The topic of an actual conversation will rarely be a pure topic in the sense that it can be named with a single label, rather conversations involve a mix of several topics that gradually changes during the discourse.

4.1.3 World knowledge

Aside from user and conversational knowledge, information from other sources can be available to a speech recogniser. Typically this will be background knowledge of a particular domain or information from other modalities that can help to focus the speech recognition. As an example, think of a hand-held computer that combines pointing and speech in a route planning application (Fitrianie *et al.* 2007).

Other modalities Psycholinguistic experiments have shown that in human speech recognition information from other modalities, in particular the visual modality, is used. For example Tanenhaus *et al.* (1995) describe a number of experiments that show how the visual context constrains acoustic processing. They built up scenes containing a number of items with different shapes and colours, such as an apple on a white towel. The eye movements of test subjects were tracked, while they listened to assignments such as 'put the apple on the towel in the box'. Tanenhaus

found that people visually focus on an item even before it is uniquely identified. For example, if there is just one apple that is on a towel the eye will focus on this apple while the word towel is uttered. Taking into account the time it takes to program and perform an eye movement it means that the item is identified before the word 'towel' is completely finished.

Another example of cross-influence between modalities is the combination of speech recognition and lip-reading. Once again psychological results show that information from both modalities is combined in human recognition. This is best demonstrated by the McGurk-effect: most people that are exposed to the acoustic sequence /ba/ while seeing a video where someone utters /ga/ will typically perceive the sound /da/.

Emotion recognition (Batliner *et al.* 2000) and speech recognition can influence each other to mutual benefit. Words have emotional content (Wojdeł 2005) that should match that of the non-verbal 'signals' (Rothkrantz *et al.* 2004). Emotion recognition is needed to deal with irony or jokes in general as such situations often cannot be deduced from the verbal content.

The examples given above illustrate that multi-modal integration can take place at different levels in the recognition process. As most modalities provide streams of information, timing information plays an important role in multi-modal integration. This is also the main source of much of the difficulties of multi-modal integration, as modalities normally do not evolve in lockstep; they have different sampling rates and may be asynchronous. For example if someone names an item with pointing at it with a mouse, the mouse cursor may have been on the item before, during or after the phrase that refers to it is uttered.

Background knowledge Whenever we engage in a conversation we carry with us all kinds of background information with regard to the way the world around us works. In addition, we have beliefs and expectations with regard to the background knowledge of our conversational partners and the course the conversation will take.

For human-machine interaction, the background of a conversation equals the knowledge provided by the application used. This includes much static knowledge, like the vocabulary used, but the state of an application may also provide dynamic constraints on what a user is likely to say. For example, many computer applications involve managing collections of objects, for example files and directories in an operating system or cubes, balls and holes in a 3D-graphical design program. Typical commands include: create, delete and move. Move and delete will take as their arguments items that are already available, while create can take any argument. In a design program spatial relations will also be relevant, for example if a phrase starts with 'the red cube on top of' then what follows must be an item that is below a red cube.

4.2 Language structure

Context influences language at various linguistic levels. In this section a brief overview of several important concepts from linguistics is given, on the one hand because a model for speech recognition may need some additional structure to allow integration of context information at the proper level, on the other hand because including linguistic structure can be useful by itself as is clear from the models discussed in chapter 3. Even though linguistic theories do not yet cover all language phenomena and even though we cannot include all linguistic information available, including part of these theories may help us to build better speech recognisers. In linguistics a distinction between several levels of knowledge in language is made (Fromkin and Rodman 1993; Jurafsky and Martin 2000):

Phonology Phonology deals with the pronunciation of words. It studies how pronunciation changes in phonetic context. Such knowledge is phrased in phonological rules. In speech recognition, the phonological level is typically not explicitly modelled, rather it is combined with phonetic information on the sounds of language. However, to model language variations such as Polder Dutch, introducing a phonological level may be desirable.

Morphology Morphology is the theory of the substructure of words in terms of meaning bearing components (morphs). Speech recognisers usually treat all inflections of a word as separate entities, as from a speech recognition point of view it is the pronunciation that matters. However, from a semantic viewpoint as taken for example when including the topic of a conversation in a model, it does not really matter whether a verb is in third person singular in the past tense or in first person plural present tense.

Syntax Syntax describes the way words group together in phrase. Syntactic structure is useful by itself, as it constrains utterances to grammatically correct utterances, but in speech recognition it has often been discarded as a useful information source for this very reason, arguing that spoken language is not grammatically correct. The power of grammar lies in its ability to name the roles that words play in a sentence and the way they relate to each other.

When it comes to including contextual knowledge in the speech recognition process syntax has another use. It identifies those phrases and words that relate to higher-level information. For example, the way function words are used does not depend upon the topic of a conversation but the nouns and verbs that are (semantic) headwords of the constituents of a sentence may. In the same way the type of conversation may influence the sentence structure. For example, read speech can contain long grammatically complex sentences, while spontaneous speech will involve short, more free formed sentences.

Semantics Semantics is the theory of meaning. In linguistics and natural language processing it is usually taken to be the subfield that deals with the meaning of individual words and how those combine to yield an interpretation for phrases and sentences.

Lexical semantics describes the meaning of individual words and defines the relations between those meanings, such as: synonyms, analogies, antonyms and sub-superclass relations. Knowledge of such classes is useful, as words in the same class tend to occur in similar contexts. It thus generalises the properties of words among classes.

Spoken language contains many idiomatic phrases or word combinations that occur more often than other word pairs, for example 'strong tea', 'stiff wind' (Manning and Schütze 1999). Often, the meaning of such idioms cannot be found by combining the meanings of the words involved. For example, there is no indication in the phrase 'to kick the bucket' that points to its meaning of dying. If these words would be treated as separate entities models based for example upon the topic of conversation will make wrong predictions.

Pragmatics Pragmatics captures the semantic relationships within a text beyond the sentence level. Much contextual knowledge, like topic and conversation style can be put at this level. Following Austin (1962) a discourse is often described as a sequence of actions. Whenever someone is uttering a sentence three different actions are performed:

1. The locutionary act, which is the physical act of uttering a sentence.
2. The illocutionary act, which is the linguistic act that a speaker performs.
3. The perlocutionary act, which is the act that results from the utterance.

Speech acts, or in the context of dialogue systems dialogue acts, refer to the illocutionary acts, examples are statements, questions, answers and requests. Speech acts help structuring a discourse, as not every speech act can follow after every other speech act, for example after a question an answer is expected. Several authors have therefore proposed to treat discourse structure as a sequence of dialogue acts and some have attempted to use such a sequence to improve speech recognition (Stolcke *et al.* 1998). The line of reasoning behind this approach is that the structure of a sentence clearly depends upon the speech act, for example in questions the subject and the direct object swap positions and backchannels are often short non-speech utterances like 'uh-huh'. However, while automatic dialogue act classification works relatively well (although still much worse than human labelling) this does not carry over to clear improvements in speech recognition (Stolcke *et al.* 2000). The likely reason is that a small number of speech acts captures the majority of all clauses, in particular 36% of all clauses in the corpus of Stolcke *et al.* (2000) are statements, thus a language model based upon a mixture of dialogue act language models comes very close to a language model that is simply trained on all data. Nevertheless, it

has also been shown that if the dialogue act is known it does help for the case of less frequent dialogue acts, therefore it might be useful in combinations with other context factors.

Syntax describes the way word sequences are structured at the sentence level. Language is also structured beyond the sentence level. Dialogue acts capture a particular aspect (the illocutionary force) of this. When looking at written text discourse structure is obvious, sentences group together in paragraphs that group together in sections that may be part of several larger sections that are in turn part of a chapter. Spoken language has a similar structure. Sentence group together in segments that are all concerned with the same items. Such segments may be subsegements of larger segments. Showing that a discourse should be organized as a stack. This especially is the case as a discourse may contain digressions to some of-the-topic items, after which the conversation continues where it left of.

Knowing the structure of a discourse is helpful as it shows which parts of the discourse history are relevant for the prediction of the current sentence. Unfortunately, much discourse structure is causal and can only be uncovered by semantic reasoning, which can be done for small well-understood domains, but is practically impossible in the general case of spontaneous speech. Some of this may be recovered by identifying topic boundaries using textual coherence properties. Moreover, some of the rhetorical structure can be found by locating cue words, e.g. 'by the way' signals a digression away from the main topic, while 'anyway' signals a return to the main topic. Which cue words are important may depend upon the type of conversation. The temporal structure of a discourse can also be used to reveal discourse structure, as tenses cannot arbitrarily be mixed (Allen 1995).

The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' (I've found it!), but 'That's funny . . . '.

Isaac Asimov

Chapter 5

Data Analysis

In which the Corpus Spoken Dutch (CGN) is introduced. To establish whether the influence of context on language can be observed in a data set, we performed an analysis of word use and sentence length in different subset of the corpus. The influence of a speakers' gender, education level and age on word use in spontaneous spoken Dutch as well as in broadcast recordings of Dutch was also investigated. This chapter presents the results of the analysis.

In the previous chapter the influence of context on language use was investigated from a theoretical perspective. The question that remains is how strong these effects are in everyday language and more specifically how strong these effects are in a limited data set that is used to construct automated language processing systems with. In addition, we would like to know the interrelationships between the context factors.

In this chapter we analyse a large corpus of spoken language to investigate if and how the type of speech influences word use (Wiggers and Rothkrantz 2007*a*;*b*). We identify features of the text that allow us to determine the type of speech. For different types of speech we look at the influence of speaker characteristics such as gender, education level and age on language variation. In all cases we compare results on Dutch as it is spoken in the Netherlands and Dutch as it is spoken in Flanders

to find out whether effects are dialect-dependent or not. We will refer to these two variations of the language as Northern Dutch and Southern Dutch respectively.

In this chapter we take a data mining approach, i.e. rather than formulating and testing hypotheses we explore the data using statistical methods in order to get more insight in the data. This will help us to identify important variables and dependencies and to detect outliers and anomalies. The results found can then be used to formulate hypotheses about the usefulness of certain variables for speech recognition on which new models can be based. In addition, insight in the types of variables these models will contain and their interplay will help to formulate requirements for a computational framework to build models in.

Note that the goal of this analysis is to see whether context effects can be found in a corpus of speech, not to form a theory of context effects in language. The analysis is performed on a limited data set, care should be taken to generalise the findings as it will be hard to separate artifacts of the data set from real context effects. Nevertheless, these results can guide the direction of future research.

5.1 The Spoken Dutch Corpus

The corpus used is the Spoken Dutch Corpus (Corpus Gesproken Netherlands or CGN) Oostdijk *et al.* (2002); Schuurman *et al.* (2003). A corpus of almost nine million words of standard Dutch as spoken in the Netherlands and Flanders. The corpus is subdivided in 15 components that contain different types of speech, ranging from spontaneous conversations to more formal speech such as sermons and read speech. Table 5.1 gives an overview of the components. It lists the total number of words

Table 5.1 – *The components of the Spoken Dutch Corpus. For each component the number of words collected in the Netherlands and in Flanders is given.*

	component	Netherlands	Flanders
a.	spontaneous conversations ('face-to-face')	1,747,789	878,383
b.	interviews with teachers of Dutch	249,879	315,554
c.	spontaneous telephone dialogues (switchboard)	743,537	465,096
d.	spontaneous telephone dialogues (mini disc)	510,204	343,167
e.	simulated business negotiations	136,461	0
f.	interviews/discussions/debates (broadcast)	539,561	250,708
g.	(political) discussions/debates/meetings	221,509	138,819
h.	lessons recorded in the classroom	299,973	105,436
i.	live (e.g. sports) commentaries (broadcast)	130,377	78,022
j.	news reports (broadcast)	90,866	95,206
k.	news (broadcast)	285,298	82,855
l.	commentaries/columns/reviews (broadcast)	80,167	65,386
m.	ceremonial speeches/sermons	5,565	12,510
n.	lectures/seminars	61,834	79,067
o.	read speech	551,624	351,419

in each category as well as the number of words collected in the Netherlands and in Flanders respectively.

5.2 Methodology

In this chapter we are mainly interested in words and part-of-speech (POS) tags that are most characteristic for a particular subset of the data. To operationalise this notion we use log-likelihood ratios (Dunning 1993) defined by:

$$G^2 = 2 \sum_i O_i \ln(\frac{O_i}{E_i}) \tag{5.1}$$

Where O_i is the frequency observed in a cell of a contingency table and E_i is the corresponding expected value:

$$E_i = \frac{N_j \sum_i O_i}{\sum_j N_j} \tag{5.2}$$

N_j is the total frequency in a column of the table and i ranges over all elements in a row of the table. The log-likelihood ratio (LLR) can be thought of as a measure of surprise and is related to the χ^2-test, but more reliable than the latter in the face of sparse data. The values of G^2 are approximately χ^2 distributed. It can be used in the hypothesis testing framework as a measure of statistical significance. However, as pointed out by Kilgarriff (2001) among others, the null hypothesis that subsets are drawn at random from the same underlying population is almost always defeated when looking at linguistic phenomena because language use clearly is not random. Outside the hypothesis testing framework likelihood ratios are nevertheless useful as a way to rank words in order of distinctiveness as shown in Kilgarriff (2001); Rayson and Garside (2000); Daille (1995).

Unless otherwise noted, before collecting statistics on word use hapax legomena are removed from the vocabulary as these are typically topic related and can be thought of as artifacts of the limited size of the data set rather than as words that are specific for a conversation type or a group of speakers.

5.3 Related work

There is a large body of work in corpus analysis, e.g. Rayson *et al.* (1997) investigates social differentiation defined by factors as gender, age and social group, in the use of English vocabulary using the British national corpus. They also look at the difference between spoken and written text. Kilgarriff (2001) is concerned with comparing corpora and looks at the difference between male and female word use, as does Harnqvist *et al.* (2003) for Swedish. The CGN is also used in Binnenpoorte *et al.* (2005) and van Gijsel *et al.* (2006). In Binnenpoorte *et al.* (2005) gender differences in speech rate and the use of fillers and nouns is investigated. They

find that male speakers use more fillers and female speakers use more pronouns. The work of van Gijsel *et al.* (2006) uses multivariate analysis to investigate lexical richness. They find that more formal speech (components k and m of the corpus) has high lexical richness, while spontaneous speech does not. In addition, they note that female speech has lower lexical richness than male speech. Hypothesising that women elaborate longer on a particular topic than men would.

5.4 Type of Speech

To investigate differences between types of speech we analyse the differences in sentence length, part-of-speech and word distributions between components of the corpus. For word distributions we remove all words that occur less than five times in the whole corpus to diminish topic dependency.

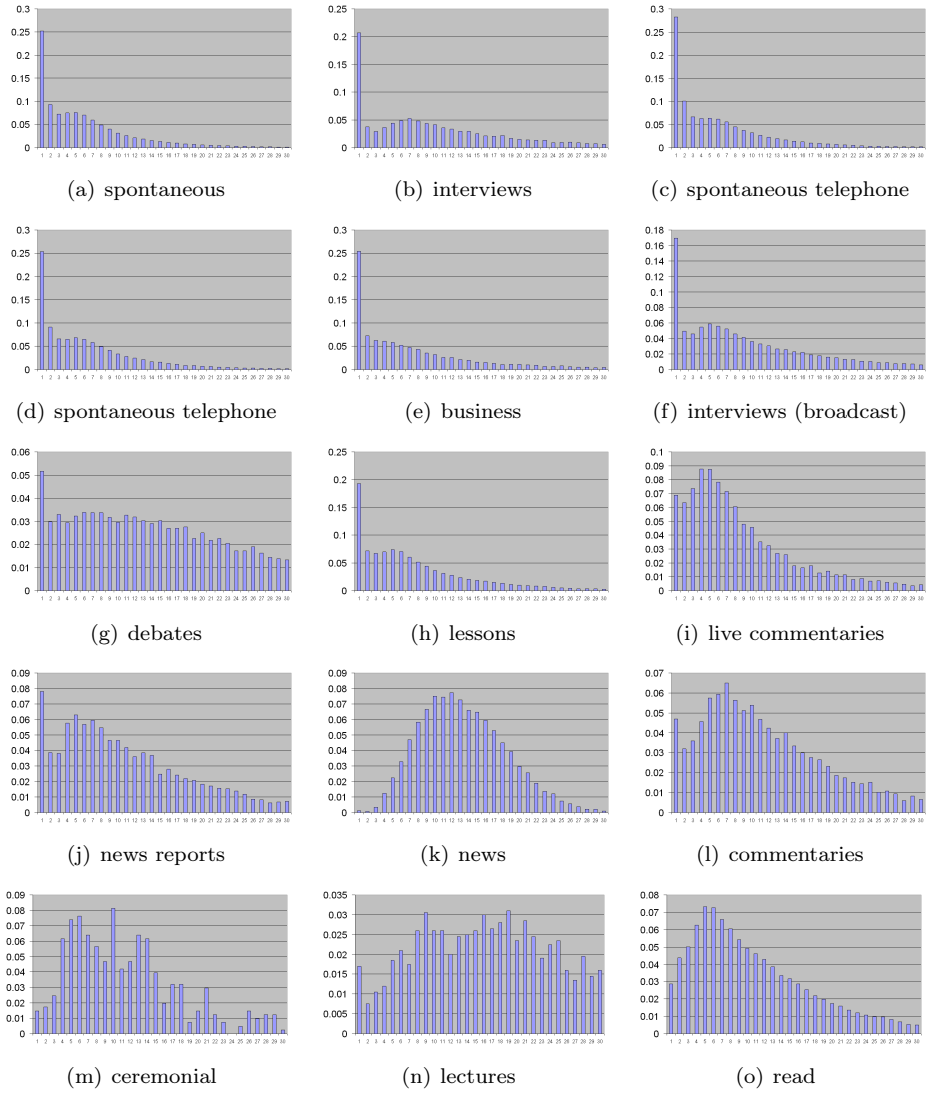
5.4.1 Sentence Length

We expect sentence length to be a good indicator of the type of speech, therefore we analyse sentence lengths in the subcorpora. Table 5.2 summarises the results for Northern Dutch. As one would expect sentences in spontaneous speech (components

Table 5.2 – *Sentence length statistics per category of the* CGN *for Northern Dutch.*

component		average	sample dev.	mode	median	IQR
a	(spontaneous face-to-face)	6.42	6.70	1	5	8
b	(interviews)	11.34	11.09	1	8	13
c	(spontaneous telephone)	6.37	6.76	1	4	8
d	(spontaneous telephone)	6.80	7.03	1	5	8
e	(business negotiations)	8.59	9.85	1	5	11
f	(interview broadcast)	11.29	11.45	1	8	13
g	(debates)	20.05	17.20	1	16	19
h	(lessons)	7.98	7.72	1	6	9
i	(live commentaries)	9.90	9.72	4	7	8
j	(news reports)	12.20	9.98	1	10	12
k	(news)	13.46	5.39	12	13	7
l	(commentaries broadcast)	13.04	10.65	7	10	11
m	(ceremonial)	11.76	7.22	10	10	9
n	(lectures)	28.15	21.16	19	23	24
o	(read)	11.51	8.61	5	9	10

a, *c* and *d*) are shorter on average than sentences in formal speech. Lectures (n) and political debates (g) contain the longest sentences on average. When taking a closer look we find that for all components the sentence length distributions are highly skewed, sentences can occasionally get very long.



(a) spontaneous (b) interviews (c) spontaneous telephone

(d) spontaneous telephone (e) business (f) interviews (broadcast)

(g) debates (h) lessons (i) live commentaries

(j) news reports (k) news (l) commentaries

(m) ceremonial (n) lectures (o) read

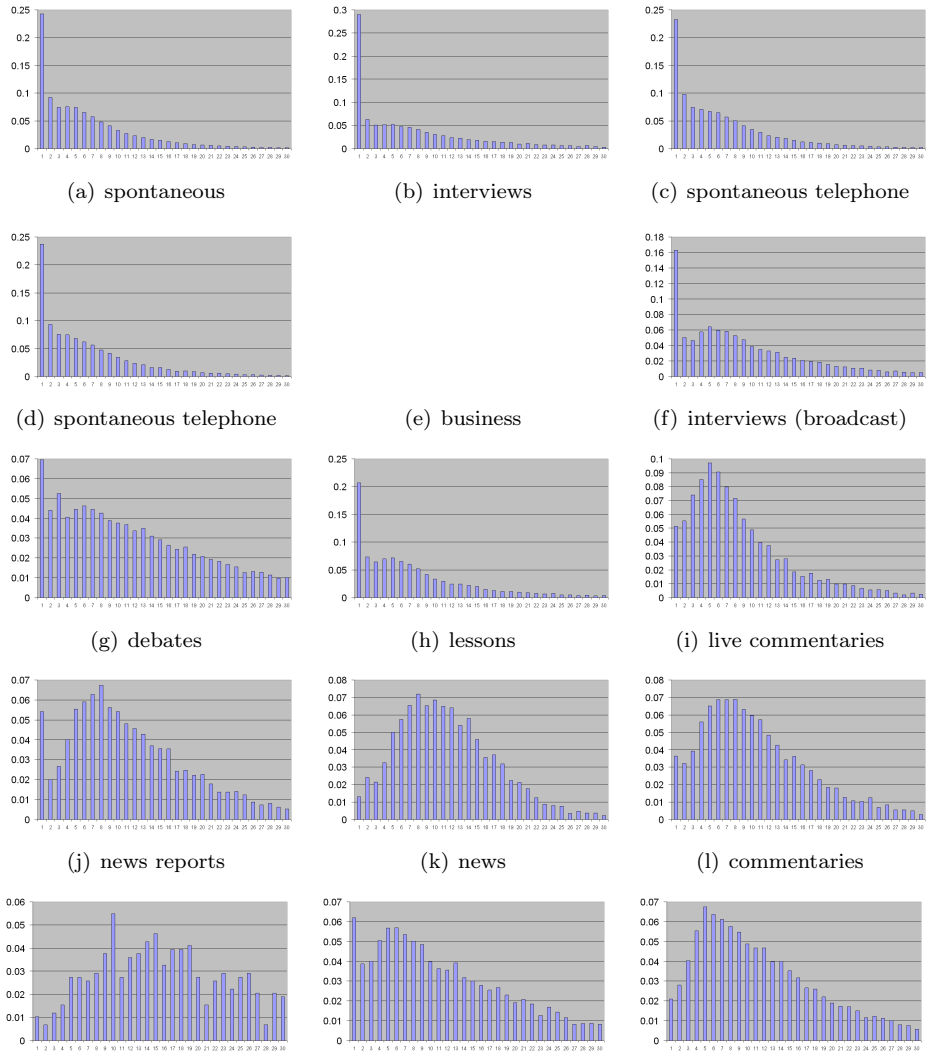
Figure 5.1 – *Sentence length distributions for components of the* CGN *(Northern Dutch).*

Therefore, the median and interquartile range (IQR) are better indicators of sentence distribution than mean and sample deviation. Figure 5.1 shows the distributions (up to sentence length 30) for all components of the Northern Dutch subcorpus. The components that contain spontaneous speech (*a*, *c* and *d*) have very similar distributions. About a quarter of all sentences are single word sentences containing interjections, yes/no answers and backchannels. For the remaining lengths there is a peak around length 6. Lessons (component *h*) and interviews (components *b* and *f*) also display the pattern of spontaneous speech. However, interviews contain longer sentences. The components containing broadcast news (*i*, *j* and *l*) have very similar sentence length distributions that are clearly different from those of spontaneous speech. It is interesting to note that reviews (component *l*) contain longer sentences than news reports (component *j*), which in turn contain longer sentences than live commentaries (component *i*). Debates *g* show a more uniform distribution over sentence length, while news *k* has a nicely balanced bell-shaped distribution, showing that anchormen and news reporters think about sentence length when writing their texts. Components *m* and *n* have very random distributions. This might be a consequence of the small size of these data sets more than anything else.

All in all, sentence length distributions differ for different types of speech. In particular the sentence length distribution of spontaneous speech is clearly different from the other sentence length distributions. Compared with spontaneous speech, the distributions of broadcast speech is closer to that of read speech. In fact, the transition is gradual, with interviews closer to spontaneous speech and reviews closest to read speech.

Table 5.3 – *Sentence length statistics per category of the* CGN *for Southern Dutch.*

component		average	sample dev.	mode	median	IQR
a	(spontaneous face-to-face)	6.65	6.85	1	2	7
b	(interviews)	8.50	9.54	1	5	11
c	(spontaneous telephone)	6.89	7.06	1	5	7
d	(spontaneous telephone)	6.91	7.33	1	5	7
e	(business negotiations)	-	-	-	-	-
f	(interviews broadcast)	10.27	10.03	1	8	11
g	(debates)	15.06	13.39	1	12	16
h	(lessons)	8.05	8.12	1	6	9
i	(live commentaries)	8.96	7.08	5	7	8
j	(news reports)	13.09	10.38	8	11	11
k	(news)	12.05	7.07	8	11	8
l	(interviews broadcast)	11.13	7.58	8	10	9
m	(ceremonies)	19.92	13.54	10	17	15
n	(lectures)	13.46	11.04	1	11	12
o	(read)	12.79	9.34	5	11	11



(a) spontaneous (b) interviews (c) spontaneous telephone

(d) spontaneous telephone (e) business (f) interviews (broadcast)

(g) debates (h) lessons (i) live commentaries

(j) news reports (k) news (l) commentaries

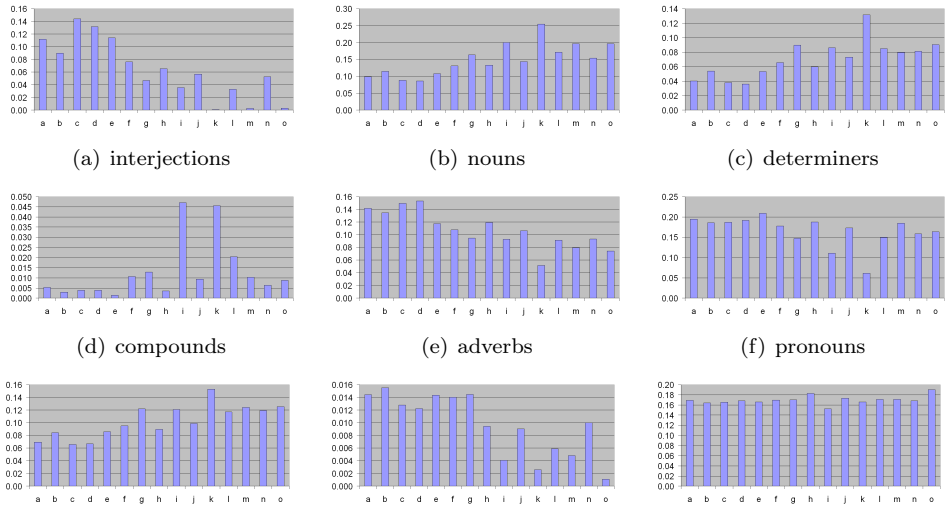
(m) ceremonial (n) lectures (o) read

Figure 5.2 – *Sentence length distributions for components of the* CGN *(Southern Dutch).*

Table 5.3 shows summary statistics for sentence length in Southern Dutch and figure 5.2 shows the sentence length distributions of all components for Southern Dutch. The patterns are very similar to those of Northern Dutch, but there are a number of small differences. News (k) has the same bell-shaped distribution, but with a shorter average sentence length. The same holds for interviews (l). The distribution of debates (g) is less uniform than in Northern Dutch. The sentence length distributions of component n is smoother for Southern Dutch than for Northern Dutch, although not as smooth as other distributions, it roughly shows the pattern of more formal speech. Component m on the other hand contains more short sentences. The distribution of component m differs a lot from its Northern Dutch counterpart, since these two sets contain different types of speech.

5.4.2 Part-of-Speech

We ranked part-of-speech (POS) categories using likelihood ratios and found that interjections, nouns, determiners, adverbs and pronouns best differentiate between subcorpora. Figure 5.3 compares the relative frequencies of the components for several POS-tags. It can be observed that interjections and adverbs as well as incomplete



(a) interjections (b) nouns (c) determiners

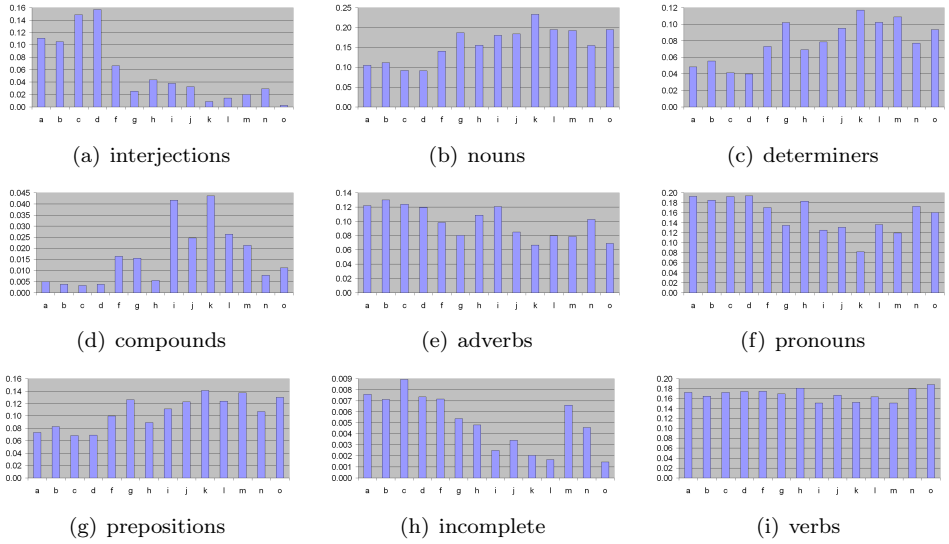
(d) compounds (e) adverbs (f) pronouns

(g) prepositions (h) incomplete (i) verbs

Figure 5.3 – *Relative frequencies of* POS*-tags in the components of the* CGN *(Northern Dutch).*

words are much more common in spontaneous speech while nouns and determiners are characteristic of more formal and read speech. Pronouns occur less in broadcast categories; in particular news contains little pronouns. Note that the relative number of verbs is more or less equal for all components.

Figure 5.4 shows how parts-of-speech are distributed in the component for Southern Dutch. The differences are similar to those for Northern Dutch.



(a) interjections (b) nouns (c) determiners

(d) compounds (e) adverbs (f) pronouns

(g) prepositions (h) incomplete (i) verbs

Figure 5.4 – *Relative frequencies of* POS*-tags in the components of the* CGN *(Southern Dutch).*

5.4.3 Words

For Northern Dutch, the following words differentiate most based on likelihood ratios: *ja* (yes), *de* (the), *het* (the), *uh* (filler), *voor* (for), *xxx* (incomprehensible), *ik* (me), *van* (of), *nee* (no), *je* (you), *nou* (well), *in* (in), *haar* (her), *hij* (he), *u* (you, polite).

Articles, interjections and (personal) pronouns make up most of this list. We further investigated the words in these classes. Figure 5.5 compares relative frequencies of some words in the components of the corpus.

As can be seen, the personal pronouns are very good indicators of the type of speech. The first person singular *ik* and its abbreviated form *'k* are most used in spontaneous speech. The colloquial second person singular *je* is used slightly more often in spontaneous speech than in formal and read speech. It is used most often in lessons. An explanation for this is that in contemporary Northern Dutch *je* is also used as an indefinite pronoun. The second person singular *jij* is used most in discussions and debates, whereas the polite form *u* is used most often in interviews and formal speech. Third person singular is used most in read speech (stories). The word *ze* which can mean either 'she' or 'they' occurs relatively often in spontaneous speech. It is interesting to note that *ie*, the colloquial version of 'he', occurs relatively

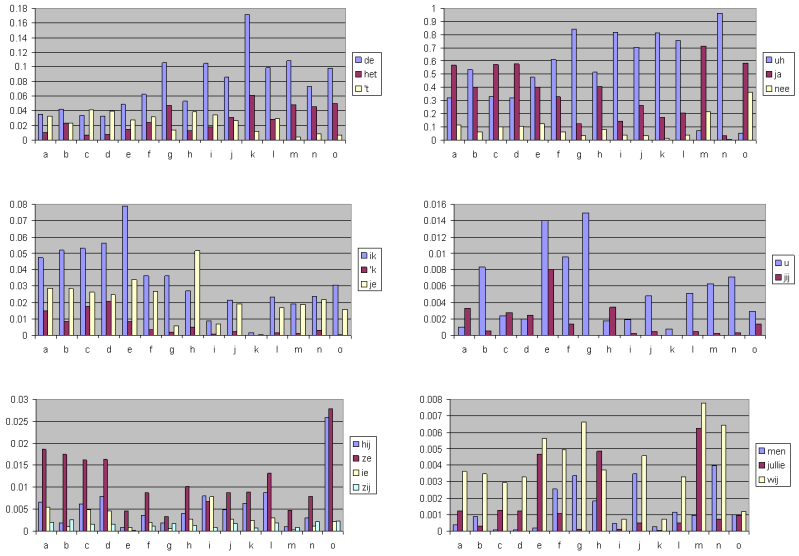


Figure 5.5 – *Distributions of common words in components of the* CGN.

often in sports commentaries. Finally, *wij* (we) is used most prominently in debates and formal speech.

5.5 Dialect

5.5.1 Part-of-Speech

The previous section showed that, although there are some differences for Northern Dutch and Southern Dutch, sentence length distributions depend on the type of speech rather than on the language. Similarities are also found in the relative use of parts-of-speech for different types of speech. Table 5.4 compares part-of-speech distributions of Northern Dutch and Southern Dutch for spontaneous speech (components *a*, *c* and *d*). The general patterns are similar, but note that in Northern Dutch 2% more adverbs and 1% more adjectives are used. Southern Dutch speakers use more conjunctions, determiners and nouns as well as more interjections. Table 5.5 compares part-of-speech distributions per language for broadcast speech. Once again, Southern Dutch uses more nouns and determiners. In this case the Northern Dutch use more interjections, and more pronouns. We might say that in both cases Northern Dutch speakers use more colloquial speech than Southern Dutch speakers.

