INTERSPEECH 2009 BRIGHTON

Example-Based Speech Recognition using Formulaic Phrases

Christopher J. Watkins, Stephen J. Cox

School of Computing Sciences, University of East Anglia, Norwich, UK

cjw@cmp.uea.ac.uk, sjc@cmp.uea.ac.uk

Abstract

In this paper, we describe the design of an ASR system that is based on identifying and extracting formulaic phrases from a corpus and then, rather than building statistical models of them, performing example-based recognition of these phrases. We describe a method for combining formulaic phrases into a bigram language model that results in a 13% decrease in WER on a monophone HMM recogniser over the baseline. We show that using this model with phrase templates in the example-based recogniser gives a significant improvement in WER compared to word templates, but performance still falls short of the HMM recogniser. We also describe an LDA decision tree classifier that reduces the search space of the DTW decoder by 40% while at the same time decreasing WER.

10.21437/Interspeech.2009-564

1. Introduction

The data-driven, probabilistic-based paradigms that have been developed for ASR (notably the hidden Markov model (HMM) framework) were highly successful in advancing the technology during the 1980s and 1990s, but they contained simplifications and assumptions that are known to be untrue (for instance, the assumption that speech is produced by a first-order Markov process, or that language can be well modelled using n-grams of words). After years of intensive incremental development of the models, it seems that these modelling assumptions are now limiting the progress of ASR, and that these techniques may not be able to provide the leap needed to move ASR performance up to the level required for it to be usable in new applications.

One area in which the conventional approach to ASR may be questioned is in the use of statistical distributions. Clearly probability density functions (PDFs) constitute an essential element of hidden Markov modelling and provide a powerful method of generalising from seen to unseen data. However, the use of PDFs does represent a potential loss of information; the detail that is present in individual data samples is sacrificed in order to pool information in a controlled fashion. In ASR, this realisation has led to a resurgence of interest in example-based recognition systems [1, 2].

Another practice that simplifies conventional ASR is the use of fixed levels of description, which enables a hierarchical and modular approach to recognition in which words can be easily constructed as sequences of phonemes and language models as n-grams of words. However, there is ample psycho-linguistic evidence that humans recognise and generate a great deal of language in ready-made chunks, which have been termed "formulaic sequences" by some linguists [3]. These sequences "appear to be pre-fabricated, that is stored and retrieved whole from memory rather than being subject to generation or analysis by the language grammar" [3]. It has been argued that these phrases serve the important purpose of avoiding processing overload in both speaker and listener: the speaker retrieves them whole from memory, and the listener is more likely to understand a message if it is in a form that he/she has heard before. Analysis shows that much commonplace language is highly formulaic, and this is especially true in ASR applications where the application context is narrow and the discourse is goal-directed, factors that apply to most telephony ASR applications that provide information provision and booking services. Although many of these phrases have the status of "carrier" phrases, and as such have a low information content associated with them [4], recognition of them is essential for segmentation of the signal and extraction of the information content.

The example-based approach to recognition fits very well with the idea that the units used for recognition can be of different lengths: rather than insisting on modelling an utterance as a sequence of phonemes, we can adjust the lengths of our modelling units according to the examples we find. This has obvious benefits in capturing the acoustic/phonetic variation in commonly occurring fragments. However, in the example-based ASR approaches reported so far, the example unit has continued to be fixed to the phone [1]. In this paper, we describe the design of an ASR system that is based on identifying and extracting formulaic phrases from a corpus and then, rather than building statistical models of them, performing example-based recognition of these phrases. The structure of the paper is as follows: in section 2, we describe the identification and acquisition of formulaic phrases and cognitively-inspired techniques for combining the phrases with non-formulaic language. Section 3 outlines the principles of example-based speech recognition and also describes novel techniques for phrase template selection prior to recognition. In section 4, the data is described, and results from both the language-modelling and the speech recognition experiments are given. We end with a discussion and conclusion in section 5.

2. Language Modelling

2.1. Phrase Acquisition

A first task is to identify fixed phrases that occur often in the data. One approach is to iteratively combine the two vocabulary items in the training data that minimise perplexity, and enter the phrase as a new token in the vocabulary. The phrase can the be combined with other words and phrases in the subsequent iterations - the algorithm iterates until a convergence is met [5]. Another approach is to build a variable n-gram tree, and essentially store phrases within the tree for varying n [6]. We decide to adopt a multigram segmentation approach that uses the frequency of phrases, to perform a maximum-likelihood (ML) segmentation of the training text [7]. An initial set of phrases is found by making counts of all word sequences in the training data of length 1 to l: in our experiments, l was set to a maximum of 7. As an example, the utterance "can i get my balance" may be segmented into "[can i get] [my balance]".

