DEGREE PROJECT IN MATHEMATICS,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2016

Speech to Text for Swedish using KALDI

EMELIE KULLMANN

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

Speech to Text for Swedish using KALDI

E M E L I E K U L L M A N N

Master's Thesis in Optimization and Systems Theory (30 ECTS credits)
Master Programme in Applied and Computational Mathematics
(120 credits)
Royal Institute of Technology year 2016
Supervisor at Swedish Radio: Paul Nygren
Supervisor at KTH: Johan Karlsson
Examiner: Johan Karlsson

TRITA-MAT-E 2016:42
ISRN-KTH/MAT/E--16/42--SE

Royal Institute of Technology
School of Engineering Sciences

KTH SCI
SE-100 44 Stockholm, Sweden

URL: www.kth.se/sci

Acknowledgement

I am very grateful to the people behind KALDI toolkit for being very responsive and helpful. Especially Daniel Povey and Jan Trmal, who have quickly answered any of my questions and helped me overcome problems along the way. I would also like to thank the author of the Danish Sprakbanken recipe, Andreas Søeborg Kirkedal, whose work is the fundamentals on which the Swedish script stands on.

To SR department of development, and in particular my supervisor Paul Nygren with colleague Christopher Bustad, for assistance and encouragement along the way.

To Johan Karlsson, for his help and guidance.

And lastly, to my family and friends for having put up with me through stressful times and supported me throughout this degree project.

Abstract

The field of speech recognition has during the last decade left the research stage and found its way in to the public market. Most computers and mobile phones sold today support dictation and transcription in a number of chosen languages. Swedish is often not one of them. In this thesis, which is executed on behalf of the Swedish Radio, an Automatic Speech Recognition model for Swedish is trained and the performance evaluated. The model is built using the open source toolkit Kaldi. Two approaches of training the acoustic part of the model is investigated. Firstly, using Hidden Markov Model and Gaussian Mixture Models and secondly, using Hidden Markov Models and Deep Neural Networks. The later approach using deep neural networks is found to achieve a better performance in terms of Word Error Rate.

Keywords: *Atuomatic Speech Recognition, Kaldi, Hidden Markov Model, Gaussian Mixture Model, Deep Neural Network*

Referat

Tal till Text, Utvecklandet av en Svensk Taligenkänningsmodell i KALDI

De senaste åren har olika tillämpningar inom människa-dator interaktion och främst taligenkänning hittat sig ut på den allmänna marknaden. Många system och tekniska produkter stöder idag tjänsterna att transkribera tal och diktera text. Detta gäller dock främst de större språken och sällan finns samma stöd för mindre språk som exemplevis svenskan. I detta examensprojekt har en model för taligenkänning på svenska utvecklas. Det är genomfört på uppdrag av Sveriges Radio som skulle ha stor nytta av en fungerande taligenkänningsmodell på svenska. Modellen är utvecklad i ramverket Kaldi. Två tillvägagångssätt för den akustiska träningen av modellen är implementerade och prestandan för dessa två är evaluerade och jämförda. Först tränas en modell med användningen av Hidden Markov Models och Gaussian Mixture Models och slutligen en modell där Hidden Markov Models och Deep Neural Networks används, det visar sig att den senare uppnår ett bättre resultat i form av måttet Word Error Rate.

Nyckelord: *Taligenkänning, Kaldi, Hidden Markov Model, Gaussian Mixture Models, Deep Neural Networks*

Contents

Abbreviations

$\Delta + \Delta\Delta$ delta + delta-delta.

AM Acoustic Model.

ASR Automatic Speech recognition.

CMVN Cepstral Mean and Variance Normalization.

DCT Discrete Cosine Transform.

DFT Discrete Fourier Transform.

DNN Deep Neural Networks.

EM Expectation-Maximization.

GMM Gaussian Mixture Model.

HMM Hidden Markov Models.

LDA Linear Discriminant Analysis.

LM Language Model.

LVCSR Large Vocabulary Continuous Speech Recognition.

MFCC Mel-Frequency Cepstral Coefficient.

MLLT Maximum Likelihood Linear Transform.

SAT Speaker Adaptive Training.

WER Word Error Rate.

WFSA Weighted Finite State Acceptors.

WFST Weighted Finite State Transducer.

1. Introduction

Human-Computer interaction is a field continuously growing and developing. Spoken dialogue is perhaps the most intuitive example of such interaction. The ability to voice command your computer or mobile device was some decades ago a utopian fantasy that has now become reality. Applications like Apple's Siri and Microsoft's Cortana, to name a few, are examples of products within the field of *Large Vocabulary Continuous Speech Recognition* (LVCSR) that have sprung from decades of research on the complex problem of trying to mathematically model human speech.

When the Swedish Radio some years ago digitalized their archives an extensive work was put into manually transferring the metadata present on the physical recordings to their corresponding digital entry in the archive. Despite the effort made, they faced the problem of having a huge number of programs in the digital archive with insufficient or no information about content. The idea sprung to have this unsearchable audio material undergo an automatic process in which *keywords* would be obtained as to *tag* the objects adequately. A first step in obtaining a satisfactory tagging of the material is to have the material being transformed from audio to text. After transcribing the object a *keyword finder* would scan the text and obtain the desired tags.

2015 a master's thesis was executed at SR with the intention of investigating how similar orginizations had solved or are working on solving the problem of transcribing audio. There are services that already offer transcription of audio, but the problem with most applications in the field of *Automatic Speech recognition* (ASR) is that of multiple language support, and often Swedish is considered too small a language to be included in the list of languages supported. Based on the research presented by Jansson in her thesis[10], it was clear to SR that to solve the process of transcribing the audio material, a complete model for automatic speech recognition in Swedish had to be developed and tested. One toolkit for building a speech recognizer that stood out in the investigations carried out by Jansson[10] is the open source toolkit Kaldi, see Section 2.2.

The knowledge of the need of an ASR model in Swedish and the possibility of development that Kaldi provides is the point of departure for this project.

1.1 The project overview and goals

The focus for this thesis has been the development and training of an ASR-model for the Swedish language. The project consists of the following parts

1. Covering the theory behind the training of an ASR-model.
2. Obtaining transcribed material to be used in training.
3. Training an ASR-model on the Swedish training material.
4. Tuning the model to achieve a satisfactory *word error rate* (WER) 2.1.5.

This is also the basic outline of the content of the report.

Some restrictions were made as

1. Only already implemented algorithms and approaches will be compared and tested.
2. The tuning of the model took place to the extent the time limit of the project allowed.

Two different approaches of carrying out the acoustic training will be compared. *Gaussian Mixture Models* with *Hidden Markov Model* (GMM-HMM) and *Deep Neural Network* with Hidden Markov Models (DNN-HMM). The robustness of the two models will also be tested on audio material from SR. This in order to compare the performance of the models on real data that differs a lot from the data used in training.

1.2 Applications

In this section some possible applications and extensions of a functioning ASR-model at SR will be presented.

1.2.1 Tagging of Archives

The SR archive consists of roughly 600 000 objects - stored SR programs - and it is expected to grow with almost 60 000 new entries every year. The problem with the archive today is that it consists of large amounts of unsearchable objects. When a

model for obtaining keywords for these unsearchable objects is fully implemented, all objects in the archive will be tagged with adequate meta data and the accessibility of the archive would be improved immensely. Not only would old entries in the archive have a consistent and satisfactory tagging but also new entries would undergo the same treatment. This would ensure a consistent tagging and also save time for producers and program hosts, who today provide each new entry in the archive with meta data manually.

1.2.2 Editing

A model for automatic speech recognition contains information about when a given word or sentence is being uttered in the audio, this is referred to as the time-word alignment. Extracting this information from the model could provide program makers and reporters with a powerful editing tool. A reporter returning to the office after making a single interview might have hours of audio material to go through. The time-word alignment editing tool would in practise give the opportunity for the reporter to edit the program on a text based manner. The reporter would mark out the sequences and sections relevant to the program from the transcribed version of the interview, the time-word editing tool would then provide information about where those particular sequences and sections are being uttered in the audio file. This information would be given to the reporter in terms of start and end times for each section of text. The same procedure would be applied when reusing parts from old programs in new productions. In both these examples the editing of a program would turn into a lot smoother process.

1.2.3 New Features for the SR-application

The ASR model can also be used in real-time applications. For example, support for voice command in the SR-application. The possibility for the user to subscribe to topics could also be implemented. Say that a user drives on the same road every day to work. Subscribing on that particular road on the traffic news would then be in his/her interest. Every time that particular road is mentioned in the traffic news, the user would receive a notification that the word has been said and could turn on the radio or open the SR-application to listen to the full program.

2. Background

Section 2.1 describes the basics of speech recognition. The preprocessing in terms of Feature Extraction, the *Acoustic Model* (AM) and *Language Model* (LM) and the process of Speech Decoding is introduced. Lastly, a measurement for evaluating the quality of an automatic speech recognition model is given. Section 2.2 will explain the Kaldi toolkit and the procedures that are specific for acoustic training and speech decoding in Kaldi.