Table 5.4 – *Part-of-speech distributions of spontaneous speech per dialect.*

part-of-speech	Northern Dutch	Southern Dutch	LLR
adverbs	0.1457	0.1222	4331
incomprehensible	0.0068	0.0130	3791
incomplete	0.0137	0.0079	2781
adjectives	0.0621	0.0527	1476
conjunctions	0.0665	0.0741	809
determiners	0.0390	0.0447	720
interjections	0.1232	0.1310	500
nouns	0.0941	0.0985	209
verbs	0.1683	0.1729	133
prepositions	0.0680	0.0710	131
foreign	0.0025	0.0020	88
compound	0.0048	0.0043	46
numerals	0.0126	0.0127	1
pronouns	0.1925	0.1929	1

Table 5.5 – *Part-of-speech distributions of broadcast speech per dialect.*

part-of-speech	Northern Dutch	Southern Dutch	LLR
interjections	0.0687	0.0505	1333
incomplete	0.0125	0.0055	1254
compounds	0.0116	0.0199	1027
nouns	0.1369	0.1589	868
determiners	0.0688	0.0828	635
pronouns	0.1736	0.1561	506
adverbs	0.1056	0.0922	450
prepositions	0.0980	0.1087	281
foreign	0.0020	0.0037	250
conjunctions	0.0671	0.0629	65
numerals	0.0119	0.0135	46
adjectives	0.0699	0.0716	10
incomprehensible	0.0028	0.0026	4
verbs	0.1704	0.1712	1

5.5.2 Words

The biggest difference between Northern Dutch and Southern Dutch is the vocabulary. Even though the two languages share the same dictionary and speakers of the two languages have no difficulty understanding each other (as long as they speak the standard languages) there are a number of language specific words that do show especially in spontaneous speech. Tables 5.6 and 5.7 show the words with the highest likelihood ratios that are used most often in Northern Dutch respectively Southern Dutch when the components containing spontaneous speech are compared for the two languages.

Table 5.6 – *The frequencies for Northern Dutch and Southern Dutch of the 20 words with the highest likelihood ratio for Northern Dutch words in spontaneous speech.*

	Northern Dutch		Southern Dutch		
word	freq.	rel.freq.	freq.	rel.freq.	LLR
nou(*now, 'well ... '*)	29052	0.0117	64	0.0000	25210
je (*you,one*)	47109	0.0190	6436	0.0046	15970
oh (interjection)	22342	0.0090	1760	0.0013	10950
uh (interjection)	76570	0.0309	23599	0.0170	7288
ie (*he*)	9193	0.0037	798	0.0006	4263
even (*'quickly'*)	5277	0.0021	213	0.0002	3336
toen (*then*)	9565	0.0039	1518	0.0011	2777
leuk (*nice*)	5476	0.0022	408	0.0003	2749
gewoon (*normal*, interjection)	11550	0.0047	2458	0.0018	2310
jij (*you*)	5376	0.0021	535	0.0004	2294
helemaal (*totally*)	4855	0.0020	518	0.0004	1977
weer (*again*)	8032	0.0032	1695	0.0012	1626
wel (*indeed*)	30554	0.0124	11300	0.0081	1530
ik (*I*)	69157	0.0280	29808	0.0215	1527
net (*just*)	2832	0.0011	221	0.0002	1388
zo'n (*like*)	4894	0.0020	802	0.0006	1371
effe (*'quickly'*)	2133	0.0009	124	0.0001	1193
lekker (*nice, good*)	2334	0.0009	172	0.0001	1177
hoor (*indeed*, interjection)	5497	0.0022	1152	0.0008	1123
hele (*whole,large*)	3173	0.0013	416	0.0002	1103

These tables contain language specific words (e.g. the Southern Dutch word 'allee'), but they also make clear that words that both languages have in common are used with different frequencies.

Table 5.7 – *The frequencies for Northern Dutch and Southern Dutch of the 20 words with the highest likelihood ratio for Southern Dutch in spontaneous speech.*

	Northern Dutch		Southern Dutch		
word	freq.	rel.freq.	freq.	rel.freq.	LLR
ge (*you*)	412	0.0002	10100	0.0073	17610
ah (interjection)	1948	0.0008	11149	0.0080	13590
allee	5	0.0001	6090	0.0044	12410
ne (*a*)	12	0.0000	3768	0.0027	7570
hè (interjection)	11504	0.0046	16633	0.0120	6279
het (*the*	6415	0.0026	11130	0.0080	5486
uhu (interjection)	117	0.0000	2894	0.0021	5044
hé (interjection)	1894	0.0008	5633	0.0041	4737
gij (*you*)	211	0.0001	2823	0.0020	4439
den (*the*)	373	0.0002	2771	0.0020	3718
nu (*now*)	4482	0.0018	7457	0.0054	3467
'k (*I*)	15895	0.0064	16543	0.0119	3093
uw (*your*)	115	0.0000	1730	0.0015	2785
nen (*a*)	29	0.0000	1341	0.0010	2491
zijt (*are*)	4	0.0000	1153	0.0008	2312
dat (*that*)	74316	0.0300	54343	0.0392	2255
hm (interjection)	5	0.0000	1053	0.0008	2097
da (*that*)	6963	0.0028	7740	0.0056	1711
's	8244	0.0033	8678	0.0063	1667
u (*you*)	969	0.0004	2092	0.0015	1325

Table 5.8 – *Summary statistics of sentence length for male and female speakers.*

gender	average	sample dev.	mode	median	IQR
male	8.8	9.1	1	6	10
female	7.5	7.8	1	5	8

5.6 Gender

To analyse the influence of user characteristics on word use we select two subsets from the corpus: spontaneous speech (components *a*,*c* and *d*) and interviews and discussion broadcasted on radio and television (components *f*, *j* and *l*). We select at random an equal number of words from an equal number of male and female speakers in order to reduce the influence of text length. It turns out that there is little difference in sentence length between male and female speakers. Men use slightly longer sentences than women on average as shown in table 5.8, but there is also more variation in sentence length for male speakers.

5.6.1 Part-of-speech

In table 5.9 the frequencies of POS-categories of spontaneous speech for male and female speakers are shown, ranked by likelihood ratio. As was also found by Bin-

Table 5.9 – *Relative frequencies of* POS*-tags in spontaneous speech for male and female speakers ordered by likelihood ratios.*

part-of-speech	male	female	LLR
adverbs	0.1389	0.1509	713
interjections	0.1291	0.1192	555
determiners	0.0423	0.0366	532
compounds	0.0056	0.0042	253
foreign	0.0031	0.0021	251
incomplete	0.0150	0.0127	234
nouns	0.0971	0.0916	216
pronouns	0.1886	0.1956	195
conjunctions	0.0643	0.0682	149
verbs	0.1650	0.1705	132
prepositions	0.0701	0.0665	128
adjectives	0.0604	0.0634	95
numerals	0.0132	0.0119	74
incomprehensible	0.0071	0.0064	43

nenpoorte *et al.* (2005) on a slightly different subset of the corpus, male speakers use more interjections. In addition, male speakers use more determiners and compound names and do not complete words more often. The latter fits with results described in the previous chapter that female speakers speak more careful.

Female speakers use more adverbs and pronouns. Detailed investigation shows that women especially use more personal pronouns (relative frequencies of 0.092 for male speakers versus 0.1 for female speakers) this is in line with results found by Argamon *et al.* (2003); Harnqvist *et al.* (2003).

Table 5.10 shows POS-tag distributions for male and female speakers of broadcast speech. Figures 5.6(a) and 5.6(b) show the the graphs corresponding to these tables.

Table 5.10 – *Relative frequencies of* POS*-tags in broadcast speech for male and female speakers ordered by likelihood ratios.*

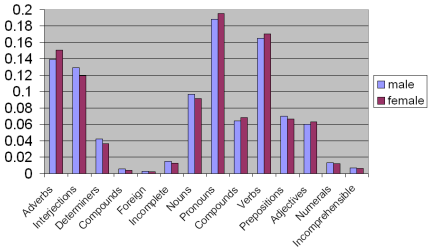
	male	female	LLR
Numerals	0.0125	0.0097	70
Nouns	0.1388	0.1300	67
Interjections	0.0672	0.0736	65
Adverbs	0.1039	0.1116	63
Determiners	0.0701	0.0643	55
Prepositions	0.0994	0.0932	44
Incomprehensible	0.0028	0.0020	28
Adjectives	0.0692	0.0726	21
Pronouns	0.1726	0.1779	20
Conjunctions	0.0666	0.0693	12
Foreign	0.0019	0.0022	6
Compounds	0.0117	0.0112	3
Verbs	0.1708	0.1692	2
Incomplete	0.0124	0.0128	1

The differences between the two types of speech are clearly visible, in the more formal broadcast speech more nouns and determiners are used. The differences between male and female speakers however remain largely the same, with one interesting exception: female speakers use more interjections in broadcast speech than male speakers.

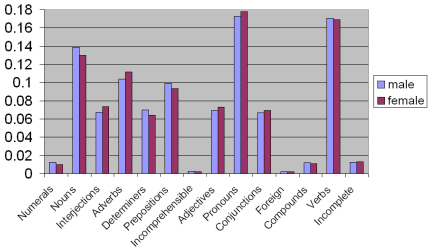
Gender specific POS-tag distributions for spontaneous speech and broadcast speech for Southern Dutch are shown in figures 5.6(c) and 5.6(d). These distributions show the same patterns as for Northern Dutch.

5.6.2 Words

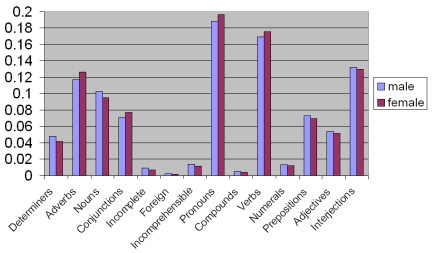
We look at the type-token ratios for male and female speakers. The type-token ratio is defined as the number of different words (types) divided by the total number of words (tokens) and can be seen as a measure of lexical richness van Gijsel *et al.* (2006). A lower type-token ratio means a relatively smaller number of different words. Unfortunately, the type-token ratio is text length dependent. The longer a text, the lower the type-token ratio will be (Van Gijsel *et al.* 2006). This is why we used randomly selected equal size subsets for male and female speakers. Table 5.11 shows type-token ratios for spontaneous and broadcast speech and for Northern Dutch and Southern Dutch. In all case the type-token ratio of male speakers is higher than for female speakers. This confirms the results of van Gijsel *et al.* (2006) where a different procedure for data selection and different subsets of the CGN corpus were



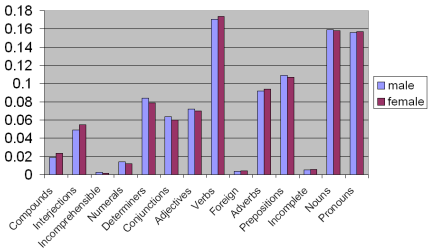
(a) spontaneous (Northern Dutch)



(b) broadcast speech (Northern Dutch)



(c) spontaneous (Southern Dutch)



(d) broadcast speech (Southern Dutch)

Figure 5.6 – POS-*tag distributions per gender for spontaneous speech and broadcast speech.*

used. Note that these results can not easily be compared across sets, as these sets are of different sizes. The top half of table 5.12 shows the ten words with the

Table 5.11 – *Type-token ratios for male and female speakers.*

subcorpus	male	female
spontaneous speech Northern Dutch	0.03011	0.02688
broadcast speech Northern Dutch	0.09227	0.08354
spontaneous speech Southern Dutch	0.03568	0.03178
broadcast speech Southern Dutch	0.12314	0.11222

highest log-likelihood values used most often by women. The bottom half of table 5.12 shows the ten words with the highest log-likelihood that are more often used by men. The tendency of women to use more personal pronouns and for men to use more fillers also shows in those lists.

5.7 Education level

Next, we look at the influence of education level of a speaker on word use. We define two education levels, high (college or university) and low. In accordance with van den Broeck (1980) we find that there is no significant difference in sentence length for these two groups. As discussed in the previous chapter sentence length seems to be related to the capabilities of the short-term memory more than to anything else.

The differences in POS-tag distributions are also smaller than for gender. Table 5.13 lists the POS-tags with the highest likelihood-ratios. The difference in type-token ratio is bigger than for gender. The ratio is 0.031 for the high education group and 0.028 for the lower education group. For Southern Dutch spontaneous speech the difference is larger: 0.049 for the high education level and 0.041 for the lower education level. Unfortunately, the broadcast categories contain too little lower educated speaker to obtain reliable statistics.

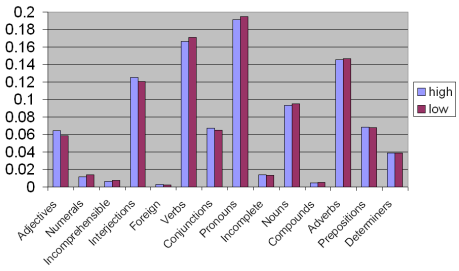
Table 5.14 lists the ten words with highest likelihood ratios that are most used by the higher education level group and table 5.15 lists the words with highest likelihood ratios that are used the most by the lower education group. It can be seen that abbreviations and slang are used often in the low education group and that the determiner *het* (the) is used more in the high education group. It is interesting to note that some interjections are used more often in the high education group. This is something worth investigating further in the future. Overall, the differences in word use between the two education level groups is small.

Table 5.12 – *Relative frequencies of high likelihood ratio words used most often by women (top) and words most often used by men (bottom).*

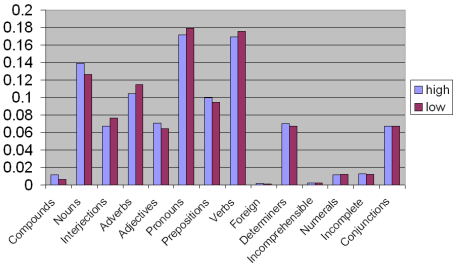
word	male	female	LLR
oh (filler)	0.0073	0.0010	435
ze (them, she)	0.0075	0.0098	324
zei (said)	0.0011	0.0020	276
leuk (nice)	0.0016	0.0026	240
want (because)	0.0044	0.0060	230
toen (then)	0.0032	0.0043	199
heel (very)	0.0033	0.0044	181
zo (therefore)	0.0077	0.0093	154
ik (I)	0.0259	0.0287	146
echt (really)	0.0035	0.0045	137
oh (filler)	0.0367	0.0260	1990
ze (them, she)	0.0182	0.0152	285
zei (said)	0.0222	0.0193	220
leuk (nice)	0.0187	0.0164	162
want (because)	0.0064	0.0056	64
toen (then)	0.0102	0.0091	60
heel (very)	0.0004	0.0002	60
zo (therefore)	0.0010	0.0007	58
ik (I)	0.0001	0.0000	54
echt (really)	0.0165	0.0153	51

Table 5.13 – *Relative frequency of* POS*-tags that differ most for higher educated and lower educated speakers as measured by likelihood ratios.*

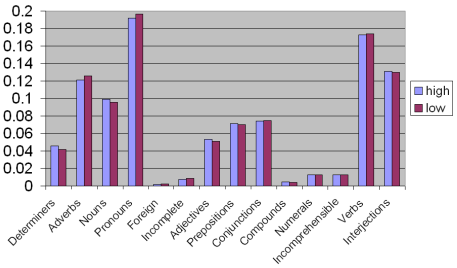
part-of-speech	high	low	LLR
adjectives	0.0641	0.0588	224
foreign words	0.0029	0.0021	140
numerals	0.0123	0.0138	85
conjunctions	0.0670	0.0650	64
verbs	0.1670	0.1710	53
incomprehensible	0.0069	0.0077	47



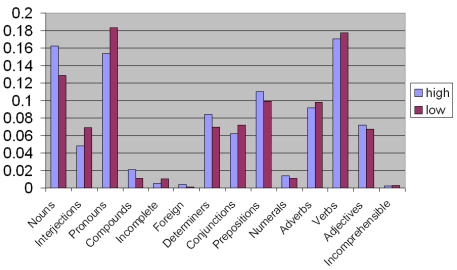
(a) spontaneous Northern Dutch



(b) broadcast speech Northern Dutch



(c) spontaneous Southern Dutch



(d) broadcast speech Southern Dutch

Figure 5.7 – POS *distributions of groups with high and low education level.*

Table 5.14 – *Relative frequencies of high likelihood ratio words used most often by higher educated speakers.*

word	high	low	LLR
het (the,it)	0.0035	0.0019	414
uhm (filler)	0.0030	0.0020	195
echt (really)	0.0046	0.0034	161
of (or)	0.0079	0.0067	91
is (is)	0.0166	0.0148	91
okay	0.0012	0.0008	86
ook (too)	0.0215	0.0195	82
volgens (according to)	0.0011	0.0007	70
precies (exactly)	0.0010	0.0007	64
gewoon (normal)	0.0050	0.0042	62
hum (filler)	0.0004	0.0002	60
dit (this)	0.0017	0.0012	58
mm-hu (filler)	0.0020	0.0015	55

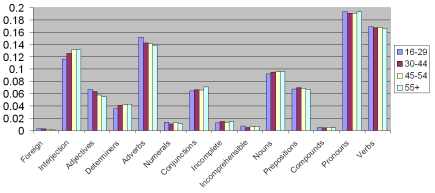
5.8 Age

To investigate the influence of age on word use we defined four age groups: 16–29, 30–44, 45–55, 55+. These groups are chosen to get reasonable sized subsets that allow us to gather reliable statistics. In particular, the CGN contains only few speaker younger than 20, therefore they were grouped together with people in their twenties.

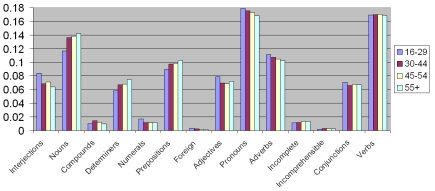
As for gender and education level, there are no significant differences in sentence length for different age groups in any of the components of the corpus.

Figure 5.8(a) shows the POS-tag distributions for age groups in spontaneous speech. The figure nicely illustrates how language use of different age groups corresponds to trends in language change. Over time, the use of interjections, determiners and nouns has decreased, while the use of adjectives and adverbs has increased. Figure 5.8(b) shows part-of-speech use by different age groups for broadcast speech. Here, younger speakers also use less nouns and determiners than older people, but the differences are much larger.

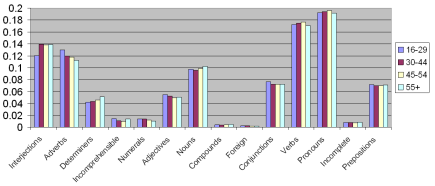
In addition to more adverbs, younger people also use more pronouns and more interjections. More research is needed to interpret these results. On one hand, we might be dealing with a trend to use less formal language. This then holds in particular for broadcast speech. On the other hand, in broadcast speech, the difference in language use may be amplified by the correlation between the type of program someone appears in and his or her age. For example, presenters on a music channel are typically younger than the host of a political talk show. Either way, a language processing system will have to deal with language differences for different age groups. In Southern Dutch similar trends can be observed as is shown in figures 5.8(d) and 5.8(d). Trends can also be observed in word use. Figure 5.9 shows the



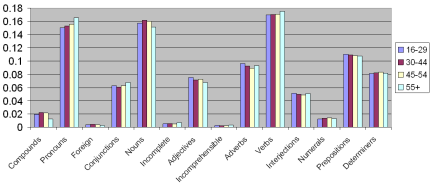
(a) spontaneous Northern Dutch



(b) broadcast speech Northern Dutch



(c) spontaneous Southern Dutch



(d) broadcast speech Southern Dutch

Figure 5.8 – *Part-of-Speech distributions for age classes in Northern Dutch.*

Table 5.15 – *Relative frequencies of high likelihood ratio words used most often by lower educated speakers.*

word	high	low	LLR
'k (I)	0.0056	0.0078	322
ze (she/they)	0.0084	0.0010	131
zei (said)	0.0013	0.0020	128
zeg (say)	0.0021	0.0029	109
zegt(says)	0.0008	0.0012	108
hè	0.0042	0.0052	103
hoor	0.0019	0.0026	100
vader (father)	0.0002	0.0004	97
toen (then)	0.0035	0.0045	93
och (filler)	0.0001	0.0003	90
effe (slang)	0.0007	0.0011	79
hé	0.0006	0.0009	73
auto (car)	0.0002	0.0005	64

words with highest likelihood ratios for spontaneous Northern Dutch. It can be seen that the words *gewoon* (normal), *echt* (really) and *wel* (indeed) are used much more by younger people than by older people. The word *ik* (I) is also used more often by younger people, as is it's abbreviated version colloquial *'k*, the word *ge* (you) on the other hand, that mainly appears in Dialect spoken in the South of the Netherlands, is not used at all by people younger than 45.

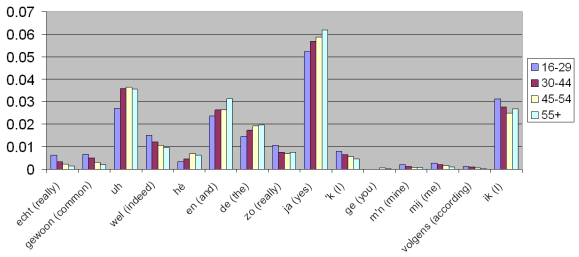
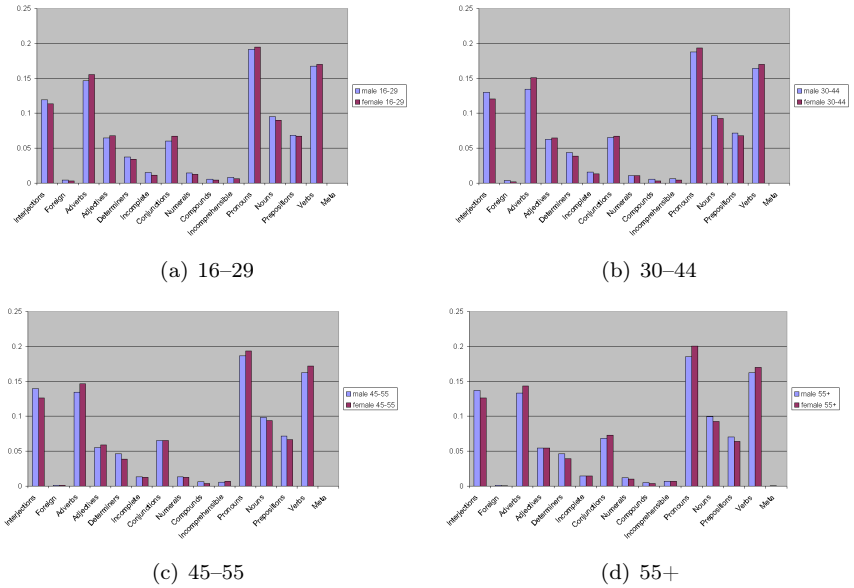


Figure 5.9 – *Distributions of words with the highest likelihood ratios in spontaneous Northern Dutch when comparing age groups.*

5.9 Combining age and gender

We also compared male and female speech for different age groups. Figure 5.9 shows POS-tag distributions for male and female speakers per age category. In all age



(a) 16–29 (b) 30–44

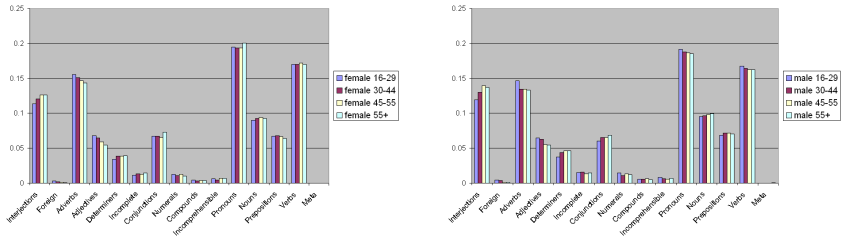
(c) 45–55 (d) 55+

Figure 5.10 – *Comparing* POS *distributions of spontaneous Northern Dutch of male and female speakers in different age groups.*

groups similar differences between male and female speakers can be observed. In the youngest age group the differences are smallest. This fits with the hypothesis that differences in speech between men and women are disappearing because social differences between men and women have become smaller in the last decades. The differences in the middle age groups are larger than the difference for people older than 55. Based on sociolinguistic theory this is unexpected (see section 4.1.1). The explanation might have something to do with emancipation, but it might as well be an artifact of the data. Figure 5.11 shows the same distributions as figure 5.9 per gender. It shows that female speech differs more between generations. Younger male speakers use more adverbs, adjectives and pronouns and fewer determiners, conjunctions and nouns. This fits in the overall trend of more informal speech, but we might also say that male speech has become more similar to female speech.

5.10 Discussion

The type of speech influences word use. In particular, it influences sentence length. It seems plausible that we can even deduce the type of speech using sentence length distributions and the distribution of personal pronouns. The characteristics of type of speech are similar for Northern Dutch and Southern Dutch. User specific factors do not influence sentence length. They do influence part-of-speech distribution and



(a) female (b) male

Figure 5.11 – POS-*tag distribution of spontaneous Northern Dutch per age group for male and female speakers.*

word use. Based on the CGN it is hard to draw any conclusions on the influence of education level, but it seems to be of less importance than gender and age. The analysis described in this chapter has only scratched the surface. We limited ourselves to a small number of context factors and looked only at word use. There are many other context factors and combination of context factors that might be of influence and one could look at word n-grams and grammar and at effects at lower linguistic levels as well. In addition, if one would want to draw conclusions on the why and how of the effects found in the data deeper analysis is needed. The analysis shows that in a limited data set one can find that context influences language use. Many of the word use patterns described in chapter 4 showed up in the data. This gives us confidence that a speech processing system could learn and use these relations. In chapter 8 the findings of this chapter will be used to develop advanced context-based language models.

The world is its own best model.

Rodney Brooks

Chapter 6

Case studies

In which two speech recognition systems that include contextual knowledge that were developed as part of this thesis work are presented. The first system includes lip-reading information to improve acoustic recognition. It is shown that this system performs better than a standard speech recogniser in noisy environments. In the second part adaptive language models that include domain specific knowledge are described. Several models that combine perceptual information and background information at different stages in the recognition process have been developed. These models are tested and compared on real data from a train table dialogue system. A confidence measure for speech recognition that has been developed in the course of this work is presented.

6.1 Case study: lip-reading

To test whether lip-reading can be of use in automatic speech recognition we experimented with several configurations of a bimodal recogniser for continuous speech (Wiggers *et al.* 2002*b*; Wiggers and Rothkrantz 2002).

The speech recogniser used in our experiments is a simplified version of our large vocabulary speaker independent recogniser described in (Wiggers *et al.* 2002*a*). It

uses continuous density Hidden Markov Models to represent phonemes. Each model has three states connected left-to-right, with single Gaussian distribution functions attached to the states. There is a total number of 45 phonemes, which include the phonemes from the SAMPA set and models for silence, optional pauses and mouth noises like, loud breath, sniffing or smacking. The original recogniser was trained on a subset of the Dutch Polyphone database (Damhuis *et al.* 1994). This is a rather large corpus containing telephone speech from 5050 different speakers from all dialect regions in the Netherlands. For these experiments the recogniser was retrained, using two iterations of Baum-Welch re-estimation, on the audio part of a small multi-modal data set that we recorded (Wojdeł *et al.* 2002).

The lip-reading part of our recogniser is based on the visual feature extraction technique described in (Wojdeł and Rothkrantz 2001; Wojdeł 2003). It uses a lip-selective colour filtering that captures both geometric and intensity related features of the mouth.

We experimented with two schemes, feature fusion and model fusion.

In feature fusion the feature vectors from both modalities are simply concatenated to generate a single vector on which a standard HMM based recogniser can then be trained. We used linear interpolation between video frames, that are extracted with 25 frames per second to match the frame rate of 100 frames per second of the audio data.

6.1.1 Feature fusion

The multi-modal recogniser was created by extending the 39 dimensional distribution functions of the (unadapted) baseline speech recogniser to 50 dimensional distributions. The additional 11 means and variances of all states of all models were initialised with the global means and variances calculated over the entire video part of the multi-modal data set. The audio part of the system was already reasonably well-trained and needed only some adaptation to the new data set, but the visual part could only be trained on the multi-modal data set. As all models initially have the same parameters for their visual features the distribution of the feature vectors during Baum-Welch re-estimation are guided by the speech features. This way a continuous multi-modal recogniser can be obtained in a few training cycles with a limited amount of training data. The speech part of the models ensure robustness while the video part may give valuable cues to differentiate between acoustically similar phonemes. The combined models were re-estimated twice, using the bimodal training data. The models in this system thus received just as much training as the models in the adapted speech only system. The recognition results of this system on a held-out test set are shown in table 6.1.

The feature fusion approach, although attractive because of its simplicity, was not able to improve upon the speech only system. One of the problems is that this approach does not take into account the reliability of the separate streams. Because of the clean audio and the well-trained speech models, the audio stream is likely to be more reliable than the video stream.

Table 6.1 – *Results audio-visual speech recognition.*

System	% of words recognised correctly
baseline speech recogniser	84.24
feature fusion	83.69
phonemes; equal weights	83.69
phonemes; audio weight 1.2	84.43
phonemes; audio weight 1.4	84.22
visemes; audio weight: 0.9	84.76
visemes; audio weight: 1.0	85.56
visemes; audio weight: 1.1	85.92
visemes; audio weight: 1.2	85.03

6.1.2 Model fusion

In model fusion two different data streams are used and these are combined within the hidden Markov model. The multi-stream HMM explicitly models the reliability of its streams. In its simplest form, the state synchronous multi-stream model, it uses separate distributions for its streams in each state. The observation likelihood of the state is the weighted product of the likelihoods of its stream components, as shown in equation (6.1), where γ_s are the weights:

$$b_j(o_t) = \prod_{s=1}^{2} \mathcal{N}(o_t, \mu_{sj}, \Sigma_{sj})^{\gamma_s}. \tag{6.1}$$

A multi-stream recogniser was build using a similar approach as with the feature fusion model. The models from the baseline speech recogniser were used for the audio stream and the distributions in the video stream were initialised with the global mean and variance of the entire video data set. This system was also re-estimated twice. A number of recognition experiments was run on the test set using different weighting schemes, the results are shown in table 6.1. The video weights and audio weights add up to two in all cases.

By setting the weights so, as to put more emphasis on the audio stream this system is capable of doing a little better than the stand-alone speech recogniser. A shortcoming of this system is that it uses phones as basic units but from a lip-reading point of view it is hard to distinguish between certain phonemes, because of similar lip movements.

To solve the problem indicated above, it was decided to use visemes for the video stream. A viseme is basically a phoneme class; we defined the following visemes:

{sil, sp}, {f, v, ʋ}, {s, z}, {ʃ, ʒ}, {p, b, m}, {t, d}, {g, k, x, n, ŋ, r, j},
{ɛ, ɛː}, {ɑ}, {a}, {ə}, {ɔ, ʏ, y, u, øː, oː, œy, œː, ɔː}, {ɪ, eː}, {ɛi}.

The use of different units for the stream was realised by tying the distribution functions of corresponding states in the second stream for phonemes that are in the same phoneme class. The limited training data problem is also partially solved this way, because there is now more data per model in the second stream available. As with the previous systems this system was also re-estimated twice before recognition experiments were conducted. Table 6.1 shows the recognition results of a number of viseme systems with different weights, once again the audio and video weights add up to 2. This system is capable of improving upon the speech recogniser even when both streams have equal weights. By giving the audio stream higher weight than the video stream the results show more improvement.

6.1.3 Noise robustness

In the experiments described in the previous section the improvements the bimodal system realised over the audio only system remained modest. This can be explained by the fact that both modalities encode similar information. If the video stream gives reason to belief that a plosive sound is uttered, and the speech recogniser has a hard time choosing between /p/ and /g/ then the bimodal system may correctly pick /p/. But if the speech recogniser already found that a /p/ was uttered then the additional information from the video data does not help much.

Because relatively clean audio was used in the experiments described so far, the speech recogniser did not need the additional information from the lip-data, most of the time. But for noisy audio the cues given by the video stream may be more valuable. To verify this hypothesis the multi-stream viseme system was tested using noisy data. This was done by adding different levels of white noise to the audio samples in the test set. The performance of the systems was measured for signal to noise ratios between 20 dB and -5 dB.

The performance of the speech-only system degrades rapidly under these conditions as can be seen in figure 6.1. The results of the bimodal system are also shown in the figure. At low noise levels the multi-modal system performs slightly better than the speech recogniser, but as the noise level increases the bimodal systems clearly outperforms the uni-modal system. Once again the multi-stream model with viseme models in the second stream shows the best results. At a signal to noise ratio of 5 dB the difference is 12%. As the noise level approaches -5 dB the audio recognition gets so poor that the visual cues can no longer provide adequate help.

Our lip reading technique seems to do especially well in discriminating between consonants (for example /f/ and /s/). Figure 6.1 also shows a system for which only the consonant weights of the visemes were gradually increased as the noise level increased (up to a noise level of 0.8 dB, for higher levels the weights were decreased again). This system impressively outperformed all other systems and shows the feasibility of bimodal speech recognition. Note that these results are obtained with a relatively small data set. If more data would be available, it might be possible to develop a more reliable lip-reader. Future research will also have to show if and how these results carry over to speech recognition in noisy environments. It is well-

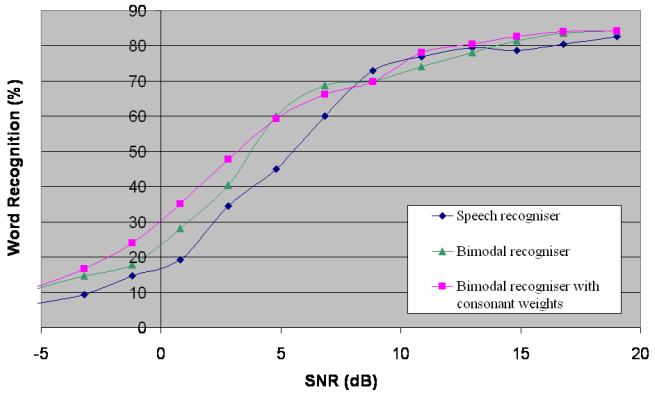


Figure 6.1 – *Percentage of words recognised correctly at different signal to noise ratios.*

known that people (unconsciously) raise their vocal intensity in the presence of noise. This phenomenon, called the Lombard effect, affects the performance of a speech recogniser and possibly that of a lip-reader.

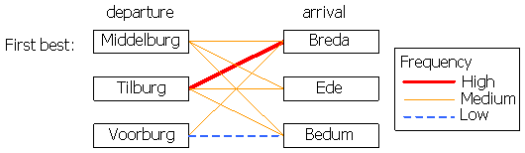


Figure 6.2 – *Using connection frequencies to select the best combination of station names from the 3 best hypotheses generated by the speech recogniser.*

6.2 Case study: domain knowledge

In many cases the context of use or other program components may provide valuable cues about the words to come. This is especially the case in systems that operate on limited domains like dialogue management systems. It is this kind of knowledge and the ways to incorporate it into the search process that we focus on in this chapter. As our domain we take a dialogue management system for train table information of the Dutch railways (Rothkrantz *et al.* 2000). Within this system we concentrate on the identities of the stations involved, as analysis of the train tables shows that there is a strong correlation between departure and arrival stations. The general idea is illustrated in figure 6.2. n-best lists for the departure and arrival stations are shown. The original system takes the first best option from both lists. However, the connection frequencies, depicted by the thickness of the lines in figure 6.2, suggest that the second best option for the departure station is very likely given the first best arrival station.

