Automatic Speech Recognition – Coursework

s1794003, s1886437

1. Introduction

This report presents and discusses the results of word recognition experiments[1] we performed using Kaldi automatic speech recognition toolkit (Povey et al., 2011). The experiments[2] were carried out on three types of models: monophone, triphone and speaker adaptive training (SAT) models.

For the monophone model, we investigated the optimal number of Gaussians that minimises the word error rate (WER). In addition, we examined the effects of using cepstral mean and variance normalisation (CMN and CMVN), as well as the effects of using the delta and delta-delta features on the performance of the model.

Using a tied-state triphone model, we performed a grid search to find the optimal numbers of Gaussians and clusters that give the best WER. We then extended our best performing triphone model to be dependent on the speakers. To achieve this, we adopted the feature-state maximum likelihood linear regression (fMLLR) transforms to normalise the variation between speakers during training (Gales, 1998; Woodland, 2001). We then decoded the model using speaker adaptation. Finally, we adopted a deep neural network (DNN) with SAT using fMLLR speaker adaptation and was able to achieve 44.48% WER when tested using our own recordings.

The next section introduces the datasets used in our experiments and describes how the performance of the models are evaluated. Section 3 gives a brief overview of the HMM-GMM model, as well as the monophone and triphone models. Section 4 outlines the experiments we conducted on the monophone, triphone and speaker adaptive training models. It also presents the results of these experiments, as well as discusses them in details. Finally, the findings of this report are summarised in Section 5.

2. Data and Evaluation Methods

2.1. Data

The training and decoding of the models in this report are carried out using the TIMIT dataset. Once the optimal hyperparameters are selected, the models are evaluated using the authors' own recordings.

[1]All experiments were performed on Google Compute. The results may vary slightly when scripts are run on DICE.

[2]All experiment results can be found in the *WorkDir*/my-local/log/ directory.

Dialect Regions	Female	Male	Both
New England	14	24	38
Northern	23	53	76
North Midland	20	56	76
South Midland	15	53	68
Southern	25	45	70
New York City	13	22	35
Western	18	59	77
Army Brat	8	14	22
Total	136	326	462

Table 1. Number of speakers within each of the eight dialect regions in the training set.

2.1.1. TIMIT dataset

The TIMIT dataset consists of speech recordings from eight major dialects of American English. The experiments in this report are trained using a subset of the TIMIT dataset and contains 3,696 utterances from 462 speakers. The number of speakers in each dialect region is summarised in Table 1. The training set contains 8 utterances from each speaker, for a total 1,088 female utterances and 2,608 male utterances.

Decoding is carried out on the TIMIT test set and consists of 192 utterances from 24 speakers. There are one female and two male speakers from each dialect region.

2.1.2. Authors' recordings

To further evaluate some of our models we create a new testing data set. The two male authors each recorded 20 English utterances. The first author (speaker 1) has a New Zealand accent, while the second (speaker 2) has an English accent. The recordings are monaural with the sampling frequency of 16kHz.

2.2. Evaluation Methods

The primary performance metric for evaluating models is the word error rate (WER) – which is derived from the Levenshtein distance between the hypothesised and the correct words. WER is defined as:

$$WER = \frac{\text{Insertions} + \text{Substitution} + \text{Deletion}}{\text{Total words in correct transcript}} \times 100. \quad (1)$$

WER is used to measure the accuracy of a speech recognition system. The lower the WER, the better the system is deemed to be.

Computation complexity of each model is measured using training and testing (decoding) times. A model with lower training and testing times is preferable because it is more likely to scale better on a large dataset.

Kaldi captures log-likelihood information during the training and testing of a model. Log-likelihood is a measure of how likely the observed data (e.g. sequence of words) is, given the model parameters. In general, higher log-likelihood indicates that a model can fit the observed data better. As a model becomes more complex, it is more likely to receive higher log-likelihood. However, this may not be desirable because the model is at risk of overfitting and will not generalise well. Furthermore, log-likelihood is dependent on the number of observations, and so the log-likelihood obtained during training cannot be directly compared to the log-likelihood obtained during testing. Due to these reasons, we focus on WER when evaluating the model performances.

