THE UNIVERSITY of EDINBURGH

Deliverable D3.4

Report describing final version of deep layered models

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 287678.

Participant no.	Participant organisation name	Part. short name	Country
1 (Coordinator)	University of Edinburgh	UEDIN	UK
2	Aalto University	AALTO	Finland
3	University of Helsinki	UH	Finland
4	Universidad Politécnica de Madrid	UPM	Spain
5	Technical University of Cluj-Napoca	UTCN	Romania

Project reference number	FP7-287678
Proposal acronym	SIMPLE4ALL
Status and Version	Complete, proofread, ready for delivery: version 1
Deliverable title	Report describing final version of deep layered models
Nature of the Deliverable	Report (R)
Dissemination Level	Public (PU)
This document is available from	`http://simple4all.org/publications`
WP contributing to the deliverable	WP3
WP / Task responsible	WP3 / Task T3.4
Editor	Oliver Watts (UEDIN)
Editor address	owatts@staffmail.ed.ac.uk
Author(s), in alphabetical order	Rob Clark, Simon King, Srikanth Ronanki, Oliver Watts
EC Project Officer	Pierre Paul Sondag

Abstract

This document describes work on three different approaches to overcoming the following problems of conventional HMM based speech synthesis. Firstly, that known linguistic structure (such as syllable structure) is not exploited directly in acoustic modelling. Work on modelling F_0 separately from spectrum in a multi-rate model tries to address this issue. Secondly, decision tree clustering is an inefficient way of tying model parameters. Work on supplementing decision tree based models with multiple regression transforms aims to alleviate this problem. Thirdly, the textual/linguistic and acoustic components of TTS systems are conventionally learned independently of one another. Work on using deep neural nets to learn representations of linguistic objects jointly with predictors of speech features is presented which aims to address to this problem.

Contents

1 Introduction

The motivation for the work in WP3 remains the same as outlined in D3.2. Briefly, the most widely-used method in statistical parametric TTS for incorporating linguistic knowledge into acoustic models is to run input text through a TTS front-end, then flatten any linguistic structure produced into a sequence of highly-context-dependent model names. Most of these model names are unique in any given training corpus, and many unseen ones are encountered at synthesis time, and so decision tree-based parameter sharing is applied to cluster similar states for better parameter estimation, and to determine appropriate parameters to use for unseen contexts at synthesis time. This approach is unsatisfying for a number of reasons.

First, it seems clear that the hierarchical phrase- and syllable-structures (for example) that are inherent in the linguistic structure of utterances should be exploited more directly in the estimation of acoustic models. We address this in work where we model different aspects of speech with different sub-models which operate at different temporal scales, described in Section 2. Specifically, we describe a method to combine F_0 generated by a syllable-level model with spectral features generated by a phone-level model.

Second, the greedy partitioning strategy employed by decision trees fails to make best use of data for modelling factorial phenomena. Section 3 describes initial work towards supplementing decision tree clustering with multiple regression matrices to integrate certain linguistic / textual features by other means.

Third, the conventional TTS approach of training the various text analysis and acoustic modelling components of the system independently means that a lot of expert knowledge is needed to specify the interface between the front- and back-end and, even then, there is no guarantee that the features output by the front end are optimal for – or even relevant to – the acoustic modelling task. Section 4 describes work which aims to overcome these problems by adopting a modelling approach which allows representations of textual units to be obtained which are guided by – and in some sense therefore optimal for – the prediction of speech features. In the work reported there, these features are annotated phrase-breaks, but the next goal is to predict acoustic features directly.

2 Explicit modelling of syllable structure

We here outline some work investigating alternative formulations of the pitch and duration models within HMM-based speech synthesis, specifically investigating models that model prosody using named syllabic contexts rather than named subsyllabic segments. The work described here was partially carried out by a SIMPLE4ALL intern at UEDIN : Srikanth Ronanki (IIIT, Hyderabad, India). In this approach, the outputs of multiple models generating at different timescales are combined. The models are still trained independently, however, so future plans include investigating ways of jointly training F_0 models that have a different underlying structure to the segment models, perhaps in a similar way to multi-rate and variable-rate modelling [2] which has been applied to ASR. This topic also features within a recently-funded SNSF (Swiss National Science Foundation) project, on which a PhD student at UEDIN has recently started (from September 2013).

2.1 Syllable-level system

An HTS system was trained using identical recipes to the ones we conventionally use to train phone-level models. The only difference was the use of syllable-level labels, which included the subset of standard HTS features that are not phone-specific, plus an extra *monosyllable* feature. These initial classes are used like monophone models in conventional HTS training recipes: they are designed to provide 'good enough' initial models that a reasonable alignment with the training data can be obtained. It is then possible to cluster them using the full set of contexts and the standard decision tree-based clustering algorithm.