Given such knowledge, or for that matter domain knowledge in general, the central questions that have to be answered are:

1. In which situations should the knowledge be used?
2. At what point in the recognition process should the knowledge be employed?
3. How can it be combined with existing knowledge sources (in this case the language model and the acoustic models)?

To find an answer to the second question we experiment with two different methods, in the first case we use a class based language model in which the probabilities are updated to influence the search process directly. In the second case the connection frequencies are used in a post-processing step on lattices produced by the recogniser. The different approaches also provide different ways to combine the knowledge sources.

The importance of the first question stems from the fact that our baseline recogniser identifies over 85% of all station names correctly. So, in many cases the use of additional knowledge is not necessary and might even do more harm than good. Essentially, this is the same discussion as for models of human speech recognition: (when) is background knowledge allowed to override perceptual information? To

decide when to utilise the additional knowledge a confidence measure based on posterior probabilities was developed and implemented.

6.2.1 Data analysis

The data set used, called the OVR log files, covers the transcribed conversations of an entire year (October 1995 - October 1996) between users and operators of the original telephone inquiry system. The total number of references to stations in questions regarding station-to-station queries is almost 13 million. Figure 6.3 plots the frequencies of station pairs mentioned in those queries. On both axes the stations are ordered based on frequency. The connection frequencies are indicated by shades. A darker shade corresponds to a higher connection frequency. Analysis of the data

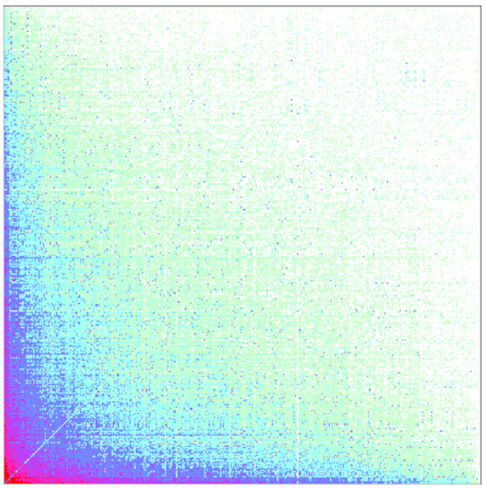


Figure 6.3 – *Plot of connection frequencies between stations. The axes display station names in order of decreasing frequency. Higher connections frequencies have a darker shade.*

revealed the following facts (van Vark *et al.* 1997):

- The connections from and to the first 50 stations (out of 377 stations) make up 50% of all connections asked for.
- Connections are highly dependent on the stations involved.
- There is a strong inverse correlation between the distance to travel and the frequency of these journeys.

Given the dependencies found in the log files it may be clear that this information can be useful in recognising the correct station name. Knowledge of either the

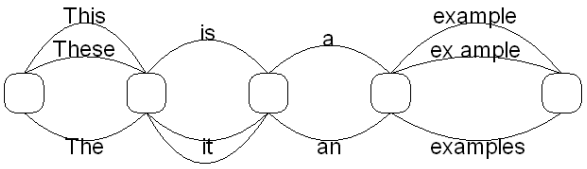


Figure 6.4 – *A confusion network that shows competing word hypotheses generated by a speech recogniser.*

departure or arrival station can help to restrict the search for the other station name considerably. But as there are a number of stations that always have high frequencies, while on the other hand the majority of the connections have relatively low frequencies, relying too much on the connection frequencies does not seem a smart thing to do. This may lead to overall improvement at expense of small stations. Therefore, it should be decided in what cases the frequency information may be beneficial and the choice of a station pair should be based on a balanced combination of all evidence available. To provide guidance for these decisions confidence measures are used.

6.2.2 Confidence measures

The confidence measures are based on a lattice output by the recogniser. The main feature used is the word posterior probability, i.e. the probability of a word in a certain position given the observation sequence. As speech recognisers try to maximise the posterior probability over an entire utterance the word level posteriors are usually not known, but they can efficiently be approximated using a variation of the forward-backward algorithm on the output lattice that provides the probability of each link in the lattice (Evermann and Woodland 2000). Usually several links may correspond to the same word, so deciding which links correspond to which word is not trivial. We chose to proceed along the lines of (Mangu *et al.* 1999) by clustering the words in the lattice based on overlap in time. Links in the same cluster that correspond to the same word are combined. An example of the resulting network is shown in Fig. 6.4. This type of network is often referred to as confusion network and explicitly shows which words in the lattice can be considered competing alternatives to each other.

Our procedure differs from the ones described in Mangu *et al.* (1999) and Tur *et al.* (2002) as we allow successive words to be in the same cluster. This may happen when two smaller words correspond to one longer word (e.g. 'ex' and 'ample' to 'example'). For further processing these sequences in a cluster are seen as a single entity. As word lattices cover only part of the entire search space the posterior probabilities are just an approximation of the real posteriors; in general they tend to overestimate the probabilities of the best alternatives, therefore a number of other confidence features is determined for each cluster and the features are combined in

Table 6.2 – *Detailed results of the confidence measure.*

Correctly classified instances			94.22%	
Class	TP rate	FP rate	Precision	Recall
Correct	98.12%	46.50%	98.12%	95.53%
Incorrect	53.50%	1.88%	53.50%	73.79%

a classifier that provides a single confidence value for each word. The following features were selected as the most discriminative subset of a large set of potentially useful features:

Posterior drop The quotient of the posterior probabilities of the first and second best option. If this value is close to one the second best alternative is almost as good as the first best, which may indicate low confidence.

n-best position The first hypothesis the word occurs in. This may differ from the word having the highest posterior probability.

Relative n-best position The number of consecutive hypotheses the word occurs in. This is a measure of the stability of a word.

Number of words in a cluster Many alternatives may imply lower confidence.

Number of words on a cluster arc As the alternatives in a cluster can actually be word sequences, this number may be larger than one. This usually means that a longer word has been recognised as a sequence of smaller words.

Language model score Words having a higher language model score may be better trained and thus better recognised.

Confidence value of the previous cluster Recognition errors tend to occur in bursts.

For combination of the features we experimented with several classifier types including neural networks and linear regression trees Like Stemmer *et al.* (2002) we found that there are no significant performance differences. For the final system a linear regression tree was chosen. Table 6.2 shows the classification results of the confidence measure on the phonetically rich sentences of the evaluation part of the Dutch Polyphone database, the word error rate of the recogniser on this set is 9%.

6.2.3 Language model structure

The marginal departure and arrival frequencies, that is, the probability that a user wants to travel to or from a certain station can directly be used in the language model. To do so, we trained a DBN bigram language model but replaced all station

names with the respective labels `from_station` and `to_station`, assuming that the words preceding a name do not depend on that particular name. Two classes were used instead of a general station class because data analysis revealed that frequencies are not completely symmetrical. For example, it was found that there are much more requests for journeys to the train station of Amsterdam Airport than for travelling from the airport to another station. A likely explanation may be that people will carefully plan their trip to the airport, but they often do not know when exactly they will be back at the station of the airport, they will simply take the first train home.

The probabilities within these classes correspond to the marginal frequencies. This approach can be generalised to obtain a class based n-gram model by assuming that all words belong to a class and the occurrence of a class only depends on the previous class. In this particular case most classes g_i contain only a single word w_i. The bigram probability of a word given the previous word now becomes:

$$P(w_i \,|w_{i-1}) = P(g(w_i)\,|g(w_{i-1}))P(w_i\,|g(w_i)). \tag{6.2}$$

Taking advantage of the connection frequencies is less straightforward as these do not fit very well in an n-gram approach, since the station names may be separated by a large, unknown number of words, be in different utterances or they may be obtained from other sources, for example a pointing device in a multi-modal system. As we are ultimately interested in the more general concept of using external knowledge in speech recognition these possibilities should be taken into account. The next sections describe two approaches that we considered: An approach that takes advantage of the class based structure of the language model to utilise the contextual knowledge in an early stage and a post-processing approach that uses the connection frequencies for rescoring the results produced by the recogniser.

6.2.4 Dynamically updating the language model

This approach uses the class based language model described above to recognise the first name. If the confidence value of this name is above a predefined threshold τ the name is assumed to be correct and the distributions in the station name classes are updated according to the conditional probabilities obtained from the frequency table. We found the best results if the conditional frequencies are interpolated with the marginal frequencies.

This approach essentially comes down to the on-the-fly creation of a context dependent language model that is interpolated with the original language model to obtain a mixture language model according to:

$$P(w_i\,|w_{i-1}) = \lambda P_{lm}(w_i\,|w_{i-1}) + (1-\lambda)P_{conn}(w_i|w_{i-1}), \tag{6.3}$$

where $\lambda \in [0,1]$. In case of low confidence for the first name no assumptions can be made about likely connections, so the standard language model is used for recognition of the second name. This approach suffers from two drawbacks, first it only

works when the arrival and departure names are mentioned in different utterances. Second, uncertainty about the first name is not resolved, even though the second name may be helpful here. As a solution a second recognition pass was introduced for the first name using a language model updated based on knowledge of the second name. This is a computationally rather expensive method, a more efficient, albeit probably less accurate method, would be to rescore the original recognition results with the updated language model. As we are only interested in station names and not in whatever other words are recognised things can be made even more efficient by using the connection frequencies in a post-processing step, which has the additional advantage that the influence of the different knowledge sources can be better controlled.

6.2.5 Using frequencies for lattice rescoring

In this approach the recogniser is set to produce n-best lattices which are transformed to confusion networks while calculating confidence values. The clusters containing station names are located and the connection frequencies are used to pick the best pair of names. This method is particularly error prone because frequent connections will dominate, thus as before the confidence measures are used to limit the use of frequency information to those cases where there remains uncertainty about one of the names. In those cases where frequency information is utilised it is combined with acoustic evidence using a Bayesian updating approach. The frequency table is interpreted as the joint probability of a connection from departure station S_1 to arrival station S_2. The word level posterior probabilities, which are already available from the confidence measure calculation, are used to represent the probability that a station name was uttered given the corresponding observation (sub)sequence. Assuming statistical independence of the second station name and the first observation sequence given the first name this can be written as:

$$P(S_1 | S_2 O_1) = \frac{P(S_1 S_2) P(S_1 | O_1)}{P(S_1) \sum_{S_1} P(S_2 | S_1) P(S_1 | O_2)}. \tag{6.4}$$

As we are only interested in relative scores the summation in the denominator can be omitted. The formula has an intuitive interpretation: the overall score will be higher when a name is more likely to be uttered and is likely to occur together with the other station name, while stations with a small marginal probability, that is the rarer stations are preferred over stations that have a high frequency in general.

Although attractive from a theoretical point of view this approach has its weaknesses. As mentioned before the lattice based posterior probabilities are a somewhat crude approximation to the real probabilities. They have a tendency to overestimate. To smooth the distributions we may resort to the same solution as with the confidence measures. Determine other features that indicate the likelihood of a word and combine them using some sort of classifier. But, as each classifier works independently, the values obtained for competing alternatives can in this case no longer be

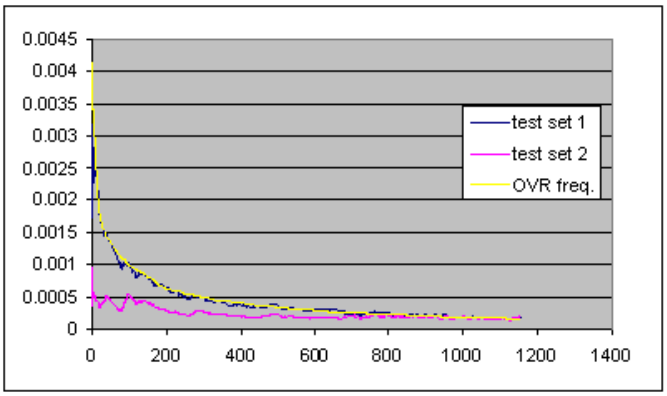


Figure 6.5 – *Distributions of names in the test sets.*

seen as probabilities, so the Bayesian framework is no longer applicable. Therefore we decided to combine the acoustic evidence now represented by local confidence measures and the connection frequencies using a weighted geometric mean, according to:

$$\mathrm{Score}(S_1S_2) = C(S_1|O_1)^{w_1} f(S_1S_2)^{w_2} C(S_2|O_2)^{w_3}. \tag{6.5}$$

While somewhat less formal than the previous method, this formula captures the same intuitive line of reasoning. A geometric mean was chosen because it has the desired effect that the overall score is high when its individual items are high, while it is rather insensitive to high values of a single attribute. This method also addresses another weakness of the Bayesian approach as it can give different weights to the knowledge sources. The score does not take into account the marginal probabilities. However, these are indirectly used as the best results were obtained by setting the weight of the connection frequencies somewhat higher than the weights of the acoustic scores, but lower for very rare stations or stations having a high marginal probability.

6.2.6 Experiments

The approaches described above were tested on a set of 325000 station pairs (Wiggers and Rothkrantz 2003*a*;*b*). These pairs were sampled from all pairs in a set of 10163 recordings of station names. The subset was selected so as to get a representative data set that exhibits roughly the same distribution as the OVR data. To test the stability of the methods another test set was selected, which has a much flatter distribution that the original data set. Unlike the first set and the OVR data where 50% of all connections have a frequency above 1000, here only 26% of all connections have a high frequency. For the rescoring approaches the recogniser produced 10-best

Table 6.3 – *Comparing word error rates for the different approaches.*

System	WER test set 1	WER test set 2
Speech recogniser	11.4%	11.4%
Highest confidence	10.9%	10.9%
Adaptive lm	10.34%	10.8%
Bidirectional lm	8.5%	9.9%
Bayesian approach	9.5%	10.6%
Geometric mean	9.0%	10.4%

lattices. Table 6.3 shows the results of these experiments. The first row shows the word error rate of the baseline speech recogniser. The second row shows the word error rate if for each name the alternative with the highest confidence value is chosen. This already gives a small improvement as the word error rate is explicitly minimised this way. Including connection information using the Bayesian approach gives some improvements; the geometric mean does even better and results in an absolute gain of 2.5%. As can be expected both methods perform less well for the second data set, but they still improve upon the baseline system. The difference in performance between the two approaches stems from the ability to weight the contributions of knowledge sources for the geometric mean. The mixture language model does less well than the post processing approaches, however when used for both station names it outperforms the other approaches. This is mainly due to the fact that in the post-processing approach the correct name may already be pruned away; in about 5% of all cases it is not in the n-best lists.

It is an interesting question whether it would be possible to reconstruct missing words in the n-best approach. For example, we might consider choosing a station name if it is known to be acoustically similar to a name that is in the n-best list and it is also strongly correlated with the name of the other station. We did some preliminary experiments in order to test this idea. For clusters with low confidence, stations names are added to the list that are acoustically similar to the names already in the list, as measured by a minimal edit distance between the phonetic transcriptions. The acoustic confidence for those new names is set equal to the lowest confidence value in the list. Furthermore, for each alternative now in the n-best list the mean acoustic distance to all other names in the list is calculated. This value is also taken into account in the final score, but only with a small weight. Using this setup we were able to get the word error rate down to 8.7% for the first test set. As this was only an ad hoc preliminary experiment it might very well be possible to outperform the early integration approach using a post-processing approach that relies on background knowledge.

There is nothing so practical as a good theory.

Kurt Lewin (1890–1947)

Chapter 7

Computational Paradigms

In which we reason that a new computational paradigm is needed for the development of context-rich models of speech recognition. Requirements for such a paradigm are formulated. Several techniques are evaluated against these requirements. It is argued that dynamic Bayesian networks are best suited as a computational technique for speech recognition.

Many sources of information discussed in previous chapters have not explicitly been used in models of speech and language processing. Models that do include linguistic or contextual knowledge typically use a single feature in isolation. The case studies presented in the previous chapter are representative for work in this domain. They prove that using contextual knowledge can improve speech recognition, but are limited to a single feature for which the standard models are adapted. The resulting models are often complex and hardly ever combined. We might argue that the area of knowledge-rich models of speech and language is largely unexplored because of the lack of a unifying framework that allows one to do so.

The focus of this and the following chapters is the development of such a general framework. In this chapter we will start by reviewing a number of computation techniques that have the potential to offer a uniform framework for speech and language processing with context information. The goal is not so much to find the ultimate representation for all speech and language processing applications —

specialised representations for different types of systems are likely to be more efficient — or to completely abandon existing theory and practices, but rather to find a framework that allows for rapid model development and experimentation. That allows us to move from an imperative approach in which new algorithms have to be implemented for every new model to a declarative approach in which we can focus on purely on model design. Based on this goal and the types of contextual knowledge described in the previous chapters a number of requirements for a computational technique can be specified:

1. The model should be able to deal with the highly ambiguous nature of language and speech.
2. It should be possible to include all context factors mentioned in the previous chapters in a model. In particular, the model should be able to deal with classes of elements, with knowledge at multiple levels where a variable number of items on a lower level can belong to an element at a higher level.
3. The model should be able to capture the dependencies and interactions between those factors.
4. It should be simple to incorporate new knowledge sources into a model.
5. It should be simple to experiment with different configurations of a model.
6. It should be possible to combine the models with existing speech recogniser components such as acoustic HMM models.
7. The model should be capable of dealing with the time dimension of speech.
8. The model should be capable of on-line processing.

In this work we will first of all stick to probabilistic modelling techniques, for two good reasons. First, the standard models of speech recognition are probabilistic and we wish to extend those models, not to replace them (requirement 6). Second, the complexity of the domains of speech and language we deal with necessitates a probabilistic approach (requirement 1). This is a less restrictive choice than it may seem as one can always turn a knowledge based model into a probabilistic model by calculating probabilities over its decision points.

7.1 Linear interpolation and back-off

In chapter 3 linear interpolation and back-off were discussed as smoothing techniques for language modelling. These methods have also been used to combine different models (Goodman 2000). Linear interpolation is simple and easy to implement and use. It is very general. Any model that can be formulated as a probability distribution can be included as a component. An advantage of interpolation is that it can never be worse than any of its components. To construct a model that includes

a number of context variables, one can simply train distributions for all subsets of variables and estimate the weights of these components on using a held-out data set.

It also provides a tool for analysis: the weights of the model reflect the relative usefulness of the components. That is, as long as the data-set is representative. It has been found that data sets for weight estimation can be of moderate size to obtain reasonable results (Rosenfeld 2000).

However, straightforward linear interpolation, that optimises weights globally, uses its components suboptimally. It does not take into account strengths and weaknesses of the components in particular contexts. Often one would like the weights themselves to depend on context. This requires handcrafting of complicated training algorithms. In addition, linear interpolation requires that all components specify distributions at the same level. For example it is hard to combine a model that works at the sentence level, such as a syntax model with a model that works at the n-gram level. To include classes into the model or to specify the interplay between context variables other techniques have to be used in addition to interpolation.

Back-off offers another simple and compact way to combine multiple models. Compared with linear interpolation it has the disadvantage that discontinuities are introduced around the point where the back-off decision is made.

7.2 Decision trees

A decision tree is a k-ary branching tree in which questions are associated with each internal node and an answer is associated with each leaf. In a statistical decision tree probability distributions are attached to the leafs of the tree. All contexts that lead to the same leaf node have the same probability distribution for the decision. Effectively, leaf nodes define equivalence classes for their values depending on the history. The leaf distributions are conditional distributions over the entire history of questions and answers from the root of the tree to the leaf. As Magerman (1994) points out there is a relation between decision trees and n-grams. If an n-gram is generalised to:

$$P(f|h_1 \dots h_{n-1}), \tag{7.1}$$

where f is some future event, e.g. the next word, that depends on $n-1$ history variables (without requiring that these range over the same vocabulary as a Markov model would). The model can be represented by a decision tree with $n-1$ questions of which the leafs define the distribution over the event f. On the other hand a decision tree can be represented by an interpolated n-gram. Each of the leafs in the tree depends on the answers to $m \leqslant n-1$ questions. If these answers are the values of the variables $h_i \dots h_m$ then the leaf defines the term $P(f|h_i \dots h_m)$ from the deleted interpolation estimate of $P(f|h_1 \dots h_{n-1})$. Given this relationship it may be clear that decision trees have no additional expressiveness over n-gram models. The advantage of decision trees is that their structure can be automatically acquired, such that the leafs will be conditioned only on the information that is available in the training data. Therefore, decision trees can handle very large modelling problems.

Although an interesting and powerful modelling technique, decision trees have their disadvantages. First of all, the tree building algorithms are usually greedy algorithms and thus do not guarantee to find the optimal tree. As the leaf distributions are empirical estimates decision trees require smoothing just as n-grams do. Furthermore, irrelevant questions or the wrong order of questions may unnecessarily fragment the data (a possible partial solution lies in node merging during the training algorithm). Finally, decision trees do not always make for intuitive models. For example, it is not straightforward — but possible, see Magerman (1994) — to define models of syntax using decision trees.

7.3 Maximum entropy models

Maximum entropy models (Lau 1994; Berger *et al.* 1996; Pietra *et al.* 1997; Ratnaparkhi 1997) define probability distributions in terms of features, which are constraints over the variables in the domain of the distribution and are usually formulated in terms of expectations of marginal distributions of these variables. The desired probability distribution should fulfil all of these constraints. Often there will be more than one distribution that fits these requirements. The principle of maximum entropy states that in this case the most uniform distribution should be chosen, as this distribution minimises for additional assumptions that are non-embeddable. The most uniform distribution is the distribution with the highest entropy. It can be proven that when the distribution is formulated as an exponential model a unique maximum entropy distribution that fulfils all features will exist, which will also be the maximum likelihood distribution of this kind over the training data. An algorithm called generalised iterative scaling exists that finds this distribution given a training set and a number of features. The algorithm allows for incremental adaptation: constraints can be added or relaxed to optimise the model.

The advantage of maximum entropy models is that they can deal with a large number of features, that may encode any kind of knowledge and that do not have to be independent. However, the generalised iterative scaling algorithm, used to learn the distribution, has a high computational complexity, it has no theoretical bound on its convergence rate, which puts a practical limit on the number of features. Furthermore, the method delivers maximum likelihood distributions, i.e. they fit the training data as best as possible but do not generalise. In fact, if information is included that is not in the training data, as some smoothing algorithms do, the existence of a unique Maximum Entropy distribution is no longer guaranteed.

7.4 Probabilistic grammars

Grammar formalisms such as probabilistic context-free grammars, that were discussed in section 3.13, offer the most natural representation for syntactic knowledge. Contextual knowledge can be incorporated in those models in the same way that

syntactic features, such as lexical heads, are incorporated. The expansion of syntactic constituents can for example be conditioned on dialect or on type of speech. However, including context information that is largely independent of syntax can become awkward. For example, it is reasonable to assume that the topic of a sentence and its structure are largely unrelated. Given that the topic should be consistent throughout the sentence we effectively need to repeat the same set of parses for every topic. Syntax models are less suited for acoustic modelling. They do not fit the time-dependent nature of speech recognition very well.

7.5 Weighted finite-state transducers

Weighted finite-state transducers (Pereira *et al.* 1994; Jurafsky and Martin 2000) can be seen as a generalisation of the hidden Markov models commonly used in speech recognition. They have been used to model phonological and morphological rules as well as simple models of grammar. As weighted finite-state transducers in turn are a subclass of dynamic Bayesian networks that are the subject section 7.7 we will not discuss them separately.

7.6 Bayesian networks

Bayesian networks (also called belief networks or causal networks) originate in artificial intelligence as a method for reasoning with uncertainty based on formal rules of probability theory (Pearl 1988; Neapolitan 1990; Russell and Norvig 1995; Cowell *et al.* 1999).

A Bayesian network represents a joint probability distribution over a set of random variables $\mathsf{X}_1, \mathsf{X}2, \ldots \mathsf{X}_\mathsf{N}$. It consists of two components: a graph that represents (causal) relationships between the variables, and a set of probability distributions that quantify the strengths of these relationships. There is a one-to-one correspondence between the nodes of the graph and the random variables in the problem domain, i.e. every node of the graph ν_i corresponds to exactly one random variable X_i and every random variable is represented by exactly one node in the graph. The directed arcs of the graph represent direct dependencies between random variables. To be precise, the absence of an arc between two variables means that the variables are not directly dependent on each other. To avoid circular reasoning directed cycles[1] in the network are not allowed. Thus, in graph theoretical terms a Bayesian network is a directed acyclic graph. How a variable X depends on the variables $\mathsf{Pa}(\mathsf{X}_\mathsf{i})$ that correspond to the parents of the node associated with X_i in G is specified by a conditional probability distribution $\mathsf{P}(\mathsf{X}_\mathsf{i}|\mathsf{Pa}(\mathsf{X}_\mathsf{i}))$ associated with it. Nodes

[1] A *cycle* is path through the graph of which the begin and end node are the same. A *path* between two nodes ν_i and ν_j in a graph is a sequence of nodes starting with ν_i and ending with ν_j such that every adjacent pair of nodes is connected by an arc and no node, except for the first and last, appears more than once.

without parents have prior distributions. Probability distributions can be continuous or discrete. The probabilities are obtained from domain experts, learned from data or a combination of both. Applying the chain rule of probability theory and the independence assumptions made by the network we can write the joint probability distribution represented by the network in factored form as a product of the local conditional probability distributions:

$$P(X_1, X_2, \ldots, X_N) = \prod_{i=1}^{N} P(X_i | Pa(X_i)). \tag{7.2}$$

The computational complexity of the right hand side of (7.2) may be much more efficient than that of the full joint representation. For example, a joint multinomial distribution of n variables with v values each requires v^n entries, so that time and space requirements increase exponentially with the number of variables. If every variable has at most k parents then the complexity reduces to $O(nv^k)$. The size of the conditional probability tables (CPTs) is still exponential in the number of parents. Thus, the number of dependencies should be kept as low as possible.

The links in the network show direct dependencies between variables, but dependencies may also be indirect. For example, if variables A and B have a common parent C, and the value of A changes, this will affect the belief about the value of C which in turn affects the belief about the value of B. Dependencies between variables can be determined based on the notion of d-separation. Two nodes are structurally independent, i.e. changes in the belief of one of these nodes will not affect the belief of the other node if they are d-separated. Two distinct variables A and B in a belief network are d-separated if for all paths between A and B, there is an intermediate variable V (distinct from A and B) such that the connection is either:

1. Serial (one arc leading in to V and one arc leading out of V) and V is instantiated to a particular value.

2. Diverging (both arcs on the path connected to V are leading out of V) and V is instantiated.

3. Converging (both arcs on the path connected to V are leading into V) and neither V nor any of V's descendants have received any evidence.

Figure 7.1 illustrates the three possibilities. Bayesian networks allow for several patterns of reasoning: *predictive*, from cause to effect, *diagnostic*, from evidence to cause and *explaining away*: if a particular cause of an effect occurs, alternative causes are less likely. Mixed patterns are also possible. Furthermore, a Bayesian network can be used to find the most likely explanation for some evidence, or to determine which additional evidence variables should be observed in order to gain useful information (Russell and Norvig 1995). Finally, one can perform sensitivity analysis to find out which aspects of the model have the greatest impact on the probabilities of the query variables.

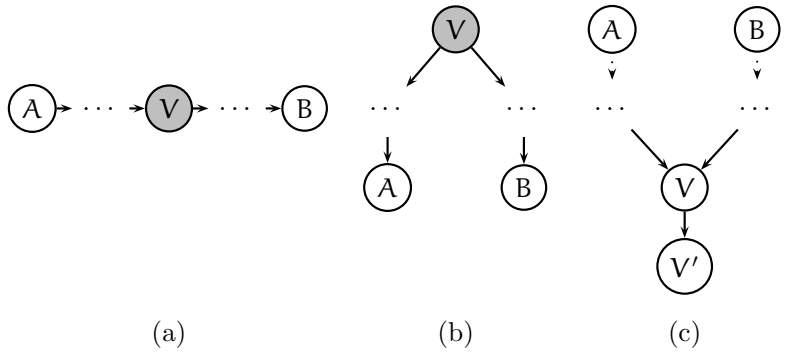


Figure 7.1 – *Variables* A *and* B *are d-separated if on every path between* A *and* B *there is a variable* V *such that* V *is instantiated (indicated by a shaded node) and the connection is* (a) *serial or* (b) *diverging or if the connection is converging* (c) *and neither* V *nor any of its descendants* V′ *are instantiated.*

7.6.1 Inference

Inference in Bayesian networks is the process of calculating the probability of one or more random variables given some evidence, i.e. computing $\mathsf{P}(\mathsf{X}_\mathsf{Q}|\mathsf{X}_\mathsf{E} = \mathsf{x}_\mathsf{E})$ where X_Q is a set of query variables and X_E is a set of evidence variables. Let X_H be the set of variables that are not part of the query and that have not received any evidence:

$$\mathsf{X}_\mathsf{H} = \{\mathsf{X}_1, \mathsf{X}_2, \ldots, \mathsf{X}_\mathsf{N}\} \setminus (\mathsf{X}_\mathsf{Q} \cup \mathsf{X}_\mathsf{E}), \tag{7.3}$$

then

$$\mathsf{P}(\mathsf{X}_\mathsf{Q}|\mathsf{X}_\mathsf{E}) = \frac{\mathsf{P}(\mathsf{X}_\mathsf{Q}, \mathsf{X}_\mathsf{E})}{\mathsf{P}(\mathsf{X}_\mathsf{E})} = \frac{\sum_{\mathsf{X}_\mathsf{H}} \mathsf{P}(\mathsf{X}_\mathsf{H}, \mathsf{X}_\mathsf{Q}, \mathsf{X}_\mathsf{E})}{\sum_{\mathsf{X}_\mathsf{H}, \mathsf{X}_\mathsf{Q}} \mathsf{P}(\mathsf{X}_\mathsf{H}, \mathsf{X}_\mathsf{Q}, \mathsf{X}_\mathsf{E})}. \tag{7.4}$$

For given evidence the denominator is a constant, therefore we can write:

$$\mathsf{P}(\mathsf{X}_\mathsf{Q}|\mathsf{X}_\mathsf{E}) = \alpha \sum_{\mathsf{X}_\mathsf{H}} \mathsf{P}(\mathsf{X}_\mathsf{H}, \mathsf{X}_\mathsf{Q}, \mathsf{X}_\mathsf{E}). \tag{7.5}$$

Inference thus comes down to marginalisation over X_H followed by normalisation. Directly applying equation (7.5) is intractable for most networks, but a number of efficient algorithms exist that exploit the independence of variables in a network.

Message passing

Pearl (1988) introduced an efficient algorithm for tree shaped networks that was later extended to singly connected networks, i.e. graphs in which there is at most one directed path between any two nodes. It exploits the fact that computations are local if a node is independent from the rest of the network given its Markov

blanket, which include its parents, its children and its children's parents. The latter have to be included because of the explaining away effect. In this approach the graph is used as a computational architecture, where the arcs act as communication channels. New information is propagated through the network by local message-passing. Nodes receive messages from neighbouring nodes, update their beliefs and send updated messages back to their neighbours. This process is guaranteed to reach an equilibrium after a number of cycles. This results in updated probability distributions, which are probabilistically consistent with the evidence.

For a typical node X with parents $U = \{U_1, \ldots, U_n\}$ and children $Y = \{Y_1, \ldots, Y_m\}$ message passing proceeds as follows: Using the notation of Pearl (1988) let e be the total evidence. As the network is singly connected information put on an evidence node always reaches X through one particular parent or a particular child. Therefore, we can split the evidence in two sets. Let

$$e_x^- = \left\{ e_{XY_1}^-, \ldots, e_{XY_m}^- \right\} \tag{7.6}$$

be the evidence that reaches X through its children, where $e_{XY_j}^-$ is the evidence connected to X through Y_j. And let

$$e_x^- = \left\{ e_{XY_1}^-, \ldots, e_{XY_m}^- \right\} \tag{7.7}$$

be the evidence that reaches X through its parents, where $e_{U_i X}^+$ is the evidence connected to X through U_i. The belief of X after receiving the evidence e is:

$$\begin{aligned} Bel(X) &= P(X|e) = P(X|e^-, e^+) && (7.8)\\ &= \frac{P(e_X^-|e_X^+, X)P(X|e_X^+)}{P(e_X^-|e_X^+)} && (7.9)\\ &= \alpha P(e_X^-|X)P(X|e_X^+) && (7.10) \end{aligned}$$

The last step follows as X d-separates e^+ and e^-. Let $\lambda(X) = P(e_X^-|X$ be the information X receives from its children and $\pi(X) = P(X|e_X^+)$ be the information X receives from its parents. The belief of a node can than be expressed as:

$$Bel(X) = \alpha\lambda(X)\pi(X). \tag{7.11}$$

Intuitively, the messages either give diagnostic or predictive support for the belief of a node.

Updating

Belief updating is now a matter of collecting messages from the parents and children of X:

$$\begin{aligned}
\lambda(X) &= P(e^-_X|x) && (7.12)\\
&= P(e^-_{XY_1},\ldots,e^-_{XY_m}|X) && (7.13)\\
&= P(e^-_{XY_1}|X)\ldots P(e^-_{XY_m}|X) && (7.14)\\
&= \prod_{i=1}^{m}\lambda_{Y_j}(X) && (7.15)
\end{aligned}$$

where $\lambda_{Y_j}(X) = P(e^-_{XY_j}|X)$ is the message X receives from Y_j.

$$\begin{aligned}
\pi(X) &= P(X|e^+_X) && (7.16)\\
&= \sum_{U_1,\ldots,U_n} P(X|U_1,\ldots,U_n)P(U_1,\ldots,U_n|e^+_X) && (7.17)\\
&= \sum_{U_1,\ldots,U_n} P(X|U_1,\ldots,U_n)P(U_1|e^+_{U_1X})\ldots P(U_n|e^+_{U_nX}) && (7.18)\\
&= \sum_{U_1\ldots U_n} P(X|U_1,\ldots,U_n)\prod_{i=1}^{n}\pi_X(U_i) && (7.19)\\
&= \sum_{U} P(X|U)\prod_{i=1}\pi_X(U_i) && (7.20)
\end{aligned}$$

where $\pi_X(U_i) = P(U_i|e^+_{U_iX})$ is the message X receives from U_i.