2.1 Automatic Speech Recognition

The purpose of an ASR model is to find the most likely word sequence given a speech input. The process of finding the most probable word sequence is referred to as *speech decoding*. More formally, the decoding process can be defined as finding the most probable word sequence $\boldsymbol{w}^*$ given a set of acoustic features $\boldsymbol{X}$,

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}}\{P(w|\boldsymbol{X})\}. \tag{2.1}$$

Computing $P(\boldsymbol{w}|\boldsymbol{X})$ directly is difficult so at this stage Bayes' Rule is applied and the expression is transformed to

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}}\{\frac{p(\boldsymbol{X}|\boldsymbol{w})P(\boldsymbol{w})}{P(\boldsymbol{X})}\} = \arg\max_{\boldsymbol{w}}\{p(\boldsymbol{X}|\boldsymbol{w})P(\boldsymbol{w})\}. \tag{2.2}$$

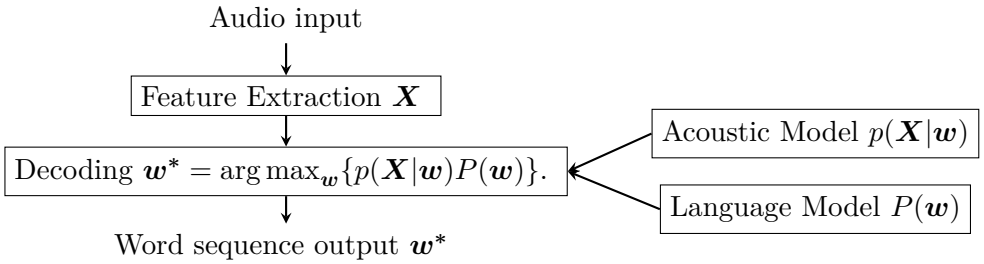

Figure 2.1. The architecture of an ASR model

The most likely word sequence does not depend on the probability of the acoustic features $P(\boldsymbol{X})$ and is eliminated in the last step of Equation 2.2. The task of acoustic modelling is to estimate parameters θ of a model so that $p(\boldsymbol{X}|\boldsymbol{w};\theta)$ is as accurate as possible. The prior $P(\boldsymbol{w})$ is what will be determined by the language model.

The structure of an automatic speech recognizer and the decoding of a audio input is shown in Figure 2.1. First the audio is sampled and processed to obtain the feature vectors $\boldsymbol{X}$ that is the input to the decoder. The acoustic features are computed over sequences of 20-30ms long windows of the audio input. For each window, referred to as a *frame*, a set of feature vectors is obtained. The decoding is performed frame by frame using beam search. The beam search expands the *hypothesis* framewise using information from previous frames when moving forward in time to the next frame. A hypothesis is defined as the acoustic model output, i.e., a probable word sequence. Probabilities for hypotheses are computed using the acoustic model and the language model. Using beam search means that only hypotheses over a certain threshold are kept and continued to be expanded. When the last frame is reached the complete hypothesis corresponding to the highest probability is what will give the word sequence output $\boldsymbol{w}^*$.

2.1.1 Feature Extraction

Feature extraction is one of the most important issues in speech recognition, aiming to solve the dimensionality problem. How can the important information contained in the acoustic data be captured in the most efficient way? One solution is to compute *Mel-Frequency Cepstral Coefficients* (MFCCs). Mel-Frequency Cepstral Coefficients were introduced by Davis and Mermelstein[2] in the 1980's and have been used in speech recognition tasks ever since. The Mel-scale is a perceptual scale of frequencies. It seeks to model the sensitivity of the human ear and it is created based on experiments in which subjects were asked about their perception of a certain frequency, for example by being asked to adjust a tone to be half the pitch of that of a comparison tone.

To calculate the MFCCs the audio is sliced into 20-30 ms long time frames using a window function which is zero everywhere except in a small region. The discretized speech signal is denoted $s_i(n)$, where i ranges over the number of frames and n ranges over the number of samples. The *periodogram* for the speech signal is calculated by taking the modulus square of the *Discrete Fourier Transform* (DFT)

$$P_i(k) = \frac{1}{N}\left|\sum_{n=0}^{N-1} s_i(n)h(n)e^{-j2\pi kn/N}\right|^2 \qquad 0 \leq k < N, \tag{2.3}$$

where $h(n)$ is a N sample long analysis window [13] and N is the size of the DFT [9]. An upper frequency f_M and a lower frequency f_0 is chosen in Hz. Between

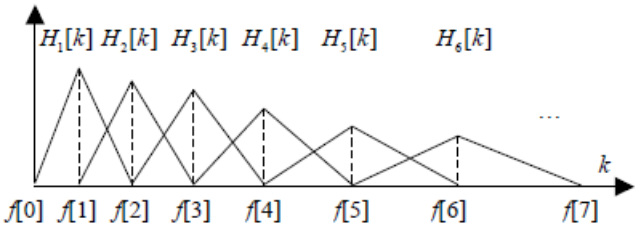

Figure 2.2. Triangular filter for center points, graph from [9]

the upper and lower frequency boundaries are M points uniformly placed in the Mel-scale,

$$f[m] = \left(\frac{N}{F_s}\right) B^{-1}\left(B(f_0) + m\frac{B(f_M) - B(f_0)}{M+1}\right). \tag{2.4}$$

F_s is the sampeling frequency in Hz, $B(f)$ is the transform from Herz scale to Mel-scale and $B^{-1}(b)$ is the inverse.

$$B(f) = 1125\ln(1 + f/700). \tag{2.5}$$

$$B^{-1}(b) = 700(\exp(b/1125) - 1) \tag{2.6}$$

The human ear captures differences and changes on lower frequencies better than on higher. Therefore in Mel-scale equally placed points in Herz-scale will be denser on the lower frequencies than the higher. A triangular filter $H_m[k]$ is applied for each point M as shown in Figure 2.2, and the so called *mel-bins* are obtained.

The output of each filter gives the average spectrum around each center frequency, thus giving an indication of how much energy there is in each triangular mel-bin.

The two final steps of computing the MFCCs include computing the log and calculating the *Discrete Cosine Transform* (DCT) for each of the M filters [12]

$$S[m] = \ln\left(\sum_{k=0}^{N-1} NP_i(k)H_m[k]\right) \qquad 0 \leq m < M \tag{2.7}$$

$$c[n] = \sum_{m=0}^{M-1} S[m]\cos(\pi n(m + 1/2)/M) \qquad 0 \leq n < M, \tag{2.8}$$

where $P_i(k)$ is given in Equation 2.3 and $H_m[k]$ the triangular filter used. Having performed these steps the result is M calculated Mel-frequency cepstral coefficients. It is shown that no order above the 13th improves the final result of decoding, therefore the first 13 coefficients are kept and the rest are disregarded [9].

Different feature transformations can be applied after this step. In this project three types are relevant and will be introduced next.

Feature Transforms

Cepstral Mean and Variance Normalization (CMVN) [27] seeks to minimize the impact of the difference in variable environments like ambient noise, recording equipment and transmission channels [19]. It is done by subtracting the mean of each coefficient and divide with the standard deviation which efficiently minimizes any stationary noise

$$\hat{x}_t(i) = \frac{x_t(i) - \mu_t(i)}{\sigma_t(i)}, \tag{2.9}$$

where $x_t(i)$ is the ith component of the original feature vector at time t and the mean $\mu_t(i)$ and standard deviation $\sigma_t(i)$ are calculated over some sliding finite window of length N

$$\mu_t(i) = \frac{1}{N} \sum_{n=t-N/2}^{t+N/2-1} x_n(i) \tag{2.10}$$

$$\sigma_t^2(i) = \frac{1}{N} \sum_{n=t-N/2}^{t+N/2-1} (x_n(i) - \mu_t(i))^2. \tag{2.11}$$

The first and second order deltas ($\Delta + \Delta\Delta$) of the MFCCs can be calculated to add dynamic information to the MFCCs. For an acoustic feature vector $\boldsymbol{x}$ the first order deltas are defined as

$$\Delta \mathbf{x}_t = \frac{\sum_{i=1}^{n} w_i(\mathbf{x}_{t+i} - \mathbf{x}_{t-i})}{2\sum_{i=1}^{n} w_i^2}, \tag{2.12}$$

where n is the window width and w_i the regression coefficients.[1] The second order delta parameters are derived in the same fashion [4]

$$\Delta^2 \mathbf{x_t} = \frac{\sum_{i=1}^{n} w_i(\mathbf{\Delta x}_{t+i} - \mathbf{\Delta x}_{t-i})}{2\sum_{i=1}^{n} w_i^2}. \tag{2.13}$$

The combined feature vector becomes

$$\mathbf{x_t} = [\mathbf{x_t} \quad \mathbf{\Delta x_t} \quad \mathbf{\Delta^2 x_t}]. \tag{2.14}$$

An alternative to the $\Delta + \Delta\Delta$ - transform is the *Linear Discriminant Analysis + Maximum Likelihood Linear Transform + Speaker Adaptive Training* (LDA+MLLT+ SAT). With this transform the original feature space is projected to a space with lower dimension using LDA [16]. It has been shown that LDA transform works best when a diagonalizing MLLT is applied afterwards [6]. For a full explanation of the LDA+MLLT transform see Gales[5]. The concept of speaker adaptive training

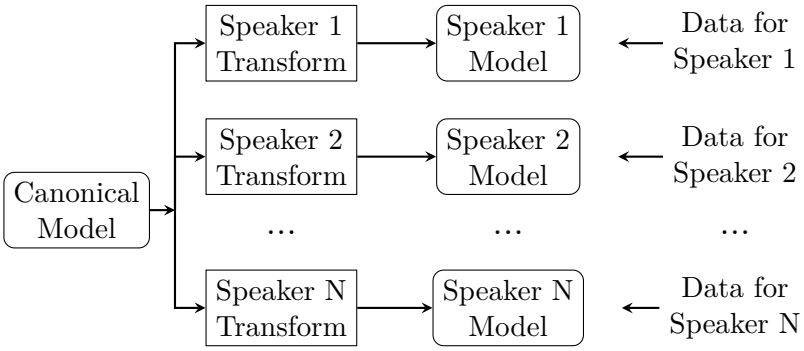

Figure 2.3. Speaker Adaptive Training, illustration from [4]

is illustrated in Figure 2.3 and it can be thought of as training a small individual acoustic model for each individual speaker in the training data.