3. Background

3.1. HMM-GMM

In speech recognition, the objective is to create a classifier that can identify which phone is uttered in each frame. In order to achieve this, the Gaussian mixture model (GMM) can be used. Specifically, for a given phone class, a GMM can be fitted using the expectation-maximisation algorithm by training on all frames where that phone is found. A new phone utterance can then be classified by determining frame-by-frame which phone class is most probable. That is, which GMM has the highest likelihood of generating that phone (or its features).

An issue with using GMM by itself is that it does not exploit temporal dependencies in the acoustic signals. That is, it assumes that for a given phone, there are no acoustic differences in the beginning, middle and end of an utterance. In order to over come this limitation, a hidden Markov model (HMM) is used to model the temporal dependencies and each phone is represented using three HMM states – beginning, middle and end. Subsequently, each state is modelled by a GMM to determine the likelihood of the observation in that state.

3.2. Monophone

Monophone is an HMM-GMM model where each phone is represented by a tri-state HMM (beginning, middle and end). It is known as a context-independent model because it assumes that each phone is uttered the same way regardless of its context. This is not true in practice and is the main drawback of this model.

3.3. Triphone

Triphone extends the monophone model to include the context of the phone. Its representation includes the phone before, as well as the one after. This allows triphone to capture variation in utterance due to that arises from the phone context.

An issue that arises with triphone is data sparsity. That is, it is unlikely that the model will see each triphone with enough observations during training. For a given phoneset of n phones, there would be n^3 possible triphones – even though not all of these triphones would ever be encountered in reality.

In order to overcome the issue of data sparsity, the tied-state triphone model can be used where the states of acoustically similar subphones are clustered (i.e. tied) together. All the subphones in the same cluster share the same GMM acoustic model. The clustering is performed in a top-down manner by a phonetic decision tree. The decision tree produces a set of states at each leaf of the tree – each of these states is modelled by the GMM.

4. Experiments

This section provides the details of the experiments and presents their results. The models that were examined are monophone, triphone and SAT models.

4.1. Monophones

Two sets of experiments were performed with the monophone model. First, we examined how the number of Gaussian mixture components affects the performance of the model. With this, we found the optimal number of Gaussian components for the monophone model. We then performed additional experiments to investigate the effects of normalising cepstral means and variances of the speakers, as well as the effects of using dynamic features (i.e. delta and delta-delta) of Mel-frequency cepstral coefficients (MFCCs).

4.1.1. Gaussian Mixture Components Experiment

A GMM is a weighted sum of Gaussian components. Given that it has enough Gaussian components, the GMM can be used to approximate any distributions. An important factor to consider when training the GMM is the number of Gaussian components to use. The model needs enough to effectively generalise all the possible observations. However, if too many are used the model can risk overfitting to the training data and performs poorly on an unseen dataset (e.g. new utterances). It will also become more computationally expensive.

To investigate this we trained a number of HHM-GMM monophone models with various numbers of Gaussian mixture components. The results can be found in Figure 1. Note that in Kaldi, training the monophone model is achieved by using the `steps/train.sh` script. By default, this script includes delta and delta-delta features. It also uses CMN to train the models. We kept these settings as they were because they do not interfere with our experiments.