Propagation

After X has updated its belief it can propagate information back to its parents and children. The message X sends to a parent U_k is calculated by:

$$\lambda_X(U_i) = \beta\sum_X \lambda(X)\sum_{U_k:k\neq i} P(X|U)\prod_{k\neq i}\pi_X(U_k). \tag{7.21}$$

The message X sends to a child Y_j:

$$\pi_{Y_j}(X) = \alpha\prod_{k\neq j}\lambda_{Y_k}(X)\pi(X) = \frac{\pi(X)\lambda(X)}{\lambda_{Y_k}(X)}. \tag{7.22}$$

Initially, all λs are set to vectors of ones. The π vectors follow from propagation of the prior probabilities of the root nodes though the network.

In multiply-connected networks multiple directed paths between two nodes are allowed, but then the message passing algorithm cannot be used as the assumptions

underlying the messages passing equations shown above may no longer be valid. As a consequence, messages might circulate indefinitely and the same information might be counted double as information with regard to its origin is lost in the propagation process.

Variable Elimination

Variable elimination (Zhang and Poole 1996; Dechter 1999; Aji 2000) is an alternative inference method that directly uses the factored representation of the joint distribution. It is based on the notion that marginalisation can be made more efficient by 'pushing sums inside of products'. By moving terms as far left as possible the amount of computational work is minimised. The restriction is that all variables appearing in a term must be in the scope of the appropriate sum operator.

The Junction Tree Algorithm

If one wants to calculate the marginals $\mathsf{P}(\mathsf{X}_i|\mathsf{X}_e)$ for all variables $\mathsf{X}_i \notin \mathsf{X}_e$, one can do so by calling the variable elimination algorithm once for every variable. As we will be recalculating many of the intermediate results several times this is rather inefficient. The related junction tree algorithm chooses its intermediate results in such a way that it can efficiently calculate the marginals for all X_i at once. It does so by a change of variables (Jensen 2001). A new tree-shaped graph is created that defines exactly the same joint probability distribution as the original graph on which a message passing scheme can then be used.

The variables in the new tree are cliques of variables in the original graph. The structure used most often is called a junction tree. In fact, variable elimination implicitly creates a junction tree. The difference is that we now save the intermediate terms in the tree and reuse them. Basically, the variables elimination process is performed in all directions at once.

The first step in junction tree construction is to explicitly represent relations between parents that share a common child by connecting them and then dropping the directionality on all arcs. This process is called moralization.

The next step is to ensure that evidence is not propagated along multiple paths to the same node, where it would be combined as separate independent pieces of evidence. This is accomplished by adding links, called fill-in edges, that break cycles of length four or greater. The result is called a triangulated graph. The triangulated graph can be found by eliminating the variables one by one and adding edges between all pairs of uneliminated neighbours. The set formed by a variable and all of its uneliminated neighbours is called an elimination set and is a clique of the original graph.

The elimination process thus finds a collection of cliques. The time complexity of inference in junction trees is exponential in the size of those cliques. To keep cliques as small as possible one should add as little fill-in arcs as possible. Unfortunately, finding an elimination order that results in the most optimal triangulation is NP-hard. However, several algorithms, such as the min-fill heuristic, the min-weight

heuristic and simulated annealing, exist that typically result in good elimination orders.

For the remaining steps only cliques that are not contained in any of the other cliques, called maximum cliques, are kept. From the set of maximal cliques a junction graph is created by connecting cliques C_i and C_j if $C_i \cap C_j \neq \emptyset$. The weight of this edge is set to $|C_i \cap C_j|$, the number of variables they have in common. Any maximal weight spanning tree of this graph is a junction tree.

The junction tree is completed by adding the probability distributions of the Bayesian network to the cliques. A distribution can be added to any clique that contains the corresponding variable. As a consequence, a clique may contain more than one distribution. The distributions belonging to a clique are multiplied together elementwise. The result is no longer guaranteed to be a probability distribution, but is called a (clique) potential.

Message passing in a junction tree

Inference in junction trees is done with a two-pass message passing scheme. When new evidence is added a collect phase is performed in which one of the junction tree nodes is seen as the root of the tree. All nodes collect messages from their children and pass a message to their unique parent. A distribution phase follows in which a node receives a message from its parent, calculates its new potential and passes it to all its children. The collect-distribute cycle is sometimes referred to as calibrating the tree. Different message passing algorithms exist. The most important of which are the Hugin or JLO algorithm (Jensen *et al.* 1990; Jensen 2001) and the Shafer-Shenoy algorithm (Shafer and Shenoy 1990; Shenoy and Shafer 1990). These algorithms are both variations on a simple message passing scheme that is often referred to as belief propagation (Murphy 2001). For a junction tree node j with one or more children and a unique parent k belief propagation works as follows. Let ψ_j be the initial clique potential of j and let S_j be the domain of this clique, i.e. the set of variables that form the clique. The node collects messages $\mu_{i\to j}$ from all its children i in the collect phase, i.e. from all its neighbours (as given by the function $\mathrm{nbr}(j)$) except its parent to obtain a new potential ϕ_j:

$$\phi_j = \left[\prod_{i\in \mathrm{nbr}(j), i\neq k} \mu_{i\to j}\right] \psi_j, \tag{7.23}$$

$$\mu_{j\to k} = \sum_{S_j \setminus S_{jk}} \phi_j, \tag{7.24}$$

where $S_{jk} = S_j \cap S_k$. In the distribution phase node j updates its potential ψ_j to its new value using the message received from its parent k and sends a message to all its children i that is of the product of the messages received from all other children, the parent node and the initial clique potential:

$$\phi_j^* = \phi_j \times \mu_{k\to j}, \tag{7.25}$$

$$\mu_{j\to i} = \sum_{S_j \setminus S_{ij}} \left(\left[\prod_{i' \in \text{child}(j), i' \neq i} \mu_{i'\to j} \right] \mu_{k\to j} \psi_j \right). \tag{7.26}$$

One of the differences with message passing that operates directly on the Bayesian network is that it is no longer straightforward to interpret these messages in terms of predictive or diagnostic support as these messages can represent the combination of diagnostic and predictive effects that the cliques of variables have on each other.

Approximate Inference

The complexity of inference using the junction tree algorithm is $O(d^{|C_{max}|})$, where d is the maximum number of states of each variable can take and $|C_{max}|$ is the size of the largest clique. For many networks this means that exact inference is intractable and one has to fall back on more efficient algorithms that only approximate the correct results. Several methods have been developed.

Loopy belief propagation (Pearl 1988) applies message passing to the graph even if it has loops. As noted before, there is a danger of counting information twice, and the algorithm may not converge or converge to the wrong answer, but in practice the method works surprisingly well (Murphy 2002).

Alternatively, stochastic sampling methods are used. These methods randomly generate a sufficiently large number of configurations from the joint distribution and then the frequency of relevant events is counted:

$$P(H = h|E = e) \approx \frac{\text{number of configurations where } H = h \text{ and } E = e}{\text{number of configurations where } E = e}. \tag{7.27}$$

The simplest form of sampling is logic sampling that generates the configurations one at a time. It starts by generating values at the root nodes using the prior probabilities and then follows the directions of the arcs, generating values according to the conditional distributions and the generated values of the parent nodes. The disadvantage of this method is that it may generate many configurations that are incompatible with the observations, this is especially troublesome in case of rare observations. In general, the fraction of useful runs deceases exponentially with the number of evidence variables (Russell and Norvig 1995).

To deal with this issue, importance sampling (Srinivasan 2002) weights every sample by the likelihood of the evidence using the conditional probabilities. Importance sampling converges much faster than logic sampling.

Stochastic sampling methods have several advantages over deterministic approximation algorithms: they are easy to implement and can deal with mixed CPDs, with variable state spaces or changing model structure while they are, in the limit of an infinite number of samples, guaranteed to converge to the exact answers. The major disadvantage is speed. Sampling methods are significantly slower that deterministic methods. In particular, it takes a long time to reach accurate probabilities for unlikely events, making them unsuitable for large models and large data sets.

Cutset conditioning selects nodes in the multiply connected network whose deletion would remove a loop. For each value of such a node a new network is constructed in which the node is treated as an observed node set to that particular value. Message passing is then performed in each of the tree-shaped networks and the results are combined using marginalization. The problem with this approach is that the number of networks grows exponentially with the number of cycles: In a network with k cycles $O(k)$ variables must be set, for binary variables this would result in $O(2^k)$ trees.

7.6.2 Learning

For Bayesian networks both the structure as well as the probability distributions can be learned from data (Jordan 1998; Heckerman *et al.* 1995). Algorithms for learning include constrained-based-learning and score-based learning. Parameters may be learned using a maximum a posteriori approach (MAP), a maximum likelihood approach (ML) or a discriminative approach. The networks may be fully observed or partially observed during learning. Networks are partially observed if some data is missing or because some nodes are hidden. Learning algorithms may follow a frequentist or Bayesian approach. Furthermore, learning can be off-line, estimating the parameters or structure from a fixed batch of data or on-line, where the parameters are sequentially updated for every new piece of data. Both supervised and unsupervised learning algorithms exist. Here we discuss parameter learning in case of partial observability using a frequentist approach.

Expectation-Maximisation Algorithm

As mentioned above, if all nodes in the network are fully observed, i.e. the examples in the data set contain values for all variables, the parameters of the network can be estimated using a simple maximum likelihood approach, similar to that discussed in section 3.2. However, if some nodes are observed or the data is incomplete or has missing values, we can no longer maximise the likelihood function directly. The expectation-maximisation algorithm (EM) circumvents this problem by taking an educated guess at values of the hidden variables and maximising the expected likelihood rather than the likelihood itself (Dempster, A. P. *et al.* 1977).

Let E be the set of observed variables an H be the set of hidden variables and $P(E, H|\theta)$ the joint distribution with parameters θ. Then as with maximum likelihood estimation we can define a log-likelihood function that is to be optimised using a data set $D = \{d_1, d_2, \ldots, d_M\}$:

$$L(\theta|E, H) = \sum_{m=1}^{M} \log P(E, H|\theta, d_m). \tag{7.28}$$

Since both E and θ are constant we can think of the likelihood as a function of the random variables H. The EM-algorithm operates in two steps. In the E-step the

value of the log-likelihood with respect to the unknown data H given the observed data E and the current parameter estimates θ' are found:

$$Q(\theta, \theta') = E\left[\log P(E, H|\theta)|E, \theta'\right]. \tag{7.29}$$

In the discrete case this can be written as:

$$Q(\theta, \theta') = \sum_m \sum_h \log P(E, h|\theta) P(h|E, \theta'). \tag{7.30}$$

In the second step of the algorithm, the M-step, the expectation calculated in the E-step is maximised:

$$\theta = \arg\max_\theta Q(\theta, \theta'). \tag{7.31}$$

These steps are iteratively repeated. Each iteration is guaranteed to increase the log-likelihood (Dempster, A. P. *et al.* 1977) and the algorithm is guaranteed to converge to a local maximum of the likelihood function. For multinomial CPDs the E-step becomes:

$$Q(\theta'|\theta) = \sum_{ijk} \sum_m P(X_i = k, Pa(X_i) = j|d_m, \theta') \log \theta'_{ijk}, \tag{7.32}$$

where $\theta_{ijk} = P(X_i = k|Pa(X_i) = j, \theta)$.
The M-step becomes:

$$\theta_{ijk} = \frac{\sum_m P(X_i = k, Pa(X_i) = j|d_m, \theta')}{\sum_{k'} \sum_m P(X_i = k', Pa(X_i) = j|d_m, \theta')}. \tag{7.33}$$

Thus learning comes down to computing $\sum_m P(X_i, Pa(X_i)|d_m, \theta')$ using an inference algorithm of choice and then use those values in the M-step to set the new parameters of the network. This is a generalisation of the Baum-Welch algorithm for HMMs. See Bilmes (1998) for a comprehensive introduction of the EM-algorithm.

7.7 Dynamic Bayesian Networks

Dynamic Bayesian networks (DBNs) model processes that evolve over time. A DBN can be defined by two Bayesian networks: a prior $P(X_1)$ and a transition model that defines how the variables at a particular time depend on the nodes at the previous time steps:

$$P(X_t|X_{t-1}) = \prod_{n=1}^{N} P(X_t^i|Pa(X_t^i)), \tag{7.34}$$

were X_t^i is the i^{th} variable in slice t. The parents of a node can either be in the current or in a previous time slice. Typically, first order Markov assuptions are made, i.e. the nodes in a time slice only depend on the nodes in the previous time slice[2]. If there is an arc from X_{t-1}^i to X_t^i the node is called persistent. Note that

[2] k-th order Markov relations can always be rewritten as first-order relations.

despite their name DBNs are not dynamic, the network topology and the CPDs do not change with time[3].

7.7.1 Inference

For DBNs several types of inference can be defined:

Filtering calculates $P(X_t|o_{1,t})$, i.e. the distribution of the current state of the model given all evidence up to the current time slice.

Smoothing computes $P(X_t|o_{1,T})$, the state distribution at time t given all evidence.

Fixed lag smoothing is similar but uses only some information from the future: $P(X_t|o_{1,t+h})$ with $h > 0$.

Prediction computes $P(X_{t+h}|o_{1,t})$ for some $h > 0$, the distribution of a future state given all evidence up to the current time slice.

Viterbi finds the probability of the sequence of states which are most likely to have generated the evidence: $\mathtt{argmax}_{x_{1,t}} P(x_{1,t}|o_{1,t})$.

As DBNs are Bayesian networks all inference algorithms for Bayesian networks can also be used for DBNs. However, this is often not feasible as for example for the junction tree algorithm one would have to unroll the entire network, which for most networks will not fit into memory, that is if one happens to know the length of the network beforehand at all. On-line algorithms have been developed that process the network slice-by-slice. At any point in time they only keep a window of a limited number of slices in memory. The next two sections discuss algorithms that have specifically been designed for on-line inference in dynamic Bayesian networks.

The Frontier algorithm

The forward-backward algorithm for HMMs exploits the fact that X_t separates the past from the future. The frontier algorithm uses as similar idea. For a DBN, the set of all hidden nodes in a time slice, $X_t^{(1,D)}$, d-separates the past from the future. The basic idea is to sweep a Markov blanket across the DBN, first forwards and then backwards.

Using notation from Zweig (1998) and Murphy (2002), let F be the set of nodes in the frontier. The nodes to the left of the frontier are denoted by L and the nodes to the right of the frontier by R. Furthermore, h_F is the set of hidden nodes in F, e_F are the evidence nodes in F, e_L refers to the evidence in L and e_R refers to the evidence in R. In the forward pass we define: $P(F) = P(h_F, e_F, e_L)$. A node X can be added to the frontier, i.e. be moved from R to F, when all its parents are already

[3]Although, one can encode dynamic behaviour in the network.

in the frontier as this will guarantee that X is independent from e_L. A node can be added by multiplying its CPD onto the frontier:

$$P(e_L, e_F, h_F, X) = P(e_L, e_F, h_F)P(X|e_F, h_F). \tag{7.35}$$

Similarly a node can be removed from the frontier (moved from F to L) when all its children are in the frontier. If X is hidden $e_{L\cup\{X\}} = e_L$ and $e_{F\{X\}} = e_F$. Thus removing a node simply means marginalising it out:

$$\begin{aligned} P\left(e_L \cup \{X\}, e_F - \{X\}, h_F - \{X\}\right) &= P\left(e_L, e_F, h_F - \{X\}\right) \\ &= \sum_X P\left(e_L, e_F, X, h_F - \{X\}\right) = \sum_X P\left(e_L, e_F, h_F\right). \end{aligned} \tag{7.36}$$

When X is observed we can skip the marginalisation, making it essentially a no-op. The procedure is equivalent to variable elimination with a specific ordering. As a matter of fact, the Frontier algorithm implicitly creates a chain-like junction tree.

In the backward pass, we define $P(F) = P(e_R|h_F, e_F)$. We can advance the frontier from slice $t+1$ to slice t by adding and removing nodes in the opposite order that we used in the forward pass. Adding a node means moving it from L to F. Removing a node means moving it from F to R.

When adding X to F, we know that all X's children are in F, because X was removed at this step in the forward pass. So X is shielded from e_R. Adding X simply means expanding the domain of the frontier to contain it, by duplicating all the existing entries, once for each possible value of X.

To remove node X, we multiply in X's CPD, this is possible since all of X's parents will be in F, since X was added at this step in the forward pass, and then marginalize out X. If X is observed, the procedure is basically the same, except we don't need to marginalize out X, since it only has one possible value.

The Interface algorithm

The frontier algorithm uses all the hidden nodes in a slice to d-separate the past from the future. This set is larger than needs to be, and hence the algorithm is sub-optimal. Murphy (2002) shows that the set of nodes with outgoing arcs to the next time-slice is sufficient to d-separate the past from the future. The interface algorithm is based on this notion. It creates a junction tree for an unrolled $1\frac{1}{2}$-slice DBN which consists of a time slice and all interface nodes from the preceding slice. It then imposes the restriction that all the nodes in the forward interface must belong to one clique. This can be ensured by adding edges between all the nodes in the interface to the moral graph during junction tree construction. The junction trees can be glued together via their interfaces. Inference can be performed in each tree separately and then messages are passed through the interfaces, first forwards and then backwards.

The Island algorithm

Even when using on-line algorithms, the state space requirements of smoothing are $O(TS)$ where S is the size of the forward message, as all forward messages have to be saved. At the other extreme one can repeat the forward calculations up to the current slice in every backward step, resulting in constant space complexity. The island algorithm chooses a point between the extremes. During the forward pass messages are saved at C island points (including the first and last slice), resulting in $C-1$ subproblems on which the algorithm is then called recursively. If messages are saved every $\sqrt{T}$ steps the space complexity will be reduced to $O(S \log_C T)$, the cost of this is an increase in time complexity from $O(T)$ to $O(T \log_C T)$.

7.7.2 Approximate inference

Approximate on-line inference algorithms exist as well. The Boyen-Koller algorithm (Boyen and Koller 1998) approximates the joint distribution over the interface as the product of smaller terms (marginals). Compared to the interface algorithm, the Boyen-Koller algorithm drops the requirement that all nodes in the interface are in the same clique. The accuracy of the algorithm depends on the number of clusters that is used to represent the interface. Using a single cluster corresponds to exact inference. The other extreme is the fully factorized representation with one cluster per variable. Boyen-Koller does exact inference in a two-slice DBN, sometimes this approximation is still intractable.

The Boyen-Koller algorithm can be thought of as a factored version of the interface algorithm, in the same way, as its name implies, the Factored Frontier algorithm (Murphy 2002) is a factored version of the Frontier algorithm. The frontier distribution is approximated as a product of marginals. Both algorithms can be seen as special cases of a single run of loopy belief propagation. Therefore, running multiple iterations improves the approximation.

As an alternative a Viterbi approximation can be used, when marginalizing the sum operator is replaced by the max operator. The rationale being that the most likely path may contain most of the probability mass.

Particle filtering is a simple and efficient sampling algorithm for DBNs that computes N samples in parallel. Essentially, particle filtering is sequential importance sampling with resampling. It starts by creating a population of N samples by sampling from the prior distribution of time slice 0. Next, the samples are propagated through the system using the transition model and each sample is weighted by the likelihood of being in the sampled state given the evidence using $P(e_{t+1}|x_{t+1})$. The whole population is then resampled according to the weights to generate a new set of the N most likely samples that are propagated to the next slice again. The algorithm thus focuses on the set of samples in the high-probability regions of the state space. One of the advantages of particle filtering is that it can be used with arbitrary CPDs. On the down side, particle filtering can be significantly slower than deterministic methods. To get the best of both worlds combinations of discrete approximation

methods and stochastic sampling also exist. Murphy (2002) describes a technique called Rao-Blackwellized particle filtering (Doucet *et al.* 2000).

7.8 Speech recognition with DBNs

DBNs have been used in speech recognition. Interesting enough Pearl (1988) already mentions speech recognition as an application area that might benefit from the 'theoretical and computational tool' of Bayesian networks. Zweig (1998) was the first to propose DBN-based acoustic modelling. His model is essentially a simplified version of the hierarchical model that will be presented in this section. Bilmes (1999) presented a similar model but focussed on modelling relationships at the feature level. Both showed that including variables between the phone state and the output distribution that represent the positions of articulators as tongue and lips can improve speech recognition performance.

Another area in which DBNs have proven their usefulness, mainly in the shape of Coupled Hidden Markov models (Brand *et al.* 1997) and Factorial Hidden Markov model (Ghahramani and Jordan 1997) is audio-visual speech recognition. Unlike the multi-stream Markov model that is often used for this task (Bourlard *et al.* 1996) such models can easily deal with different frame rates and asynchrony between input streams.

To see how speech recognition can be performed with DBNs it is illustrative to look at the relationship between DBNs and HMMs. In figure 7.2 a 3-state hidden Markov Model is shown, together with an equivalent dynamic Bayesian network that is unrolled for T time slices.

Figure 7.2 also shows a trellis diagram that displays all temporal paths through the HMM. The diagram is obtained by 'unfolding' the HMM in time, explicitly showing the transitions that can be made in every time step. By folding back the trellis in vertical direction rather than in horizontal direction the DBN is obtained. Each of the variables q_t in the DBN may assume one of three states $\{1, 2, 3\}$ corresponding to the states of the HMM at time t. The probability that q_t will be in a particular state at time t, depends upon the probabilities that its predecessor q_{t-1} is in a certain state and upon the transitions probabilities $P(q_t = i|q_{t-1} = j)$ encoded by the link between the two nodes. These link probabilities equal the transition probabilities a_{ij} in the HMM. Graphically, the arrows between time slices in the trellis, each of which represents a single probability, are collapsed into one link in the DBN that has a probability distribution attached to it[4].

Thus, in theory, speech recognition can thus be performed with a model as shown in figure 7.2. However, this is rather cumbersome as all knowledge on state transitions, phoneme transitions and word transitions has to be be encoded into a single probability distribution function $P(q_t|q_{t-1})$. To see how to do better one first should realise that the models used in speech recognition are in fact hierarchical: sentences are sequences of words, words are sequences of phones and phones are sequences

[4]Strictly speaking the distribution is attached to the state rather than to the arc.

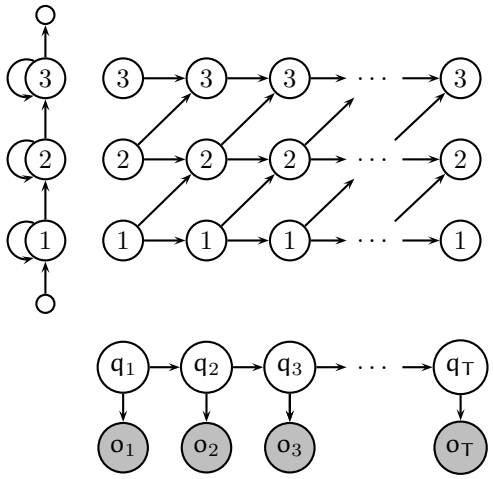


Figure 7.2 – *An* HMM *and a corresponding* DBN. *The random variable* q_t *takes the states of the* HMM *at time* t *as its values. The shaded* o_t *nodes represent the observations.*

of HMM states. For practical purposes this hierarchy is normally converted in to a flat HMM structure before recognition but it could explicitly be represented using a Hierarchical HMM (Fine *et al.* 1998), an example of which is shown in figure 7.3. A path through this model is found as follows: if a state has vertical transitions one of these is taken according to some probability distribution before a horizontal transition is taken. When the end state of a submodel is reached control is returned to the state at the previous level that called the model and one of its horizontal (or self loop) transitions is taken as usual in an HMM. The model is thus traversed in a depth-first fashion. Only the leaf nodes generate observation symbols.

State tying is realised in such a model by letting several nodes point to the same submodel. Murphy (2002) shows how a Hierarchical HMM can be represented as a dynamic Bayesian network, which results in a faster inference routine. Figure 7.4 shows two time slices of the HMM corresponding to the model of figure 7.3.

In this model the W nodes represent words, the P nodes represent phones, S nodes correspond with HMM states and the square O nodes represent observation vectors. The F nodes are switch nodes that can take the values 'off' and 'on' and signify that the level indicated by their subscript has just finished. They correspond to the small end nodes of figure 7.3. A trace trough the network shows how it works. In the first time slice the W node will be in state 'he' and the P node will be in state /h/ and the S node will be in state 1 of the HMM corresponding to /h/. All F nodes will be in the off state. Now at every level a transition to another state is only allowed if the F node below it is turned on, otherwise the transition has to be to the same state. Thus in the examples of Figures 7.3 and 7.4 initially only the S variable is allowed to change state, until it decides that it has finished (reached the small

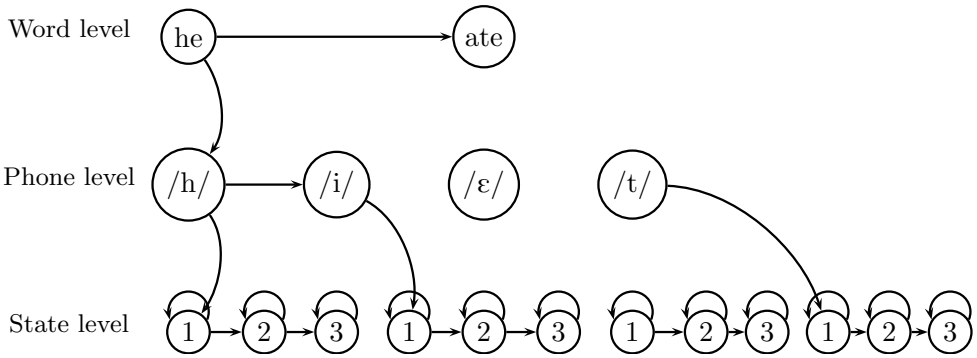


Figure 7.3 – *A hierarchical* HMM *for speech recognition. Links to lower levels are processed before links at the same level. Only the lowest level has state-dependent observation distributions.*

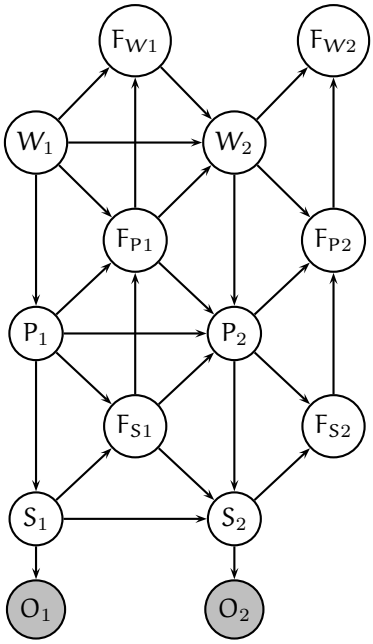


Figure 7.4 – *Two timeslices of a* DBN *representation of a hierarchical* HMM. O_t *is the observation vector at time* t, W_t *is the word at time* t, P_t *the phone at time* t *and* S_t *corresponds to the* HMM *state at time* t. *The* F *nodes are binary nodes that signal when a level is allowed to change state.*

end node in figure 7.3), then it switches its F node on, meaning that in the following time slice the P node should change state and thus transition to the next phoneme and the S node in the next slice should be the first node of the HMM corresponding to this phoneme. In the same way the last phone model of a word will indicate that the word level should make a transition to the next word.

Figure 7.4 shows the most general way to represent a hierarchical HMM as a DBN. As the models of speech recognition contain many independence assumptions, e.g. that a state usually belongs to a single unique phone, these models often can be simplified. Murphy (2002) and Zweig (1998) discuss several simplifications and variations on this model that are tailored to the task of speech recognition. Bilmes (1999) developed several Bayesian network models for acoustic processing of speech.

The models discussed here are equivalent to HMMs, but the real power of the DBN shows when more information is included in the model. Zweig (1998) mentions including user information like gender and age and describes a successful experiment with a system that conditions its choice of acoustic mixtures upon a single node that is supposed to capture such information. This node has no specific a priori meaning; it simply learns its behaviour during training.

7.9 Conclusion

To easily experiment with different configurations, a uniform representation for all models is preferable, therefore our main weapon of choice will be dynamic Bayesian networks. The advantage of these models is that they are very flexible and subsume the Hidden Markov models and n-gram models normally used in speech recognition, but allow for factored state spaces and thus more interesting and efficient models.

On the down side DBNs are less powerful than probabilistic grammars when it comes to representing hierarchical structure and dependencies always have to be cast in terms of causal relationships between variables.

Dasein ist nie »zunächst« ein gleichsam in-seins-freies Seiendes, das zuweilen die Laune hat, eine »Beziehung« zur Welt aufzunehmen. Solches Aufnehmen von Beziehungen zur Welt ist nur möglich, *weil* Dasein als In-der-Welt-sein ist, wie es ist. Diese Seinsverfassung entsteht nicht erst dadurch, daß außer dem Seienden vom Charakter des Daseins noch anderes Seiendes vorhanden ist und mit diesem zusammentrifft. »Zusammentreffen« kann dieses andere Seiende »mit« dem Dasein nur, sofern es überhaupt innerhalb einer *Welt* sich von ihm selbst her zu zeigen vermag.

Sein und Zeit, Martin Heidegger

Chapter 8

DBNs for Speech and Language Processing

In which several well-known language models are reformulated in terms of dynamic Bayesian networks. Subsequently, a number of new language models that include context knowledge is defined and it is shown how context can be included in acoustic models. Finally, DBN *models of syntax are defined.*

Although DBNs have been used in speech recognition their use in language modelling and in natural language processing in general remains rather limited, probably because other techniques such as grammars and weighted finite state transducers are customary in those areas. This chapter will show how several well-known language models can be formulated as DBNs and how context information can be included in language models as well as in acoustic models.

8.1 N-grams

Figure 8.1 shows a DBN representation of a trigram language model. Every time slice contains a single word variable that is connected to its two predecessors. Although

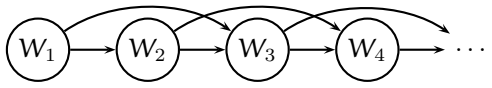


Figure 8.1 – *A conceptual trigram dynamic Bayesian network. Every time slice contains a random variable W that takes the words in the vocabulary as its states.*

conceptually this is all there is to it, there are a number of issues we have to deal with to obtain a proper language model.

8.1.1 Separating observations and control statements

The first n words pose a problem for n-grams, as the history used by these models is not completely available yet. The standard solution to this problem is the use of dummy states in front of a sentence. The first word of the sentence then depends on n dummy states (Manning and Schütze 1999). The same can be done in DBNs, but there is a more elegant, conceptually clearer, solution that separates control information from the observation distributions. A variable N is introduced that counts the words in a sequence. Based on the value of this counter a word distribution is chosen. For the first slice the a priori distribution is used, for the second slice a distribution that conditions the word only on the previous word and from the n-th slice on the standard n-gram distribution is used. Note that for an n-gram model with a vocabulary V the parameter space is reduced from $|V+1|^n$ to $|V|^n + |V|^{n-1} + \ldots + |V| + n$, given that the counter variable needs to have only n different states. For a trigram over a modest sized vocabulary of 1000 words, this reduces the number of parameters from 1,003,003,001 to 1,001,001,002. Replacing dummy states by control variables has other advantages. As dummy states only occur at the beginning of a sentence most n-gram involving dummy states cannot occur and thus will have zero probability. So, unless we take special measures, such impossible states will get a non-zero probability and thus steal away probability mass from other events when applying a smoothing algorithm.

8.1.2 All good things must come to an end

Most language models also use a dummy state that signals the end of a sentence. In fact, they must do so in order to be proper language models. As described in chapter 3 the task of a language model is to assign a probability to every sentence in a language. However, the chain rule of (3.1) that is repeated below, only deals with sentences of a particular length, rather than with sentences of all possible lengths:

$$P(w_{1,t}) = P(w_1)P(w_2|w_1)P(w_3|w_1w_2)\ldots P(w_n|w_{1,t-1}). \tag{8.1}$$

To correct this problem, the end state should be a sink state, i.e. it only has an outgoing transition to itself.

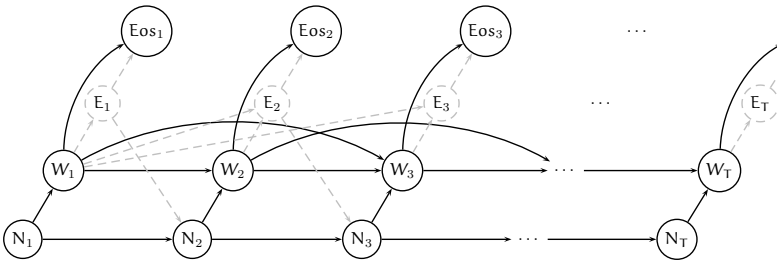


Figure 8.2 – *A trigram* DBN. *The* N *variables give the positions of the words in a sentence. The* E *nodes are binary variables that indicate the end of a sentence.* Eos *is a binary variable that indicates the end of the utterance.*

To deal with this in DBN models we introduce a control variable EOS that signals the end of a sentence. Requiring that this variable is true at the end of the sentence (and false in all other cases) ensures that the probabilities the model assigns to all sentences in the language will sum to one.

In speech recognition an utterance may span several sentences. In this case it is useful to differentiate between an end-of-sentence node E and an end-of-sequence node EOS. The word counter N can be conditioned on the end-of-sentence variable to let it restart for every sentence. For the first words of a sentence the a priori distribution can be used again, so it will not depend on the last words of the previous sentence. We may also choose to use a specific sentence transition distribution. Figure 8.2 shows a model with end-of-sequence nodes and optional end-of-sentence nodes. Note that the end-of-sentence and end-of-sequence nodes do not have to be conditioned on the previous n nodes as would be the case with dummy states in an n-gram, but can be conditioned on anything that is deemed useful.

While language models that are used for machine translation or augmentative communication can use punctuation to find the end of a sentence, in speech recognition the end of a sentence is typically not known. Nevertheless, including end-of-sentence nodes can be useful, as it will provide a better model of language. In this case the end-of-sentence variable is treated as a hidden variable.

Unless otherwise stated, the remaining models discussed in this chapter all include positions counters, end-of-sentence nodes and end-of-sequence nodes. To keep the figures clear, these nodes will not always be shown.

8.1.3 Smoothing

As discussed in chapter 3 data sparseness presents an ever-present problem to language modelling. The standard solution is smoothing. To realise smoothing in DBNs one can simply incorporate smoothed distributions in the model. As an alternative, many smoothing algorithms can be modelled directly in the DBN. This hold in particular for interpolation schemes, which include many popular schemes such as

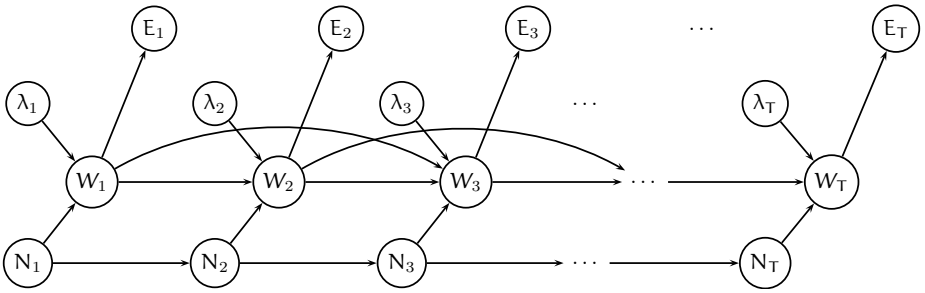


Figure 8.3 – *An interpolated trigram. The hidden variable* λ *is used to decide on which other parents the word depends.*

Kneser-Ney.