2.1.2 Acoustic Modelling

The *Acoustic Model* (AM) estimates the likelihood $p(\boldsymbol{X}|\boldsymbol{w};\theta)$. The model's parameters θ is found through training. There is a level of uncertainty in the training due to the exact word - time alignment in an utterance not being known. The *Hidden Markov Model* (HMM) is a popular choice in speech recognition, being able to model this uncertainty between acoustic features and corresponding transcription.

To be able to handle natural speech and large vocabulary speech recognition tasks, the word is not a feasible choice of training unit. The number of words that would have to be known by the model would make the problem of dimensionality too large to handle. Instead the *phone* is introduced and defined as the smallest unit of speech and the unit to be used in training.

The remaining of this section will be dedicated to explaining concepts central to acoustic modelling, beginning with the phone.

The Phone

Each word is composed by a sequence of phones. The Swedish language has approximately 40 different phones. A full map is given in Table 3.2 with corresponding sound. The training utterance's transcription is converted from words to phones using a phonetic dictionary. Below is a short example of two words and their phonetic transcription.

[1]In Kaldi to ensure that the same number of frames is maintained after adding delta and delta–delta parameters, the start and end elements are replicated to fill the regression window[4].

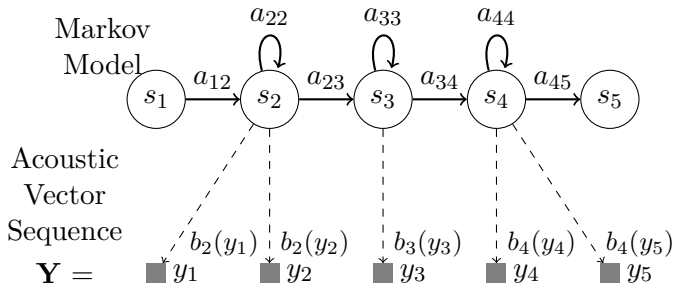

Figure 2.4. HMM-based phone model.

HI	`h a I`
THERE	`D e @`

Better than training on single phones, referred to as *monophones*, is training on *triphones*. A triphone is a sequence of three phones and it captures the context of the single middle phone very efficiently. If there are N base phones then there are N^3 potential triphones. To reduce the dimensionality, acoustically similar triphones are tied together in a process called *state-tying*. In Kaldi the state-tying is performed using *decision trees*, see Section 2.2.1. Each triphone is modelled by a *Hidden Markov Model*.

Hidden Markov Model

The Hidden Markov Model is a statistical model. In speech recognition the hidden Markov model provides a statistical representation of the sound of words. The architecture of an arbitrary HMM in speech recognition is given in Figure 2.4. The Hidden Markov model consists of a chain of states. In a HMM the current state is hidden and only the output from each state can be observed. Each state in the HMM corresponds to one frame in the acoustic input. The model parameters that are estimated in the acoustic training are $\theta = [\{a_{ij}\}, \{b_j()\}]$, where $\{a_{ij}\}$ corresponds to transition probabilities and $\{b_j()\}$ to output observation distributions. The transition probability a_{ij} is the probability of changing from state i to state j. An important feature with the HMM is the self loops a_{ii} which makes the HMM able to model variable length of phones. When making a transition and entering a new stage in the HMM a feature vector is generated using the distribution associated with that particular state in the HMM. The first state and last state in the HMM are called *non emitting* states. For the example in Figure 2.4 s_1 is the entry state and s_5 the exit state. They are used as the entry and exit to the model and simplifies the concatenation of HMMs, phone models, to form words. In this project two possible choices for estimating the output distribution $\{b_j()\}$ will be presented. Firstly, output generated by a *Gaussian Mixture Model* (GMM) and secondly, by a

Deep Neural Network (DNN).

Gaussian Mixture Models

A common choice of output distribution is a mixture of Gaussians. Speaker, accent and gender differences tend to create multiple modes in the speech data. Gaussian mixture models are able to describe such multimodality thus making GMMs a powerful tool in speech recognition [3]. The expression for the output observation becomes,

$$b_j(\boldsymbol{x}) = \sum_{m=1}^{M} c_{jm}\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}^{(jm)}, \boldsymbol{\Sigma}^{(jm)}), \tag{2.15}$$

where c_{jm} is the prior probability for component m of state $\mathbf{s}_j$ and $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}^{(jm)}, \boldsymbol{\Sigma}^{(jm)})$ is the Gaussian (or normal) disitribution with parameters $\boldsymbol{\mu}^{(jm)}$ & $\boldsymbol{\Sigma}^{(jm)}$ corresponding to the mean and covariance of state $\mathbf{s}_j$ respectively. The prior probabilities satisfy the probability mass function constraints

$$\sum_{m=1}^{M} c_{jm} = 1, \quad c_{jm} \geq 0. \tag{2.16}$$

If M is set to equal one, a single Gaussian distribution is obtained [4]. In the training of a GMM the aim is to update the mean $\boldsymbol{\mu}^{(jm)}$ and covariance $\boldsymbol{\Sigma}^{(jm)}$.

Deep Neural Networks

An alternative to mixture of Gaussians is to use a deep neural network[8]. A deep neural network is a feed-forward, artificial neural network that has more than one hidden layer between the input layer and the output layer, as illustrated in Figure 2.5. The nodes along the path have weights attached to them and the output at every node is computed by an *activation function* a_{ut}. Typically the input to a node at a layer in the DNN is computed from the layer below by,

$$x_j = b_j + \sum_i y_i w_{ij}, \tag{2.17}$$

where b_j is the bias of unit j, i is an index over units in the layer below and w_{ij} is the weight on a connection to unit j from unit i in the layer below. The output to the above layer is then computed by

$$y_j = a_{ut}(x_j). \tag{2.18}$$

The hidden layers makes the deep neural network able to model non-linear and complex relationships in the data. For multiclass classification, output unit j convert

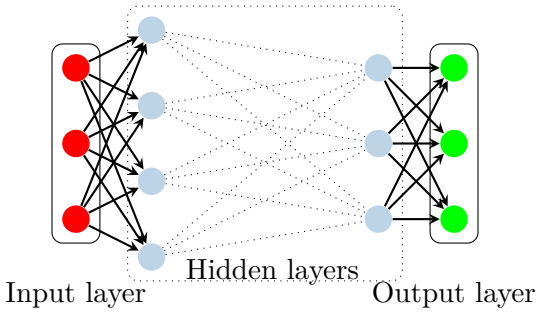

Figure 2.5. A deep neural network

its total input x_j into a probability using a *softmax function*[8]. In Kaldi the function used to estimate the posterior probabilities for the HMM is

$$y_{ut}(s) \triangleq \mathrm{P}(s|\boldsymbol{o}_{ut}) = \frac{\exp\{a_{ut}(s)\}}{\sum_{s'} \exp\{a_{ut}(s')\}}, \tag{2.19}$$

where $\boldsymbol{o}_{ut}$ denotes the observation at time t in utterance u and a_{ut} is the activation function at the output layer corresponding to state s [26].

The objective in training is to optimize an *objective function* and update the weights of the internal nodes based on some information that is passed to the model. In training one important parameter is the *learning rate*. The greater the learning rate, the faster but less accurate the training is.

2.1.3 Language Modelling

$P(\boldsymbol{w})$ is determined by the language model. The language model contains information of the likelihood of words to co-occur. The prior probability $P(\boldsymbol{w})$ for a word sequence $\boldsymbol{w} = w_1, ..., w_K$ is given by

$$P(\boldsymbol{w}) = \prod_{k=1}^{K} P(w_k|w_{k-1}, ..., w_1). \tag{2.20}$$

For large vocabulary speech recognition, the probability of a word is often modelled using the n-gram language model. The n-gram model assumes that the probability of a word to occur is dependent on the n number of words occurring before it. For large vocabulary, n typically ranges between two and four

$$P(\boldsymbol{w}) = \prod_{k=1}^{K} P(w_k|w_{k-1}, w_{k-2}, ..., w_{k-n+1}). \tag{2.21}$$

The model is trained by counting occurrences of word sequences in the training data and estimating probabilities from the maximum likelihood.

$$P(w_k|w_{k-1}, w_{k-2}) \approx \frac{C(w_{k-2}w_{k-1}w_k)}{C(w_{k-2}w_{k-1})}, \tag{2.22}$$

where $C(w_{k-2}w_{k-1}w_k)$ represents the total number of occurrences of the word sequence $w_{k-2}w_{k-1}w_k$ in the data and $C(w_{k-2}w_{k-1})$ the total number of occurrences of $w_{k-2}w_{k-1}$ respectively. A problem with the n-gram model is sparsity of data. This can be solved by pruning, for example by only storing n-grams with counts greater than some threshold, or smoothing [9].