The number of unique syllables as described by the concatenation of their phone sequences can vary from 1000 to 5000 depending on the language. The *monosyllable* inventory of basic syllable types is designed to be much smaller than this so that syllable models can be initialised. To produce monosyllable labels, syllables are divided

into the classes defined by concatenating the binary linguistic features listed in Table 2.1a, which are obtained from standard HTS contexts.

Table 2.1a: Features used for characterising syllables in initial models.

Features	Representation
Stress	(0/1)
Accent	(0/1)
Word Initial	(0/1)
Word Final	(0/1)
Phrase Initial	(0/1)
Phrase Final	(0/1)

Each unique combination of values is given an arbitrary name (*Syll1*, *Syll2*, etc). Table 2.1b shows an example of monosyllables for several English words. Aside from the level at which modelling takes place and the need for these extra syllable classes, training is identical to normal phone-level HTS training. The standard configuration of acoustic feature streams and HMM topology are used.

2.2 System combination

Given a syllable-level system trained as described above and a standard voice with phone-level units, we synthesise in a hybrid fashion, using the phone models to produce spectral features and the syllable-based one for prosodic features. Our first method to synthesise speech from text in this way makes use of a dynamic programming algorithm. The algorithm aligns the spectral features from source (phoneme) to target (syllable) spectral features. The F_0 contour taken from the syllable models is combined with the aligned spectral features from the phone models, and together these set of acoustic features are passed as input to either an MLSA filter or the STRAIGHT vocoder to synthesize speech. Durations are taken from the phone-level model. The process is depicted in Figure 2.2a.

The sub-models of the hybrid system described were trained on approximately 90 minutes of speech taken from each of the four audiobooks released for the Blizzard challenge 2012. The phone-based sub-model was also used in the normal way to generate conventional HTS speech for comparison. The two systems were evaluated objectively using both RMSE and correlation of synthetic F_0 contours against those of held-out natural speech utterances. Results are shown in Table 2.2a. A subjective evaluation is now in progress.

3 Multiple Regression HMM for integration of continuously-valued representations of text into an HMM-based speech synthesiser

In earlier work we have developed TTS systems which make use of high-dimensional vector representations of words and other textual units. An attractive property of such representations is that they can be learned in an unsupervised manner from unannotated text. This allows us to exploit potentially vast amounts of electronic text to

Table 2.1b: Example of monosyllables

Words	Syllables	Feature representation	Segments
All	(olw)	(010101)	Syll23
librivox	(lib, rIv, Qks)	01010101	Syll8, Syll12, Syll7
recordings	(rI, kod, INz)	01010101	Syll4, Syll3, Syll42

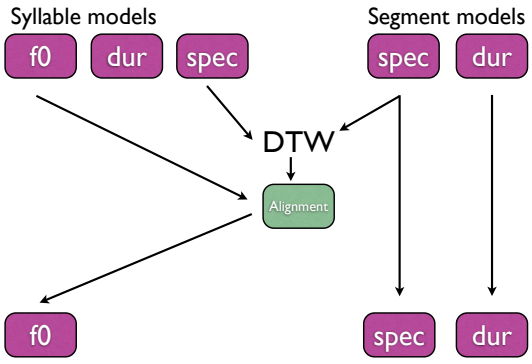

Figure 2.2a: *Current generation from syllable models.*

Table 2.2a: Results of objective evaluation

Metrics	Phone-based system	Hybrid system
RMSE	16.62	16.07
CORR	0.61	0.60

create representations which – on some tasks at least – have been shown to be effective substitutes for part-of-speech tags, for example.

When used directly as context features for acoustic modelling – which involves partitioning the vector space using decision trees – the usefulness of these features diminishes. We hypothesise that this is due firstly to the fact that decision tree clustering for state tying effectively causes the continuous features to be discretised. Secondly, the features themselves contain substantial amounts of noise and are somewhat redundant and this type of feature is known to degrade the quality of decision trees, which are rather weak feature selectors.

It was therefore decided to experiment with an alternative means of integrating these vector space model features into acoustic models: they are to be used as a word-level 'exogenous variable' in a Multiple Regression HMM (MRHMM). The MRHMM has been used in the past for TTS as a way of manipulating articulation, and also speaking style, by introducing utterance-level exogenous variables representative of these things. This would the first time to our knowledge that the MRHMM has been used to integrate *lexical* information.