Figure 8.3 shows an interpolated trigram. The λ variables implement smoothing. Depending on its value the current word does either not depend on it predecessors at all, only on the previous word, or on the previous two words. As λ is a hidden node, the result is a mixture of distributions implementing deleted interpolation:

$$\mathsf{P}_\lambda(W_t|W_{t-1},W_{t-2}) = \lambda_1\mathsf{P}(W_t) + \lambda_2\mathsf{P}(W_t|W_{t-1}) + \lambda_3\mathsf{P}(W_t|W_{t-1},W_{t-2}). \quad (8.2)$$

The values of λ can be found by training on a held-out data set. The interpolation variables can be conditioned themselves on other variables to implement generalised interpolation. The construction of variations on n-grams such as distant n-grams in DBNs is simply a matter of linking the variables involved.

8.2 Class-based language models

Class-based language models group words into classes in order to generalise to unseen words and to obtain more reliable statistics. n-gram probabilities over classes are used to predict the class of a word which is then used to predict the word. Figure 8.4 shows the DBN counterpart of a part-of-speech model as introduced in section 3.5. The POS-tags are added to the model as hidden variables (P) that are connected in time. Compared to n-grams, class-based models achieve better generalisation and a smaller parameter set at the cost of less fine-grained modelling. To get the best of both worlds class-based models and n-grams are often combined through interpolation. This is particularly easy to accomplish in a DBN as is shown in figure 8.5 which combines a POS-model with an interpolated trigram. There is no need to derive or implement special algorithms for this model, the general purpose Bayesian network algorithms are all that is needed.

We can further improve the class-based model by adding first and second order relations on the class level as is depicted in figure 8.5. Additionally, we can let the class nodes depend on previous words or the words on previous tags.

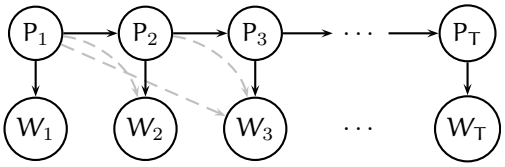


Figure 8.4 – *A class-based* POS*-model. The* P*-variables take part-of-speech tags as their states and are interconnected through time, the word variables only depend on the* POS*-classes.*

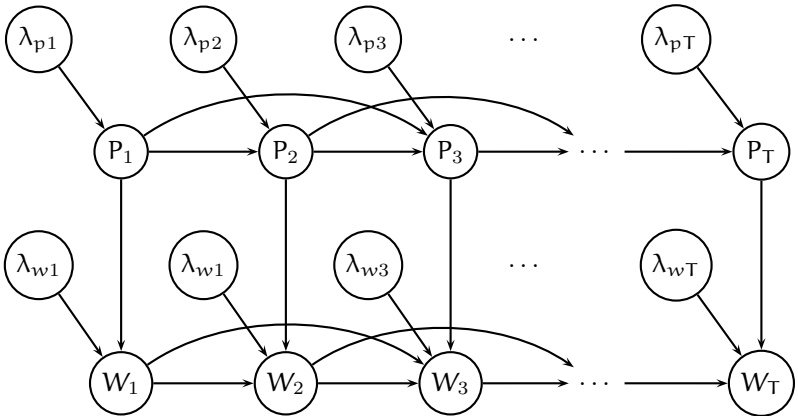


Figure 8.5 – *A combined* POS*-trigram model, words depend on* POS*-classes and on previous words. As before* λ*s are interpolation weights that implement smoothing.*

8.3 Cache-based language models

Using DBNs to construct cache-based language models is not straightforward, as cache-based models directly manipulate probabilities. One option would be to construct the cache distribution as usual and incorporate it as an a priori variable in the DBN slices. Another solution phrased completely in terms of Bayesian networks is to use a cache with a limited number of words or n-grams. The cache is then implemented by a queue. This queue consists of a number of word variables as shown in figure 8.6. At every time step the variables in the cache take on the previous value

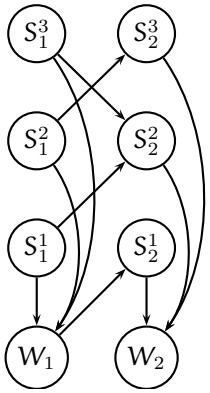


Figure 8.6 – *A cache implemented as a queue. The S variables represent the positions in the queue. At every time step a new word is shifted in the queue. All words in the queue shift up one position.*

of the variable on the next lower level. The first variable takes the value of the word that is observed in the previous timeslice. The word is conditioned on all words in the queue. The idea of a decaying cache can be implemented in this scheme through the position of the words in the queue.

8.4 Modelling context

The models discussed above are all based on existing models. Compared to the original models they have the advantages that they clearly separate control variables and observation variables, that they are all formulated within the same framework and are therefore easy to combine and that there is no need to derive special purpose inference algorithms. However, the real power of the approach is that we can add new variables to the models to obtain more accurate, context-dependent language models.

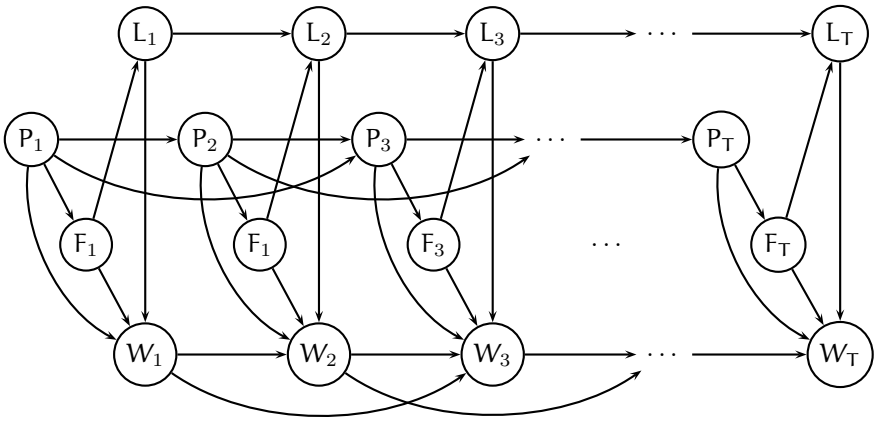


Figure 8.7 – *Modelling dependencies between content words. The* P *nodes are random variables over part of speech tags, the* F *nodes are binary variables that indicate whether the word is a content word or a function word. The* L *nodes are lemmas corresponding to content words.*

8.4.1 Modelling long-distance relationships

Often, there will be semantic relations between the content words in a sentence. This cannot be modelled with n-grams as the number of function words that separate the content words varies. Figure 8.7 shows a belief network that can model such relations. This model was first presented in (Wiggers and Rothkrantz 2006). The model is based on the trigram model with part-of-speech classes of figure 8.5. A lemma variable (L) is added that takes as its states the lemmas of all content words. We decided on using lemmas rather than the content words themselves to alleviate data sparseness somewhat. When the model moves to the next slice it will first predict the POS-tag, from which follows whether the word is a function word or a content word. This is indicated by the binary variable F. In case of a function word the word is predicted based on its POS-tag and the previous words without using the lemma as usual. The lemma in this slice will simply be a copy of the previous lemma; this way the last content lemma seen is memorised. If the POS-tag indicates a content word, the lemma will be predicted based on the previous lemma and is subsequently used in the prediction of the word.

8.4.2 Sentence length

In an n-gram language model, the probability of a sentence gets lower as the sentence gets longer. This is an artifact of the model rather than a feature. The sentence length distributions in chapter 5 clearly show that this behaviour is not entirely correct. In a DBN the sentence length can be modelled explicitly. Earlier the counter

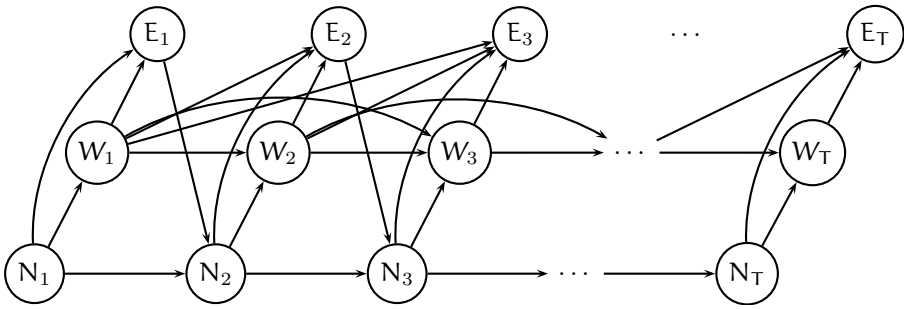


Figure 8.8 – *A trigram* dbn *with explicit sentence length encoding.* N *variables indicate the position in a sentence,* E *nodes signal the end of a sentence.*

variable N was already introduced. If this variable does not only keep track of n-gram counts, but simply all counts, the end-of-sentence variable can be conditioned on this counter to obtain a proper sentence length distribution. Figure 8.8 shows the idea.

8.4.3 Type of speech

In chapter 5 it was shown that the sentence length distributions of different types of speech are very different. To incorporate this in the model a type-of-speech node can be added. The end-of-sentence variable is conditioned on the sentence length as well as on the type-of-speech variable.

The results in chapter 5 suggest that part-of-speech distributions should be conditioned on the type of speech as well, as can the word (n-gram) distributions. Rather than conditioning all words on the type-of-speech, a subset of the words can be conditioned on the type of speech, e.g. all function words or all pronouns and determiners. This can be done by using the value of the part-of-speech variable on which the word is already conditioned.

If the type-of-speech is known, the value can be set in all slices, but if the type-of-speech is not known beforehand, it is treated as a hidden variable. In this case the type-of-speech nodes have to be connected in time. As the type-of-speech will not change much, these connections can be deterministic. If the type-of-speech can change the links can be probabilistic at particular points in time, such as the sentence boundaries.

8.4.4 User knowledge

User knowledge can be incorporated in the model in the same way as the type-of-speech. For example, part-of-speech and word distributions can be conditioned on age and gender. Word distributions can also be conditioned on dialects. An other

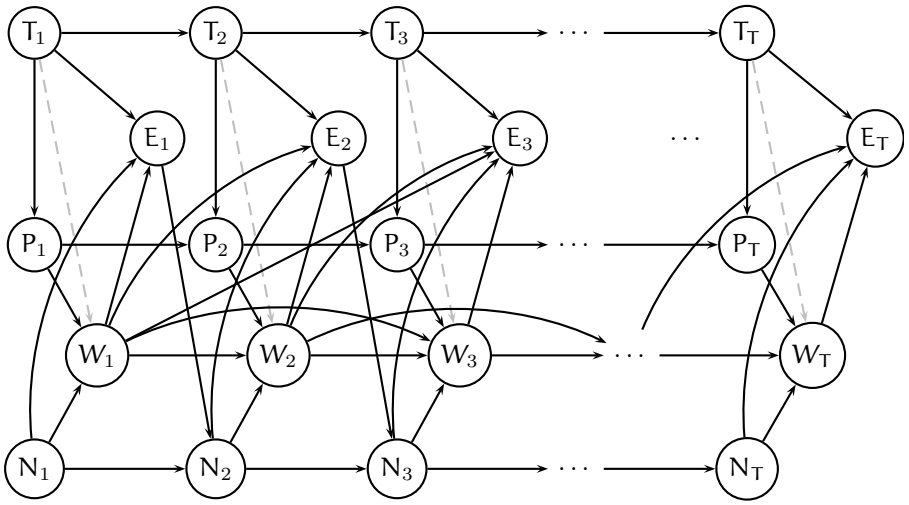


Figure 8.9 – *A trigram* DBN *with type-of-speech information* (T) *and explicit sentence length encoding.*

possibility is to add a concept level and condition the realisation of the concept as a word on dialect (among other things).

8.4.5 Context in acoustic models

User knowledge can also be incorporated in acoustic models. Figure 8.10 shows an example of such a model. It shows only the lowest two layers of one time slice of a model such as shown in figure 7.4. Next to the direct link from the state to the observation an indirect connection is added via a node M. This variable selects a Gaussian distribution that generates the observation from a pool of Gaussians associated with S. As M is always hidden the net effect is that the observation is generated by a mixture of Gaussians, where the probabilities $P(M = m)$ represent mixture weights. This is equivalent to the way Gaussian mixtures are typically modelled in speech recognition as HMM substates.

Now by making M dependent on the user knowledge U, a different set of mixtures or a different weighting scheme will be selected for different types of users. If user characteristics are known beforehand, the U nodes can be instantiated, thereby making the system more specific. If user knowledge is not available or uncertain, the U nodes can be treated as hidden variables. In this case they should be connected across time. During the recognition process part of the uncertainty may be resolved. As user characteristics usually do not change for an entire session a scheme somewhere between these extremes would be ideal. In this case the U variables would adapt themselves to the user at the start of the conversation and when

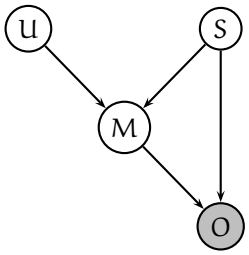


Figure 8.10 – *A mixture of Gaussians conditioned on user information, where* S *corresponds to the state of a phone level* hmm*, the observed variable* O *represents the observation distribution and* M *is a mixture weight.* U *represents the user information.*

they are reasonably constant they are treated as a given, probably until there are indications that something has significantly changed. This could be accomplished by using special network structure at the start of the conversation or by including meta-reasoning using for example information theoretic measures such as Kullback-Leibler divergence.

The scheme of figure 8.10 is well suited for incorporating knowledge such as gender and age of a speaker. Other context variables may exert more influence on other levels. For example, words are often pronounced differently in different dialects. As long as this is a matter of different pronunciation of single phonemes this variable may be set to influence the choice of mixtures or the choice of state variable. If complete words are pronounced differently in the sense that another phone sequence is used, the dialect variable should be linked to the phone level. A completely different context variable that may be relevant here is the type of conversation. Normally, people will speak slower and articulate more accurate when reading aloud. Spontaneous speech on the other hand will often be more animated and contain many phone deletions. Figure 8.11 shows some of these ideas. The model will become rather complex as more and more knowledge is added, resulting in a large state space and large conditional probability tables. This in turn may make the model inefficient and its hunger for training data unsatisfiable. Therefore, the influence of potential context variables should be assessed and only those that really make a difference should be included. In addition, the conditional probability distribution should be robust and be encoded efficiently. For example in Figure 8.11 the choice of mixtures is conditioned upon three variables. It is not likely that every combination of their states will make sense, rather there will be clusters of states, this can be implemented by introducing an intermediate variable or by encoding the cpd using a probabilistic decision tree. On the other hand backing off schemes can be introduced to ensure robust statistics.

Many have argued that state duration should be modelled explicitly in speech recognition (e.g. van Dalen *et al.* 2005; 2006). In dbns this can be done by introducing a duration variables between the phone level and the state level that are

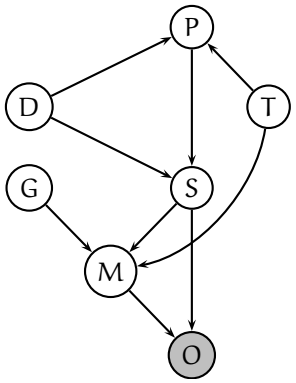


Figure 8.11 – *An acoustic model with contextual knowledge.* P *takes phones as its states,* S *is the substate within a phone and* O *is the observation distribution.* M *is a mixture weight.* G *gives the gender of the speaker,* D *the dialect and* T *the type of speech.*

connected in time just like the word counter in language modelling.

As a last remark in this section, it should be noted that the schemes shown in figure 8.10 and figure 8.11 would be very hard to realise in terms of HMMs, thus justifying the DBN view of acoustic modelling. The only way Markov models can deal with additional conditioning information is by expanding the state space, which basically means copying the model set for every combination of context variables, leading to a very large and inefficient model. Contextual knowledge would be smeared out across the entire model. Obtaining an HMM equivalent to the model of figure 8.11 would involve careful data selection and handcrafting many state-tyings.

8.5 Shallow parsing DBN language models

Including syntactic knowledge into a speech recogniser is useful by itself, but as argued in chapter 4, may also be beneficial in order to link high-level contextual knowledge to the word level. Unlike (statistical) grammars, dynamic Bayesian networks cannot deal with unlimited recursion. This essentially means that DBNs cannot model the syntax of natural languages. That is, unless one is prepared to drop the objective of finding a full and preferably correct parse of a sentence. In this section it will be shown that DBNs can be used for shallow parsing.

As spoken language is not grammatically correct putting more structure into a model can be seen as a trade off between robustness and correctness. Therefore, one might argue that for language modelling shallow parsing will do just fine.

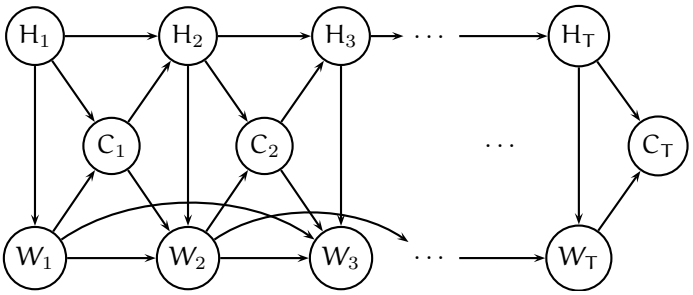


Figure 8.12 – *A chunking language model. Ws are word variables, the* H *nodes are the head words of chunks in the sentence and the* C *nodes indicate chunk boundaries.*

8.5.1 Chunking models

The shallowest way of parsing, apart from POS-tagging which would lead to a class based language model, is chunk-based parsing. Chunks (Abney 1991) include noun groups, verb groups, proper noun phrases and in some cases prepositional phrases. (1) shows an example.

(1) | He | ate | his white rice | with chopsticks |.

Chunking can efficiently be accomplished by tagging the spaces between words with tags like START, CONTINUE, END, BETWEEN and NULL. The latter two tags signify respectively a boundary between two chunks and between words that are not part of any chunk at all. This can be accomplished by the DBN of figure 8.12. In this model the W nodes represent words, the H nodes represent chunk headwords and the C nodes the chunking tags. A word is conditioned on its two predecessors as in a trigram, but also upon the headword of the chunk and the tag. The tag may be used to determine how strong the influence of the preceding words should be. By introducing additional links and a counter variable that keeps track of the length of the current chunk a word can also be conditioned upon other words in the chunk, while for example between chunk relations can be represented by simple bigrams. As an aside, in practice it would be preferable to obtain this effect by introducing additional deterministic memory nodes in every time slice. This would lead to more efficient inference, but clutters up the diagram.

8.5.2 Parsing with a fixed number of levels

To go beyond chunk parsing more and more levels may be introduced into the model, leading to yet another incarnation of the Hierarchical HMM discussed in the previous chapter. An example is shown in figure 8.13. Inclusion of contextual knowledge is straightforward, functioning basically in the same way as in the the other models discussed. But there are a few details that require some attention here. First of all,

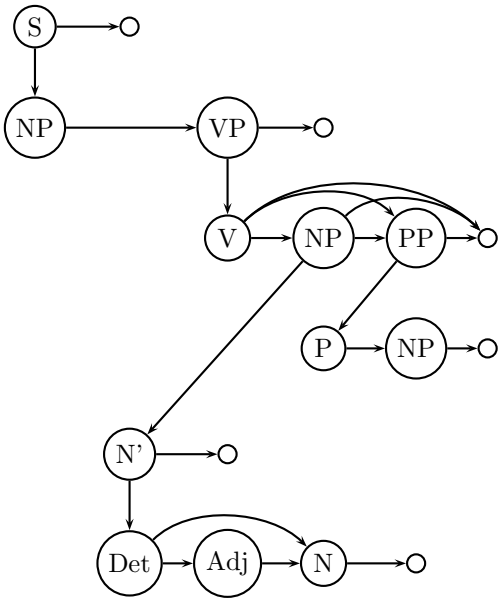


Figure 8.13 – *A hierarchical* HMM *for parsing, states correspond to nonterminals.*

an end-of-sequence node should be used to ensure that the probabilities of all strings should sum to one. The second problem is related to the explicit representation of levels. In a grammar-based model some constituent, say a noun phrase (NP), always behaves the same regardless of the level in the parse tree it is on, in fact it has no knowledge of such levels. In a DBN on the other hand noun phrases at different levels are completely different things. There may be reasons to accept this as is, arguing that phrases may behave differently at different levels, but if this behaviour is not acceptable, parameter tying should be introduced. This can be done as long as the parameter spaces for the variables are equal. In fact the shared parameters can be introduced as nodes in the network.

The basic hierarchical HMM model conditions a node upon its parent or predecessor. Actually, if no parameter tying is used a node is indirectly conditioned upon all nodes on the vertical path from the root done to its parent. However, there is no fundamental reason why conditioning should be limited this way, in theory a node can be conditioned on the whole partial analysis up to the current timeslice. In the real world one has to deal with intractability and data sparseness. Naturally, this severely limits the amount of information a node should be conditioned on. Fortunately, the situation is not different for probabilistic grammars and the literature on statistical parsing (Black *et al.* 1992; Charniak 1993; Magerman 1995; Stolcke 1995; Collins 1996; 1997; Charniak 1997; 1999; 2001; Chelba and Jelinek 1998; Roark and Johnson 1999; Collins 1999; Chelba 2000; Roark 2001; Uytsel *et al.* 2001) provides

many thoughts on which structural relations to include. As discussed in chapter 3, many have shown that parent, grandparent and sibling constituents provide a large part of the information required. Furthermore c-commanding structures can be helpful (Roark 2001), as can categorisation frames and explicit coordination flags (Collins 1999). Charniak (2001) has shown that it is beneficial to predict the POS tag of a headword before predicting the headword itself. A possible generalisation of this would be to predict a head concept first and use this to predict the headword.

Morphological knowledge can be included in the syntactic model by introducing several levels below the word level, which has the advantage of reducing the number of lexemes. At higher levels morphemes may be used instead of headwords, as it can be argued that the semantic relations between headwords do not really depend on the inflections. For example there is no semantic difference between 'eat a banana' and 'eat some banana'. As the inflections do matter for other words, e.g. 'a' versus 'some' in the previous example a word can be represented as a feature vector or frame.

8.5.3 Where do the probabilities come from?

Training of grammar models and DBN shallow grammars proceeds along the same lines. In both cases the general training algorithms are incarnations of the expectation-maximisation algorithm. In theory this algorithm can be used to learn structure, in this case grammar rules. As discussed in chapter 3 in practice grammar rules are usually learned from an annotated dataset, the same can be done for DBNs.

The rules for headword propagation are normally taken to be deterministic and can be found in Magerman (1995). Morphological information can automatically be extracted from a lexicon and be added to data transcriptions.

Once the parameters of a model are estimated from an annotated corpus the EM algorithm may be used to re-estimate the parameters on actual speech recogniser output to make it more robust to recogniser errors and non-speech sounds. An interesting thought here would be to switch to probabilistic headword propagation, this way the language model could learn to correct errors introduced by the acoustic model.

8.6 Combining the language model and the acoustic model

In HMM-based speech recognition the language model is typically run for every hypothesis produced by the acoustic model separately. The models discussed here define proper language models and can thus readily be used in this fashion.

However, if the acoustic model is also defined in terms of DBNs the models may directly be connected at the word level. The main difference here is that the probability of a word depends on all paths that contain the word, rather than on a single path. There is a catch, however, that shows up when doing recognition. Imagine for

example a shallow parsing language model that is connected to a standard acoustic model. As usual, recognition is performed by running the Viterbi algorithm in order to find the most likely word sequence given the observation sequence (i.e. we are interested in the values that the variable representing words takes). However, this would mean that only a single 'path' through the language model, that corresponds with this word sequence is used, in case of the shallow parsing DBN this single path corresponds to the most likely parse of this word sequence. In itself this may be a perfectly acceptable solution, as long as we realise that we are not getting the most likely word sequence (and the underlying segmentation) but most likely word sequence - parse pair, where the word sequences in those two cases are not necessarily the same.

But most of the time, for speech recognition this is not what we want as from a speech recognition point of view the syntactic structure is a hidden variable that is used in determining the overall probability of a word sequence. Its benefit is that it will favour grammatically plausible sentences of implausible sentences and not that it can find a particular parse of a sentence. So, what we are really after is the sum of all parses that yield a particular word sequence. The situation is even more pressing for the topic-based language model that will be discussed in chapter 10 for which the idea of summing over a hidden variable (in this case the topic of conversation) is at the very core of its functioning.

All in all, it can be concluded that to do proper inference (recognition) with a combined DBN model is should be possible to do max-marginalisation in one part of the model, while doing sum-marginalisation in other parts.

Another option to combine the models that avoids this complication is the use of n-best lists or lattices. In this case the acoustic model would be combined with a simple n-gram model and would produce a large number of alternative hypotheses. The more advanced DBN or grammar model would then be used to rescore these hypotheses. Within the rescoring approach the interface of the models does not have to be at the word level, instead it can also be situated at some intermediate level such as the phoneme level. An interesting variation on this rescoring scheme is to clip states in the Bayesian network that correspond to parts of the utterance or background information that is relatively certain and subsequently use a forward-backward pass over the model to fill in missing words.

Chapter 9

A Computational Framework

In which the implementation of a framework for language processing with Bayesian networks is described. The requirements that speech and language processing applications impose on Bayesian networks are given. Algorithms and data structures that fulfil these requirements are discussed. In particular a fast algorithm for inference with probability tables, lazy evaluation of probability tables, algorithms for calculations with tree-shaped distributions and a generalisation of dynamic Bayesian networks that we developed are introduced.

The previous chapter shows how models for speech and language processing can be formulated in terms of dynamic Bayesian networks. To make these models work an inference engine is needed. Many different algorithms for inference in DBNs exist as well as several toolkits that implement these algorithms. From the models in the previous chapters we can deduce the properties that an inference engine for speech and language modelling should have. Below these requirements will be discussed.

Allow k-th order Markov assumptions Almost all models defined in the previous chapter use k-th order Markov assumptions. Many algorithms for dynamic Bayesian networks are formulated in terms of first-order Markov models and implemented as such in most toolkits. From a theoretical point of view, this poses no problem, as any k-th order model can always be reformulated as a first order model, but on the practical side, we do not want to put that burden on a model designer.

Deal with large state spaces While the number of states in a Bayesian network is typically small e.g. Boolean, the vocabulary of a language model contains tens of thousands of words. As a consequence probability tables may get very large, causing memory overflow. In fact, the models may become intractable because the time complexity of inference in Bayesian networks depends on the number of states in the network as well. Fortunately, as many combinations of values do not occur

in practice most of these tables are extremely sparse. By taking this into account average space and time complexity can be lowered to make inference in models of speech and language feasible.

Deal with small probabilities The number of time slices in an utterance may get large when processing speech, while on the other hand the probabilities of feature vectors and words are small. For a practical system this makes the danger of numeric underflow very real.

Provide discrete and continuous distributions Speech recognition requires continuous observation distributions. In particular Gaussian mixtures. Other language processing tasks such as language modelling require discrete distributions such as multinomial distributions.

Allow clipping of values In some situations the value of a hidden node in the network might be or become known. For example an application might explicitly ask a user to enter his gender and age or a system can deduce such information in the first seconds of a conversation. For the remainder of the conversation the value of gender and age nodes can than be clipped to a specific value.

Provide mechanisms for parameter tying Substructures of a model may reappear at multiple points in a model. For example, in speech recognition several acoustic models may share the same mixtures, states or transition matrix. In language modelling, history dependent weights can be tied to form bins and as discussed in the previous chapter and in parsing the parameters of constituents are tied across levels in the parse tree.

Deal with switching variables Many models described in the preceding chapters use switching parents, i.e. variables of which the values are used to decide on which of its (other) parents a variable will be conditioned. Interpolation weights are an example of switching parents. Technically, there is nothing special about switching parents. The dependency of a variable on its parents can be implemented using a simple joint probability distribution. However, switching parents are often used to combine distributions for subspaces of the state space of a distribution. For example, in language modelling it is common to combine unigram and bigram models. To represent this in a joint distribution the unigram has to be repeated for every value of the preceding word. Saving the two components separately can result in significant space savings and more efficient inference. In fact, to properly deal with parameter tying component distributions have to be stored as separate entities.

Provide a rich network structure Dynamic Bayesian networks are usually defined as an a priori Bayesian network that is used in the fist time slice and a temporal

Bayesian network that includes links to previous time slices that is repeated as often as necessary, i.e. except for the first slice, all slices have the same structure.

This is a rather simplified view of temporal processes. There are many cases it which the network structure should change over time. For example, a speech recogniser that includes user information may need only a few seconds to identify user features, after which these features can be kept constant.

Provide pruning mechanisms Speech recognisers typically prune large parts of the search space to achieve real-time performance. The same is done for parsing. For Bayesian networks on the other hand, pruning is usually not used. Speed ups are achieved using approximate algorithms such as the Boyen-Koller algorithm that factors the state space. The advantage of pruning over such methods is that pruning focuses on the high probability paths through a network given the observations, whereas Boyen-Koller is applied without any knowledge of the inputs. Stochastic approximate inference techniques such as particle filtering do focus on high probability paths. However, to achieve real-time performance the number of paths through the model that will be sampled will be sparse. Given the stochastic nature of the algorithm the chance that the correct path is missed is higher than for pruning.

Provide filtering, smoothing and Viterbi inference To build language models that can be integrated with existing speech recognisers conditional probabilities such as $\mathsf{P}(w_i|w_{i-1}\ldots)$ need to be calculated, i.e. the inference engine also has to be able to perform prediction. In general the tools should be flexible enough to enable us to inspect any set of variables in any time slice when doing inference.

As made clear in the discussion of combining acoustic models and language models in the previous chapter, it should be possible to marginalise out some of the variables, while maximising others when doing inference. For example, when recognising speech we are interested in the values of the word variables on the most probable path, but not in the values of other variables. It should thus be possible to specify the variables of which the values should be stored during inference.

Provide parameter learning algorithms An implementation of EM training should be available. To allow for efficient incremental model improvement, e.g. update parameter weights, it should be possible to specify which variables should be updated in a training run and which should not.

Extensible It is very likely that future uses of DBNs for speech and language processing will add new requirements to the list. It should be possible to incorporate these in the framework. In other words the source code of a framework should be available.

9.1 Related work

Several toolkits that implement dynamic Bayesian networks or generalisations thereof already exist. See (Korb and Nicholson 2004; Hulst 2006) for overviews of these and other toolkits.

dHugin (Kjaerulff 1995) adds temporal reasoning to the popular commercial Hugin (Andersen *et al.* 1989) shell. It assumes that DBNs obey the Markov property, i.e. a variable only depends on variables in the current or in the previous time slice. The stucture of time slices can vary. An exact junction-tree based inference routine that unrolls the network for k slices at time as well as forward sampling are provided.

The Bayes net toolbox (BNT) (Murphy 2001) is a free, open-source library intended for research purposes. It is implemented in Matlab because of the ease with which it can handle Gaussian random variables. On the down side, this choice for a high level language makes the toolbox slow and limits the size of the networks that can be processed. A DBN is represented with a prior and a transitional network, so that only first-order Markov processes can be modelled. Several inference algorithms for static Bayesian networks are provided, each of which makes different trade offs between accuracy, generality, simplicity and speed. The conditional probabilities of the defined variables can be continuous or discrete. Parameter learning is supported as well. Currently, the toolbox lacks online inference and learning, and does not include prediction.

The probabilistic network library (PNL) (Intel Corporation 2004) is a C++ version of BNT implemented by Intel's research lab in Saint Petersburg. It does not yet support the whole functionality of the Bayes net toolbox.

The graphical models toolkit (GMTK) (Bilmes and Zweig 2002; Bilmes 2002*b*) is a freely-available toolkit written in C++ that is specifically designed for DBN-based speech recognition. Models have to be defined in GMTKL a flexible but complex specification language. In this language a DBN definition consists of several frames each of which can define different variables and relations between variables. The first N frames form the prologue and are used at the beginning of the network. The last M frames, called the epilogue, are used for the final M frames in the network. The frames between N and M form the repeating substructure. The toolkit has many desirable features, such as sparse representations of CPDs, tree-shaped CPDs, continuous observation distributions, switching parents, beam search, parameter tying and generalised EM training. It supports smoothing and Viterbi inference using the online Frontier algorithm. But because of the epilogue in the network definition the length of an input sequence has to be known in advance, making real-time, on-line processing impossible. Sampling of networks is possible, but the toolkit does not directly implement algorithms for approximate inference.

The Structural Modeling, Inference and Learning Engine (SMILE) (Druzdzel 1999; 2005) is a platform independent library of C++ classes that implements Bayesian networks and influence diagrams. Recently, support for temporal reasoning has been added (Hulst 2006).

None of the existing frameworks covers all of the requirements formulated above.

Therefore, a framework specifically designed for use in speech and language processing was implemented as part of this thesis work.

9.2 Design of the framework

The framework we developed consists of a set of general purpose tools for inference with arbitrary DBNs. It implements a generalisation of DBNs that allows the structure of the model to change over time. A model designer does not have to worry about algorithms but can specify models in XML format.

The tools meet all of the requirements discussed above. They can deal with very small probabilities and a large number of states. Data representation and inference algorithms have been optimised for sparse distributions. Tree-shaped distributions can exploit reoccurring substructure and implement interpolation and smoothing schemes. Other features include: parameter sharing, lazy evaluation, pruning and multiple inference engines. EM learning in log-space allows for large input sequences. An optional damping factor improves learning convergence.

Tools for corpus processing that format data (e.g. remove punctuation, filter out-of-vocabulary words, including all words that occur only a limited number of times in the training data, apply stop lists) and can construct (smoothed) distributions directly from the data are also included.

The functionality behind the tools is implemented in a common library that makes it easy to change or reuse parts of the software and to experiment with different algorithms and data structures. The library has a layered structure. At the lowest level it provides general purpose classes, among which are input and output routines that translate words to an internal numerical representation and a class that can represent very small probabilities. The core of the systems is formed by classes that implement efficient mathematical operations on multidimensional probability tables. The top layer is responsible for construction of and inference in dynamic Bayesian networks. The library and tools have been designed with language processing in mind, but can be used for many other applications such as multi-modal fusion.