2.1.4 Speech Decoding

Speech decoding is solving the equation

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}}\{P(w|\boldsymbol{X})\}. \tag{2.23}$$

The decoder searches phones sequences which corresponds to words. The phones typically being represented as triphones in the acoustic model and the word hypothesis being limited to the words listed in the phonetic dictionary. The language model validates the word sequence hypothesis put forward by the acoustic model. In decoding the impact of the language model on the final output is regulated using *Language Model Weight*, w_{lm}. The equation then becomes

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}}\{P(w|\boldsymbol{X})\} = \arg\max_{\boldsymbol{w}}\{p(\boldsymbol{X}|w)P(w)^{w_{lm}}\}. \tag{2.24}$$

Finding the corresponding word sequence that maximizes Equation 2.24 is a search problem where the solution is the domain of the decoder. This problem can be tackled and solved using search algorithms. Kaldi solves the search task using *word lattices*, and how it is done is described in detail in Section 2.2.4.

2.1.5 Measuring ASR Performance

A common measurement used when comparing the performance of ASR models is the *Word Error Rate* (WER). It is computed at the word level and uses the *Levenshtein distance* between words to compute a matching and a score. The Levensthtein Distance is the minimal number of substitutions, deletions and insertions to make two strings equal [21]. The Levenshtein distance allows *substitutions*, *deletions* and *insertions*. In a simple metric all these operations are valued equally and have the cost of 1. An example of a string alignment between a reference string and a potential ASR output string is given below. Errors are marked with S, D and I. A correct transcription is marked with C.

ref	it	is	great		seeing	you	all	here	today
hyp	let's	***	great	to	see	you	all	here	today
error	S	D	C	I	S	C	C	C	C

After an alignment is made, the WER is computed by

$$WER = \frac{S_{tot} + D_{tot} + I_{tot}}{N} \tag{2.25}$$

where

S_{tot} is the total number of *substitutions*,

D_{tot} is the total number of *deletions*,

I_{tot} is the total number of *insertions*,

C_{tot} is the total number of *corrects*,

N is the number of words in the reference ($N = S_{tot} + D_{tot} + C_{tot}$)

So for the above example where $S_{tot} = 2, D_{tot} = 1, I_{tot} = 1, C_{tot} = 5$ the WER would be

$$WER = \frac{2+1+1}{2+1+5} = \frac{4}{8} = 0.5$$

The intuitive WER is a very straight forward measurement and sadly this naive error analysis can fail to understand the underlying pattern of spoken language. Example from [7]

ref	it	is	great		seeing	you	all	here	today
hyp1	**let's**	***	great		**see**	you	all	here	today
hyp2	**let's**	***	great	**to**	**see**	you	all	here	today

Here hyp2 better captures the nature of spoken language although it gives a higher WER than hyp1. The WER is also completely insensitive to the level of error in a substitution. Since it matches word with word the substitution *cat-cats* would yield as much of an error as *cat-hat*. A human reading the ASR output would probably not pay much notice to the first substitution whereas the second substitution would probably result in a very illogical sentence and be disturbing to the context.

The WER might not be an optimal measurement when it comes to measuring the quality of an ASR system, but it is a sufficient metric to use to compare the performance between models and evaluate a general accuracy.

2.2 Kaldi

Kaldi is an open source toolkit intended for use by speech recognition researchers [22]. The development started in 2009 and the toolkit now provides C++ implementations of most standard language modelling- and speech recognition algorithms, including the different transforms and all models mentioned in Section 2.1.

Kaldi is released under the Apache license version 2.0.

The following sections will describe how Kaldi deals with the triphone state tying, acoustic training and the speech decoding.

2.2.1 Triphone State-Tying

Kaldi uses *cross-word triphones*, which gives best accuracy for large vocabulary speech recognition tasks [28]. The phonetic transcription of the phrase "Look left!" using the marker `sil` to indicate silence in the beginning and end of the utterance would be `sil l U k l E f t sil .` Using cross-word triphone, this would instead be modelled as

`sil-l+U l-U+k U-k+l k-l+E l-E+f E-f+t f-t+sil ,`

where `-` indicates that the phone `b` occurs after `a` and `+` indicates that `b` occurs before the phone `c` in the phone sequence `a b c`. Notice that the two instances of phone `l`, `sil-l+U` and `k-l+E`, are represented by different HMM because their contexts are different.

As mentioned in Section 2.1.2 having triphones as training unit increases the dimensionality. To reduce the dimensionality parameter tying is introduced as illustrated in Figure 2.6 [29]. In Kaldi this is done at the state-level using *decision trees* [4] to map acoustically similar triphones together to the same HMM state.

For each phone a binary decision tree is generated seeking to cluster all of its associated triphones. The associated triphones for a phone are defined as all possible triphones generated by changing the left and right neighbour. The questions in the tree have the form of asking if the left neighbour *L-phone* or right neighbour *R-phone* belongs to a certain set X. The set X can be for example the set of Nasals or Fricative (see 3.2) or singleton sets like $\{l\}$ or $\{m\}$. Where to split the tree is decided by choosing the node and asking the question which maximizes the difference in log likelihood.

$$\Delta L_q = L(\mathbf{S}_y(q)) + L(\mathbf{S}_n(q)) - L(\mathbf{S}) \tag{2.26}$$

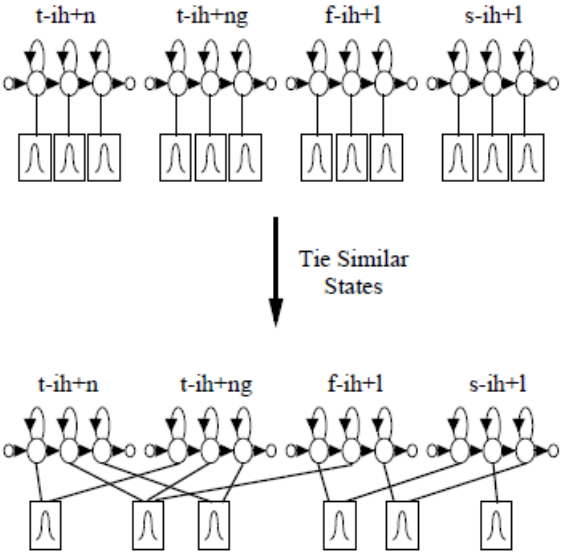

Figure 2.6. State Tying,figure from [28]

where $L(S)$ is the log likelihood of the set of HMM states S,

$$L(\mathbf{S}) = -\frac{1}{2}(\log[(2\pi)^n|\Sigma(\mathbf{S})|] + n)\sum_{s\in S}\sum_{f\in F}\gamma_s(\mathbf{o}_f) \tag{2.27}$$

where n is the dimensionality of the data and $\gamma_s(\mathbf{o}_f)$ is the *a posteriori* probability of the observed frame $\mathbf{o}_f$ being generated by state s in the HMM [29]. Depending on each answer the state makes its way down the tree until a leaf node is reached. All states reaching the same leaf node are then tied. These tied states are acoustically indistinguishable [28].

2.2.2 Acoustic Training

This section will explain how Kaldi handles the training of the two acoustic models, the GMM-based model and the DNN-based model.

Gaussian Mixture model

In Kaldi the acoustic training uses *Viterbi training* [1] for updating the Gaussian variables and model parameters θ.

Given a set of training observations, utterance and corresponding transcription, $O^r, 1 \leq r \leq R$ and an HMM state sequence $1 < j < N$ the observations sequence

is aligned in time to the state sequence via Viterbi alignment. This alignment is found by maximizing

$$\phi_N(T) = \max_i \left[\phi_i(T)a_{iN}\right], \qquad 1 < i < N, \tag{2.28}$$

where

$$\phi_j(t) = b_j(o_t) \max \begin{cases} \phi_j(t-1)a_{jj} \\ \phi_{j-1}(t-1)a_{j-1j} \end{cases} \tag{2.29}$$

with initial conditions, $\phi_1(1) = 1$ and $\phi_j(1) = a_{1j}b_j(o_1)$, for $1 < j < N$. Using the definition in 2.15 the output observations are defined as

$$b_j(o_t) = \sum_{m=1}^{M_j} c_{jm}\mathcal{N}(o_t; \mu_{jm}, \Sigma_{jm}). \tag{2.30}$$

The model parameters are updated based on the single-best alignment of individual observations to states and Gaussian components within states. Let A_{ij} denote the total number of transitions from state i to state j. Then the transition probabilities are estimated from the relative frequencies as

$$\hat{a}_{ij} = \frac{A_{ij}}{\sum_{k=2}^{N} A_{ik}} \tag{2.31}$$

The parameters of the Gaussian distribution are updated using an indicator function $\psi_{jm}^r(t)$ which is 1 if o_t^r is associated with mixture component m of state j and zero otherwise.

$$\hat{\mu}_{jm} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r} \psi_{jm}^r(t) o_t^r}{\sum_{r=1}^{R}\sum_{t=1}^{T_r} \psi_{jm}^r(t)} \tag{2.32}$$

$$\hat{\Sigma}_{jm} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r} \psi_{jm}^r(t)(o_t^r - \hat{\mu}_{jm})(o_t^r - \hat{\mu}_{jm})'}{\sum_{r=1}^{R}\sum_{t=1}^{T_r} \psi_{jm}^r(t)} \tag{2.33}$$

and the mixture weights are computed based on the number of observations allocated to each component

$$c_{jm} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r} \psi_{jm}^r(t)}{\sum_{r=1}^{R}\sum_{t=1}^{T_r}\sum_{l=1}^{M} \psi_{jl}^r(t)} \tag{2.34}$$

Deep Neural Network

One common choice for supervised training is error propagation where the output achieved at an iteration is compared to the desired output and the error is propagated backwards through the network and accounted for. For LVCSR with hundreds of hours of acoustic training data, such methods become too computationally heavy

[23]. A popular approach is to instead use gradient based methods and update the nodes according to the gradient of the optimization problem object function.