2.2. Combining phrases with n-grams

The phrases themselves can be used to generate a language model, without the use of the phrase classes. Each phrase appears as an item in the lexicon, and an n-gram language model of phrases is trained in exactly the same way as one trains a word-level n-gram language model i.e. by counting n-grams of phrases and applying an appropriate smoothing technique (we used Good-Turing smoothing). A phrase-bigram (PB) model was trained in this way, as was a word-bigram (WB) for comparison purposes. The topologies of the WB and PB models are shown in Figure 1, although backoff and unigram transitions are omitted.

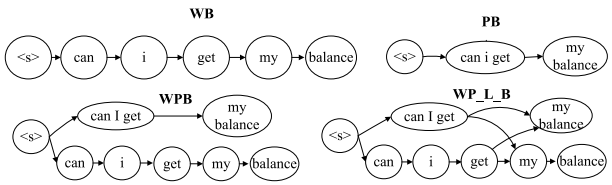

Figure 1: *Different language model topologies.*

A language model consisting of n-grams between phrases provides only an approximate generalisation to the unseen data (test-set), as many formulaic phrases allow variations within the phrases. For example, the following utterance occurs in the test-set, but not in the training-set: "hi can i get my membership card balance". It might be that in the training set, "hi" and "can i get my" have been seen, and thus the first part of the utterance is modelled accurately. But for the second part of the utterance, although "membership" occurs in the training-set, it does not do so in the context "my membership card balance": the closest matches in the phrase lexicon are "my card balance" and "my credit card balance". The introduction of "membership" into the utterance will make it impossible for the recogniser to decode it accurately.

One view from language theory is that humans use a *dual processing* [3] approach to production and perception of language. Language is constructed using a combination of formulaic phrases and constituents that are assembled at a higher level using grammatical principles. This process can be modelled using an extension to the word bigram language model, where word bigrams are combined alongside with the phrase bigrams in the language model. We term this extended model WPB (word bigrams + phrase bigrams), and it is shown schematically in Figure 1. WPB allows phrases and words to be combined in the decoding: however, a disadvantage of this approach is that to allow a transition from a phrase to a word (and vice versa) requires that the decoder "backs off" to the unigram probability. Consider, for instance, the WPB model, shown in Figure 1, processing the input utterance "can i get my statement". If the current state of the decoder is at the end of the word "get" within the phrase model for "can i get", the decoder must then exit the phrase model and move to the first state of the word model for "my" if it is to successfully decode the input (this presumes that the bigram "my statement" was seen in the training-set). The only way it can do this is by "backing-off" and using the unigram transition (not shown in Figure 1).

Our solution to this is a technique that we term the *word phrase link bigram* (WP_L_B). This model includes word and phrase bigrams as before, but also includes bigrams of the final words of phrases and the phrases themselves (this can also be extended to higher order n-grams by taking the $n-1$ words before and after a phrase). For the example in Figure 1, this model now provides a transition from "can i get" to "my", and also from "get" to "my balance".

3. Example-Based Recognition

De Wachter et al. have adapted traditional template-based dynamic programming (DP) techniques to deal with large-vocabulary speaker-independent recognition and hence compete in performance with current HMM based systems [1]. In our work, we have used two of these methods: a new class-based distance measure, and the *time filter* algorithm (see Section 3.1). The class-based distance measure, which is an adapted version of the Mahalanobis distance, uses a HMM state alignment to group reference frames into classes, from which a diagonal covariance matrix is estimated (see [1]).

3.1. Acoustic pre-selection of possible template candidates

Even for a small dataset, using phrase templates makes the search space for a Viterbi decoder (with DTW) vast: for example, the vocabulary size of our data (Section 4.1) is approximately 1500 words, which results in about 65,000 templates. Using this search space would be similar to using 65,000 HMMs in a conventional system, whereas actual current systems use a few thousand context-dependent phoneme models. De Wachter et al. have developed the *time filter* algorithm [1] which is an acoustic pass over the input that seeks aproximate diagonal matches i.e. it attempts find potential templates for the decoding of the input, with relaxed constraints, resulting in a large reduction in the search space.