The remainder of this section discusses these layers in more detail, focusing on the algorithms and data structures that were developed as part of this thesis work.

9.2.1 Dealing with small probabilities

At the lowest layer two data structures implement the mathematical concepts of probabilities and likelihoods and their operations. In this context a likelihood is defined as a positive real value that can result from an operation on probabilities. Compared to a simple floating point representation these representations have the advantage that they are more robust in the face of underflow, which is a huge problem for models, such as HMMs and DBNs, that multiply long sequences of probabilities together. They lift the burden of normalising or worrying about underflow from

higher layers.

The first method accomplishes this in the classical way of representing a probability as its logarithm (see e.g. Van Alphen 1992). This has the additional advantage that multiplication and division become faster since they reduce to addition and subtraction. However, addition and subtraction themselves become more complex and rather slow.

The second approach simply extends its range compared to double precision floating points. It represents its value as a mantissa in the interval $[0.5, 1)$ and an integer exponent. The value follows from: $\text{mantissa} * 2^{\text{exponent}}$. The second approach takes more space than the first and is slower for multiplication and division, but when it comes to addition and subtraction it is considerably faster.

Both methods suffer from the typical floating point problems, i.e. a limited range and a limited precision. In these respects the logarithm does worse than double precision floating points, while the extended range does better.

9.2.2 Likelihood tables

The second layer of the library introduces multidimensional tables of likelihoods. These tables are the core of the library. They are used to implement multinomial probability tables, to store intermediate calculations and as accumulators in learning. To make low level parameter sharing possible, tables consist of two parts: the table itself with an associated list of cardinalities and a domain of variables. As long as the cardinalities match and values are not altered the same table can be used with multiple domains. The dimension of a table is determined by the variables in its domain.

Tables are implemented in several ways. The straightforward implementation is an ordered list with one entry for every tuple of values of variables in its domain. For small tables this works well and can be reasonably efficient as one does not explicitly have to save the indices of the table and any value can be found in constant time.

However, for many tables this representation will take too much space. As mentioned before, probability tables in speech recognition often are very sparse. Saving only non-zero entries together with their indices can realise enormous space savings at the cost of some time for searching. To minimise the latter indices are stored in order. This allows our inference algorithm that processes tables as a whole to accesses the indices (semi)sequentially. It is possible to specify default values different from zero for sparse tables. Only probabilities that are different from the default value will be saved.

Deterministic variables have only one possible value for every configuration of their parent variables (that thus must have probability 1). As a consequence, there is no need to save the probabilities, only the non-zero values have to be saved. There are several ways to do so. The current implementation is similar to that of sparse tables, with the difference that it only saves values. The resulting structure is strictly speaking more general than described above, as it is possible to have more than one value for the same 'parent configuration'. This allows to represent so-called

0–1 potentials. 0–1 potentials have no distribution counterpart, but can be used to represent evidence that only certain states are possible.

Calculations with likelihood tables proceed as described by (Jensen 2001). Multiplication of two tables is defined as pairwise multiplication of elements that have equal values for all common variables. Equation (9.1) gives an example:

A	0.2
	0
	0.8

$\times$

	A		
B	0.3	0.5	0.7
	0.3	0.5	0.1
	0.4	0	0.2

$=$

	A		
B	0.06	0	0.56
	0.06	0	0.08
	0.08	0	0.16

. (9.1)

Marginalisation projects a table to a lower dimension, by summing over all values of variables that are marginalised out for every tuple of values of the remaining variables. Equation (9.2) shows how a three-dimensional table is projected to a one-dimensional table:

$\sum_{BC}$

		A		
B	C	0.012	0	0.056
		0.048	0	0.504
	C	0.036	0	0
		0.024	0	0.08
	C	0.04	0	0.16
		0.04	0	0

$=$

A	0.62
	0.14
	0.24

. (9.2)

An inference algorithm has been developed to multiply any number of tables and project the result to a subdomain of the joint domain in one operation. The algorithm takes into account the domains of the tables to deal with overlapping variables. It also takes into account observed values tot avoid unnecessary calculations and to keep the size of the resulting table as small as possible. Since probability table manipulation makes up the bulk of the work in probabilistic inference a lot of attention has been dedicated to optimising this part of the library. For example special memory pools are used for fast allocation and deallocation of probability tables.

The algorithm systematically iterates through all values of the joint domain of the input tables. Variables are ordered in all tables. All values of a lower ranking variable are processed before the value of a higher ranking variable is increased. For example, in (9.3) the order might be A, B, C, D. On multiplication of these tables the upper left corners will be multiplied first, then D will increase its value. As D has only two values, it will be reset to its start position in the third step, while the value of C is increased by one.

	A	
C	0.2	0.4
	0.3	0.1

,

	B	
D	0.1	0.3
	0.2	0.4

(9.3)

As a consequence, all input tables are processed in sequential blocks (non-overlapping input tables are processed sequentially). This approach is fast in the face of buffers,

caches and paging. It guarantees that intermediate results never take more space than the end result. In particular if no projection to the output domain is needed, the output table will be built sequentially.

To limit the amount of searching in the tables, a reference to its direct parent within the domain of a table is kept for every variable, as this is the starting point for this variable within the set of values of lower ranking variables. For example, if in (9.3) the value of C is increased, D should be reset. It should start at the first position with the current value of B. Observed variables are skipped completely.

The algorithm only processes values that are non-zero in the joint table. Rather than increasing the value of a variable by one in every step every input table is consulted to find the next common non-zero probability within the current set of parent values. For summation the algorithm moves to the next value for which one of the tables has a non-zero probability.

9.2.3 Lazy evaluation

Inference in Bayesian networks comes down to a sequence of multiplications and marginalisations of likelihood tables. Most inference techniques determine the order of operations based on the structure of the network, without taking the shape of the probability distributions into account. In addition, this is typically done offline, before any evidence, that may introduce additional independence relations, has been observed. As a consequence, the order is not always efficient. In Madsen and Jensen (1999) the Lazy Hugin algorithm was introduced. This variation on the Hugin algorithm for inference in junction trees uses lazy evaluation to make better use of evidence and probability table structure.

We decided to use a similar lazy evaluation approach, but at the level of likelihood tables. This has the advantage that it can be used with any inference algorithm. In this approach multiplication of tables is deferred until it really is necessary, i.e. if a variable in the domain of the table is marginalised out. Rather than a table a set of tables is used. Multiplication is a very fast operation as it simply adds the table to the set. Upon marginalisation or summation all tables in the set whose domains contain variables that are marginalised out are selected. Multiplication of these tables and marginalisation are done in a single operation to avoid the construction of (large) intermediate tables. The result is added to the set. Note that if the variables of a particular subtable are never looked at, that table will never really be processed. In Bayesian networks this may occur if certain subparts of the network are not needed in an inference process (barren nodes).

9.2.4 Tree-shaped likelihood tables

If a table contains some reoccurring substructure or if several tables share the same substructure an even more efficient representation of tables is possible. The n-ary decision trees discussed in 7.2 can be used for this purpose (Boutilier *et al.* 1996; Bilmes and Zweig 2002; Bilmes 2002*a*). The questions in the decision tree are random

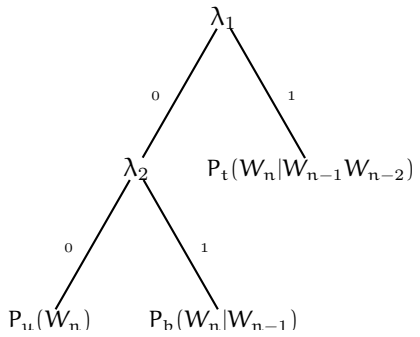


Figure 9.1 – *A decision tree representation of an interpolated trigram.* W_i *corresponds to the* i*-th word of a sentence.* λ_1 *and* λ_2 *are interpolation weights.*

variables and the answers in the tree are values. As this is the means by which switching variables in the Bayesian networks are realised the questions are called switches in this context. Every switch has a list of values that are connected to either another switch or to a set of tables as described above. A table or subtree of tables can be shared by several trees, allowing for very flexible parameter sharing. Figure 9.1 shows a tree representation of an interpolated trigram $P(W_i|W_{i-1}W_{i-2}\lambda_1\lambda_2)$. If $\lambda_1 = 0$ the word variable W_i is independent of W_{i-2}, if $\lambda_2 = 0$ as well W_i is also independent of the previous word W_{i-1}.

Unlike (Boutilier *et al.* 1996; Pfeffer 2001) who use the tree-structures to alter the network structure we use the tree-shaped CPDs directly. Although altering the network structure may lead to smaller cliques and hence to faster processing, our approach, in combination with lazy evaluation, has the advantage that whole sequences of operations may be skipped as on observation of a value of a switching variable all subtrees associated with other values can be removed at once. We developed algorithms that use decision tree manipulations for multiplication and marginalisation of these switching tables. Representing tables as trees does not only save space, but also leads to very efficient processing as often entire subtrees can be pruned without evaluation. For example on observing a particular value of a switching variable all subtrees associated with other values can be removed at once.

Tree multiplication Multiplication and division of trees comes down to merging trees. Multiplication creates a new tree by attaching the right-hand-side tree below every leaf of the left-hand-side tree and subsequently removing unreachable paths. Subtrees are non-reachable when their root question already appears higher up in the tree with a different answer value.

The new tree has the tables of the right-hand-side tree as its leafs. To obtain the product the corresponding leafs of the left-hand-side tree are multiplied onto the leafs of the new tree (The construction of the new tree guarantees that there is

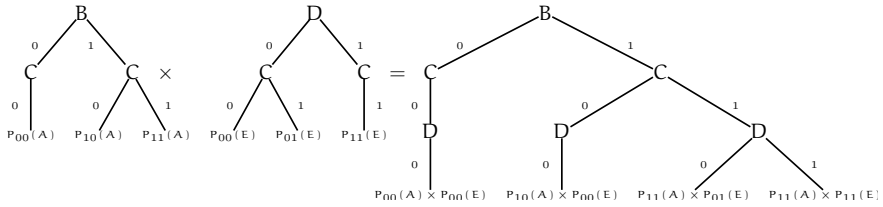


Figure 9.2 – *Multiplication of trees representing probability distributions* P(ABC) *and* P(CDE).

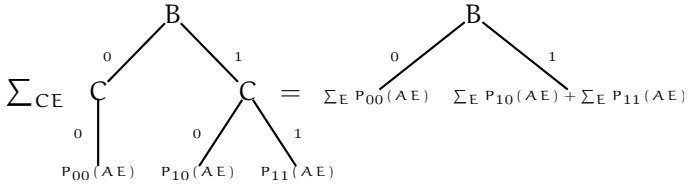


Figure 9.3 – *Marginalising variables* C *and* E *out of a tree representing probability distribution* P(ABCE). *Marginalisation of leafs proceeds as usual, marginalisation of intermediate nodes requires summation of subtrees.*

exactly one path in the old tree for every path in the new tree). Figure 9.2 illustrates tree multiplication. A missing link, such as the link corresponding to $B = 0, C = 1$ in the left-most tree in the figure, means that all corresponding probabilities are zero.

Tree marginalisation Marginalisation processes the tree in (depth first) post-order. First, leaf tables are marginalised as usual. If a intermediate node in the tree has to be marginalized out, it is removed by attaching the sum of its subtrees directly to its parent. Since leafs are marginalised before their parents these node variables, that act as observed variables in the leafs, will be removed from the leaf tables. All leaf tables will thus have the same domain the moment the node is marginalised. An example of tree marginalisation is given in figure 9.3.

For projection the same algorithms as for marginalisation are used, with the only difference that nodes are marginalised when they are not in the specified domain.

Tree summation The order of the switches of the subtrees may be different. In fact subtrees do not necessarily contain the same switches. Summing over trees therefore not only involves summing the leaf distributions, but also properly combining the trees. This is done by traversing the leafs of one of the trees and keeping track of the path that leads to a leaf. Next this path is used to find corresponding paths in the other tree using a recursive procedure. If the root of the tree occurs in the path and the value of this variable equals one of the branches of this switch

we can mark this variable in the path. If the branch leads to another switch we can recursively call the algorithm with this switch. If the branch leads to a leaf distribution we have to insert the unmarked part of the path between the branch and the leaf and sum the leafs. If the root variable is not in the path at all we call the procedure recursively for every branch of the switch.

9.2.5 Potentials and distributions

Tree-shaped tables are used to represent both potentials and multinomial distributions. Potentials are tables of likelihoods that represent intermediate results in an inference procedure. Multinomial distributions on the other hand can only contain probabilities. In addition, distributions can have accumulators that are used when learning the parameters of a network. Accumulators can be shared across distributions. This is for example used when learning the parameters of a DBN. The accumulators are shared across time slices. But parameters can also be tied for learning across other variables.

Only observed variables can have a continuous distribution, such as Gaussian mixtures for speech recognition. Upon observation a conditional Gaussian provides one probability for every set of parent values. Hence the result is a potential over the parent variables that can be uses as any other potential.

9.2.6 Network structure

The upper layer of the library implements a generalisation of dynamic Bayesian networks. To increase the flexibility of the system this layer consists of three components: an abstract definition of DBNs that is not bound to any inference algorithm, an interface to the outside world that implements high level inference technique independent functionality such as learning and an inference engine. Different inference engines can be plugged into the system without changing any of the other structures.

Whereas standard DBNs have a repeating substructure, these networks can have any number of subnetworks, called chapters, that each have a repeating substructure. The last chapter is allowed to repeat indefinitely, all other chapters must have a predefined length. The substructure that repeats within a chapter may span several time slices. In addition, a chapter can define static variables, that do not have temporal dynamics. It is also possible to define static variables at the network level. Those correspond to the contemporal nodes of Hulst (2006). Following Hulst, we also provide a separate chapter with static variables that is attached at the end of the network.

To every variable in the network a distribution is attached. Variables with the same name in different time slices and chapters share the same distribution. The parents of a variable can be in any preceding slice or chapter, i.e. k-th order Markov relations are allowed.

The user of the system provides the chapters of the network and a template of the repeating slices in the network as well as the (tree-shaped) distributions in XML

format. The DBN interface is responsible for expanding (unrolling) this definition into a network and checking it for consistency. Figure 9.2.6 shows an example of a generalised DBN definition for a language model in which the words W are conditioned on their part of speech P, such as verb, noun or determiner, that are in turn conditioned on the type of speech T, i.e. whether it is conversational speech or more formal speech. The rectangles in the figure denote chapters. The top-most chapter contains global static nodes. It is assumed that the type of speech is constant for the whole word sequence. Therefore the type of speech random variable is placed in this chapter. The other chapters are combined left-to-right in the expanded network. The first chapter defines slices of the first two time steps, represented by rectangles with dotted lines. The number in the upper right corner gives the length of this chapter. Because it has a length of two, the slices in it will not be repeated. After the first two time steps, the expanded network will continue with the second chapter. This chapter contains a single slice that will be repeated as long as there are input values. The dotted circles are place holders for parents of a node that are defined in other slices in the expanded network. For example all dotted T nodes refer to the global T variable and the P_{-1} node in the second chapter refers to the part of speech node in the previous slice. Note that for the third time step this is a variable in the first chapter and after that it is a node in a previous instantiation of the second chapter. When the whole word sequence is processed the final chapter is attached to the network, this contains an end of sequence node, as discussed in the previous chapter. Figure 9.2.6 illustrates the advantage of subdividing a network in chapters. There is no need to repeat the type of speech and end of sequence nodes in every time step as is the case in standard DBNs. This is not just a convenience for the designer of the network, but also allows for faster inference.

9.2.7 Inference engine

Filtering, smoothing, prediction and Viterbi inference is possible. For the latter it can be specified which variables should be marginalised (summed) out and which should be maximised. In addition it is possible to run any of these algorithms in interactive mode to inspect any variables in any time slice during inference.

We currently implemented the Frontier algorithm and the interface algorithm for exact online inference and the Boyen-Koller (Boyen and Koller 1998) and Factored Frontier (Murphy and Weiss 2001) algorithms for approximate inference. The generalised structure using chapters and k-order Markov relations complicates things a bit as for different time steps a variable (in a particular chapter) may have a different set of children. In addition, as the same variable may appear in different chapters, the last child of a variable may change. This is solved by splitting up the network in additional chapters in such a way that the variables in a particular chapter have the same children in all future slices that can be reached from slices in the chapter.

In a preprocessing step the Frontier engine transforms the DBN definition in a set of operations for every slice of every chunk. It basically establishes the elimination order based on a number of rules:

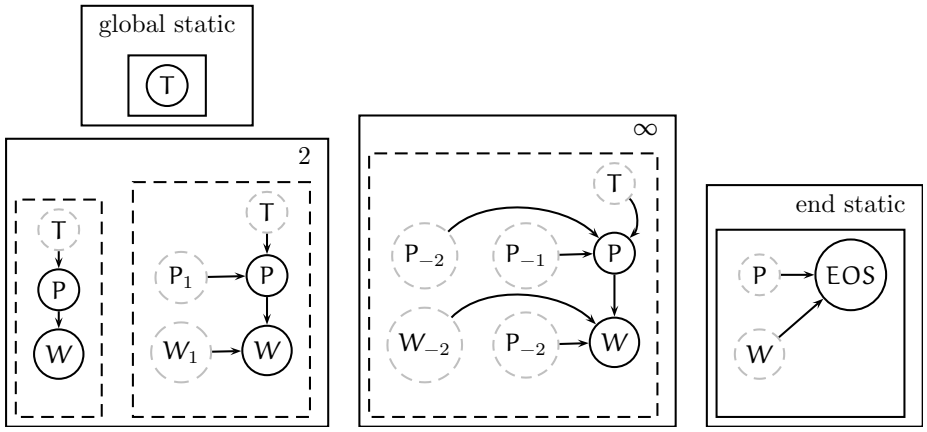


Figure 9.4 – *A generalised* DBN *definition of a word trigram in which the words* W *are conditioned on their part of speech* POS *and on the type of speech* TOS*. The outer rectangles represent chapters, dashed rectangles represent slices in non-static chapter that have repeating substructure. Dotted nodes are references to parent nodes in the expanded network.*

- A node can be added to the frontier when all of its parents are in the frontier.
- A node can be removed when all its children are in the frontier.
- Sum marginalisations are added before max marginalisations.

Heuristics or a search algorithm can be used to order unconstrained nodes.

When an input sequence is processed these operations are added to a processing queue. Different operations such as filtering, prediction and smoothing may alter the order of the operations in the queue. Furthermore, if one wants to consult the value of particular variables the order of the operations in the queue can be altered on the fly.

We extended the algorithm with beam pruning. At every slice boundary the likelihoods in the frontier that are smaller than the largest likelihood by some predefined percentage are set to zero.

9.2.8 Learning

For parameter learning the expectation maximisation algorithm is used. It has been implemented using a forward and a backward pass, enabling the use of any inference engine that implements smoothing. At every time slice intermediate results have to be saved in the forward pass, so the space requirements of the algorithm may quickly get out of hand. Therefore, the island algorithm (Zweig and Padmanabhan 2000), that is discussed in section 7.7.1 has been implemented.

The learning tool can work in distributed mode. Each processor learns a part of the data. A central thread of the program combines the accumulators of the other programs.

9.2.9 Data preparation and processing

An additional library module provides a basic framework for corpus processing. For a given corpus one has to provide a simple EBNF-style grammar that can parse the data format of the corpus, the framework then provides all the functionality needed to translate the data to the format used by the toolkit, to remove or replace tokens such as stop words or punctuation, to construct a vocabulary file and a mapping of words to unique numerical identifiers automatically as well as to calculate basic corpus statistics and extract (smoothed) distributions over the data. In addition, one can provide special purpose classes to calculate additional statistics or to write to other output formats. A tool for automatic subset selection from a corpus is also provided.

9.3 A few words on the implementation

The toolkit is platform independent. It has been implemented in C++ following the generic programming paradigm in which algorithms and data structures are designed to be as general as possible, thereby allowing for flexibility and code reuse.

Generic programming uses compile time decisions to generate fast code. For example decision trees are used to implement probability distributions. The underlying decision tree class however contains no knowledge of random variables or probability distributions. It only uses abstract concepts: question (in the nodes), answer (on the branches) and leafs. Within the generic programming paradigm these abstract place holders will be bound to particular types, i.e. random variables, states and probability tables in this particular case, during compile time. This places no additional burden on the run time of the program, as a typical object oriented solution using polymorphic classes would, while providing a software developer with much flexibility. For standard data types such as vectors, lists, sets, date and time manipulations, interacting with the file system, graphs, mathematical functions, random number generators, handling of regular expressions and XML parsing the library relies on the C++ standard template library (STL) and on the peer-reviewed BOOST libraries (`www.boost.org`).

Things that try to look like things often do look more like things than things. Well known fact. But I don't hold with encouraging in it!

Granny Weatherwax in *Wyrd Sisters*, Terry Pratchett

Chapter 10

A Topic-based Language Model

In which we define a novel, adaptive language model that combines topic information and structure information together with a procedure for unsupervised learning of the model. We relate the model to sentence level mixture models, thereby giving a Bayesian explanation of these models. We also relate the model to the concept of spreading activation as used in models of human speech processing. Experimental results are reported.

A discourse is not a random collection of phrases. Following the argument of Jurafsky and Martin (2000): if we would pick at random a number of sentences from this thesis and put them together we would certainly not get anything that looks like a proper paragraph. The difference is, that any discourse will display some coherence. It will contain semantically related words, and particular phrases, typically content words will reappear in the text. It is this property that information retrieval systems use to find documents relating to a query. In other words, a discourse has a topic.

As discussed in chapter 1 there is evidence that human speech recognition greatly benefits from contextual coherence (Gill-Günzburger 1979). Eye-tracking experiments show that words that fit the context are read faster (Ledoux *et al.* 2006). At the same time a peak of electrical activity can be observed in the brain about 400 ms after the onset of a word that is semantically incoherent (Osterhout *et al.* 2002; Halgren *et al.* 2002).

On the other hand, the common practice in language modelling to collect statistics on average word use in a very large corpus completely ignores contextual coherence and topics. In fact, to make a language model robust data from as many different contexts as possible is combined. Although this ensures that reliable parameter estimates can be found, the assumption that the relative frequency of a word combination is the same for all conversations is clearly incorrect. A word may be far more likely in a particular conversation than on average (in a corpus) and even more likely than in conversations about other topics.

In chapters 4 and 5 we have identified other determinants of language use, in particular user characteristics and the type of speech. Many existing systems using topic information (Iyer and Ostendorf 1996; Seymore and Rosenfeld 1997; Seymore *et al.* 1998; Chen *et al.* 1998; Florian and Yarowsky 1999; Mahajan *et al.* 1999; Khudanpur and Wu 1999; Gildea and Hofmann 1999) do not differentiate between topic and type of speech. Rather a subset of a corpus is selected that is assigned a topic. Such a subset may contain utterances that are all about a particular subject, say speech recognition, but it might also contain documents that are all in a particular genre or type of speech, for example lectures. We claim that type of speech and topic are complementary. People can talk about the same topic using different types of speech. For example, there is a clear difference between a news anchorman presenting the latest political scandal and people talking about that same scandal in a pub. The difference is in the use of formal and informal language, but the words or at least the concepts that belong to the topic will remain the same.

Although the notion of a topic is very intuitive, giving a definition of topic is not easy. Unless one is willing to specify many details, it is hard to name the exact topic of a conversation. We might end up with a unique topic for every conversation. Rather, a conversation is a mix of several topics, where some are more prominent than others. This mix can change over time; topics can become less important or disappear completely and new topics can be introduced.

Obviously, words can belong to multiple topics. But when is a word excluded from a topic? For example, when talking about cars the word 'chopstick' is not very likely. Nevertheless, we might hear a sentence like: 'Japanese cars are just like chopsticks.' Now the word chopstick has something to do with Japan, as do cars, but in many conversations about Japan neither will occur. We might argue that the word 'car' is central to the topic 'cars', but less important in the topic 'Japan'.

10.1 The basic model

The last remarks already hinted at a view of topics that can deal with the issues mentioned above. We define a topic as a probability distribution over the vocabulary. For a given topic some words are likely while others are not. The idea that a given discourse contains a mixture of topics can then be implemented as a mixture of topic distributions. In the language of Bayesian networks this can be put as shown in figure 10.1. As before W represents a word. It is conditioned on the topic T.

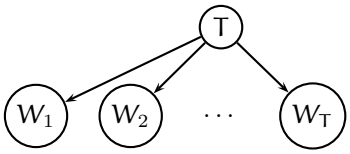


Figure 10.1 – *Conceptual view of a topic model. The word variables are influenced by the topic* T.

When the topic of a discourse is known, assigning a probability to this discourse is straightforward. However, more often than not, the topic of a conversation is not known beforehand. Imagine observing the words in a conversation one by one. Initially, all topics are possible, so it is reasonable to assume that they are equally likely or distributed according to some a priori distribution (people tend to talk about the weather more often than they talk about speech recognition). Now when a word is observed, it can be used to update the belief of the topic node. Topics in which the word has a high probability will become more likely. For the next word this updated belief of the topic node is used. Therefore words that belong to the same topics as the first word have become more probable. The second word will in turn influence the topic distribution as well. In the course of the conversation, the belief of topics will go up and down, but given that a text is reasonably coherent, the topics that are present in the text will eventually be identified and can be used to guide the prediction of future words.

What is going on here is belief revision or uncertainty reduction. Every new word that is observed is propagated back through the network as evidence, supporting those topics that predicted it with a high probability and moderating the influence of other topics. In other words, the distribution over the topics shifts towards the observed words. If a text is reasonably coherent this will lead to a better prediction of future words. If the observed words themselves are uncertain, evidence from future words can be used to disambiguate them by also running a backwards pass over the model.

To summarise, the model of figure 10.1 can naturally deal with mixtures of topics, it can identify the topics of a conversation in the course of that conversation and can adapt when the topic of a conversation changes. One can think of the topic nodes as a means to capture long-distance dependencies: the exact words are not remembered, but the topic mixture provides a summary of the history.

Figure 10.1 uses a single global topic node that is seen as the common cause of all the word variables. The changes in the topic belief over time can also be modelled explicitly as shown in figure 10.2. To be completely compatible with the previous model the links between the topics nodes should be deterministic. There are no transitions from a topic to any other topic. Using this representation we can get a better insight in the workings of the model. Notice that the model is equivalent to an HMM where the topics are the states and the words the observations. As discussed in

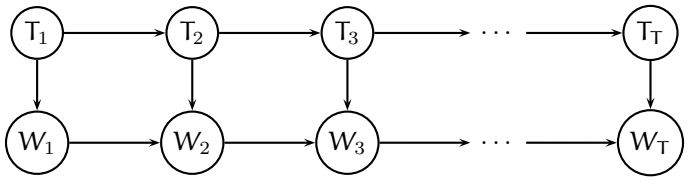


Figure 10.2 – *The basic topic model.* W_t *is the word at time* t, T_t *the topic at time* t.

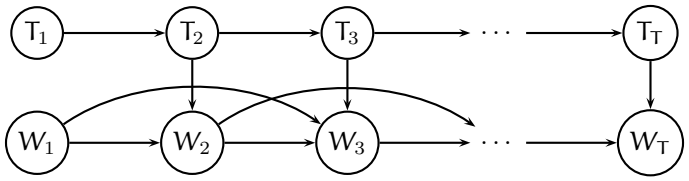


Figure 10.3 – *Topic model with* n*-gram constraints.*

chapter 2 the forward algorithm that gives the probability of an observation sequence in an HMM can be visualised by a trellis. As topics have only self-transitions, the trellis consists of a set of unconnected chains. The total probability of the observation sequence is the sum of the probabilities of the individual sequences. This model will thus assign a higher probability to a coherent text than to a text that contains words that have high probabilities in different topics. Note that in general, every word must have a non-zero probability for every topic. Otherwise, a word that has zero probability for a particular topic would set the belief of that particular topic to zero even if all other words have a high probability in this topic.

10.2 The relation with mixture models

Since the topic model defines a distribution over word sequences, it can directly be used as a language model. But taking the trigram as a benchmark it comes of rather poorly, as it actually defines unigram probabilities. It is a bag-of-words model. Obviously, it can be combined with standard n-gram models to get the best of both worlds. The question that remains is how these models should be combined. The simplest option is to condition a word on the joint of its parents or put differently to condition n-grams on the topic. Phrased like that, the model is equivalent to the topic mixture model of (Iyer *et al.* 1994) discussed in section 3.9. So, we can now understand the sentence mixture model from a Bayesian perspective and use this insight to extend the model.

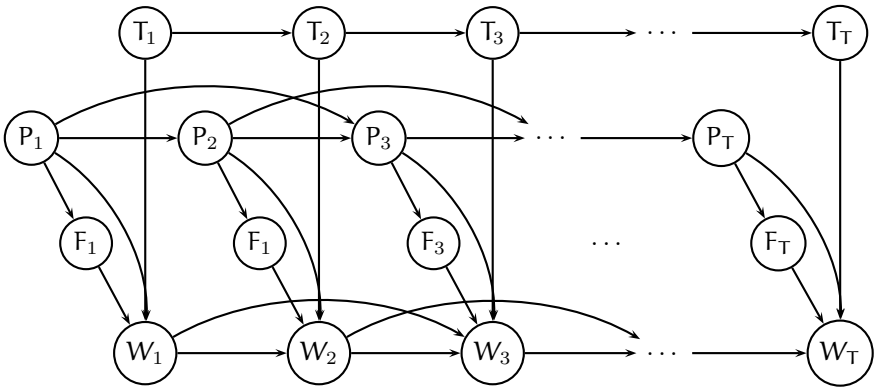


Figure 10.4 – *A topic model that uses part of speech information* (P). *The* F_t *variable signals whether the word* W_t *is a content word in which case it is conditioned on the topic* T_t *as well. Function words are only conditioned on previous words and part of speech tags.*

10.3 A more advanced topic model

Making n-grams dependent on the topic may unnecessarily fragment training data. What such models are capturing is not so much the topic of a conversation but rather the distribution of a particular subset of the data, combining information on genre and topics. It seems reasonable to assume that content words are much more topic dependent than function words, whereas function words are more dependent on the type-of-speech as shown in chapter 5. To be precise, complete word sequences, such as 'global heating' can be topic dependent. To deal with this the topic model can be combined with the chunking model presented in section 8.5.1. In the remainder of this chapter we will stick to a simpler model that only conditions content words on the topic. This implies that the type-of-speech of a word has to be predicted before the word itself is predicted. Therefore, we built our topic model on top of the POS-model described in 8.2. The resulting model is shown in 10.4.

In every time slice of this model first the part-of-speech P is predicted. The F node is a binary variable that indicates whether the word is a function word or a content word. In case of a content word the word is not only conditioned on its POS-tag and previous words, but also on the topic. In case of a function word the topic distribution is simply a copy of the topic distribution of the previous slice.

Note that we could have simplified the model by making the function words dependent on the topic as well. Given that function words are more or less equally distributed in all topics the model would still work. However, the topic distribution would be much more uniform and hence less predictive and, as will become clear in section 10.5, harder to train.

Up to now, we have assumed that the transitions between topics are deterministic. And with good reason, as this guarantees that the model prefers coherent texts.

Nevertheless, we could allow probabilistic transition between topics. As long as the probability of a self-transition is much larger than that of transitions to other topics there is still a preference for coherent text. The gain would be additional flexibility. The model might learn common topic transitions or relations between topics from the training data. A higher belief of a topic then increases the belief of related topics. In theory, probabilistic topic transitions also remove the need for every word to have a non-zero probability in every topic. In practice however, it turns out that it is better to use this constraint. We experimented with several configurations and found that a model that has only self-transitions between topics within a sentence but allows probabilistic transitions to other topics at sentence boundaries performs best. The probability distribution at topic boundaries typically shows clusters of related topics. Many variations on this model are possible. For example user knowledge can be added to model topic preference. Other modalities, such as computer vision, or background information can influence the topic distribution as well.

10.4 Relating topics and spreading activation

In chapter 1 the concept of spreading activation as a model of human language processing was explained. In this model words activate semantically related words that will then be recognised faster when they do occur. The topic-based model was developed as an attempt to place this idea in a probabilistic framework. The topic distributions mimic the spreading activation algorithm. The further a word is from the core of the topic, i.e. the words that have a high probability in a cluster, the lower the probability will be. Of course the spreading activation model is more fine-grained, as it matters which particular word has been activated, while in a topic-based network it only matters whether a topic has a high probability of generating a word. But by making the topics more specific, spreading activation can be approximated. In the ultimate case there would be one topic per word, but in fact it is not necessary to go any further than the cliques in a network of semantically related words, as those will consists of words that will mutually activate each other in one activation step. Moreover, as a semantic network of word occurrences typically contains subnetworks of strongly connected words and these subnetworks are only weakly connected to each other (Bordag and Bordag 2003; Veronis 2004) — an effect that is called the local world property — a topic network with a relatively small number of topics will do almost as well.

10.5 Training the topic model

If a data set is annotated with topic information, any of the methods discussed in chapter 3 can be used to calculate smoothed estimates of the topic dependent word distributions. If training sequences contain multiple topics, topic transitions can also be estimated this way.

Unfortunately, most data sets do not contain topic information as it it very labour intensive to create such a set. As argued above it is very hard to decide on a good set of topics. Even if a set is annotated, the question remains whether that annotation is useful for language modelling.

In theory, this should not be a problem, as the topic based language model can learn its parameters from unannotated data using the expectation maximisation algorithm. In case of topic learning the algorithm proceeds as follows: The model starts with random distributions. In every iteration, the training algorithm will try to maximise the probability of the data. Imagine that one of the topics provides a relatively high probability for several content words that occur in the training text. The expectation step of the algorithm will now conclude that this topic has a high probability of producing the text. The maximisation step will use this information to assign to all words in the text a higher probability of being produced in this text. Some of those words may also occur in other texts together with other words and therefore get a higher probability in other topics. Conversely, the words that are really related to the content words the topic started with will mainly co-occur with those words and therefore get their highest probability from this topic. In the next iteration, those words in turn may make the probabilities of other texts and other words higher for this topic. Essentially, this approach implements a soft-clustering of words in topics.

The major drawback of the EM algorithm is that it only guarantees to find a local maximum. We experimented with this approach and found that the algorithm is very sensitive to its initial distributions and does not find good topic distributions.

There are several solution to this problem. One interesting variation is to introduce the document as a variable in the model to function as a constraint on the topic variable. $\mathsf{P}(\mathsf{T}|\mathsf{D})$ is initialised in such a way that for every document only a limited number of topics has a high probability. This method is strongly related to probabilistic semantic analysis (Hofmann 1999) but adds the time dimension to this model.