The objective function used in Kaldi is the so called *Cross-Entropy*

$$\mathcal{F}_{CE} = -\sum_{u=1}^{U}\sum_{t=1}^{T_u} \log y_{ut}(s_{ut}), \tag{2.35}$$

where s_{ut} is the reference state label at time t for utterance u and $y_{ut}(s)$ is defined in Equation 2.19 [26]. The gradient is

$$\frac{\partial \mathcal{F}_{CE}}{\partial a_{ut}(s)} = -\frac{\delta \log y_{ut}(s_{ut})}{\delta a_{ut}(s)} = y_{ut}(s) - \delta_{s;s_{ut}}. \tag{2.36}$$

Kaldi uses a Natural Gradient for Stochastic Gradient Descent intraining. It is described in [24] and [11]. The strategy in stochastic gradient descent is to perform the iterations

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta_t \boldsymbol{g}_t \tag{2.37}$$

where η_t is the scalar learning rate and $\boldsymbol{g}_t$ is the gradient from Equation 2.36. What is special for the implementation by Povey[24] is that it does not use a scalar learning rate, but instead a matrix-valued. The property of the matrix is that the learning rate is reduced in dimensions where the derivatives have a high variance; this will stop parameters from moving too fast in any direction.

2.2.3 Weighted finite State Transducers

This section will describe the building of a *Weighted Finite State Transducer* (WFST) [17]. Kaldi uses WFST to combine the information from the acoustic model and the language model.

A weighted finite state transducer maps symbols or objects of an *input alphabet* to an object in the *output alphabet*.

The HMM model used for acoustic modelling and the n-gram language model are special cases of *Weighted Finite State Acceptors* (WFSAs). The conditions for a WFSA is that the model is in only one state at a time, called the *current state*. It can change from one state to another - make a *transition* - when invoked upon by a *triggering event*. When the transitions between states have some kind of cost attached to them, for example associated probabilities, the acceptor is called *weighted*. In Kaldi the arc weights are negative log probabilites. By adding information about the final output in the WFSAs states, the weighted finite state acceptor can be interpreted as a weighted finite state transducer, see Figure 2.7 By doing so it is possible to combine the acoustic model and the language model into one integrated transducer containing all information required for an ASR model [17].

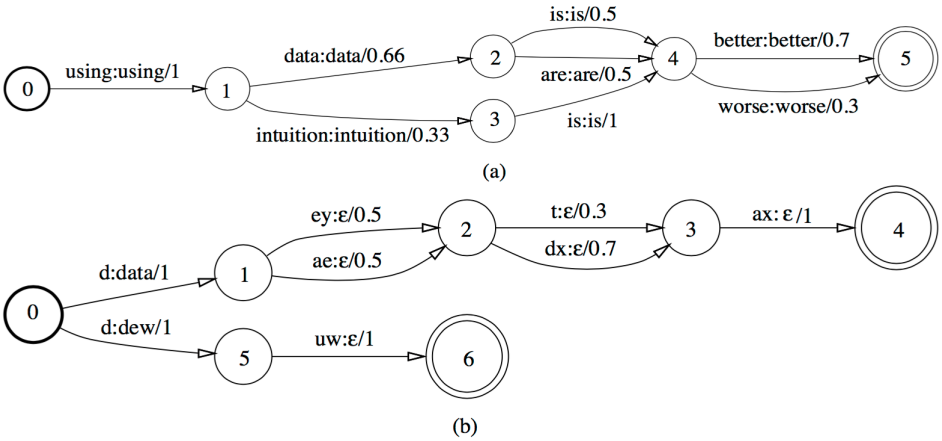

Figure 2.7. Examples of weighted finite state transducers, the labels on the arcs are on the form $i : o/p$ where i is input, o output and p probability. (a) An acceptor that has been turned into a transducer by setting the output labels equal to the input labels. (b) A phone acceptor turned into a transducer. ϵ is the mark for no output and is put on all nodes except the initial where the output label is set to the word output, ϵ is put as output because a word pronunciation may be a sequence of several phones.

To combine and concatenate two transducers three criteria must be fulfilled. If the transducer T is the composite transducer defined as $T = T_1 \circ T_2$, where T_1 and T_2 are transducers, then

(1) its initial state is the pair of the initial states of T_1 and T_2;

(2) its final states are pairs of a final state in T_1 and a final state in T_2, and

(3) there is a transition t from (q_1, q_2) to (r_1, r_2) for each pair of transitions t_1 from q_1 to r_1 and t_2 from q_2 to r_2 such that the output label of t_1 matches the input label of t_2.

The transition t takes its input label from t_1, its output label from t_2, and its weight is the $\otimes$-product of the weights of t_1 and t_2 when the weights correspond to negative log probabilities. A transducer is *deterministic* if there for every state only exists one possible transition for any given input label, otherwise it is *non-deterministic* [18].

This is applied on the speech model. The grammar G learned in the n-gram language model is combined with the phonetic dictionary also referred to as the *pronunciation lexicon* L,

$$L \circ G \tag{2.38}$$

resulting in a transducer that maps from phones to word sequences restricted to the grammar G. As mentioned earlier a transducer is deterministic if for every given input only one transition is possible. Problems arise in the ASR transducer for *homophones*, words that sound the same but have different meaning and perhaps also different spelling. This can be solved by adding auxiliary symbols in the lexicon L. In Kaldi this is solved by adding *transition-ids*, containing information about the PDF, the phone and the arc (transition) within the topology for that specific phone [22].

The transducer $L \circ G$ is context-independent. Thankfully composition provides a convenient mechanism for applying context-dependency to ASR transducers. By using the knowledge from the triphone model a context-independent chain of states can be composed with its corresponding context-dependent chain. It will result in a context-dependent transducer having the context-dependant labels on the input side and corresponding context-independent labels on the output side. Let C denote this transducer then

$$C \circ L \circ G \tag{2.39}$$

is a transducer that maps from context-dependent phones to word sequences restricted to grammar G.

Lastly the representation of HMM-states is integrated in the transducer

$$H \circ C \circ L \circ G \tag{2.40}$$

resulting in a transducer that maps from distributions to word strings restricted to G [18].

2.2.4 Word Lattices

The generation of word lattices is a powerful method for dealing with the large dimensional search problem that speech decoding is. There is no general single definition of a word lattice and generation algorithms are closely bound to the decoder used, in this case the above previously described WFST. This specific procedure of decoding and definition of word lattices is described by Povey in [23].

For a set of feature vectors obtained by computing the MFCCs from the audio input, the problem of obtaining the transcription is the equivalence problem of finding the most likely path through the WFST, i.e the path that yield minimum final cost. The search graph for an utterance is defined as

$$S \equiv U \circ HCLG, \tag{2.41}$$

where U is a WFSA corresponding to the utterance. The WFSA consists of $T + 1$ states, where T is the number of frames for the utterance, and an arc for each

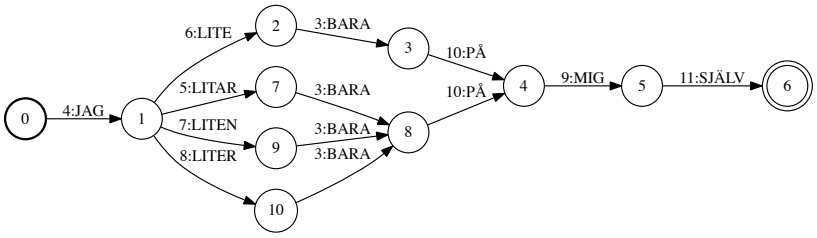

Figure 2.8. Example of a time aligned word lattice

combination of (time, context-dependant HMM state). The cost for each arc is the corresponding acoustic likelihood. With this notation, finding the transcription is the equivalence of finding the best path through S. A popular search algorithm that is applied in this step is *Viterbi decoding* in combination with beam-pruning. The process as described by Lingyun[14] can be summarized

For each time frame all parallel arcs are compared directly. The paths whose probabilities are much less than that of the most likely path are discarded. Let α be the beam parameter. If the most likely path have the score p_{MAX}, paths with individual score p_i that fulfil

$$|p_i - p_{MAX}| > \alpha \tag{2.42}$$

will be pruned away.