As seen in from the plot, the WER is lowest when there are about 6,000 Gaussian components. It is important to

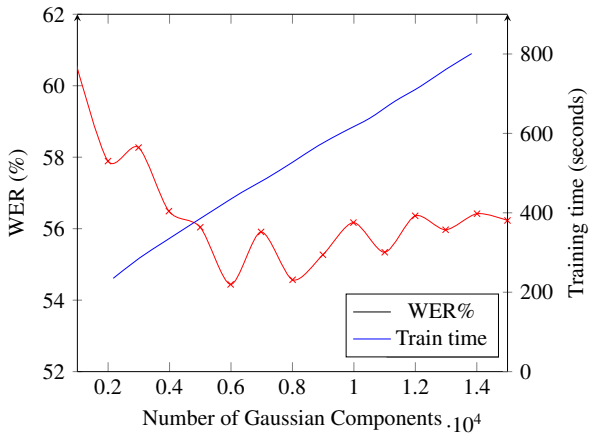

Figure 1. Effects of the number of Gaussian components (actual) on WER and training time in monophone models.

note that the actual number of Gaussian components is not the same as the number of components we specified. For example, when we specified 6,000 Gaussian components, the actual number of Gaussians was 5,988.

From the plot, we also observed a linear relationship between the training time of the monophone models and the number of Gaussian components. This shows that the computation complexity of adding an additional Gaussian component is approximately linear. This finding is consistent with the fact that GMM is a weighted sum of the Gaussian components.

The average log-likelihood during decoding is -8.51 for 1,000 specified Gaussians, and increases to -8.37 for 15,000 Gaussians. This may indicate that a model with more Gaussian components is able to fit the observed data better. However, as mentioned in Section 2.2, log-likelihood is not a good performance metric. Increasing log-likelihood while WER decreases is likely a result of overfitting as the model complexity increases when more Gaussians are added.

4.1.2. Effects of Dynamic features and CMN/CMVN

So far we have trained our models with input feature vectors of 39 dimensions. These 39 dimensions are:

- 12 MFCCs + 1 energy
- 13 first-order delta (12 MFCCs + 1 energy)
- 13 second-order delta (12 MFCCs + 1 energy)

Two thirds of the feature vector is occupied by the dynamic delta features. In this section, we investigate whether these dynamic features help to improve the performance of the model.

CMN and CVMN are techniques used to add robustness to the model via addressing the noises that are introduced by channel characteristics, e.g. using different microphones. Intuitively, removing these variability should increase the performance of the model because the signal-to-noise ratio would increase as a result. Therefore, we also investigate how CMN/CMVN influence the performance of the model in this section.

WER (%)	None	CMN	CMVN
No delta	82.49	81.21	79.11
First-order delta	63.32	61.73	62.11
Second-order delta	58.02	**54.44**	55.46
Training time (sec.)	None	CMN	CMVN
No delta	302	299	303
First-order delta	379	380	381
Second-order delta	444	449	448
Testing time (sec.)	None	CMN	CMVN
No delta	414	444	451
First-order delta	261	284	289
Second-order delta	200	204	196
Decode LL	None	CMN	CMVN
No delta	-4.19	-4.17	-1.30
First-order delta	-6.82	-6.79	-1.05
Second-order delta	-8.42	-8.39	0.20

Table 2. WER, training time, testing time and decode log-likelihood for monophone models with no delta, with first-order delta, or with first- and second-order delta features; and with no mean/variance normalisation (None), with cepstral mean normalisation (CMN), or with cepstral mean and variance normalisation (CMVN). The best WER is shown in bold.

We conducted a combined set of experiments to test each setting of dynamic features, as well as each setting of cepstral normalisation. The experiments were performed using the previously obtained optimal number of Gaussian components of 6,000. The results are presented in Table 2.

As seen from Table 2, the WER decreases as higher order of delta features are added. The inclusion of delta-delta features adds more context to the feature space which improves the accuracy of the prediction.

The increase in feature dimensions also increases the training time as more computations are required to handle extra features. However, this results in faster decoding. This is possibly because decoding with Viterbi algorithm can involve a beam search. The inclusion of dynamic features result in more low-probability paths that are pruned out in the process. This ultimately results in fewer possible matches and speeds up the decoding process.