In D3.2, it was noted that a version of HTS extended with MRHMM capability being prepared external to SIMPLE4ALL was due to be publicly released as part of HTS 2.3 in December 2012. Since D3.2, this plan was changed, and so instead some effort within SIMPLE4ALL was necessary to merge an existing MRHMM implementation into HTS 2.3, and this is planned for public release as part of HTS in December 2013. Some work on the code still remains to be done before it can be used to handle sub-utterance level contexts: as it stands, the current code can only deal with utterance-constant control variables, suitable for representing such factors as speaking style. However, MRHMM technology (and the software implementation in HTS 2.3) forms as a key element of the recently-started INSPIRE ITN (`http://www.inspire-itn.eu`), and we are beginning mutually-beneficial co-work with the new INSPIRE-funded researcher at UEDIN (Gustav Henter) to extend the codebase. The plan is then to use the code to 'steer' the synthesiser using numerical word-representations as control variables. We anticipate having results in time to submit a paper to Interspeech 2014.

4 Neural net word representations for TTS

This work has been submitted to ICASSP 2014: the paper is appended to the current document, and will be summarised briefly here.

As noted already, previous work of ours has used of high-dimensional vector representations of words as a substitute for conventional part of speech tags, for the task of predicting phrase-breaks from text for TTS. The paradigm used is a semi-supervised one: word representations are learned from unannotated text in an initial unsupervised step, and then used as independent variables for a classifier in a second step of supervised learning. The weakness of this approach is the separation of these 2 steps: there is no guarantee that the representations learned in the first step will be useful for the second.

To address this weakness, we have experimented with a *multitask learning* approach to mitigate the effect of this separation. First, a neural net language model of the type described in [1] is trained; this corresponds to our initial unsupervised step of learning a vector space model (VSM) from text. Then, the word embeddings learned as a by-product of the language modelling task are tuned on the supervised task of phrase-break prediction. The separation between the 2 stages is lessened by allowing the representations themselves to be influenced by the supervised task: our hypothesis was that this would lead to a performance improvement. To test this, several experimental systems were built (alongside several benchmarks), including:

System U This is the neural net system most similar to our previous work with VSMs: word representations were learned distributionally on the language-modelling task but not updated on the supervised task.

System F This system updates the word representations learned on the language-modelling task on the supervised task: we hypothesised that this would improve performance relative to system U.

System R This system is designed to test the importance of pre-training the word representations on the language modelling task: word representations are randomly initialised and learned only on the supervised task, without first training the language model.

The systems were evaluated on a task of interest we have used before: prediction of phrase-breaks, as hand-annotated in the MARSEC corpus [7]. Evaluation did not support our hypothesis: fine-tuning on the phrase-break task actually worsened performance, with system U outperforming system F. An explanation for this is that, because of the small size of the MARSEC corpus and the small fraction of word types that are represented in it, fine-tuning only a subset of the large language model weights on the phrase-break task leads to inconsistencies between the parts of the model that have been fine-tuned and those that have not. Ongoing work seeks to test this explanation, and if it proves to be correct, to alleviate the problem by, for example, tuning a shared-weight adaptation layer instead of the word representations themselves [8, 6], or by using a different training curriculum, such as one where batches of phrase-break examples and language modelling examples are interleaved and presented to the system during training.

However, the original hypothesis that system F would outperform system U is only one facet of this experiment. There is still the very positive result that system U outperformed all our previous experimental systems (using VSMs) trained with the same amount of text data, and performed as well as the best benchmark systems, showing the usefulness of representations learned within a neural net language model for TTS tasks.

Given this positive result, the larger plan is therefore to develop a TTS system with shared representations for all units of interest (letters, phonemes, words, phrases, etc.) which are trained jointly with all system components (relating both to text processing and acoustic modelling). The infrastructure needed to conduct extensive experimentation with neural network models is coming in to place through a collaboration between SIMPLE4ALL and several other projects at UEDIN , where expertise in the software environment and the hardware for GPU-based computation are rapidly coming together.

We have already taken advantage of these overlapping interests in some initial experiments with neural networks for *acoustic* modelling [4]. This is important groundwork for the next phase, in which we plan to train our shared representations of textual units to optimise acoustic-based objective functions.

5 Adaptation techniques for paralinguistic contexts

In [3] we describe an alternaive way of handling contexts – in this case, paralinguistic rather than linguistic – by a cascade of jointly-learned transforms. Using speaker adaptive training with cascaded transforms allows us to factor out speaker and speaking style differences. We note that – in the same way as for the MRHMM – this approach could be used more generally to handle not only paralinguistic contexts, but also linguistic ones.

[5] describes some initial attempts at predicting sentiment polarity from text, which would allow us to control speaking style in a TTS system. All this work will be described more fully in D5.2, where it belongs.

References

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[2] O. Cetin and M. Ostendorf. Multi-rate and variable-rate modeling of speech at phone and syllable time scales. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, volume 1, pages 665 – 668, 18-23, 2005.