The surviving paths are then stored in a word lattice that satisfy the following conditions

- The lattice should have a path for every word sequence within α of the best-scoring one.
- The lattice should not contain duplicate paths with the same word sequence.
- All scores and alignments in the lattice correspond to actual paths through S

A graphic interpretation is given in Figure 2.8 for a lattice corresponding to the Swedish utterance 'JAG LITAR BARA PÅ MIG SJÄLV' where each word is represented by an in time aligned node. The number before the word on the arcs corresponds to the ID in the dictionary.

The most likely path through the word lattice is again a search problem and again any fast dynamic search algorithm would solve this. A popular choice being the Viterbi Algorithm 1, also used in Kaldi and described by Huang[9]

Algoritm 1 The Viterbi Algorithm

Step 1: Initialization
$V_1(i) = \pi_i b_i(X_1) \qquad 1 \leq i \leq N$
$B_1(i) = 0$
Step 2: Induction
$V_t(j) = \max_{1 \leq i \leq N} [V_{t-1}(i)a - ij]b_j(X_t) \qquad 2 \leq t \leq T; 1 \leq j \leq N$
$B_t(j) = \arg\max_{1 \leq i \leq N} [V_{t-1}(i)a_{ij}] \qquad 2 \leq t \leq T; 1 \leq j \leq N$
Step 3: Termination
The best score $= \max_{1 \leq i \leq N} [V_t(i)]$
$s_T^* = \arg\max_{1 \leq i \leq N} [B_T(i)]$
Step 4: Backtracking
$s_t^* = B_{t+1}(s_{t+1}^*) \qquad t = T-1, T-2, \ldots, 1$
$\boldsymbol{S}^* = (s_1^*, s_2^*, \ldots, s_T^*)$ is the best sequence

$V_t(i)$ is the probability of the most likely state sequence $\mathbf{S} = (s_1, s_2, \ldots, s_T)$ at time t, which has generated the observation X_1^t (until time t) and ends in state i. B is a back-pointer.

3. The Speech Data

3.1 Training Set

The data used for training is retrieved from a speech database collected by Nordisk Språkteknologi (NST). The database consists of roughly 500 hours of Swedish speech. The speech recordings are split into training and testing. Out of the 500 hours, ~400 hours is used for training and ~100 hours is used in testing.

Purpose	Lines	Persons	Recordings	Size(GB)
Training	312	920	287040	96.5
Testing	987	80	78960	22.7

The lines the speaker is asked to read vary from just being some letters to a word to at most a complete sentence. There is no detailed statistics about the speakers in the corpus but NST states that both genders are represented and that the ages span from 18 to 70. The database contains speakers from different dialectal areas in Sweden split into 10 regions

- Stockholm
- Eastern Southern-Sweden
- Western Southern-Sweden
- Västergötland
- Östergötland
- West-Sweden
- Dalarna with surroundings
- Göteborg

- Central Sweden
- Norrland

The technical details of the speech recordings are given below.

Signal encoding:	linear PCM
File format:	headerless raw
Sampling rate:	16 kHz
Resolution:	16 bit
Format:	Intel PCM
Channels:	2(stereo)

Apart from providing the recordings along with corresponding transcriptions, NST has also created a lexicon containing 927 167 entries.

3.2 Phonetic Dictionary

The lexicon produced by NST is described in detail in [20]. 26.95% of the lexicon is manually transcribed and 73.05 % of the words in the lexicon are generated by an inflector and has not been manually checked. An inflector automatically generates transcriptions for the different forms of a word based on its base form.

To motivate the usage of such an inflector a comparison between Swedish and English nouns is given. As can be seen in Table 3.1, inflecting nouns in Swedish can be complicated. This is just one example but there are at least five different classes of endings for when inflecting nouns in Swedish all result in similar tables like Table 3.1. Generally the corresponding English versions are often solved simply using *the* and *-s*.

Every entry in the NST lexicon has 51 fields. The first field is devoted to the words orthographic form, the subsequent fields contain information from morpho-

	Singular	**Plural**
Indefinite	*(en) stol* a chair	*stol**ar*** chairs
Definite	*stol**en*** the chair	*stol**arna*** the chairs

Table 3.1. Swedish declination vs English

Phonetic Category	XSampa	Example	XSampa	Example
Vowels	i:	s**i**l	y:	s**y**l
	I	s**i**ll	Y	s**y**ll
	u0	f**u**ll	e:	h**e**l
	}:	f**u**l	e	h**e**rr
	a	m**a**tt	2:	n**ö**t
	A:	m**a**t	9	m**ö**tt, f**ö**rra
	u:	b**o**t	o:	m**å**l
	U	b**o**tt	O	m**o**ll, h**å**ll
	E:	h**ä**l		
	E	h**ä**ll		
Dipthongs)	a*U	**au**tomat	E*U	**eu**ropa
Stops	p	**p**ol	d	**d**op
	b	**b**ok	d`	bor**d**
	t	**t**ok	k	**k**on
	t`	bo**rt**	g	**g**od
Nasals	m	**m**od	N	lå**ng**
	n	**n**od	n`	fo**rn**a
Fricatives	f	**f**ot	s	**s**ot
	v	**v**åt	S	**sj**ok
	s'	**kj**ol, **tj**ugo	h	**h**ot
	s`	fo**rs**		
Approx.	r	**r**ov	l`	po**rl**a
	l	**l**ov	j	**j**ord

Table 3.2. Swedish phone table

logical and grammatical information to lexicon id and inflector rule used. The most important field being the field containing the phonetic transcription.

The phonetic transcriptions are made using SAMPA (Speech Assessment Methods Phonetic Alphabet) [15] standards. The phones used are listed in Table 3.2 with corresponding sound. The original SAMPA phone `\x` had to be replaced by `S` in the implementation because of code issues caused by the backslash symbol \. The transcriptions also contain information about stress. Out of the stress markings in the NST lexicon only the symbols for *primary stress* `"` and *secondary stress* `%` are used in the final dictionary. The parsing of the dictionary will be described in Section 4.1.

3.3 Test Sets

The main set used for testing consist of the 100 hours separated for testing purposes in the NST speech data. From this set one subset of test sentences is taken and will be referred to as the `wordlooptest`[1]. This test set is created to be able to compare the acoustic model trained in Kaldi with a model trained in another toolkit but on the same speech data, as described by Vanhainen[25]. Another subset of the full NST test set that is used in testing is a set consisting of 120 randomly chosen utterances for each speaker in the test set. This set consists of approximately 9000 utterances in total and will be referred to as `test120`. This smaller test set `test120` was used when comparing different feature transforms in order to decrease the decoding time and arrive at an optimal set up to test on the full data faster.

Separate from the NST speech data is 20 hours of speech data collected and transcribed by SR. The SR material consists of 80 news broadcasts. Each news broadcast is 15 minutes long and approximately five minutes of the program is recorded outside a studio environment. The audio is mainly read by the same single speaker but interviews and out of studio elements often include another speaker or multiple speakers. This test set differs a lot from the training data in both terms of recording length as well as the environmental and surrounding aspects. It is used to test the robustness of the models, GMM-HMM and DNN-HMM, and to see their performance on actual SR data.

[1] Published on `ftp://ftp.nada.kth.se/TMH/asr/testlistsLREC2014.tgz`

4. Model Building

Apart from providing implementations of algorithms and transforms, as mentioned in Section 2.2, Kaldi also provides so called 'recipes'. Recipes are start-to-end implementations of models for speech recognition, trained on different speech databases, some which have to be bought and some which are free to use. It is also possible to follow any of these recipes to train a model using arbitrary speech data.

Nordisk Språkteknologi (NST) has apart from collecting Swedish speech data also collected speech data in Danish and Norwegian. Using the Danish database a Kaldi-recipe was developed by author Andreas Søeborg Kirkedal. The training of this model follows the standard procedure as presented on the Kaldi website [1]. The Danish script also provides code for downloading the speech material from the Språkbanken site and parsing it to fit with the Kaldi-code.

The Swedish recipe is developed by modifying the Danish recipe. During the development of the Swedish recipe the results produced by the Danish recipe on the Danish speech data also provided some validation to the results produced by the Swedish model on the Swedish speech data at the early stages of training.

In this section the building of the model is described. Firstly, the necessary preparations of the speech data is described and lastly the process of training the acoustic model.

4.1 Preparing the Training Data

For Kaldi to be able to generate the language model a directory called `dict`, for dictionary, has to be provided by the user.[2] In this directory the following files must all exist

[1] `http://www.danielpovey.com/kaldi-docs/tutorial.html`

[2] As described on kaldi website `http://www.danielpovey.com/kaldi-docs/data_prep.html#data_prep_lang_creating`

```
extra_questions.txt  lexicon.txt  nonsilence_phones.txt
optional_silence.txt  silence_phones.txt
```

The `nonsilence_phones.txt` is the list of all phones used in the transcriptions. Variations of the same phone, different stress, is put on the same line in the file. See Appendix for all files above except `lexicon.txt`. The `silence_phones` and `optional_silence` is the marker or markers for silent or unknown segments in the audio. The markers used are `SIL` and `SPN`. The sound `SPN`, meaning spoken noise, is paired with $<$ `UNK` $>$, meaning unknown, in the dictionary and Kaldi maps all words that appear in training data but not in the lexicon to $<$ `UNK` $>$. The file `extra_questions` contains additional questions to be added to the automatically generated questions by Kaldi. These are used in the decision tree splitting phase as described in Section 2.2.1. Since the different stress versions of a phone are on the same line in the `nonsilence_phones.txt` they share a root node when generating the phonetic decision trees. In order to separate them extra questions need to be asked, thus the file `extra_questions.txt` has one line for each stress mark used.