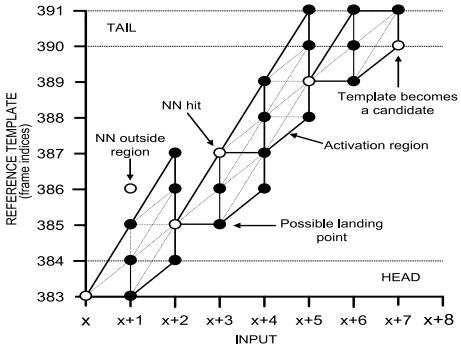

Figure 2: *Time filter algorithm. The NN hits create four activations, terminating in the tail - the template becomes a candidate.*

The time filter algorithm works first by finding the *k nearest neighbours* (KNN) of the reference database to the current input frame. De Wachter et al. use a KNN Roadmap [1], which is organised by frame class (from the HMM state alignment). We have replaced this step with vector quantisation (VQ): each class is quantised into a certain number codewords, with the mean frame used as the comparison to the input frame. Codewords are ranked in order of closeness to the input frame, with each frame in the closest codewords loaded into the KNN list until K is reached. The nearest neighbours (NN) generate a sequence of activation regions within a template (Figure 2), and if these sequences cover the length of a template $\vec{Y}$, then the template becomes a candidate for recognition, for a start time t with an acoustic distance $D(\vec{Y})$. The reader should refer to [1] for more details.

The time filter algorithm has, to-date, only been applied to phoneme templates [1]. Our novel application to phrase-length templates, requires a further extension. Because the templates are so varied in length (a short template representing "a" might be 5 frames long, while a longer template representing "i would like to know" might be 70 frames long), a normalisation method is needed for their acoustic distance, which is accumulated over the time filter process. As the time filter algorithm is based on the DTW, a template's distance is based on the distance between input frames and template frames, as well as additional penalties for skips and stalls. This means that longer templates will garner more penalties as the input is processed (unless they are perfect matches). Our solution to this problem is the normalised distance, $D_\lambda(\vec{Y})$:

$$D_\lambda(\vec{Y}) = \frac{D(\vec{Y})}{|\vec{Y}|(1+e^{\lambda|\vec{Y}|})} \quad (1)$$

where, $\vec{Y}$ is the reference template, $|\vec{Y}|$ represents the length of template $\vec{Y}$, $D(\vec{Y})$ is the pre-normalised acoustic distance of $\vec{Y}$, and λ is a user-defined weight. The normalised distance measure uses a form of the *sigmoid* function [8], of general form $S(t) = 1/(1+e^{-t})$, where the weight λ can be used to define the smoothness of normalisation i.e. the growth of the normalising factor as the template length increases. A typical value for λ in our experiments is 0.05.

3.2. An LDA-based filtering of template candidates

As a post-processing stage, the template candidates are filtered using a process that combines a 2-level *decision tree* with *linear discrimant analysis* (LDA) [9], in an attempt to increase the ratio of correct to incorrect templates for any utterance. First of all, the time filter is run on the training utterances, and a 3-d feature vector is extracted for each template candidate (defined in Section 3.2.1) for all training utterances (resulting in approximately 55 million training vectors for our data). The training vectors are labelled as either *correct* or *incorrect* (i.e. a two-class problem), by comparing the start times and template labels (e.g. "i'd like to") to the annotation of the training utterances (from a HMM flat start). We then apply the LDA twice to the training vectors using a decision tree (see Figure 3) and then transform the training vectors to a 1-d space. The threshold for classifying templates as correct or incorrect can be placed anywhere in this space to control acceptance/rejection of candidate templates: in the experiments reported here, it is placed at the optimum decision point where the distributions cross. To filter the test candidates, features are extracted, and both sets of eigenvectors and thresholds from the training sequence are used to transform the test vectors and classify them using the same decision tree.