[3] Jaime Lorenzo-Trueba, Roberto Barra-Chicote, Junichi Yamagishi, Oliver Watts, and Juan M. Montero. Towards speaking style transplantation in speech synthesis. In *8th ISCA Workshop on Speech Synthesis*, pages 179–183, Barcelona, Spain, August 2013.

[4] Heng Lu, Simon King, and Oliver Watts. Combining a vector space representation of linguistic context with a deep neural network for text-to-speech synthesis. In *8th ISCA Workshop on Speech Synthesis*, pages 281–285, Barcelona, Spain, August 2013.

[5] Ioana Muresan, Adriana Stan, Mircea Giurgiu, and Rodica Potolea. Evaluation of Sentiment Polarity Prediction using a Dimensional and a Categorical Approach. In *Proc. of the 7th International Conference on Speech Technology and Human-Computer Dialogue (accepted)*, October 2013.

[6] Junho Park, Xunying Liu, Mark J. F. Gales, and Philip C. Woodland. Improved neural network based language modelling and adaptation. In *INTERSPEECH*, pages 1041–1044, 2010.

[7] Peter Roach, Gerry Knowles, Tamas Varadi, and Simon Arnfield. Marsec: A machine-readable spoken english corpus. *Journal of the International Phonetic Association*, 23(2):47–54, 1993.

[8] Holger Schwenk. Continuous space language models. *Computer Speech & Language*, 21(3):492 – 518, 2007.

Appendix: Submitted Conference Paper

NEURAL NET WORD REPRESENTATIONS FOR PHRASE-BREAK PREDICTION WITHOUT A PART OF SPEECH TAGGER

O. Watts, S. Gangireddy, J. Yamagishi, S. King, S. Renals

The Centre for Speech Technology Research
University of Edinburgh, UK

A. Stan, M. Giurgiu

Communications Department
Technical University of Cluj-Napoca, Romania

ABSTRACT

The use of shared projection neural nets of the sort used in language modelling is proposed as a way of sharing parameters between multiple text-to-speech system components. We experiment with pretraining the weights of such a shared projection on an auxiliary language modelling task and then apply the resulting word representations to the task of phrase-break prediction. Doing so allows us to build phrase-break predictors that rival conventional systems without any reliance on conventional knowledge-based resources such as part of speech taggers.

Index Terms— Speech synthesis, TTS, unsupervised learning, neural net language modelling, multitask learning.

1. INTRODUCTION

Neural networks (NNs) have re-emerged as a popular paradigm for the construction of text-to-speech (TTS) systems in recent years [1, 2, 3], much of their popularity being due to their successful use in learning 'deep' representations of data. In TTS (as in related work in speech recognition [4]) the emphasis has been on learning representations of acoustic data. For example, [3] uses Restricted Boltzmann Machines to operate directly on spectral envelopes, in effect replacing conventional representations of speech derived with expert knowledge (mel cepstral coefficients, line spectral pairs, etc.) with ones learned by the deep model. In this paper we focus instead on the text part of TTS, which has so far received less attention, and treat NNs as a way of obtaining representations of text which are optimised directly for the prediction of speech features.

Our previous work [5, 6] on obtaining features in an unsupervised way for use in TTS systems adopted a vector space model (VSM) approach, using a two-stage approach to semi-supervised learning. In the first – unsupervised – stage, we extract high-dimensional continuous-valued features to characterise textual or linguistic units of interest (letters, phonemes, words, utterances, etc.). This is done by compiling matrices of cooccurrence counts and then applying low rank matrix factorisation to obtain lower-dimensional distributional representations of the items in question. Then in a second supervised step, these features are used in place of conventional features (part of speech (POS) tags, phonetic categories etc.) as predictors for various supervised tasks such as phrase-break prediction and acoustic state clustering. The advantage of this approach is that the first stage allows the system to take advantage of large quantities of unannotated text. However, although we have shown empirically that the representations obtained in the first step in many cases improve the predictors trained in the second, the weakness of this approach is that the two steps are nonetheless unlinked, and there is no guarantee that the features from the unsupervised phase will be useful in the supervised one.

In the current work, we investigate the use of *shared projection feed-forward NNs* of the sort that have become popular for language modelling [7] to learn word representations. This architecture has the potential to overcome the problems of our VSM approach as it allows word representations to be learned jointly with a classifier on some supervised task of interest. However, we want to retain the benefit of being able to exploit large text resources which has been a key feature of our previous work, and so consider an approach similar to that of [8, 9]. There, word representations obtained within a neural network language model are subsequently used for a variety of natural language processing tasks in a *multitask learning* (MTL) framework. The idea of MTL [10] is to train predictors for several related tasks in such a way that some parameters of the various task-specific models are shared, and can therefore be estimated on more data. Classically, some classifiers are devoted to *auxiliary tasks*: the predictions of these classifiers are of no direct interest, and they are learned purely to improve the estimation of the shared parameters. In [8, 9] the shared parameters are word representations from a NN language model (NNLM): the language modelling task is an auxiliary task whose only role is to initialise word representations which are subsequently refined on the other tasks. This is an attractive paradigm for TTS where multiple system components are trained on small amounts of expensive manually-labelled (usually disjoint) data and optimised in isolation. One goal of our on-going work is to develop a TTS system with shared representations for all units of interest (letters, phonemes, words, phrases, etc.) which are trained jointly with all system components (relating both to text processing and acoustic modelling) in the manner of [11].