The `lexicion.txt` is a modified version of the lexicon created by NST. Firstly, only the fields containing the orthographic and the phonetic information were extracted. The phonetic transcription was stripped of any additional markers, those not being the primary and secondary stress markers, and any occurrence of `\x` was replaced with `S`. The word and transcription were separated by `tab` and phones in the transcription were separated with blank space. Any duplicates were removed and essential words that were in the training speech data transcription but not in the original NST lexicon were added. The final number of entries in the `lexicon.txt` is 822 747. A sample from the dictionary is given below

```
BARNBOK              "b A: n`%b u k
BARNBOKEN            "b A: n`%b u k e n
BARNBOKENS           "b A: n`%b u k e n s
BARNBOKS             "b A: n`%b u k s
BARNBOKSBILDER       "b A: n`b u k s %b I l d e r
BARNBOKSFÖRFATTARE   "b A: n`b u k s f 9 r %f a t a r e
BARNBOKSFÖRFATTAREN  "b A: n`b u k s f 9 r %f a t a r e n
...
```

When all information is provided and correct, Kaldi provides code for building the L transducer.

The n-gram with $n = 4$ is trained using a put together version of the unique lines from the transcriptions of the training data. From this the transducer G is generated.

To be able to train the acoustic model the speech data and transcriptions have to

be prepared and a `train` directory have to be created. Files needed in the training directory are

`spk2utt  text  utt2spk  wav.scp`

`spk2utt` and `utt2spk` are each others mirrors and the format for `spk2utt` follows `<speaker-id> <utterance-id1> <utterance-id2> ...`

`utt2spk` is a list of all utterances and corresponding speaker and the format is `<utterance-id> <speaker-id>`

`text` contains the transcript of each utterance and is on the format `<utterance-id> <TRANSCRIPTION>`. When information about speaker is present the speaker-id should be put as a prefix of the utterance-id so that the utterance-id will be `<speaker-id>-<utterance-id>`. In the below sample from the file `text` four utterances from the speaker `003` is listed.

```
003-r4670003-100  UTVÄNDIGT I DELAD LEDNING IN I SISTA SVÄNGEN
003-r4670003-101  DET FINNS ALLTSÅ INGET ATT HÄMTA ÄVEN OM MAN VINNER
003-r4670003-102  FRAMFÖR ALLT HAR VERKSTADSINDUSTRIN HAFT EN STARK ÖKNING
003-r4670003-103  F A L U N
```

The `wav.scp` contains the path way to the `.wav` file for each utterance. The format is `<utterance-id> path/to/utterance.wav`

The existence of the above files are also a requirement for the speech data used for testing. The test directory is prepared in the exact same way.

For decoding arbitrary speech data the files needed are `spk2utt`, `utt2spk` and `wav.scp`.

Now that the necessary preparations have been made the actual training can begin.

4.2 Training the Acoustic Model

First, the training of an acoustic model using Gaussian mixture models to estimate the HMM output observations is described. In the subsequent section the training of an acoustic model that uses deep neural networks is described. The stages of training will be given and the corresponding Kaldi script used stated.

4.2.1 Using Gaussian Mixture Models

1. To extract the acoustic feature vectors the script `make_mfcc` is called.

2. Cepstral mean variance normalization is then applied to the feature vectors by `compute_cmvn_stats`
3. An initial model trained on mono phones is computed by the script `train_mono`
4. An initial triphone model is computed by the script `train_deltas`, on top of this model are two models using different feature transforms trained
   a) One triphone model is trained using first and second order MFCC deltas, $\Delta + \Delta\Delta$, computed by the script `train_deltas`
   b) The other triphone model is trained using the LDA+MLLT+SAT transform computed by `train_lda_mllt` followed by `train_sat`

At each call of the triphone training scripts the parameter `number of Gaussians` and `number of leaves` need to be specified. The number of Gaussian is the number of mixture models that the training should aim to achieve and the number of leaves is the number of leaf nodes that should be aimed for in the process of state tying. A first guess on suitable parameter values are based on how many hours the training data consists of. There is no general rule for finding the optimal value of the number of Gaussians nor the number of leaves, it depends and varies with the nature of the training data and they are found through testing. The parameters are increased until the performance drops.

The parameters used for each triphone training is given in Table 4.1.

Training	Model name	# Leaves	# Gaussians
Initial triphone training	`tri1`	5 800	96 000
$\Delta + \Delta\Delta$	`tri2a` $\Delta + \Delta\Delta$	7 500	125 000
LDA+MLLT + SAT	`tri3b` LDA+MLLT + SAT	7 500	125 000
LDA+MLLT + SAT	`tri4a`	13 000	300 000

Table 4.1. Set up for different GMM-HMM training

4.2.2 Using Deep Neural Network

The deep neural network set up and parameters will be given in this section. The network used has four hidden layers and 1024 nodes in each layer, including the input and output layer. The choice of number of hidden layers and nodes in layers are based on the size of the training data. The activation function for producing the output of the nodes is the tanh function

$$y_j = 1 - \frac{2}{(e^{2x_j} + 1)}. \tag{4.1}$$

The initial learning rate is set to 0.01 and the final learning rate to 0.001.

The motivation behind choosing this set up is that the Danish recipe uses this set up and the Danish training data is approximately of the same size as the Swedish training data.

The deep neural network training is initialized with `train_tanh` and the input features are MFCC + LDA + MLLT + SAT.

5. Results and Analysis

5.1 Acoustic Training

The decoding of the test set `wordlooptest` when using the `tri2a` $\Delta + \Delta\Delta$ model gives a WER of 23.43%. A similar model is trained in a different toolkit but on the same training data by Vanhainen[25]. The corresponding WER achieved by Vanhainen in his report[25] is 23.98%. This validates the training of the GMM-HMM model in Kaldi. Using the `tri4a` model when decoding makes the WER on `wordlooptest` further reduce to 18.47%.

The Table 5.1 shows the WER obtained by the different models described in Section 4.2.1. The test set used is `test120`.

Model	WER
`mono`	48.86 %
`tri1`	24.16%
`tri2a` $\Delta + \Delta\Delta$	23.86%
`tri2b` LDA+MLLT	22.66%
`tri3b` LDA+MLLT + SAT	20.19%
`tri4a`	19.06%

Table 5.1. Result for different GMM-HMM training

The largest improvement can be seen when switching from monophones to triphones as the unit used in training. The best result for the acoustic model trained using GMM-HMM is obtained for the `tri4a` model.

5.2 Gaussian Mixture Models vs Deep Neural Networks

The GMM-HMM (model `tri4a`) and DNN-HMM is tested on the 100h NST test set. An investigation of convergence is performed using the same smaller subset for

training the models on fewer hours of training. It can be seen that the DNN model improves much faster and reaches a lower WER with fewer hours of training than the GMM-HMM model. At 80h of training is the DNN-HMM outperforming the GMM-HMM with 400h of training.

Acoustic data (h)	~20	~40	~80	~400
DNN-HMM	21.03%	19.55%	18.64%	15.97%
GMM-HMM	24.70%	24.03%	23.57%	18.88%

Table 5.2. Result for DNN-HMM and best GMM-HMM on all test data

It is not possible to draw any conclusion on whether or not the models have converged in performance. It can be noted that between 80 and 400 hours of training the GMM-based model changes with 4.69 percentage and the DNN-based model with 2.67, which means that the GMM-HMM could perhaps improve more than the DNN-HMM if more hours of training were added. The final performance for the GMM-HMM does still not get close to the final performance of the DNN-HMM for 400h which is 15.97%.

5.3 SR-material vs NST Test-material

The test set consisting of SR news broadcasts is decoded with the GMM-HMM (`tri4a`) and the DNN-HMM model and compared to the NST test set results.

Test set	GMM-HMM	DNN-HMM
SR test	55.46%	51.28 %
NST test	18.88%	15.97 %

Table 5.3. Result for test sets with GMM-HMM decoding

A significantly higher WER is obtained for the SR test data than for the NST test data for both models. This can have multiple explanations. Firstly, the SR test data is very different from the NST data. The decoding length for the SR test data is 15 minutes compared to NST data where the audio recordings rarely exceeds ten seconds. Secondly, the SR news broadcast often include multiple speakers in the broadcast. This means that applying cepstral mean variance normalization does not help with reducing the impact of the environmental differences. The CMVN transform is applied per speaker and with multiple speakers in one recording the effects of CMVN is erased. The segments in the news broadcast that are recorded outside a studio environment are often noisy and not ideal for speech recognition tasks. Sometimes the out of studio elements can also be in a different language.

The high WER can also be caused by the unfortunate properties of the WER measurement. When manually going through the transcription the impression is that the general content of the program is well captured. The decoder succeeds in finding the long, complicated keywords but fails in out of studio environment and decoding geographic locations and names. An estimate is that the out of studio segments make up one third of the news broadcast, that is five minutes out of 15. When five minutes of the file completely fails in decoding the remaining ten minutes of the program would have to achieve a very low WER to make up for this loss. Another problem with WER very specific for the Swedish language is the lack of a good strategy for compound words.

ref	socialdemokrat	
hyp	social	demokrat
error	S	I

This example gives a word error rate of $\frac{2}{1} = 200\%$. It is an incorrectly decoded sentence, but this example shows that one should bear in mind that a high WER does not necessarily mean that the decoder is wildly guessing.