Both CMN and CMVN improve WER as they reduce the noise through mean normalisation. We did not observe any improvements when cepstral variance normalisation is applied. This contradicts with the results of Viikki & Laurila (1998). However, it could be explained by the fact that the utterances in the TIMIT dataset are short, and so there is not enough data for parameter estimation (Prasad & Umesh, 2013).

WER (%)		Number of Gaussian components					
		10,000	**12,000**	**14,000**	**16,000**	**18,000**	**20,000**
Number of clusters	**1,000**	46.45	48.24	46.84	46.84	46.84	47.41
	1,500	45.18	45.50	**44.41**	45.05	46.65	45.37
	2,000	46.26	46.01	45.81	45.75	45.62	46.45
	2,500	45.75	46.39	45.75	45.69	45.05	44.86
	3,000	46.52	47.48	47.22	46.58	47.99	46.84
	3,500	46.77	46.96	47.09	47.03	47.73	46.58
	4,000	46.71	46.65	47.22	45.94	45.50	45.75
	4,500	46.90	46.33	46.39	46.77	47.03	46.90

Table 3. Coarse grid search for the optimal numbers of Gaussian components and clusters in tied-state triphone model. The best WER is shown in bold.

4.1.3. Evaluating best system

The best monophone model is specified with 6,000 Gaussian components, and is trained with delta and delta-delta features, as well as CMN. Under these conditions, the monophone tested on the authors dataset has WER of 63.08% (59.60% for speaker 1 and 64.54% for speaker 2). These WERs are higher than the WER obtained through decoding on the TIMIT dataset. The higher WERs could be explained by the fact that the authors have different accents from the speakers in the TIMIT dataset which the models were trained on.

4.2. Triphones

4.2.1. Effects of numbers of clusters and numbers of Gaussian mixture components

In this section, we investigate how number of clusters (i.e. leaves) and the number of Gaussians affect the performance of a triphone model. More clusters allows more variability in phone and prevents acoustically dissimilar sounds from being tied together. However, having too many clusters may result in sparse data with insufficient observations in each cluster. In addition, the number of Gaussians are required to estimate the PDFs in each cluster.

In order to train the triphone model, the monophone model must first be cloned to triphones. To achieve this, we used the best monophone model obtained from Section 4.1 which has 6,000 Gaussian components, and is trained with delta and delta-delta features with CMN.

We first performed a coarse grid search to get an intuition of where the local optima might be. In our search, we specified between 10,000 and 20,000 Gaussian components with increments of 2,000; and between 1,000 and 4,500 clusters with increments of 500. The lowest WER is obtained when the numbers of Gaussians and clusters are 14,000 and 1,500, respectively. We then performed a fine grid search around these settings, in attempt to find the local optima. However, the optimal settings remained unchanged. The complete WER results are summarised in Tables 3 and 4 for coarse and fine search, respectively.

From the grid search conducted, the triphone model appears to be insensitive to the changes in the parameters. The the average number of Gaussians per cluster, as computed by $\frac{\text{num Gaussians}}{\text{num clusters}}$, ranges from ≈ 2 to ≈ 20 Gaussians per cluster. In this range, the best and the worst models differ in WERs by about 3%.

WER (%)		Number of Gaussian components				
		13,000	**13,500**	**14,000**	**14,500**	**15,000**
Num. clusters	**1,300**	45.18	45.81	46.90	46.01	46.96
	1,400	46.39	46.65	46.20	45.75	46.01
	1,500	46.26	46.52	**44.41**	45.56	46.71
	1,600	44.86	45.37	44.98	45.88	45.50
	1,700	45.62	46.52	45.56	45.30	44.98

Table 4. Fine grid search for the optimal numbers of Gaussian components and clusters in tied-state triphone model. The best WER is shown in bold.