Another approach, the one that we took, is to find better initial distributions for the model. We initialised topics with clusters of semantically related recordings. To obtain these clusters, we created a vector in lemma space for every recording. With every document a weight vector is associated. The length of the vector corresponds to the number of different semantically salient lemma types in the vocabulary, that were found by removing all function words and common content words from the vocabulary. We used lemmas rather than words, as inflections are not important for topicality. The entries of the vectors are weights that indicate the relation between the document and the lemmas. We used term frequency-inverse document frequency (TF-IDF) weights as widely used in information retrieval (see e.g. Baeza-Yates and Ribeiro-Neto 1999):

$$\text{weight}(\mathsf{i},\mathsf{j}) = \begin{cases} (1+\log(\mathtt{tf}_{\mathsf{ij}}))\log(\frac{\mathsf{N}}{\mathtt{df}_{\mathsf{i}}}) & \mathtt{tf}_{\mathsf{ij}} > 0, \\ 0 & \mathtt{tf}_{\mathsf{ij}} = 0, \end{cases} \tag{10.1}$$

where N is the number of documents and the term frequency $\mathtt{tf}_{\mathsf{ij}}$ counts the number

of times lemma i occurs in document j. High frequency lemmas are thought to be characteristic for the document. Higher counts reflect more saliency of a word for a document but the scale is not linear. Observing a word twice as much does not mean that it is twice as important. Therefore, term frequencies are logarithmically scaled.

This quantity is weighted by the inverse document frequency df_i which gives the number of different documents lemma i occurs in. The idea is that lemmas that occur in many documents are semantically less discriminating. This component is also logarithmically weighted. Note that a word that occurs in all documents will get weight zero.

Together the word vectors span a high-dimensional space in which each dimension corresponds to a lemma. To measure semantic similarity between documents we apply a metric.

To find clusters of related documents we used agglomerative clustering with the cosine as a similarity measure. Agglomerative clustering is a greedy iterative algorithm in which every vector initially has its own single-element cluster. In every iteration, the two most similar clusters are merged. The similarity of two clusters with multiple elements is defined as the distance between the two least similar elements (complete-link clustering).

Every document is annotated with the cluster number of its corresponding vector. This annotated data was used to estimate the initial parameters of the topic model using simple maximum likelihood estimation. The parameters are then interpolated with the global distribution over content words to make sure that all words have a non-zero probability for all topics. One could leave it at this, but then we would not be using the full potential of the topic model. Agglomerative clustering assigns very document to a single cluster but documents may contain several topics. The topic model can represent soft clustering. Therefore, we only selected the most similar half of the documents of every cluster for initialisation of the topic model. The idea is that this will result in relatively coherent topic distributions, documents that can belong to multiple clusters are not used for initialisation. Next the topic model was retrained on all training data using the EM algorithm. All documents are assigned to all topics weighted by the likelihood of the topics resulting in soft clustering rather than in hard clustering. Recall that our model allows for probabilistic topic transitions at sentence boundaries. This transition distribution, simultaneously trained, enables dealing with documents that contain a sequence of topics.

Monitoring the perplexity of a development test set we found that the soft clustering step does result in a better model than using the clusters directly to estimate topic distributions, but the model quickly overtrains. Therefore, we introduced a damping factor (Murphy 2002):

$$P_t(W|T) = (1-\delta)\tilde{P}_t(W|T) + \delta P_{t-1}(W|T) \quad 0 \leqslant \delta \leqslant 1, \tag{10.2}$$

where $\tilde{P}_{i+1}(W|T)$ is the result of EM iteration $i+1$ and $P_i(W|T)$ is the topic distribution obtained in the previous iteration. Depending on the weight δ distributions

Table 10.1 – *Perplexity results*

language model	perplexity
interpolated bigram	296.49
interpolated trigram	280.76
topic-based model with 64 topics	242.92

are only partially updated in every iteration. If $\delta = 0$ this amounts to normal updating. $\delta = 1$ would result in no updating at all. Because the initial distributions were smoothed by interpolation with the global distributions, the damping factor has additional advantage that it avoids zero probabilities for words that do not occur in the training data in subsequent training iterations.

10.6 Experiments

We tested the topic model on component *f* of the CGN corpus described in chapter 5. This set contains interviews and discussions broadcasted on radio and television. The set contains a total of 790.269 words, 80% of which we use for training, 10% for development testing and tuning and the remaining 10% for evaluation. All words that occur only once in the training set are treated as out-of-vocabulary, resulting in a vocabulary size of 17833 words and 257 POS-tags. The POS-tags include attributes such as number, degree and tense. Table 10.1 gives the perplexity of the model on the evaluation set. As a baseline the perplexities of standard interpolated bigram and trigram on the same set with the same vocabulary are also shown. In all cases the interpolation weights have been optimised on the same development test set. The topic model clearly outperforms the standard models.

10.7 Doing it differently: conditional models

The topic model described in this section is a generative model, i.e. it models the (hidden) causes that have generated the observed words. In this it follows common practice in both HMM based speech recognition and Bayesian belief propagation in probabilistic expert systems (Pearl 1988). Generative models have the advantage that they can easily generalise; they provide a probability for every observed string. Furthermore, they often make for simple and elegant models. On the downside, generative models can seldomly exclude anything at all. In the topic based language model this shows in the fact that every word has to have a non-zero probability in every topic.

The alternative way of looking at things is from a conditional point of view. In this case the observed variables are seen as features that help to identify the hidden variables. Specifically, for the topic-based model this would mean that the topic at

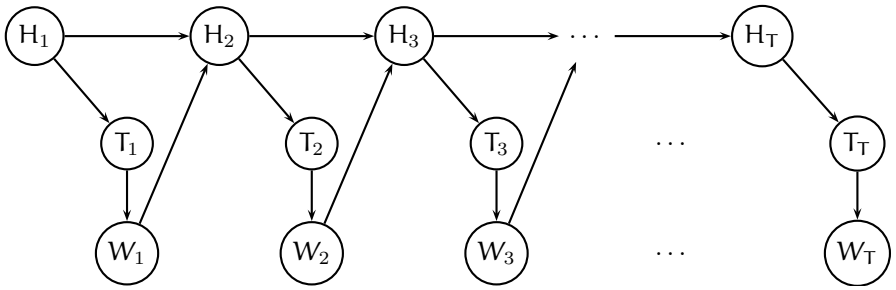


Figure 10.5 – *A conditional topic-based language model. The topic* T_t *is conditioned on the history* H_t *which is a function of the history in the previous time slice and the previous word.*

some point in time is directly conditioned upon the previously observed words. In practice the topic will be conditioned upon a deterministic function of the history rather than on the complete history itself to obtain reliable statistics. In fact almost all topic based language models proposed in literature (Seymore and Rosenfeld 1997; Mahajan *et al.* 1999; Zhang and Rudnicky 2002; Gildea and Hofmann 1999; Khudanpur and Wu 1999) are of this kind. In terms of DBNs such a scheme can be realised by introducing an explicit, deterministic history variable as shown in figure 10.5. Instead of a DBN with history nodes the probability $P(w_i|T)P(T|w_1 \dots w_{i-1})$ can be modelled directly as the topic no longer depends on information from previous time slices.

Acta est fabula.

Chapter 11

Conclusion

In which the research questions formulated in chapter 1 will be answered.

The statistical approach based on hidden Markov models forms the state of the art in speech recognition. Having been introduced in the 1970s (Jelinek 1976) it has been dominating the field since the early 1990s. In this time great leaps forward in speech recognition have been made and some applications have become feasible. However, compared with human speech recognition, the performance of its automatic counterpart is still an order of magnitude behind. More importantly, speech recognition is not yet robust enough to function as a general man-machine interface. For example, dictation systems require extensive training to adapt to the voice of a particular user and the acoustical properties of the environment before they become useful.

Much of the recent progress in speech recognition has come from carefully crafting systems for particular application domains, i.e. by keeping the context of use in mind when defining the vocabulary of a recogniser and when selecting or fabricating training data. Recognisers that deal with multiple contexts often select models from a predefined set. For example systems for broadcast news transcription have different acoustic models for different environments, ranging from the anchorman in the studio to interviews with an eyewitness in the street, as wells as item specific language models, each with its own vocabulary.

Based on this practice and by drawing inspiration from models of human speech recognition, I propose to include information about the context of use directly in the models of speech recognition.

This information will allow a general purpose recogniser to focus on a particular situation. The complex task of person independent recognition of any speech about any topic reduces to a sequence of domain and person specific recognition tasks. The difference with a domain specific recogniser is that such a system can dynamically adapt to context changes. It is not limited to predefined context-dependent models, but adjusts its parameters by reasoning over the context. As a consequence, this approach also reduces the need for domain specific data. Systems for new domains do not have to be developed from scratch, but can exploit similarities between domains. For example, the differences between formal and informal speech are constant across all conversation topics. Therefore, rather than needing a model for formal conversations about politics and a model for informal conversations about politics, a system in which the type of speech and the topic of a conversation are included as variables can deal with an informal conversation about politics even if it has only been trained on formal conversations about politics and informal conversations about other subjects.

Using findings in sociolinguistics, psychology and linguistics, I define three types of context: conversational knowledge, user knowledge and world knowledge. Conversational knowledge tells whether it is a conversation between a user and a system or a conversation between multiple speakers that is transcribed by a system. In addition, it includes the type of speech, i.e. whether we are dealing with for example spontaneous speech, with broadcast speech or with more formal speech such as a debate or read speech. It also includes the topic of a conversation which is a strong determinant of the vocabulary used. User knowledge comprises speaker characteristics such as dialect, gender, age and education level. All of which have been shown to influence speech production and hence are relevant for speech recognition. World knowledge is subdivided in knowledge provided by other modalities and domain specific knowledge.

To find out whether context and language use do or do not correlate I analysed a large corpus of spoken Dutch. This research confirmed that word use indeed differs for different types of speech. In particular, I found that sentence length distributions are very different for different types of speech as is the relative frequency of parts of speech. It turns out that the relative frequencies of personal pronouns are characteristic for the type of speech. I also found influence of speakers' gender, education level and age on word use.

The data analysis suggests that modelling context in speech recognition should help. To further investigate this I conducted two case studies. The first case study looked at the usefulness of information from other modalities in speech recognition. In particular, I integrated a data stream obtained from an automated lipreader in a speech recogniser. Experiments with different configurations show that integration within the acoustic models of the recogniser gives better results than early integration of the audio and video signals. A model that uses different classes for the two input

streams, phonemes for audio and visemes for video, outperformed the audio-only speech recogniser. In case of background noise the audio-visual recogniser performs considerably better than the standard speech recogniser. The recognition rates of the bimodal system were up to 15% higher.

The second case study was about the use of domain knowledge in a speech recogniser. A train table dialogue system was chosen as the domain of use. By analysing transcriptions of conversations with such a system we found a correlation between departure and arrival station. I used this finding to dynamically set the language model probabilities with which a recogniser predicts a station name if the other name is recognised. I found that use of this information directly in the speech recogniser results in better performance than using this information in a later stage. But even if the information is used afterwards, by rescoring several hypotheses output by the recogniser using the context information the approach outperforms a standard recogniser that does not use domain knowledge. I also found that context information should not override perceptual information. The best results were achieved if context information was only used if there was low confidence in the recognition result. To decide when to use contextual information a confidence measure was developed as part of this research.

From the data analysis and the case studies we can conclude that modelling context does indeed improve speech recognition. However, putting contextual information in hidden Markov models is far from simple. Our case studies fit in with much research in speech recognition dedicated to developing variations on the hidden Markov model to include a particular piece of additional knowledge that is thought to be of help. Typically, these model variations require specialised inference routines. Even though one might expect that many of the small improvements presented in literature are additive, the specialised nature of the models makes them difficult to integrate. Therefore, I argue that a new computational paradigm for speech recognition is needed, in which context as well as other information can be included. I compared several paradigms that have proven their worth in speech and language processing before, and conclude that dynamic Bayesian networks provided a good computational paradigm for speech recognition. Dynamic Bayesian networks can be seen as a generalisation of the hidden Markov models and n-gram models typically used in speech recognition. They retain the greatest strength of these models, their statistical nature, but allow for a much richer representation of model state in terms of random variables and the relations between those variables. Earlier research already showed how dynamic Bayesian networks can be used for acoustic modelling (Zweig 1998; Bilmes 1999). In this thesis I show how a complete speech recogniser can be formulated in terms of dynamic Bayesian networks. In particular, I reformulate existing language models in terms of dynamic Bayesian networks, and show that in some cases the resulting models are even more efficient and conceptually clearer than in the original formulation. I discuss how such models can be combined and extended without the need to develop new algorithms.

The most important contributions of this thesis are a number of new language models that include sentence length, context information and syntactic structure.

Specifically, I designed a new adaptive language model that includes topic information.

The assumption behind this model is that every conversation displays coherence. Simply put, it has a topic. The model relies on Bayesian updating to find and track the topic of a conversation. A topic is modelled as a distribution over words. At any point in time the actual topic of a conversation is seen as a mixture of these topic distributions. The composition of this mixture can change as topics may come and go. The model thus focusses on words that are likely in the context. Unlike existing topic-based models, the model separates the topic and the structure of a sentence. The topic is used in the prediction of future content words in the discourse, while part-of-speech information is used to better model the word sequences in which these content words can appear and to detect sentence boundaries at which a chance of topic is allowed. I developed an unsupervised learning algorithm for this model, that extracts topics from a data set. Experiments show that this model performs better, in terms of perplexity, than standard language models. The model can be seen as an extension of sentence level mixture models and thus provides an explanation of such models in terms of Bayesian updating.

From a computational point of view, speech recognition is challenging because it searches through a huge state space. The time and space complexity of dynamic Bayesian networks is exponential in the number of states. Therefore, straightforward implementations of dynamic Bayesian networks cannot be used for speech recognition. Luckily, the search space in speech recognition is very sparse, because many of its states do not occur in practice. I combined ideas scattered throughout literature on Bayesian networks, hidden Markov models and speech recognition to design algorithms and data-structures that make continuous speech recognition with dynamic Bayesian networks possible. Novel contributions include a highly efficient algorithm for inference with probability tables, the combination of lazy evaluation at the probability table level and tree-shaped distributions that exploit independencies in the network that are not used by junction tree based inference algorithms such as the frontier and forward algorithm and a generalisation of dynamic Bayesian networks that consists of several chapters each of which contains a repeating substructure that can span several slices in the time domain. These algorithms were implemented in a toolkit to be used for rapid model construction and experimentation and opens up the way for research in context-rich models for speech and language processing. A particularity exciting direction for the future is to build on the large body of literature available on structure learning of Bayesian networks to learn new models for speech recognition from data.

Bibliography

S. P. Abney (1991), 'Parsing by chunks', in R. C. Berwick, S. P. Abney, and C. Tenny (eds.), *Principle-Based Parsing: Computation and Psycholinguistics*, pp. 257–278, Kluwer, Dordrecht, URL `citeseer.ist.psu.edu/abney91parsing.html`.

R. Aji, S.M. McEliece (2000), 'The generalized distributive law', *IEEE Transactions on Information Theory*, 46(2), pp. 325–343.

J. Allen (1995), *Natural language understanding (second edition)*, Benjamin/Cummings, Menlo Park, CA.

P. van Alphen (1992), *HMM-based continuous-speech recognition*, Ph.D. thesis, University of Amsterdam.

S. K. Andersen, K. G. Olesen, F. V. Jensen, and F. Jensen (1989), 'HUGIN—a shell for building Bayesian belief universes for expert systems', in *Eleventh International Joint Conference on Artificial Intelligence (ijcai89)*, vol. 2, pp. 1080–1085.

S. Argamon, M. Koppel, J. Fine, and A. R. Shimoni (2003), 'Gender, genre, and writing style in formal written texts', *Text*, 23(3).

J. Austin (1962), *How to Do Things with Words*, Oxford University Press, New York.

R. Baeza-Yates and B. Ribeiro-Neto (1999), *Modern Information Retrieval*, Addison Wesley.

L. R. Bahl, P. F. Brown, P. V. DeSouza, and R. L. Mercer (1989), 'A tree-based statistical language model for natural language speech recognition', *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 37(7), pp. 1001–1008.

L. R. Bahl, F. Jelinek, and R. L. Mercer (1983), 'A maximum likelihood approach to continuous speech recognition', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2), pp. 179–190.

A. Batliner, R. Huber, H. Niemann, E. Nöth, J. Spilker, and K. Fischer (2000), 'The recognition of emotion', in W. Wahlster (ed.), *Verbmobil: Foundations of Speech-to-Speech Translations*, pp. 122–130, Springer.

T. Bayes (1763), *An Essay Toward Solving a Problem in the Doctrine of Chances*, vol. 53, reprinted in *Facsimiles of two papers by Bayes*, Hafner Publishing Company, New York, 1963.

A. Bell (1984), 'Language style as audience design', *Language in Society*, 13, pp. 145–204.

J. R. Bellegarda (1998), 'A multispan language modeling framework for large vocabulary speech recognition', *IEEE Transactions on Speech and Audio Processing*, 6(5), pp. 456–467.

A. L. Berger, S. D. Pietra, and V. J. D. Pietra (1996), 'A maximum entropy approach to natural language processing', *Computational Linguistics*, 22(1), pp. 39–71, URL `citeseer.ist.psu.edu/berger96maximum.htm`.

R. van Bezooijen (1985), 'Een vergelijkende stemkwaliteitsbeschrijving van vier groepen Amsterdammers.', *Spectator*, 13(3), pp. 182–192.

R. van Bezooijen (1995), 'Sociocultural aspects of pitch differences between Japanese and Dutch women.', *Language and Speech*, 38, pp. 253–266.

J. Bilmes (1998), 'A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models', Tech. Rep. ICSI-TR-97-021, University of Berkeley, URL `citeseer.ist.psu.edu/bilmes98gentle.html`.

J. Bilmes (1999), *Natural Statistical Models for Automatic Speech Recognition*, Ph.D. thesis, Dept. of EECS, University of California, Berkeley, URL `citeseer.ist.psu.edu/bilmes99natural.html`.

J. Bilmes (2002*a*), *GMTK: the Graphical Models Toolkit*, University of Washington.

J. Bilmes (2002*b*), 'What HMMs can do', Tech. Rep. UWEETR–2002–0003, Department of Electrical Engineering, University of Washington.

J. Bilmes and G. Zweig (2002), 'The graphical models toolkit: An open source software system for speech and time-series processing', in *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, URL `citeseer.ist.psu.edu/bilmes02graphical.html`.

D. Binnenpoorte, C. van Bael, E. den Os, and L. Boves (2005), 'Gender in everyday speech and language: a corpus-based study', in *Interspeech 2005*, pp. 2213–2216.

E. Black, F. Jelinek, J. D. Lafferty, D. M. Magerman, R. L. Mercer, and S. Roukos (1992), 'Towards history-based grammars: Using richer models for probabilistic parsing', in *Proceedings DARPA Speech and Natural Language Workshop*, pp. 134–139, Morgan Kaufmann, Harriman, New York.

S. Bordag and D. Bordag (2003), 'Advances in automatic speech recognition by imitating spreading activation', in *Text, Speech and Dialogue 2003*.

H. Bourlard, S. Dupont, and C. Ris (1996), 'Multi-stream speech recognition', Research Report 96-07, IDIAP.

C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller (1996), 'Context-specific independence in Bayesian networks', in *Uncertainty in Artificial Intelligence*, pp. 115–123, URL `citeseer.ist.psu.edu/article/boutilier96contextspecific.html`.

T. Boves and M. Gerritsen (1995), *Inleiding in de sociolinguïstiek*, Uitgeverij Het Spectrum, Utrecht.

X. Boyen and D. Koller (1998), 'Tractable inference for complex stochastic processes', in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pp. 33–42, URL `citeseer.ist.psu.edu/article/boyen98tractable.html`.

M. Brand, N. Oliver, and A. Pentland (1997), 'Coupled hidden Markov models for complex action recognition', in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, 407.

J. van den Broeck (1980), 'Beperkte en geelaboreerde stijl in 'formele' en 'informele' interviews in maaseik', in G. Geerts and A. Hagen (eds.), *Sociolinguistische studies 1. Bijdragen uit het Nederlandse taalgebied*, Wolters-Noordhoff.

D. Brouwer (1989), *Gender variation in Dutch*, Ph.D. thesis, KU Nijmegen, Dordrecht.

D. Brouwer (1991), *Vrouwentaal. Feiten en verzinsels*, Aramit, Bloemendaal.

P. F. Brown, J. Cocke, S. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin (1990), 'A statistical approach to machine translation', *Computational Linguistics*, 16(2), pp. 79–85, URL `citeseer.ist.psu.edu/brown90statistical.html`.

P. F. Brown, V. J. D. Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer (1992), 'Class-based n-gram models of natural language', *Computational Linguistics*, 18(4), pp. 467–479, URL `citeseer.ist.psu.edu/brown90classbased.html`.

D. Bruce (1958), 'The effects of listeners' anticipations in the intelligibility of heard speech', *Language and Speech*, 1, pp. 79–97.

R. Carlson, J. Edlund, M. Heldner, A. Hjalmarsson, D. House, and G. Skantze (2006), 'Towards human-like behaviour in spoken dialog systems', in *Proceedings of Swedish Language Technology Conference (SLTC 2006)*, Gothenburg, Sweden.

E. Charniak (1993), *Statistical Language Learning*, The MIT Press.

E. Charniak (1997), 'Statistical parsing with a context-free grammar and word statistics', in *AAAI/IAAI*, pp. 598–603, URL citeseer.ist.psu.edu/charniak97statistical.html.

E. Charniak (1999), 'A maximum-entropy-inspired parser', Tech. Rep. CS-99-12, Brown University, URL citeseer.ist.psu.edu/article/charniak99maximumentropyinspired.html.

E. Charniak (2001), 'Immediate-head parsing for language models', in *Meeting of the Association for Computational Linguistics*, pp. 116–123, URL citeseer.ist.psu.edu/charniak01immediatehead.html.

C. Chelba (2000), *Exploiting Syntactic Structure for Natural Language Modeling*, Ph.D. thesis, The Johns Hopkins University, URL citeseer.ist.psu.edu/chelba00exploiting.html.

C. Chelba and F. Jelinek (1998), 'Exploiting syntactic structure for language modeling', in C. Boitet and P. Whitelock (eds.), *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, pp. 225–231, Morgan Kaufmann, San Francisco, URL citeseer.ist.psu.edu/chelba98exploiting.html.

C. Chelba and F. Jelinek (1999), 'Recognition performance of a structured language model', in *Proceedings of Eurospeech '99*, URL citeseer.ist.psu.edu/chelba99recognition.html.

S. F. Chen, D. Beeferman, and R. Rosenfeld (1998), 'Evaluation metrics for language models', in *DARPA Broadcast News Transcription and Understanding Workshop*.

S. F. Chen and J. Goodman (1996), 'An empirical study of smoothing techniques for language modeling', in A. Joshi and M. Palmer (eds.), *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pp. 310–318, Morgan Kaufmann Publishers, San Francisco, URL citeseer.ist.psu.edu/article/stanley98empirical.html.

K. Church and W. Gale (1991), 'A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams', *Computer Speech and Language*, 5, pp. 19–54.

P. Clarkson and A. J. Robinson (1997), 'Language model adaptation using mixtures and an exponentially decaying cache', in *Proc. ICASSP '97*, pp. 799–802, Munich, Germany, URL citeseer.ist.psu.edu/clarkson97language.html.

P. Clarkson and T. Robinson (1999), 'Towards improved language model evaluation measures', in *Proceedings of EUROSPEECH 99, 6th European Conference on Speech Communication and Technology*, vol. 5, pp. 1927–1933, URL citeseer.ist.psu.edu/clarkson99toward.html.

N. Coccaro and D. Jurafsky (1998), 'Towards better integration of semantic predictors in statistical language modeling', in *ICSLP-98*, vol. 6, pp. 2403–2406, Sydney.

M. Collins (1997), 'Three generative, lexicalized models for statistical parsing', in P. R. Cohen and W. Wahlster (eds.), *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pp. 16–23, Association for Computational Linguistics, Somerset, New Jersey, URL citeseer.ist.psu.edu/article/collins97three.html.

M. Collins (1999), *Head-Driven Statistical Models for Natural Language Parsing*, Ph.D. thesis, University of Pennsylvania, URL citeseer.ist.psu.edu/collins03headdriven.html.

M. J. Collins (1996), 'A new statistical parser based on bigram lexical dependencies', in A. Joshi and M. Palmer (eds.), *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pp. 184–191, Morgan Kaufmann, San Francisco, URL citeseer.ist.psu.edu/collins96new.html.

T. M. Cover and J. A. Thomas (1991), *Elements of Information Theory*, Wiley Series in Telecomunication.

R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter (1999), *Probabilistic Networks and Expert Systems*, Springer.

B. Daille (1995), 'Combined approach for terminology extraction: lexical statistics and linguistic filtering.', Technical Report 5, Lancaster University.

R. C. van Dalen (2005), *Lexical Stress in Speech Recognition*, Master's thesis, Delft University of Technology.

R. C. van Dalen, P. Wiggers, and L. J. M. Rothkrantz (2005), 'Modelling lexical stress', in *Text, Speech and Dialogue*, *Lecture Notes in Artificial Intelligence*, vol. 3658, pp. 211–218, Springer Verlag.

R. C. van Dalen, P. Wiggers, and L. J. M. Rothkrantz (2006), 'Lexical stress in continuous speech recognition', in *Proceedings of the Ninth International Conference on Spoken Language Processing (Interspeech 2006 - ICSLP)*, pp. 2382–2385.

M. Damhuis, T. Boogaart, C. In 't Veld, M. Versteijlen, W. Schelvis, L. Bos, and L. Boves (1994), 'Creation and analysis of the Dutch Polyphone corpus', in *Proceedings of the International Conference on Spoken Language Processing, ICSLP'94*, pp. 1803–1806, Yokohama, Japan.

R. Dechter (1999), 'Bucket Elimination: A unifying framework for reasoning', *Artificial Intelligence*, 113(1–2), pp. 41–85, URL citeseer.ist.psu.edu/article/dechter99bucket.html.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977), 'Maximum likelihood from incomplete data via the em algorithm', *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1), pp. 1–38, URL http://links.jstor.org/sici?sici=0035-9246%281977%2939%3A1%3C1%3AMLFIDV%3E2.0.CO%3B2-Z.

A. Doucet, N. de Freitas, K. Murphy, and S. Russell (2000), 'Rao-Blackwellised particle filtering for dynamic Bayesian networks', in *UAI '00 (Uncertainty in AI)*.

M. J. Druzdzel (1999), 'SMILE: Structural modeling, inference, and learning engine and GeNIe: a development environment for graphical decision-theoretic models', in *AAAI '99/IAAI '99*, pp. 902–903, Menlo Park, CA, USA.

M. J. Druzdzel (2005), 'Intelligent decision support systems based on SMILE', *Software 2.0*, 2, pp. 12–33.

T. Dunning (1993), 'Accurate methods for the statistics of surprise and coincidence', *Computational Linguistics*, 19(1), pp. 61–74, URL citeseer.ist.psu.edu/dunning93accurate.html.

G. Evermann and P. Woodland (2000), 'Posterior probability decoding, confidence estimation and system combination', in *Proceedings of Speech Transcription Workshop*.

E. Filisko and S. Seneff (2003), 'A context resolution server for the Galaxy conversational systems', in *EUROSPEECH-2003*, pp. 197–200, URL http://www.isca-speech.org/archive/eurospeech_2003/e03_0197.html.

S. Fine, Y. Singer, and N. Tishby (1998), 'The hierarchical hidden Markov model: Analysis and applications', *Machine Learning*, 32(1), pp. 41–62.

S. Fitrianie, R. Poppe, T. Bui, A. Chitu, D. Datcu, R. Dor, D. Hofs, P. Wiggers, D. Willems, M. Poel, L. Rothkrantz, L. Vuurpijl, and J. Zwiers (2007), 'A multimodal human-computer interaction framework for research into crisis management', in *ISCRAM2007*.

R. Florian and D. Yarowsky (1999), 'Dynamic nonlocal language modeling via hierarchical topic-based adaptation', in *ACL-99*, pp. 167–174, ACL, College Park, MD.

V. Fromkin and R. Rodman (1993), *An introduction to language*, New York : Harcourt Brace Jovanovich, 5th edition ed.

S. Furui (2001), *Digital Speech Processing, Synthesis and Recognition, Second Edition Revised and Expanded*, Marcel Dekker.

S. Furui (2005), '50 years of progress in speech and speaker recognition research', *Transactions on Computer and Information Technology ECTI-CIT*, 1(2), pp. 64–74.

W. A. Gale and G. Sampson (1995), 'Good-Turing frequency estimation without tears', *the Journal of Quantitative Linguistics*, 2, pp. 217–237.

M. Gales (1998), 'Maximum likelihood linear transformations for hmm-based speech recognition', *Computer Speech and Language*, 12.

F. Gallwitz, M. Aretulaki, M. Boros, J. Haas, S. Harbeck, R. Huber, H. Niemann, and E. Nöth (1998), 'The Erlangen spoken dialogue system EVAR: A state-of-the-art information retrieval system', in *Proceedings of the 1998 Symposium on Spoken Dialogue (ISSD 98)*.

J. Gao and J. T. Goodman (2002), 'Exploring asymmetric clustering for statistical language modeling', in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 183–190, Philadelphia.

Z. Ghahramani and M. Jordan (1997), 'Factorial hidden Markov models', *Machine Learning*, 29, pp. 245–275.

S. van Gijsel, D. Speelman, and D. Geeraerts (2006), 'Locating lexical richness: a corpus linguistic, sociovariational analysis', in J. Viprey (ed.), *Proceedings of the 8th International Conferene on the statistical analysis of textual data (JADT)*, pp. 961–971, Besan con, France.

D. Gildea and T. Hofmann (1999), 'Topic-based language models using EM', in *Proceedings of the 6th European Conference on Speech Communication and Technology (EUROSPEECH)*, URL `citeseer.ist.psu.edu/gildea99topicbased.html`.

D. Gill-Günzburger (1979), *Linguistic expectancy and word-initial consonant clusters*, Ph.D. thesis, Utrecht, Univ., Literatuurwetenschappen.

I. Good (1953), 'The population frequencies of species and the estimation of population parameters', *Biometrika*, 40, pp. 237–264.

J. Goodman (2000), 'A bit of progress in language modeling', Tech. rep., Microsoft Research, 56 Fuchun Peng, URL `citeseer.ist.psu.edu/goodman01bit.html`.

Y. Gotoh and S. Renals (1999), 'Topic-based mixture language modelling', *Natural Language Engineering*, 5(4), pp. 355–375.

F. Grosjean (1980), 'Spoken word recognition processes and the gating paradigm', *Perception and Psychophysics*, 28, pp. 267–283.

E. Halgren, R. P. Dhond, N. Christensen, C. V. Petten, K. Marinkovic, J. D. Lewine, and A. M. Dale (2002), 'N400-like magnetoencephalography responses modulated by semantic context, word frequency, and lexical class in sentences', *NeuroImage*, 17, pp. 1101–1116.

A. Hämäläinen, L. ten Bosch, and L. Boves (2006), 'Pronunciation variant-based multi-path HMMs for syllables', in *Interspeech 2006*, pp. 1579–1582.

T. Harley (2001), *The Psychology of Language: From Data to Theory*, Psychology Press, Hove.

K. Harnqvist, U. Christianson, D. Ridings, and J.-G. Tingsell (2003), 'Vocabulary in interviews as related to respondent characteristics', *Computers and the Humanities*, 37, pp. 179–204, URL `http://www.ingentaconnect.com/content/klu/chum/2003/00000037/00000002/00396067`.

D. Heckerman, D. Geiger, and . D. Chickering (1995), 'Learning Bayesian networks: The combination of knowledge and statistical data.', *Machine Learning*, 20(3), pp. 197–243.

P. A. Heeman (1999), 'POS tags and decision trees for language modeling', in *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-99)*, pp. 129–137, ACL, College Park, MD.

V. J. van Heuven, R. van Bezooijen, and L. Edelman (2005), 'Pronunciation of /ei/ in avant-garde Dutch: A cross-sex acoustic study', in M. P. Markku Filppula, Juhani Klemola and E. Penttilä (eds.), *Dialects Across Borders; Selected papers from the 11th International Conference on Methods in Dialectology (Methods XI)*, pp. 185–210, John Benjamins Publishing Company, University of Joensuu / University of Tampere.

T. Hofmann (1999), 'Probabilistic latent semantic analysis', in *Proc. of Uncertainty in Artificial Intelligence, UAI'99*, Stockholm, URL `citeseer.ist.psu.edu/hofmann99probabilistic.html`.

J. Hu, M. K. Brown, and W. Turin (1996), 'HMM based on-line handwriting recognition', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10), pp. 1039–1045.

J. Hulst (2006), *Modeling physiological processes with dynamic Bayesian networks*, Master's thesis, Man-Machine Interaction Group, Delft University of Technology.

Intel Corporation (2004), *Probabilistic Network Library - User guide and reference manual*.

Interspeech (2006), *Proceedings of the Ninth International Conference on Spoken Language Processing (Interspeech 2006 – ICSLP)*, Pittsburgh.

Interspeech (2007), *Proceedings of Interspeech 2007 - Eurospeech, 10th European Conference on Speech Communication and Technology*, Antwerp.

R. Iyer and M. Ostendorf (1996), 'Modeling long distance dependence in language: Topic mixtures vs. dynamic cache models', in *Proc. ICSLP '96*, vol. 1, pp. 236–239, Philadelphia, PA, URL `citeseer.ist.psu.edu/iyer96modeling.html`.

R. Iyer, M. Ostendorf, and M. Meteer (1997), 'Analyzing and predicting language model improvements', in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding.*, URL `citeseer.ist.psu.edu/iyer97analyzing.html`.

R. Iyer, M. Ostendorf, and J. R. Rohlicek (1994), 'Language modeling with sentence-level mixtures', in *HLT '94: Proceedings of the workshop on Human Language Technology*, pp. 82–87, Association for Computational Linguistics, Morristown, NJ, USA.

I. Jacobi, L. C. Pols, and J. Stroop (2005), 'Polder Dutch: Aspects of the /Ei/-lowering in standard Dutch', in *Eurospeech 2005*, pp. 2877–2880, ISCA, Lisbon.

F. Jansen (1981), *Syntaktische konstrukties in gesproken taal.*, Huis aan de drie grachten, Amsterdam.

H. Jeffreys (1948), *Theory of Probability*, Clarendon Press, Oxford, second edition ed.

F. Jelinek (1976), 'Continuous speech recognition by statistical methods', *Proceedings of the IEEE*, 64(4), pp. 532–557.