6. Conclusion

In this thesis a speech recognizer for the Swedish language was successfully developed using the Kaldi toolkit. Two different approaches, one using a mixture of Gaussians to model the output of the Hidden Markov Model, and the other using a deep neural network, were developed and compared. It was shown that the model using a deep neural network both learned faster, in terms of achieving a lower WER with fewer hours of training, as well as achieving a lower WER on the full NST test set when trained an all 400 hours. The performance of the two models, GMM-HMM and DNN-HMM, were also tested on SR material. 20 hours of news broadcasts were transcribed by both models. Again, the model using deep neural networks achieved a lower WER on the SR news transcriptions than the GMM-HMM. In terms of WER it seems that the approach using a deep neural network is advantageous over the model using a mixture of Gaussians.

When comparing the models' performances on the SR and NST test data sets it could be seen that the WER on the SR test set for both models were higher. Based on the discussion carried out in Section 5.3 this could be due to the more challenging out of studio segments that are present in all news broadcast as well as the problem the models have with transcribing names and compound words.

The results presented in this thesis show that both the GMM-HMM and the DNN-HMM model are able to transcribe Swedish speech, the DNN-HMM performing better on all test sets. The fact that the DNN-HMM model is capturing the context of an SR news broadcast quite well means that the model produces transcriptions that contain enough information to generate metadata. Concerning the possible applications that would sprung from a working ASR model, as mentioned in the Introduction, Kaldi provides code for both implementing real-time transcription as well as providing the time-word alignment information of a audio file and its transcription. The models developed in this project show that it is possible to implement those features using the speech data from NST and tools provided by Kaldi.

6.1 Future Work

To achieve a better performance of the model focus should be put on the phonetic dictionary. Firstly, asserting that the compound words the model misses out on are present in the dictionary. Secondly, to reach a lower WER on the SR material the dictionary should be updated with the names of geographic locations and names of program hosts and program producers at SR.

In the process of decoding it would be interesting to extract some kind of confidence score of the models as well as to mark out the words in the text the models are very insecure about. This to be able to better evaluate the performance of a model. As has been discussed in previous sections, using the WER as a performance measurement is not optimal. Finding another metric to use to evaluate the models' performances and compare that result with the result when using WER would be preferred.

Bibliography

[1] Senaka Buthpitiya, Ian Lane, and Jike Chong. A parallel implementation of viterbi training for acoustic models using graphics processing units. In *Innovative Parallel Computing (InPar), 2012*, pages 1–10. IEEE, 2012.

[2] Steven B Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):357–366, 1980.

[3] Li Deng Dong Yu. Automatic speech recognition, a deep learning approach. 2015.

[4] Mark Gales and Steve Young. The application of hidden markov models in speech recognition. *Foundations and trends in signal processing*, 1(3):195–304, 2008.

[5] Mark JF Gales. Semi-tied covariance matrices for hidden markov models. *Speech and Audio Processing, IEEE Transactions on*, 7(3):272–281, 1999.

[6] Ramesh A Gopinath. Maximum likelihood modeling with gaussian distributions for classification. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 2, pages 661–664. IEEE, 1998.

[7] Xiaodong He, Li Deng, and Alex Acero. Why word error rate is not a good metric for speech recognizer training for the speech translation task? In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5632–5635. IEEE, 2011.

[8] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

[9] Xuedong Huang, Alex Acero, Hsiao-Wuen Hon, and Raj Foreword By-Reddy. Spoken language processing: A guide to theory, algorithm, and system development. 2001.

[10] Annika Jansson. Tal till text för relevant metadatataggning av ljudarkiv hos sveriges radio, 2015.

[11] KALDI. DNN. `http://kaldi-asr.org/doc2/dnn2.html`, 2016. [Online; accessed 14-July-2016].

[12] KALDI. Feature Extraction. `http://kaldi.sourceforge.net/feat.html`, 2016. [Online; accessed 29-April-2016].

[13] KALDI. Wondow function. `http://kaldi-asr.org/doc/structkaldi_1_1FeatureWindowFunction.html`, 2016. [Online; accessed 28-June-2016].

[14] Xie Lingyun and Du Limin. Efficient viterbi beam search algorithm using dynamic pruning. In *Signal Processing, 2004. Proceedings. ICSP'04. 2004 7th International Conference on*, volume 1, pages 699–702. IEEE, 2004.

[15] University College London. Sampa. `https://www.phon.ucl.ac.uk/home/sampa/`, 1999-2015.

[16] Václav Matoušek. *Text, Speech and Dialogue: 10th International Conference, TSD 2007, Pilsen, Czech Republic, September 3-7, 2007, Proceedings*, volume 4629. Springer Science & Business Media, 2007.

[17] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002.

[18] Mehryar Mohri, Fernando Pereira, and Michael Riley. Speech recognition with weighted finite-state transducers. In *Springer Handbook of Speech Processing*, pages 559–584. Springer, 2008.

[19] Sirko Molau, Florian Hilger, and Hermann Ney. Feature space normalization in adverse acoustic conditions. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 1, pages I–656. IEEE, 2003.

[20] Gisle Andersen (Nasjonalbibloteket). Leksikalsk databaser for svensk. `http://www.nb.no/sbfil/dok/nst_leksdat_se.pdf`, 2011.

[21] Gonzalo Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001.

[22] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr

Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

[23] Daniel Povey, Mirko Hannemann, Gilles Boulianne, Lukáš Burget, Arnab Ghoshal, Miloš Janda, Martin Karafiát, Stefan Kombrink, Petr Motlicek, Yanmin Qian, et al. Generating exact lattices in the wfst framework. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4213–4216. IEEE, 2012.

[24] Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur. Parallel training of dnns with natural gradient and parameter averaging. *arXiv preprint arXiv:1410.7455*, 2014.

[25] Niklas Vanhainen and Giampiero Salvi. Free acoustic and language models for large vocabulary continuous speech recognition in swedish. *training*, 965(307568):420–8, 2014.

[26] Karel Veselỳ, Arnab Ghoshal, Lukás Burget, and Daniel Povey. Sequence-discriminative training of deep neural networks. In *INTERSPEECH*, pages 2345–2349, 2013.

[27] Olli Viikki and Kari Laurila. Cepstral domain segmental feature vector normalization for noise robust speech recognition. *Speech Communication*, 25(1):133–147, 1998.

[28] Steve Young. A review of large-vocabulary continuous-speech. *Signal Processing Magazine, IEEE*, 13(5):45, 1996.

[29] Steve J Young, Julian J Odell, and Philip C Woodland. Tree-based state tying for high accuracy acoustic modelling. In *Proceedings of the workshop on Human Language Technology*, pages 307–312. Association for Computational Linguistics, 1994.

A. The Contents of the Dictionary Directory

silence_phones.txt

```
SIL
SPN
```

optional_silence.txt

```
SIL
```

extra_questions.txt

```
"2:  "9   "A:  "E  "E*U  "E:  "I  "O  "S  "U"d    "Y  "a  "a*U  "b  "d  "d`  "e  "e:  "f  "g  "h  "i:  "j  "k  "l  "l`  "m  "n`  "o:  "p  "r  "s  "s`  "s'  "t  "t`  "u0  "u:  "v:  "y:  "}:
%2:  %9   %A:  %E  %E*U  %E:  %I  %O  %S  %U%d    %Y  %a  %a*U  %b  %d  %d`  %e  %e:  %f  %g  vh  %i:  %j  %k  %l  %l`  %m  %n`  %o:  %p  %r  %s  %s`  %s'  %t  %t`  %u0  %u:  %v:  %y:  %}:  %"j
2:
9
A:
E    E*U  E:
I
N
O
S
U
Y
a    a*U
b
d    d`
e    e:
f
g
h
i:
j
k
l    l`
m
n    n`
o:
p
r
s    s`   s'
t    t`
u0   u:
v
y:
}:
```

nonsilenece_phones.txt

```
2:   "2:   %2:
9    "9    %9
A:   "A:   %A:
E    "E    %E
E*U  "E*U
E:   "E:   %E:
I    "I    %I
N
O    "O    %O
S    "S    %S
U    "U    %U
Y    "Y    %Y
a    "a    %a
a*U  "a*U  %a*U
b    "b    %b
d    "d    %d
d`   "d`   %d`
e    "e    %e
e:   "e:   %e:
f    "f    %f
g    "g    %g
h    "h    %h
i:   "i:   %i:
j    "j    %j    %"j
k    "k    %k
l    "l    %l
l`   "l`   %l`
m    "m    %m
n`   "n`   %n`
o:   "o:   %o:
p    "p    %p
r    "r    %r
s    "s    %s
s`   "s`   %s`
s'   "s'   %s'
t    "t    %t
t`   "t`   %t`
u0   "u0   %u0
u:   "u:   %u:
v    "v:   %v
y:   "y:   %y:
}:   "}:   %}:
```

TRITA -MAT-E 2016:42
ISRN -KTH/MAT/E--16/42--SE