The training time and testing time appear to be higher when the number of Gaussians per cluster is higher. Furthermore, the test log-likelihood appears to increase as the number of Gaussians per cluster increases. These observations are consistent with the fact that more Gaussians per cluster increases the complexity of the model, resulting in higher computation time and better fit to the observations (possibly overfit).

4.2.2. Evaluating best system

The best tied-state triphone model is specified with 14,000 Gaussian components and 1,500 clusters. Under these conditions, the tied-state triphone model tested on the authors dataset has WER of 53.43% (46.89% for speaker 1 and 58.45% for speaker 2). This shows $\approx 10\%$ overall improvement in WERs over the monophone model and provides evidence that context is important in acoustic modelling.

4.3. Speaker Adaptive Training and Speaker Adaptation

We extended the speaker-independent triphone model to carry out speaker adaptive training (SAT) and speaker adaptation using fMLLR. SAT involves estimating and applying fMLLR transform matrix to the model (mean and variance of Gaussians) during training in order to perform speaker normalisation. That is, to remove variability from data that arises due to having different speakers. On the other hand, speaker adaptation aims to reduce the mismatch between the train and test data. This enables the model to perform well on the unseen speakers in test data.

We first performed SAT under HMM-GMM. We then modified the model to use a deep neural network (DNN) to estimate the PDFs.

4.3.1. HMM-GMM

The experiments in this section involve running SAT on top of the best tied-state triphone (delta and delta-delta) model using HMM-GMM acoustic modelling. The procedure can be summarised as:

1. fMLLR transforms are estimated and applied to the

model parameters during training to normalise any variability between speakers.

2. Speaker adaptation is performed during decoding to reduce the mismatch between train and test data.

A grid search was carried out to find the optimal numbers of Gaussians and clusters (i.e. leaves) that minimise the WER. We trained our SAT model by specifying between 10,000 and 20,000 Gaussians in 2,000 increments, and between 1,000 and 4,000 clusters in 500 increments. It is important to note that under these settings, the final number of PDFs estimated by GMM are constant at 2,560 across all models. This would explain why the training and testing times remained relatively unchanged across all models (313 seconds for training and 187 seconds for decoding, on average).

The lowest WER of 39.49% is achieved on the TIMIT test dataset when we specified the numbers of Gaussians and clusters to be 16,000 and 1,500, respectively. When we ran this model using the authors dataset, we obtained WER of 48.67% (39.27% for speaker 1 and 56.61% for speakers 2). The decrease in WERs provides across TIMIT and authors' dataset provides some indications that fMLLR transforms is able to remove speaker-related variability from data.

Relevant scripts:

- `exp_task3_sat.sh` contains the script to train and decode the HMM-GMM SAT model using TIMIT dataset.
- `run_task3_sat.sh` contains the script to test the best performing HMM-GMM SAT model on the authors' recordings.

4.3.2. HMM-DNN

In this section, we added a DNN in order to train a new SAT model with fMLLR speaker adaptation. The steps to train and test can be summarised as follows:

Training:

1. Create alignment on previously trained HMM-GMM SAT model using training data.
2. fMLLR-transform the features of training set.
3. Train with DNN using fMLLR features as inputs.

Testing:

1. Create alignment on previously trained HMM-GMM SAT model using test data.
2. fMLLR-transform the features of test set.
3. Decode with DNN using fMLLR features.

WER (%)	TIMIT	Authors	Speaker 1	Speaker 2
Monophone	54.44	63.08	59.60	64.54
Triphone	44.41	53.43	46.89	58.45
HMM-GMM SAT	39.49	48.67	39.27	56.61
HMM-DNN SAT	32.08	44.48	38.70	46.54

Table 5. Summary of test results on TIMIT and authors dataset (including individual results for speakers 1 and 2) for each model in this report.

We experimented with different numbers of hidden layers, as well as the number of dimensions in each hidden layer in order to search for the best DNN architecture that minimises the WER. We trained the HMM-DNN model using 6, 7 and 8 hidden layers, where the hidden layers have 512, 1,024 or 1,536 dimensions. The learning rate of 0.0008 was used in all experiments (default in Kaldi).