F. Jelinek (1990), 'Self-organized language modeling for speech recognition', in A. Waibel and K.-F. Lee (eds.), *Readings in Speech Recognition*, Morgan Kaufman Publishers, Inc.

F. Jelinek (1999), *Statistical Methods for Speech Recognition (Language, Speech, and Communication)*, MIT Press.

F. Jelinek and R. L. Mercer (1980), 'Interpolated estimation of Markov source parameters from sparse data', in E. S. Gelsema and L. N. Kanal (eds.), *Proceedings, Workshop on Pattern Recognition in Practice*, pp. 381–397, North Holland, Amsterdam.

F. Jelinek and R. L. Mercer (1985), 'Probability distribution estimation from sparse data', *IBM Technical Disclosure Bulletin*, 28, pp. 2591–2594.

F. Jelinek, B. Merialdo, S. Roukos, and M. Strauss (1991), 'A dynamic language model for speech recognition', in *HLT '91: Proceedings of the workshop on Speech and Natural Language*, pp. 293–295, Association for Computational Linguistics, Morristown, NJ, USA.

F. V. Jensen (2001), *Bayesian Networks and Decision Graphs*, Statistics for Engineering and Information Science Series, Springer Verlag.

F. V. Jensen, S. L. Lauritzen, and K. G. Olesen (1990), 'Bayesian updating in causal probabilistic networks by local computations', *Computational Statistics Quaterly*, 4, pp. 269–282.

M. I. Jordan (ed.) (1998), *Learning in Graphical Models*, MIT Press.

B. H. Juang and L. R. Rabiner (2005), 'Automatic speech recognition–a brief history of the technology', in *Elsevier Encyclopedia of Language and Linguistics, Second Edition*, Elsevier.

D. Jurafsky and J. H. Martin (2000), *Speech and Language Processing*, Prentice Hall.

D. Jurafsky, C. Wooters, J. Segal, A. Stolcke, E. Fosler, G. Tajchman, and N. Morgan (1995), 'Using a stochastic context-free grammar as a language model for speech recognition', in *Proc. ICASSP '95*, pp. 189–192, Detroit, MI, URL `citeseer.ist.psu.edu/article/jurafsky95using.html`.

S. M. Katz (1987), 'Estimation of probabilities from sparse data for the language model component of a speech recogniser', *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3), pp. 400–401.

T. Kawahara, A. Lee, T. Kobayashi, K. Takeda, N. Minematsu, S. Sagayama, K. Itou, A. Ito, M. Yamamoto, A. Yamada, T. Utsuro, and K. Shikano (2000), 'Free software toolkit for japanese large vocabulary continuous speech recognition', in *ICSLP-2000*, vol. 4, pp. 476–479.

S. Khudanpur and J. Wu (1999), 'A maximum entropy language model integrating n-grams and topic dependencies for conversational speech reconition', in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Phoenix, AZ, URL `citeseer.ist.psu.edu/khudanpur99maximum.html`.

A. Kilgarriff (2001), 'Comparing corpora', *International Journal of Corpus Linguistics*, 6(1), pp. 1–37.

B.-W. Kim, D.-L. Choi, Y. Um, and Y.-J. Lee (2006), 'Phone vector DHMM to decode a phone recognizer's output', in *Interspeech 2006*, pp. 1591–1594.

U. Kjaerulff (1995), 'dhugin: a computational system for dynamic time-sliced Bayesian networks', *International Journal of Forecasting*, 11(1), pp. 89–111.

R. Kneser and H. Ney (1993), 'Improved clustering techniques for class-based statistical language modelling', in *EUROSPEECH-93*, pp. 973–976.

R. Kneser and H. Ney (1995), 'Improved backing-off for m-gram language modeling', in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 1, pp. 181–184.

R. Kneser and V. Steinbiss (1993), 'On the dynamic adaptation of stochastic language models', in *Proceedings of ICASSP-93, Minnapolis(USA)*, vol. II, pp. 586–589.

K. B. Korb and A. E. Nicholson (2004), *Bayesian Artificial Intelligence*, Chapman&Hall/CRC.

R. Kuhn and R. de Mori (1990), 'A cache-based natural language model for speech recognition', *EEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6), pp. 570–583.

W. Labov (1972), *Sociolinguistic patterns*, University of Pennsylvania Press.

P. Ladefoged (1996), *Elements of Acoustic Phonetics*, University of Chicago, Chicago, IL, second Edition.

R. Lakoff (1975), *Language and woman's place*, Harper & Row, New York.

R. Lau (1994), *Adaptive Statistical Language Modelling*, Master's thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, URL `citeseer.ist.psu.edu/lau94adaptive.html`.

K. Ledoux, C. C. Camblin, T. Y. Swaab, and P. C. Gordon (2006), 'Reading words in discourse: The modulation of lexical priming effects by message-level context', *Behav Cogn Neurosci Rev.*, 5(3), pp. 107–127.

G. J. Lidstone (1920), 'Note on the general case of the Bayes-Laplace formula for inductive or a posteriori probabilities', *Transactions of the Faculty of Actuaries*, 8, pp. 182–192.

P. Lieberman (1963), 'Some effects of semantic and grammatical context on the production and perception of speech.', *Langauge and Speech*, 6, pp. 172–187.

A. Madsen and F. Jensen (1999), 'Lazy propagation: a junction tree inference algorithm based on lazy evaluation', *Artificial Intelligence*, 113, pp. 203–245.

D. M. Magerman (1994), *Natural Language Parsing as Statistical Pattern Recognition*, Ph.D. thesis, Stanford University.

D. M. Magerman (1995), 'Statistical decision-tree models for parsing', in *Meeting of the Association for Computational Linguistics*, pp. 276–283, URL `citeseer.ist.psu.edu/magerman95statistical.html`.

M. Mahajan, D. Beeferman, and X. Huang (1999), 'Improved topic-dependent language modeling using information retrieval techniques', in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1, pp. 541 – 544.

D. N. Maltz and R. A. Borker (1982), 'A cultural approach to male-female miscommunication', in J. J. Gumperz (ed.), *Language and social identity*, Cambridge University Press.

L. Mangu, E. Brill, and A. Stolcke (1999), 'Finding consensus among words: Lattice-based word error minimization', in *Proc. Eurospeech'99*, pp. 495–498, Budapest, URL `citeseer.ist.psu.edu/284590.html`.

C. D. Manning and H. Schütze (1999), *Foundations of statistical natural language processing*, MIT Press.

K. Markov and S. Nakamura (2006), 'Forward-backwards training of hybrid HMM/BN acoustic models', in *Interspeech 2006*, pp. 621–624.

W. Marslen-Wilson (1984), 'Spoken word recognition: A tutorial review', in H. Bouma and D. Bouwhis (eds.), *Attention and performance X: Control of language processes*, pp. 125–150, Lawrence Erlbaum Associates Ltd.

W. Marslen-Wilson (1987), 'Functional parallelism in spoken word recognition', *Cognition*, 25, pp. 71–102.

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

E. Mays, F. J. Damerau, and R. L. Mercer (1991), 'Context based spelling correction', *Information Processing and Management*, 27(5), pp. 517–522.

J. L. McClelland and J. L. Elman (1986), 'Interactive processes in speech perception: The TRACE model', in J. L. McClelland, D. E. Rumelhart, and the PDP Research Group (eds.), *Parallel Distributed Processing Volume 2: Psychological and Biological Models*, pp. 58–121, MIT Press, Cambridge, MA.

A. Miguel, E. Lleida, A. Juan, L. Buera, A. Ortega, and O. Saz (2006), 'Local transformation models for speech recognition', in *Interspeech 2006*, pp. 1598–1601.

G. Miller, G. Heise, and W. Lichten (1951), 'The intelligibility of speech as a function of the text of the test materials.', *Journal of Experimental Psychology*, 41, pp. 329–355.

R. K. Moore (2003), 'A comparison of the data requirements of automatic speech recognition systems and human listeners', in *EUROSPEECH-2003*, pp. 2581–2584.

K. Murphy (2002), *Dynamic Bayesian Networks: Representation, Inference and Learning*, Ph.D. thesis, University of California, Berkeley.

K. P. Murphy (2001), 'The Bayes net toolbox for Matlab', *Computing Science and Statistics*, 33, pp. 331–350.

K. P. Murphy and Y. Weiss (2001), 'The factored frontier algorithm for approximate inference in dbns', in *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pp. 378–385, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

R. E. Neapolitan (1990), *Probabilistic Reasoning in Expert Systems - Theory and algorithms*, John Wiley & Sons, Inc.

H. Ney, U. Essen, and R. Kneser (1994), 'On structuring probabilistic dependencies in stochastic language modelling', *Computer Speech and Language*, 8, pp. 1–38.

L. Nguyen, X. Guo, R. Schwartz, and J. Makhoul (2002), 'Japanese broadcast news transcription', in *ICSLP-2002*, pp. 1749–1752.

T. Niesler, E. Whittaker, and P. Woodland (1998), 'Comparison of part-of-speech and automatically derived category-based language models for speech recognition', in *ICASSP '98*, Seattle, URL `citeseer.ist.psu.edu/niesler98comparison.html`.

T. Niesler and P. Woodland (1996), 'Combination of word-based and category-based language models', in *Proc. ICSLP '96*, vol. 1, pp. 220–223, Philadelphia, PA, URL `citeseer.ist.psu.edu/niesler96combination.html`.

N. Oostdijk, W. Goedertier, F. V. Eynde, L. Boves, J. Martens, M. Moortgat, and H. Baayen (2002), 'Experiences from the Spoken Dutch Corpus project', in M. G. R. C. P. S. Araujo (ed.), *Proceedings of the third International Conference on Language Resources and Evaluation*, pp. 340–347.

L. Osterhout, M. D. Allen, J. Mclaughlin, and K. Inoue (2002), 'Brain potentials elicited by prose-embedded linguistic anomalies', *Memory & Cognition*, 30(8), pp. 1304–1312.

S. Oviatt, G. Levow, M. MacEachern, and K. Kuhn (1996), 'Modeling hyperarticulate speech during human-computer error resolution', in *Proc. ICSLP '96*, vol. 2, pp. 801–804, Philadelphia, PA, URL `citeseer.ist.psu.edu/432560.html`.

J. Park and H. Ko (2006), 'A new state-dependent phonetic tied-mixture model with head-body-tail structured hmm for real-time continuous phoneme recognition system', in *Interspeech 2006*, pp. 1583–1586.

J. Pearl (1988), *Probabilistic Reasoning in Intelligent Systems - Networks of Plausible Inference*, Morgan Kaufmann Publishers, Inc.

K. Pellom, B.; Hacioglu (2003), 'Recent improvements in the cu sonic asr system for noisy speech: the spine task', in *Proceedings of ICASSP '03*, vol. 1, pp. I–4–I–7 vol.1.

F. Pereira, M. D. Riley, and R. Sproat (1994), 'Weighted rational transductions and their applications to human language processing', in *ARPA Human Language Technology Workshop*, pp. 262–267, Morgan Kaufmann, Plainsboro, NJ.

C. van Petten, S. Coulson, S. Rubin, E. Plante, and M. Parks (1999), 'Time course of word identification and semantic integration in spoken language', *Journal of Experimental Psychology: Learning, Memory and Cognition*, 25, pp. 394–417.

A. Pfeffer (2001), 'Sufficiency, separability and temporal probabilistic models', in *UAI*.

M. Phillips (2006), 'Plenary talk: Creating speech interfaces for mass market applications', in *Ninth International Conference on Spoken Language Processing (Interspeech 2006)*.

S. D. Pietra, V. J. D. Pietra, and J. D. Lafferty (1997), 'Inducing features of random fields', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4), pp. 380–393, URL `citeseer.ist.psu.edu/article/pietra97inducing.html`.

R. Plamondon and S. Srihari (2000), 'Online and off-line handwriting recognition: a comprehensive survey', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22, pp. 63–84.

L. Rabiner and B. H. Juang (1993), *Fundamentals of Speech Recognition*, Prentice Hall, Englewood Cliffs, N.J.

P. Ramesh and J. G. Wilpon (1992), 'Modeling state durations in hidden Markov models for automatic speech recognition', *Proceedings of* ICASSP, 1, pp. 381–384.

A. Ratnaparkhi (1997), 'A simple introduction to maximum entropy models for natural language processing', Tech. rep., Institute for Research in Cognitive Science, University of Pennsylvania, URL `citeseer.ist.psu.edu/128751.html`.

P. Rayson and R. Garside (2000), 'Comparing corpora using frequency profiling', in *proceedings of the workshop on Comparing Corpora*, pp. 1–6, URL `citeseer.ist.psu.edu/rayson00comparing.html`.

P. Rayson, G. Leech, and M. Hodges (1997), 'Social differentiation in the use of English vocabulary: some analyses of the conversational component of the British National Corpus', *International Journal of Corpus Linguistics*, 2(1), pp. 133 – 152.

R. Reddy (2006), 'Plenary talk: Speech recognition: The unfinished agenda', in *Ninth International Conference on Spoken Language Processing (Interspeech 2006)*.

A. Rietveld and V. van Heuven (1997), *Algemene fonetiek*, Coutinho.

B. Roark (2001), 'Probabilistic top-down parsing and language modeling', *Computational Linguistics*, 27(2), pp. 249–276, URL `citeseer.ist.psu.edu/roark04probabilistic.html`.

B. Roark and M. Johnson (1999), 'Efficient probabilistic top-down and left-corner parsing', in *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics (ACL '99)*, pp. 421–428, URL `http://acl.ldc.upenn.edu/P/P99/P99-1054.pdf`.

R. Rosenfeld (1994), *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*, Ph.D. thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, URL `citeseer.ist.psu.edu/rosenfeld94adaptive.html`.

R. Rosenfeld (2000), 'Two decades of statistical language modeling: Where do we go from here', in *Proceedings of the IEEE*, vol. 88, pp. 1270–1278, URL `http://www.cs.cmu.edu/~roni/papers/survey-slm-IEEE-PROC-0004.pdf`.

L. J. M. Rothkrantz, R. J. van Vark, A. Peters, and A. C. Andeweg (2000), 'Dialog control in the ALPARON system', in *Text Speech and Dialogue*.

L. J. M. Rothkrantz, P. Wiggers, J. W. A. van Wees, and R. J. van Vark (2004), 'Voice stress analysis', in *Lecture Notes in Artificial Intelligence 3206: Text, Speech and Dialogue*, pp. 449–456, Springer, Berlin-Heidelberg-New York.

M. J. Russell and R. K. Moore (1985), 'Explicit modelling of state occupancy in hidden Markov models for automatic speech recognition', *Proceedings of* ICASSP, 10, pp. 5–8.

S. Russell and P. Norvig (1995), *Artificial Intelligence: A Modern Approach*, Prentice Hall, Englewood Cliffs, NJ.

S. Sakti, K. Markov, and S. Nakamura (2006), 'The use of Bayesian network for incorporating accent, gender and wide-context dependency information', in *Interspeech 2006*, pp. 1563–1566.

C. Samuelsson and W. Reichl (1999), 'A class-based language model for large vocabulary speech recognition extracted from part-of-speech statistics', in *IEEE ICASSP'99*, pp. 537–540.

I. Schuurman, M. Schouppe, H. Hoekstra, and T. van der Wouden (2003), 'CGN, an annotated corpus of spoken Dutch', in *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*, Budapest, Hungary.

S. Seneff (1992), 'Tina: A natural language system for spoken language applications', *Computational Linguistics*, 18(1), pp. 61–86.

K. Seymore, S. Chen, and R. Rosenfeld (1998), 'Nonlinear interpolation of topic models for language model adaptation', in *ICSLP-98*, vol. 6, pp. 2503–2506, Sydney.

K. Seymore and R. Rosenfeld (1997), 'Using story topics for language model adaptation', in *Proc. Eurospeech '97*, pp. 1987–1990, Rhodes, Greece, URL `citeseer.ist.psu.edu/seymore97using.html`.

G. R. Shafer and P. P. Shenoy (1990), 'Probability propagation', *Annals of Mathematics and Artificial Intelligence*, 2, pp. 327–351.

C. E. Shannon (1951), 'Prediction and entropy of printed English', *Bell System Technical Journal*, 30, pp. 50–64.

P. P. Shenoy and G. Shafer (1990), 'Axioms for probability and belief-function propagation', in *Readings in uncertain reasoning*, pp. 575–610, Morgan Kaufmann, San Francisco, CA, USA.

R. N. V. Sitaram and T. Sreenivas (1997), 'Incorporating phonetic properties in hidden Markov models for speech recognition', *Acoustical Society of America Journal*, 102, pp. 1149–1158.

M. Siu and M. Ostendorf (2000), 'Variable n-grams and extensions for conversational speech language modeling', *IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING*, 8(1), pp. 63–75.

R. Srinivasan (2002), *Importance sampling - Applications in communications and detection*, Springer-Verlag, Berlin.

G. Stemmer, S. Steidl, E. Nöth, H. Nieman, and A. Batliner (2002), 'Comparison and combination of confidence measures', in *Proceedings of Text Speech and Dialogue 2002*, pp. 181–188.

A. Stolcke (1995), 'An efficient probabilistic context-free parsing algorithm that computes prefix probabilities', *Computational Linguistics*, 21(2), pp. 165–202.

A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. V. Ess-Dykema, and M. Meteer (2000), 'Dialogue act modeling for automatic tagging and recognition of conversational speech', *Computer Linguistics*, 26(3), pp. 339–373, URL `citeseer.ist.psu.edu/stolcke00dialogue.html`.

A. Stolcke, E. Shriberg, R. Bates, N. Coccaro, D. Jurafsky, R. Martin, M. Meteer, K. Ries, P. Taylor, and C. Van Ess-Dykema (1998), 'Dialog act modeling for conversational speech', in J. Chu-Carroll and N. Green (eds.), *Applying Machine*

Learning to Discourse Processing. Papers from the 1998 AAAI Spring Symposium. Tech. rep. SS-98-01, pp. 98–105, AAAI Press, Stanford, CA.

J. Stroop (1990), 'Towards the end of the standard language in the Netherlands', in J. Leuvensteijn van and J. Berns (eds.), *Dialect and Standard Language in the English, Dutch, German, Norwegian Language Areas. Proceedings of he Colloquium 'Dialect and the Standard Language'*, pp. 162–177, Amsterdam.

M. Tachibana, T. Nose, J. Yamagishi, and T. Kobayashi (2006), 'A technique for controlling voice quality of synthetic speech using multiple regression HSMM', in *Interspeech 2006*, pp. 2438–2441.

M. K. Tanenhaus, M. J. Spivey-Knowlton, K. Eberhard, and J. Sedivy (1995), 'Integration of visual and linguistic information in spoken language comprehension', *Science*, 268(5217), pp. 1632 – 1634.

G. Tur, J. Wright, A. Gorin, G. Riccardi, and D. Hakkani-Tür (2002), 'Improving spoken language understanding using word confusion networks', in *Proceedings of ICSLP 2002*.

J. P. Ueberla (1995), 'More efficient clustering of n-grams for statistical language modeling', in *EUROSPEECH-1995*, pp. 1257–1260.

D. H. V. Uytsel, F. V. Aelten, and D. V. Compernolle (2001), 'A structured language model based on context-sensitive probabilistic left-corner parsing', in *NAACL '01: Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001*, pp. 1–8, Association for Computational Linguistics, Morristown, NJ, USA.

R. van Vark, J. de Vreught, and L. Rothkrantz (1997), 'Analysis of the OVR log files progress report', Alparon report 97-07, Delft University of Technology.

J. Veronis (2004), 'Hyperlex: lexical cartography for information retrieval', *Computer Speech & Language*, 18(3), pp. 223–252, URL `http://www.sciencedirect.com/science/article/B6WCW-4CKNCXH-1/2/6f989b44a9c26c68c36bf2e67957bea1`.

T. Virtanen (2006), 'Speech recognition using factorial hidden Markov models for separation in the feature space', in *Interspeech 2006*, pp. 89–92.

R. A. Wagner and M. J. Fischer (1974), 'The string-to-string correction problem', *Journal of the Association for Computing Machinery*, 21, pp. 168–173.

W. Wahlster, W. Reithinger, and A. Blocher (2001), 'Smartkom: Multimodal communication with a life-like character', in *Proceedings of Eurospeech 2001*, Aalborg, Denmark.

X. Wang (1997), *Duration modelling in* HMM*-based speech recognition*, Ph.D. thesis, University of Amsterdam.

R. M. Warren (1970), 'Perceptual restoration of missing speech sounds', *Science*, 167, pp. 392–393.

R. M. Warren and R. P. Warren (1970), 'Auditory illusions and confusions', *Scientific American*, 223, pp. 30–36.

P. Wiggers (2001), *Hidden Markov models for automatic speech recognition and their multimodal applications*, Master's thesis, Delft University of Technology, Delft.

P. Wiggers and L. J. M. Rothkrantz (2002), 'Integration of speech recognition and automatic lip-reading', in *Lecture Notes in Artificial Intelligence 2448, Text, Speech and Dialogues*, pp. 205–212, Springer, Berlin-Heidelberg-New York.

P. Wiggers and L. J. M. Rothkrantz (2003*a*), 'Improving speech recognition by utilizing domain knowledge and confidence measures', in *Lecture Notes in Artificial Intelligence 2807: Text, Speech and Dialogues 2003*.

P. Wiggers and L. J. M. Rothkrantz (2003*b*), 'Using confidence measures and domain knowledge to improve speech recogntion', in *Proceedings of Eurospeech 2003*, Geneva, Switzerland.

P. Wiggers and L. J. M. Rothkrantz (2006), 'Dynamic Bayesian networks for language modeling', in *Lecture Notes in Artificial Intelligence 4188: Text, Speech and Dialogue 2006*.

P. Wiggers and L. J. M. Rothkrantz (2007*a*), 'Exploratory analysis of word use and sentence length in the Spoken Dutch Corpus', in V. Matousek and P. Mautner (eds.), *Lecture notes in Artificial Intelligence 4629: Text, Speech and Dialogue 2007*.

P. Wiggers and L. J. M. Rothkrantz (2007*b*), 'Exploring the influence of speaker characteristics on word use in a corpus of spoken language using a data mining approach', in *XII International Conference Speech and Computer (SPECOM'2007)*.

P. Wiggers, J. C. Wojdeł, and L. J. M. Rothkrantz (2002*a*), 'Development of a speech recognizer for the Dutch language', in *Proceedings of 7th annual scientific conference on web technology, new media, communications and telematics theory, methods, tools and applications (*EUROMEDIA*)*, pp. 133–138.

P. Wiggers, J. C. Wojdeł, and L. J. M. Rothkrantz (2002*b*), 'Medium vocabulary audio-visual speech recognition', in *the International Conference on Spoken Language Processing (ICSLP) 2002*, Denver, USA.

A. Wojdeł (2005), *Knowledge Driven Facial Modelling*, Ph.D. thesis, Delft University of Technology.

J. C. Wojdeł (2003), *Automatic Lipreading in the Dutch Language*, Ph.D. thesis, Delft University of Technology, Delft.

J. C. Wojdeł and L. J. M. Rothkrantz (2001), 'Robust video processing for lipreading applications', in *EUROMEDIA'2001*, pp. 195–199, Valencia, Spain.

J. C. Wojdeł, P. Wiggers, and L. J. M. Rothkrantz (2002), 'An audio-visual corpus for multimodal speech recognition in Dutch language', in *the International Conference on Spoken Language Processing (ICSLP) 2002*.

P. C. Woodland, J. Odell, and S. J. Young (1994), 'Large vocabulary continuous speech recognition using HTK', in *Proceedings of ICASSP 1994*, pp. 125–128.

S. Young (1995), 'Large vocabulary continuous speech recognition: A review', in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 3–28, Snowbird, Utah, URL `citeseer.ist.psu.edu/article/young96large.html`.

H. Zen, Y. Nankaku, K. Tokuda, and T. Kitamura (2006), 'Speaker adaptation of trajectory HMMs using feature-space MLLR', in *Interspeech 2006*, pp. 1141–1144.

N. L. Zhang and D. Poole (1996), 'Exploiting causal independence in Bayesian network inference', *Journal of Artificial Intelligence Research*, 5, pp. 301–328, URL `citeseer.ist.psu.edu/article/zhang96exploiting.html`.

R. Zhang and A. I. Rudnicky (2002), 'Improve latent semantic analysis based language model by integrating multiple level knowledge', in *Proc. of ICSLP 2002, Denver, Colorado*, pp. 893–896.

G. Zweig (1998), *Speech Recognition with Dynamic Bayesian Networks*, Ph.D. thesis, Computer Science Division, University of California at Berkeley.

G. Zweig and M. Padmanabhan (2000), 'Exact alpha-beta computation in logarithmic space with application to map word graph construction', in *Proceedings of ICSLP'00*, Beijing, China, URL `citeseer.ist.psu.edu/zweig00exact.html`.

Curriculum Vitae

Pascal Wiggers was born on July 29, 1977 in Den Helder, The Netherlands.
In 1991 he moved to Braunschweig, Germany where he attended the Martino-Katharineum, the same high school once attended by Carl Friedrich Gauss. In 1996, he graduated at the Jan Arentsz College in Alkmaar, The Netherlands. In the same year, he registered as a student of the Faculty of Technical Mathematics and Informatics of Delft University of technology.

In 2001 he finished his master's thesis work 'Hidden Markov models for automatic speech recognition and their multimodal applications' under supervision of Leon Rothkrantz. He graduated with honours in August 2001 and was awarded as his year's best graduate in Computer Science in 2002.

From September 2001 to April 2006 he worked as a Ph.D student in the Man-Machine Interaction Group of the Department of Electrical Engineering, Mathematics and Computer Science of Delft University of Technology. Since August 2006 he has been working as a Lecturer in the same group.

Samenvatting

Het modelleren van context voor automatisch spraakherkenning,
door Pascal Wiggers.

Het proefschrift begint met een overzicht van mogelijke toepassingen van automatische spraakherkenning. De stand van de techniek en het onderzoek wordt beschreven, waarbij wordt aangegeven dat, hoewel automatische spraakherkenning een enorme ontwikkeling heeft doorgemaakt, de meeste toepassingen nog niet haalbaar zijn. Vervolgens wordt vanuit een drietal invalshoeken onderzocht hoe automatische spraakherkenning verbeterd kan worden. Ten eerste worden fouten, gemaakt door een herkenner, onder de loep genomen. Vervolgens wordt expliciet gemaakt wat spraakherkenning moeilijk maakt en als derde wordt gekeken naar menselijke spraakherkenning. Uit deze analyse volgt de conclusie dat spraakherkenners te kampen hebben met ambiguïteit veroorzaakt door gebrek aan informatie. De in de praktijk gekozen oplossing is veelal het domein waarvoor de herkenner gebruikt wordt zodanig te beperken dat deze ambiguïteiten grotendeels verdwijnen. In dit proefschrift wordt een alternatieve oplossing voorgesteld, namelijk context expliciet mee te nemen in spraakherkenningsmodellen. Dit heeft het voordeel dat een systeem zich kan aanpassen aan veranderende gebruiksomstandigheden zoals de stem van een nieuwe spreker, een overgang van gespreksonderwerp, of achtergrondgeluiden.

De gedachte dat spraakherkenning verbeterd kan worden door toevoeging van kennis is niet nieuw. Toch beperkt onderzoek in deze richting zich vaak tot kleine uitbreidingen op het standaardmodel. De oorzaak lijkt te liggen in de beperkingen van de huidige wiskundige modellen voor spraakherkenning. Deze zijn ongeschikt om context mee te modelleren. Aan het eind van hoofdstuk 1 wordt derhalve gesteld dat om contextgevoelige modellen te realiseren een krachtiger modelleertechniek nodig is.

De huidige aanpak van spraakherkenning, gebaseerd op hidden Markov modellen wordt in hoofdstukken 2 en 3 besproken. Hoofdstuk 2 behandeld akoestische modellen die onderdeel uitmaken van een spraakherkenner, hoofdstuk 3 taalmodellen die aangeven hoe waarschijnlijk het is dat een herkende reeks woorden een zin vormt in de te herkennen taal. Variaties op de standaardmodellen en eerdere pogingen om meer kennis in deze modellen mee te nemen worden behandeld.

In hoofdstuk 4 wordt het begrip context nader gedefinieerd als kennis van de

gebruiker, kennis van de conversatie en kennis van de wereld. Voor ieder van deze categorieën wordt vervolgens mede aan hand van (socio)linguïstische en psychologische theorieën onderzocht hoe deze van belang kunnen zijn voor spraakherkenning. Verder wordt een kort overzicht van taalkundige begrippen en constructies gegeven die van belang kunnen zijn om contextinformatie te koppelen aan de woorden en klanken die herkend worden.

Hoofdstukken 5 en 6 beschrijven empirisch onderzoek naar de in hoofdstuk 4 op theoretische basis bepaalde invloeden van context op spraak en taal. Hierbij is de vraag niet alleen of dergelijke invloeden gevonden kunnen worden, maar vooral ook of deze gevonden kunnen worden in het soort gegevens dat gebruikt wordt bij de ontwikkeling van spraakherkenners en of deze kennis nuttig is voor spraakherkenning.

Hoofdstuk 5 presenteert de resultaten van een statistische analyse van de invloed van het conversatietype en sprekerskenmerken op woordgebruik en zinslengte in een corpus van ruim 8 miljoen gesproken woorden. Het blijkt dat de verdeling van zinslengten sterk verschilt voor verschillende conversatietypen zoals spontane spraak, debatten, nieuwsberichten en voorgelezen teksten. Opvallender is dat er een sterk verband is tussen frequentie van gebruik van lidwoorden en persoonlijke voornaamwoorden en het conversatietype. Het woord 'ik' is bijvoorbeeld prominent aanwezig in spontane spraak, terwijl 'je', 'jij' en 'u' veel frequenter zijn in debatten, discussies en lessen en de derde persoon vooral gebruikt wordt in verhalen. De analyse toont eveneens aan dat woordgebruik verschilt voor sprekers van verschillende leeftijden, van verschillend geslacht en voor sprekers met verschillende opleidingsniveaus.

In hoofdstuk 6 wordt een tweetal casestudies besproken die in het kader van dit onderzoek zijn uitgevoerd om te bepalen of, wanneer en hoe context informatie nuttig is voor spraakherkenning. De eerste casestudy betreft toevoegen van informatie uit een automatische liplezer aan een spraakherkenner. Dit blijkt in omgevingen met veel achtergrondruis de herkenning te verbeteren. De tweede casestudy beschrijft een dialoogsysteem voor treinreisinformatie dat gebruik maakt van reisfrequenties om de verwachting van de stationsnamen die een gebruiker noemt bij te stellen. Voor een spraakherkenner is bijvoorbeeld het onderscheid tussen de namen 'Middelburg' en 'Tilburg' niet altijd duidelijk. Analyse van reisfrequenties toont echter aan dat bijvoorbeeld veel vaker van Breda naar Tilburg dan naar Middelburg gereisd wordt. Als het vertrekstation Breda herkend wordt, kan dit dus helpen bij het bepalen van het aankomststation. Deze aanpak blijkt inderdaad het herkenningsresultaat ten goede te komen, met name als deze informatie al vroegtijdig in het herkenningsproces gebruikt wordt.

De casestudies en de data analyse geven aan dat spraakherkenning baat kan hebben bij context informatie. Integreren van extra informatie in spraakherkenningsmodellen is echter moeilijk. De voor de casestudies gevolgde aanpak om modellen en bijbehorende algoritmen te ontwikkelen om specifieke informatie te kunnen toevoegen is algemene praktijk in spraakherkenningsonderzoek. Deze verbeterde modellen leiden vaak tot kleine verbeteringen in de herkenning. Men mag aannemen dat veel van deze verbeteringen elkaar aanvullen en gezamenlijk een aanzienlijke herkenningsverbetering kunnen realiseren. Helaas zijn de gespecialiseerde, complexe modellen

moeilijk te combineren, zodat deze stap in de praktijk uitblijft.

In hoofdstuk 7 wordt de noodzaak van een krachtiger modelleerparadigma bepleit en worden aan hand van de bevindingen uit voorgaande hoofdstukken eisen aan dit paradigma opgesteld. Verschillende paradigmas, die hun bruikbaarheid in automatische spraak- en taalverwerking hebben bewezen, worden aan deze eisen getoetst. De keuze valt op dynamische Bayesiaanse netwerken, een techniek die kan worden gezien als een generalisatie van de hidden Markov modellen en n-gram modellen die gebruikelijk zijn in spraakherkenning.

In hoofdstuk 8 wordt getoond hoe bestaande modellen voor spraakherkenning in termen van dynamische Bayesiaanse netwerken kunnen worden geformuleerd en vervolgens zonder ontwikkeling van nieuwe algoritmen kunnen worden gecombineerd. Daarnaast worden een aantal nieuwe modellen, waarin context informatie wordt meegenomen, opgesteld. In het bijzonder wordt in hoofdstuk 10 een nieuw taalmodel gedefinieerd dat het onderwerp van gesprek in de berekening van de waarschijnlijkheid van zinshypothesen meeneemt. In dit model wordt aan hand van de woorden in een gesprek de kans op het onderwerp bepaald. Het onderwerp maakt vervolgens gerelateerde woorden waarschijnlijker en ongerelateerde woorden onwaarschijnlijker. Deze cyclus van bijstellen van de waarschijnlijkheden van onderwerpen aan hand van al verwerkte woorden en het bijstellen van de waarschijnlijkheden van woorden aan hand van het onderwerp herhaalt zich voor elk nieuw woord. Het gevolg hiervan is dat het model, en daarmee een spraakherkenner, de voorkeur geeft aan zinshypothesen die inhoudelijke samenhang vertonen. Experimenten tonen aan dat dit model beter werkt dan standaard taalmodellen.

Spraakherkenning is een rekenintensief proces. Een groot aantal mogelijke klanken zinshypothesen wordt door de computer vergeleken. Hoofdstuk 9 bespreekt algoritmes en datastructuren die spraakherkenning met dynamische Bayesiaanse netwerken mogelijk maken. Deze technieken combineren ideeën uit de literatuur over spraakherkenning, hidden Markov modellen en Bayesiaanse netwerken. Nieuwe bijdragen in dit proefschrift zijn: een efficient algoritme voor het rekenen met kanstabellen, het gebruik van lazy evaluation voor kanstabellen onafhankelijk van de inferentiemethode, algoritmen voor het rekenen met boomvormige kanstabellen en een uitbreiding van dynamische Bayesiaanse netwerken die k-de orde Markov relaties toestaat en het mogelijk maakt de netwerkstructuur te laten variëren voor verschillende tijdstappen.