3.2.1. Features for LDA

For this initial work, we have chosen a 3-d feature vector to represent each template activation candidate from the output of the time filter algorithm. The first feature is a count-based probability, and a variant of the relative frequency probability:

$$Pr(\vec{Y}|t) = \frac{C^*_{window}(W_1, t)}{C^*_{window}(W \in V, t)} \quad (2)$$

where, $\vec{Y}$ is the template candidate, t is the start time (frame number) of template $\vec{Y}$, W_1 is the first word of $\vec{Y}$ which has a label string defined by W_1^j where j is the number of words that $\vec{Y}$ represents, and $W \in V$ represents any template label string in the vocabulary V. The C^*_{window} function counts template labels within a window of size l, centred on time t, and weights the counts using an un-normalised Gaussian radial basis function (RBF), where the mean, μ, of the Gaussian is at time t (at the center of the window), and the standard deviation, σ, is user-defined (typically $\sigma = 3$ in our experiments).

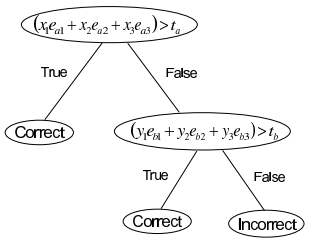

Figure 3: *The LDA decision tree. The LDA is first applied to all training vectors, $\vec{x}$, then to all vectors $\vec{y} \in \vec{x} < t_a$. $\vec{e}_a$ and $\vec{e}_b$ are the eigenvectors for each LDA.*

The second feature, is the *z-score* of the template distance scores, defined as $z(\vec{Y}) = (D_\lambda(\vec{Y}) - \mu_{D_\lambda})/\sigma_{D_\lambda}$, where the μ_{D_λ} and σ_{D_λ} are calculated over the entire time filter output for an utterance. The third feature is the length of the template activation, $L(\vec{Y})$, giving the feature vector for template $\vec{Y}$, at time t as $F(\vec{Y}, t) = [Pr(\vec{Y}|t)\ z(\vec{Y})\ L(\vec{Y})]$.

4. Experiments and Results

4.1. Data and Experimental Procedure

The data used in this study consisted of transcriptions of telephone calls to an experimental call-routing system. Customers were invited to call up the system and to make the kind of enquiry they would normally make when talking to an operator. Only their initial, query utterance was transcribed. The transcriptions were divided into a training-set of 4800 utterances and a testing set of 1000. The training set vocabulary size is approximately 1500 words, and the test-set size is 900 words after removal of any utterances that contained out-of-vocabulary (OOV) words. The utterances themselves were of low quality because of factors such as restricted bandwidth, noise and distortions. Because the low accuracy obtainable from recognizing this material could disguise the effects of language modelling that we were investigating here, a single speaker re-recorded the transcriptions using high quality recording equipment - all original disfluencies in the speech, such as pauses, repetitions, and grammatical errors, were retained in the recordings. The training-set utterances consisted of approximately 3.5 hours of speech and the test-set 40 minutes.

The acoustic models for the HMM speech recognition experiments were built using HTK. They were speaker dependent models using three state monophone HMMs with 20 Gaussian mixture components per state. Templates for the example-based recogniser are defined by the multigram segmentation of Section 2.1, leading to approximately 65k phrase templates. We used the CMU toolkit [10] to train all n-gram probabilities for the language models, and in our experiments, we used bigram estimation - trigrams actually gave an increase in word error rate (WER) when tested on the baseline HMM recogniser.

4.2. Results and Discussion

Lang. Model	Train WER	Test WER	Rel. Reduction
WB	8.43	14.87	n/a
PB	3.29	14.82	-0.34
WPB	2.16	15.46	+3.98
WP_L_B	4.05	12.94	-12.98

Table 1: *WER for different language models using phoneme HMM recognition. Relative reduction is on the test data and w.r.t. the WB baseline.*

Table 1 gives WER for WB, WPB, and WP_L_B using the phoneme HMM recogniser. WP_L_B outperforms all other methods on the test-set, and the matched-pairs test [11] confirms that it is a statistically significant improvement over all other methods reported. WPB decreases the WER on the training-set over the WB and PB models, but gives a reduction in performance on the test-set: this may be due to unseen phrases in the test-set that cause the decoder to favour the back-off connections between phrases and words. WP_L_B shows an increase in WER on the training-set, unlike PB and WPB, but it decreases on the test-set: the increase on the training-set is most likely due to the increased perplexity (i.e. there are more connections between states, see Figure 1) of WP_L_B compared with PB and WPB, but this richer topology leads to better performance on unseen data.