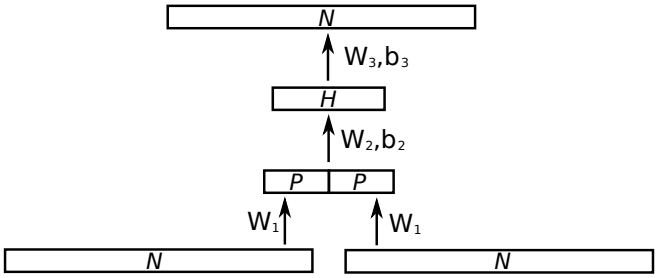

Fig. 1. Topology of neural net language model used.

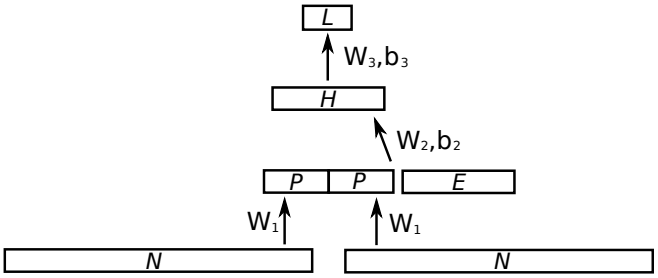

Fig. 2. Topology of phrase-break predictor used.

The scope of the current work is limited in comparison to the longer-term plans outlined in the previous paragraph, but is a first step in that direction. In [5] we took the task of phrase-break (PB) prediction for rapid testing of VSM features before applying them to other tasks in later work. Here we take the same task: its credible objective measure and the small size of its training data enable a faster turnaround of experiments than would be allowed, for example, by the training and subjective evaluation of acoustic models. We take advantage of the relatively small scale of the PB task to pose some questions which it seems desirable to answer before progressing to more complex systems. Firstly, are features taken from a standard feed-forward NNLM equally as useful for our task as VSM features were previously shown to be? Secondly, how big an increase in performance is afforded by updating word representations on the supervised task? Conversely, what happens if we learn representations from scratch on the supervised task?

2. SHARED PROJECTION FEED-FORWARD NNS

What we term a *shared projection feed-forward NN* was used for letter-to-sound conversion in [12], and has become popular for language modelling [7, 13, 14]. In all cases, shared weights in an initial hidden *projection layer* (in [12] termed a *self-organising code layer*) map from orthogonal inputs encoding the identities of textual units (letters, words) in some history or context to continuous representations of them. The fact that weights are shared between all units in the context or history means that the representations of those units are invariant to their positions in the context or history. Furthermore, the representations are learned along with the other parameters of the network so that the letter/word representations which are obtained are optimised on the task of interest (predicting the correct phoneme in [12], predicting the following word in [7]). In [12] a direct comparison is made with the knowledge-based coding of letters used in the NETSpeak system: the learned representations outperformed the knowledge-based coding.

Figure 1 shows the topology of a feed-forward trigram NNLM. N denotes the size of the vocabulary: input consists of a $2N$ sparse vector encoding the 2 word trigram history. Shared weights $\boldsymbol{W_1}$ transform the inputs for individual tokens into their P-dimensional representations: the concatenation of the representations of the words in the n-gram history constitute the output of the projection layer. A hidden layer with H units and a non-linear activation function transforms the outputs of the projection layer. Finally, an N-dimensional output layer produces n-gram probabilities for each word in the vocabulary given the input history. Use of the *softmax* normalisation function to constrain the network's outputs to be in the range [0, 1] and to sum to 1 ensures a valid probability distribution.

There is nothing about this model topology which restricts it to language modelling: as already mentioned, in [12] it is used for LTS conversion. We here use the slightly modified version of it shown in Figure 2 for phrase-break prediction. Here, two words of context are also used, although in the current experiments these are the words preceding and following a possible break, in contrast to the language model history. Indeed, the use of the shared projection means that words' representations are invariant to position in context, so that arbitrarily different contexts (in terms of the words' relative position to the target and their number) can be used for different tasks within the same experiment. The reason we restrict the context to preceding and following words is to ensure results which are comparable to those of previous work, where the same context was used. Another difference between the language model and PB predictor is that the size L of the output layer is not the size of the vocabulary, and is determined by the number of possible values of the variable to be predicted. In the experiments presented here, L is 2 to encode *break* and *no break* (which we note is equivalent to using a single sigmoidal unit). A final difference is that an extra E inputs are fed directly into the second hidden layer: these extra inputs encode additional contextual information beyond the 2 words' context such as distance to punctuation (see the following section for details).