The lowest WER of 32.08% is achieved on the TIMIT test dataset for DNN with 7 hidden layers and 1,024 hidden dimensions. Overall, as the number of hidden layers and hidden dimensions increase, the training and test time increase. This is because the network has to estimate more parameters (weights). Furthermore, the log-likelihood increases as the number of hidden dimensions increase.

When we ran the HMM-DNN model using the best settings on our own recordings, we obtained WER of 44.48% (38.70% for speaker 1 and 46.54% for speakers 2). The results indicate that using DNN for SAT and speaker adaptation with fMLLR transforms can further remove variability between speakers.

Relevant scripts:

- `exp_task3_nn2.sh` contains the script to train and decode the HMM-DNN SAT model using TIMIT dataset.
- `run_task3_nn2.sh` contains the script to test the best performing HMM-DNN SAT model on the authors' recordings.

The summary of test results for the models in this report are summarised in Table 5. Overall, HMM-DNN SAT improved WER on TIMIT dataset by 22.36% and on authors' recordings by 18.6%.

5. Conclusion

Using Kaldi, we performed various experiments on monophone, tied-state triphone and SAT/speaker adaptation word recognition models. The key findings are as follows:

- Training with delta and delta-delta features can improve the WER of a monophone model by adding more context to the features. This comes at a cost of increased training time.
- CMN improves the WER of a monophone model. However, we did not see an improvement in WER

when cepstral variance normalisation was applied. This is possibly a result of having speech data with short utterances. An alternative technique to CMVN that works with short utterances is called vector Taylor series (VTS) (Moreno et al., 1996). VTS can be tested in future experiments on TIMIT dataset – although the method is computationally expeensive (Obuchi & Stern, 2003).

- By estimating and applying fMLLR transforms, we can normalise the variability between speakers. In addition, applying speaker adaptation during decoding can reduce the mismatch between train and test data – resulting in better WER performance. Using DNN for SAT/speaker adaptation can further improve the WER.

- The datasets used in the experiments are small. Future experiments can be carried out on larger datasets. This would likely improve WER results.

- Overall, our final HMM-DNN SAT model achieved WER of 32.08% on TIMIT dataset and 44.48% on our own recordings. This is an improvement of WER over the monophone model by 22.36% on TIMIT dataset and by 18.6% on our own recordings.

References

Gales, Mark JF. Maximum likelihood linear transformations for hmm-based speech recognition. *Computer speech & language*, 12(2):75–98, 1998.

Moreno, Pedro J, Raj, Bhiksha, and Stern, Richard M. A vector taylor series approach for environment-independent speech recognition. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 2, pp. 733–736. IEEE, 1996.

Obuchi, Yasunari and Stern, Richard M. Normalization of time-derivative parameters using histogram equalization. In *Eighth European Conference on Speech Communication and Technology*, 2003.

Povey, Daniel, Ghoshal, Arnab, Boulianne, Gilles, Burget, Lukas, Glembek, Ondrej, Goel, Nagendra, Hannemann, Mirko, Motlicek, Petr, Qian, Yanmin, Schwarz, Petr, et al. The kaldi speech recognition toolkit. Technical report, IEEE Signal Processing Society, 2011.

Prasad, N Vishnu and Umesh, Srinivasan. Improved cepstral mean and variance normalization using bayesian framework. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 156–161. IEEE, 2013.

Viikki, Olli and Laurila, Kari. Cepstral domain segmental feature vector normalization for noise robust speech recognition. *Speech Communication*, 25(1-3):133–147, 1998.

Woodland, Phil C. Speaker adaptation for continuous density hmms: A review. In *ISCA Tutorial and Research Workshop (ITRW) on Adaptation Methods for Speech Recognition*, 2001.