	Phrase Templates	Word-only Templates
class. acc %	66.96	44.31
CER %	3.99	0.00094
IER %	49.12	97.46
Temps. Kept %	62.77	80.40
WER %	19.93	22.24
Unfilt. WER %	20.59	22.14

Table 2: *LDA filter results on test data for phrase and word template recognisers using WP_L_B model.*

Table 2 shows the performance of the LDA filter for phrase templates and word only templates. The word-only templates, although having an excellent correct error rate (CER), have a very high incorrect error rate (IER) which means that most incorrect templates are misclassified as correct, which is confirmed by the fact that only about 20% of the templates are filtered out. For the phrase templates, although the CER is slightly higher, the IER is much lower (49%), which results in almost 40% of the templates being filtered, and an increased classification accuracy. It can be seen that reducing the number of templates for the phrase-based recogniser has had positive effect on WER, with a relative reduction of -3.21% w.r.t. the unfiltered templates, even though 4% of correct templates are lost, while for the word-based recogniser, the WER actually increases—we assume that the template length feature for the word-template filter has a negligble effect, and may be a factor in the reduction in performance.

Finally, Table 3 summarises the best WERs from our experiments and gives the relative WER w.r.t to the baseline measure, the word bigram HMM recogniser. Both example-based systems have WERs that are higher than the HMM recogniser (a result also observed in [1]), which might be expected at an early stage of developing these techniques. However, it is encouraging that using phrase templates gives a very significant improvement over the word-template recogniser.

	WER	Rel. reduct.
Base HMM	14.87	n/a
WP_L_B HMM	12.94	-12.98
Word template (unfilt.)	22.14	+48.89
Phrase template (filt.)	19.93	+34.03

Table 3: *WER Comparison for the recognition systems.*

5. Conclusions

This paper has described an ASR system that uses phrases in an example-based framework. Although its performance is some way short of the performance of a conventional HMM recogniser, we have shown that using phrase templates does, in fact, increase performance compared to a word template recogniser. We have also shown that reducing the number of template candidates using an LDA decision tree improves WER.

Future improvements to this work can be made on both language and speech components. The inherent problem in modelling phrases is small variations—quite often a phrase will vary by one or two words because of disfluencies or different contexts of use, and this ensures that it cannot be decoded correctly in the current system. For the language component, a system which allows open slots within phrases for variable words could guide the selection of speech units for the recogniser. We can also seek to improve the template filter by choosing better features—using the prior probability of a template, based on the number of examples of a phrase in the reference database is a starting point. In the long-term, a successful recognition strategy may be to combine the modelling power of conventional HMMs with the ability of example-based systems to decode longer span events [1].

6. References

[1] M. D. Wachter, M. Matton, K. Demuynck, P. Wambacq, R. Cools, and D. V. Compernolle, "Template-Based Continuous Speech Recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 15, no. 4, pp. 1377–1390, May 2007.

[2] G. Aradilla, J. Vepa, and H. Bourlard, "Improving Speech Recognition Using a Data-Driven Approach," in *Proc. of Interpseech*, 2005, pp. 3333–3336.

[3] A. Wray, *Formulaic Language and the Lexicon*. Cambridge University Press, 2002.

[4] Q. Huang and S. Cox, "Task independent call routing," *Speech Comm.*, vol. 48, no. 3–4, pp. 374–389, 2006.

[5] E. P. Giachin, "Phrase Bigrams for Continuous Speech Recognition," in *Proc. of ICASSP*, 1995, pp. 225–228.

[6] M. Siu and M. Ostendorf, "Variable N-Grams and Extensions for Conversational Speech Language Modelling," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 8, no. 1, pp. 36–75, 2000.

[7] S. Deligne and F. Bimbot, "Language Modelling by Variable Length Sequences: Theoretical Formulation and Evaluation of Multigrams," in *Proc. of ICASSP*, 1995, pp. 169–172.

[8] T. Mitchell, *Machine Learning*. WCB-McGraw-Hill, 1997, ch. 4.

[9] A. Webb, *Statistical Pattern Recognition*. Wiley, 2002, ch. 4.

[10] P. Clarkson and R. Rosenfeld, "Statistical Language Modeling Using the CMU–Cambridge Toolkit," in *Proc. of Eurospeech*, 1997, pp. 2707–2710.

[11] L. Gillick and S. J. Cox, "Some statistical issues in the comparison of speech recognition algorithms," in *Proc. of ICASSP*, vol. 1, 1989, pp. 532–535.