3. EXPERIMENTS

3.1. Data

The data used for the supervised part (PB prediction) of present experiments is identical to that described in [5]: a subset of 39 stories from the SEC/MARSEC corpus [15, 16, 17] which consists of annotated radio broadcast transcriptions. Punctuation marks and two levels of PB (break/no break) were associated with the token that precedes them as a feature of that token. The data consist of c.35,000 non-punctuation tokens; 11% are from the 'overlap' section (annotated prosodically by both the corpus's transcribers) which we use as in [5] as a test set. No single fixed development set was used in

Table 1. Summary of benchmark systems built

Neural net systems	Decision tree systems	Feature description
B	B′	**B**asic
G	G′	**G**uessed POS
T	T′	**T**opline POS

the current work as in [5]; instead, partition into training and validation sets was carried out randomly per model built, as explained in Section 3.2.

For the unsupervised pretraining part of the present experiments (NNLM training) we used the same 1.2 million tokens of Wall Street Journal text from which VSM features were obtained in [5]. The tokenised text was lowercased, but no sophisticated text normalisation was applied. A small set of tokens was found using the procedure described in [5] and rewritten with the <*unk*> token used to handle unseen words at validation and run time.

3.2. Systems built

Benchmark systems Table 1 summarises the benchmark systems used in this work. Systems B, G and T are NN benchmark systems which in terms of the features they use are directly comparable with decision tree-based systems of the same names from previous work [6], which are here denoted B′, G′ and T′ to disambiguate.[1] None of systems B, G or T in effect makes use of the network's projection layer – for those systems, an input identical for each example is fed into the two words' context input of the NN, in effect resulting in a standard feed-forward NN with a single hidden layer.

All systems make use of the following basic punctuation and positional features:

- The identity of the word's punctuation symbol;
- The number of words {since, until} a word with a *strong punctuation* mark (i.e. excluding quotes);
- The number of words {since, until} the beginning/end of the utterance;

For training NNs, the punctuation feature was represented using 1-of-k coding, and the positional features were normalised to have zero mean and unit variance.

System B makes use of these features only.All other systems additionally make use of some representation of the chosen *context words*: this was limited to the words preceding and following a possible phrase-break for consistency with our previous work.

Despite the great variety of machine learning methods that have been used for PB prediction in the past, all of them have used POS tags as independent variables. A system using POS tag representations of the context words is therefore an appropriate topline system, and **system T** supplements the basic features with the output of a high-quality POS tagger (*TnT*: [18]). The tagger's output for the context words was collapsed to the 23-tag set used in [19, 20, 5] and was presented (coded as two 1-of-k vectors) to the network along with the basic features of system B as the E extra inputs shown in Figure 2.

A full POS tagger can be approximated with a few simple rules; **system G** makes use of such rules, consulting 9 lists of different types of function words at run time. Words not found in any of the lists are tagged as *content* words; others are replaced with the tag associated with the list in which they appear (e.g. *pronoun*, *modal verb*, etc.). This is a suitable benchmark against which to compare unsupervised methods as it is a system built with only minimal expert knowledge. [2]

Table 2. Summary of systems built with induced word representations; ●and ○indicate *true* and *false* respectively.

NN system	DT system	Feature description	Distributional pretraining	Representations tuned for PBs	Vocabulary from PB data
U	U′	**U**nsupervised	●	○	○
R	–	**R**andomly init'd	○	●	●
F	–	**F**ine-tuned	●	●	○
S	–	**S**ubset vocabulary	●	●	●

Experimental systems Experimental systems U, R, F and S are summarised in Table 2. They all make use of the projection layer as well the E extra inputs shown in Figure 2, which are used to input the basic positional and punctuation features already described.

Several different configurations using induced word representations were explored. Most of them used the training of a trigram NNLM as an auxiliary task: for this the 1.2M tokens of text described in Section 3.1 were used. **System U** made use of weights $\boldsymbol{W}_1$ which were pretrained on the full vocabulary of the unsupervised (language modelling) task, but kept them frozen during training on the PB task. Unseen words are handled with the <*unk*> token of the NNLM's vocabulary. This system is the nearest equivalent to the decision tree system (here called U′) from [6] using a VSM (learned on the same 1.2 million tokens of text) and where the classifier was not able to transform individual words' representations on the PB task.

The full vocabulary and projection weights $\boldsymbol{W}_1$ were similarly taken from the trained NNLM and used to initialise the projection layer weights $\boldsymbol{W}_1$ of **system F**. Instead of then being kept frozen, however, these weights were then updated with backpropagation along with the NN's other weights when the NN is trained on the PB data. Comparison of the results of systems U and F therefore will allow us to test the

[1] Results for the slightly refined systems presented in [6] are used, rather from the nearly identical systems presented in [5]; note that the system here called T′ corresponds to system Tt (using the TnT tagger) in [6].

[2] External benchmarks: [21, 19, 20] report F scores of 74.4%, 78.3%, and 81.6% on MARSEC data, although possible differences of data preparation mean this comparison needs to be made with caution.

usefulness of fine-tuning existing word representations on the task of interest.

Systems R is designed to determine the usefulness of pretraining weights $\boldsymbol{W}_1$ on the language modelling task. Its projection weights $\boldsymbol{W}_1$ were randomly initialised and then tuned only on the PB task. The vocabulary is therefore limited to that of the PB training set: to handle unseen words at testing time, words with a single occurrence in the training data were found, and $n\%$ of them were randomly selected and replaced with the $<unk>$ token. Three versions of this configuration were tried, with the value of n set to 100, 50 and 10.

To investigate the impact of the mismatch caused by fine-tuning only the representations of tokens seen in the PB training data (and not the tokens seen in the LM data but absent in the PB training data), **system S** was built. This is identical to F, except that it uses only a subset of the language model vocabulary, limited to the words seen in the PB training data. The representation of the $<unk>$ token – initialised in the NNLM and updated on the PB task – is therefore used to handle all words absent from the PB training data at validation and test time.

3.3. Network training

Weights and biases of all models were initialised uniformly at random according to the *normalised initialisation* suggested by [22] (except weights $\boldsymbol{W}_1$ of systems U, F and S where pretrained weights are used). 10% of the training examples were chosen at random for each model trained for use as a validation set. The remaining 90% were resampled to balance class probabilities – resampling was done with a new random seed for each model trained. Stochastic gradient descent with minibatches was used to train NNs using negative log likelihood of the training examples as the cost function. A learning rate was used which decayed exponentially when improvement in validation set negative log likelihood between successive epochs fell beneath a prespecified threshold. Training finished when the negative log likelihood of this validation set stopped decreasing, or when 15 epochs had been completed. A small L_2 regularisation term was used for training the NNLM, but no explicit regularisation was used in training the PB predictors. In all NNs built, projection dimension P was fixed at 50 (consistent with the dimension of the VSM features used in previous work). For NNLM training, hidden layer size H was set to 100. For PB predictors, 5 different values of H (10, 50, 100, 150, 200) were tried. Five PB predictors were trained with different weight initialisations for each combination of configuration and H value.

3.4. Results

Results are reported as F scores on phrase-breaks. Mean F scores on validation sets were used to determine 50 as the optimal setting of parameter n in system R. The optimal setting for H for all configurations was also found in this way. A single set of 5 systems in each of these 7 configurations was then evaluated on the test set; means and standard deviations of F scores are plotted in Figure 3 alongside corresponding results from earlier work (where the statistics summarise the scores of 10 systems per configuration).

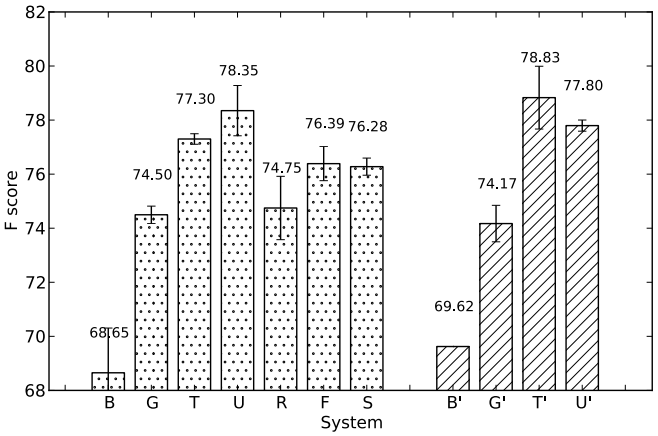

Fig. 3. Mean and st. deviation of F scores (test set) of final systems (left) & decision tree systems for reference (right).

4. DISCUSSION

The overall trend among baseline systems B, G and T is the same as that of decision tree benchmarks (B′, G′, T′), except that T makes less good use of the POS features than the corresponding decision tree system T′. Adding the word representations pretrained on the LM task and keeping them fixed (U) gives the best performance, outperforming T and equalling that of T′. Thus using NNLM features allows us to close the gap in performance between topline and unsupervised word feature systems while using only 1.2M tokens of unannotated text data. This is a pleasing outcome: in previous work [6] 20M tokens of text were needed to close this gap when using VSMs. Pretraining representations in the NNLM is clearly beneficial to training them from scratch on the PB task, although doing only that allows system R to rival the minimally supervised system G. The slightly inferior performance of system F is surprising, as a major motivation for this work was the expectation that updating the word representations on the PB task directly would yield an improvement. We hypothesise that this is due to the mismatch between the fine-tuned vocabulary and the items whose NNLM representations were 'left behind' during fine-tuning. Using these 'left behind' representations gives similar results to the case where they are simply replaced with the $<unk>$ token (system S). If true, this has important implications for MTL as it suggests that a large and consistent inventory of representations is better than one which has been partially tuned on a task of interest. Several ways to enforce representations' consistency during fine-tuning could be proposed, such as tuning a shared-weight adaptation layer instead of the word representations themselves [14, 23], or by using a different training curriculum, such as interleaving batches of PB examples and LM examples.

5. ACKNOWLEDGEMENT

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement № 287678.

6. REFERENCES

[1] Shiyin Kang, Xiaojun Qian, and Helen Meng, "Multi-distribution deep belief network for speech synthesis," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013, pp. 8012–8016.

[2] Heiga Zen, Andrew Senior, and Mike Schuster, "Statistical parametric speech synthesis using deep neural networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013, pp. 7962–7966.

[3] Zhen-Hua Ling, Li Deng, and Dong Yu, "Modeling spectral envelopes using Restricted Boltzmann Machines and Deep Belief Networks for statistical parametric speech synthesis," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 21, no. 10, pp. 2129–2139, 2013.

[4] G. Hinton, Li Deng, Dong Yu, G.E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[5] Oliver Watts, Junichi Yamagishi, and Simon King, "Unsupervised continuous-valued word features for phrase-break prediction without a part-of-speech tagger," in *Proc. Interspeech*, Florence, Italy, Aug. 2011.

[6] Oliver Watts, *Unsupervised Learning for Text-to-Speech Synthesis*, Ph.D. thesis, University of Edinburgh, 2012.

[7] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[8] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *International Conference on Machine Learning, ICML*, 2008.

[9] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa, "Natural language processing (almost) from scratch," *CoRR*, vol. abs/1103.0398, 2011.

[10] Rich Caruana, "Multitask learning," *Machine Learning*, vol. 28, pp. 41–75, July 1997.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[12] Kåre Jean Jensen and Søren Riis, "Self-organizing letter code-book for text-to-phoneme neural network model," in *INTERSPEECH*, 2000, pp. 318–321.

[13] Holger Schwenk and Jean-Luc Gauvain, "Connectionist language modeling for large vocabulary continuous speech recognition," in *International Conference on Acoustics, Speech and Signal Processing*, 2002, pp. 765–768.

[14] Holger Schwenk, "Continuous space language models," *Computer Speech & Language*, vol. 21, no. 3, pp. 492 – 518, 2007.

[15] Gerry Knowles, Briony Williams, and L. Taylor, *A Corpus of Formal British English Speech: The Lancaster/IBM Spoken English Corpus*, Longman, 1996.

[16] Gerry Knowles, Anne Wichmann, and Peter Alderson, *Working with Speech: Perspectives on Research into the Lancaster/IBM Spoken English Corpus*, Longman, 1996.

[17] Peter Roach, Gerry Knowles, Tamas Varadi, and Simon Arnfield, "Marsec: A machine-readable spoken english corpus," *Journal of the International Phonetic Association*, vol. 23, no. 2, pp. 47–54, 1993.

[18] Thorsten Brants, "TnT: a statistical part-of-speech tagger," in *Proc. 6th Conf. Applied Natural Language Processing*, 2000, pp. 224–231.

[19] Paul Taylor and Alan W. Black, "Assigning phrase breaks from part-of-speech sequences," *Computer Speech & Language*, vol. 12, no. 2, pp. 99 – 117, 1998.

[20] Ian Read and Stephen Cox, "Stochastic and syntactic techniques for predicting phrase breaks," *Computer Speech and Language*, vol. 21, no. 3, pp. 519 – 542, 2007.

[21] Bertjan Busser, Walter Daelemans, and Antal van den Bosch, "Predicting phrase breaks with memory-based learning," in *Proc. 4th ISCA Speech Synthesis Workshop*, 2001.

[22] Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks.," *Journal of Machine Learning Research - Proceedings Track*, vol. 9, pp. 249–256, 2010.

[23] Junho Park, Xunying Liu, Mark J. F. Gales, and Philip C. Woodland, "Improved neural network based language modelling and adaptation," in *INTERSPEECH*, 2010, pp. 1041–1044.