Learning continuous-valued word representations for phrase break prediction
Anandaswarup Vadapalli, Kishore Prahallad
Speech and Vision Lab, IIIT Hyderabad
anandaswarup.vadapalli@research.iiit.ac.in, kishore@iiit.ac.in

Abstract
Phrase break prediction is the first step in modeling prosody for text-to-speech systems (TTS). Traditional methods of phrase break prediction have used discrete linguistic representations (like POS tags, induced POS tags, word-terminal syllables) for modeling these breaks. However these discrete representations suffer from a number of issues such as fixing the number of discrete classes and also such a representation does not capture the co-occurrence statistics of the words. As a result, the use of continuous valued word representation was proposed in literature. In this paper, we propose a neural network dictionary learning architecture to induce task specific continuous valued word representations, and show that these task specific features perform better at phrase break prediction as compared to continuous features derived using Latent Semantic Analysis (LSA).

1. Introduction

In the context of TTS, the first step in modeling of prosody is phrase break prediction. Phrase breaks are manifested in the speech signal in the form of several acoustic cues like pauses as well as relative changes in the intonation and duration of syllables. In this paper, we restrict ourselves only to pauses in speech, and limit our phrase break models to predicting the locations of pauses while synthesizing speech. For details, please refer to [1–4].

Typically phrase break prediction has been achieved by using machine learning models like regression trees or HMMs in conjunction with data labeled with linguistic classes (such as part-of-speech (POS) tags, phrase structure etc.) [5–11]. These methods assume the availability of labeled data, and thus cannot be used for languages where such resources are not readily available.

In view of the above limitations, a lot of effort has been directed towards unsupervised methods of inducing word representations, which can be used as surrogates for POS tags / linguistic classes. Parlikar and Black [12] used the Ney-Esken clustering algorithm [13] to automatically induce POS tags. These induced POS tags are automatically generated from text using the frequency analysis of the words. In [14], a set of morpheme tags units were manually identified and used to model phrase breaks in Telugu TTS systems. In [3], the authors showed that word-terminal syllables in Indian languages (last syllable of the word) discriminate between words based on syntactic meaning, and can be used for phrase break prediction in Indian language TTS systems.

All the methods mentioned so far use discrete linguistic representations of words (like POS tags or word-terminal syllables). Such a representation requires a hard classification of words into a set of discrete classes. This raises issues when there is ambiguity in the linguistic representation of the word. For example, the English word plant can be categorized as a noun or as a verb depending upon the context it occurs in. Also such a representation does not take into account the distributional behavior of words. To address these issues, there have been efforts towards deriving or inducing continuous dimensional representations of words, i.e., representing words as points in a continuous dimensional space, for various natural language processing (NLP) applications. These continuous dimensional representations have several advantages over conventional discrete word representations. The primary benefits being, continuous valued representations make no assumptions about the granularity of the discrete categories and the use of these features allows delaying a hard decision about assigning words to a particular category.

Schütze [15] described a method for representing words in a continuous dimensional space using a technique derived from Latent Semantic Analysis (LSA). In this approach, words are mapped to points in a continuous dimensional space defined by the rows of a word co-occurrence matrix. A transformation is then applied to this matrix, to project it into a lower dimensional space (called the latent space). This lower dimensional latent space matrix captures the distributional behavior of the words in as few dimensions as possible. In [15], these word representations were then clustered to produce another discrete set (induced POS-like categories). However this step is not necessary and the continuous features can directly be used for further processing as in [2, 4], where the authors omit the final quantization step and instead use the continuous features directly in a regression tree framework to predict phrase breaks.

In [16,17], the authors describe a unified multitask architecture for NLP that learns features relevant to the tasks at hand. Their approach utilizes a deep convolutional neural network architecture trained in an end-to-end fashion. The input sentence is first processed through several layers of feature extraction, and these features are then automatically trained by backpropagation to be relevant to the task. They report results on the following six standard NLP tasks: POS tagging, Named Entity Recognition, Semantic Role Labeling, Language Modeling and Semantically Related Words.

Following the work of Collobert et al., [17], we propose a neural network based dictionary learning architecture to induce task specific word embeddings for phrase break prediction. We show that our proposed architecture combines feature induction and phrase break prediction in a single framework and thus avoids the two stage process required while using LSA based features for the same task. We experiment our approach on audiobook data, and compare the results obtained for phrase break prediction with the results obtained using LSA based word features.
2. Unsupervised word features using Latent Semantic Analysis

To derive unsupervised continuous valued word representations using Latent Semantic Analysis (LSA), we follow the same method as in [4]. For the sake of completeness, we describe the method below.

This method is based on the first POS induction method in [15] (POS induction based on word type), where a word type is characterized distributionally, i.e., in terms of the words with which it co-occurs in a body of text. The steps involved in generating this representation are:

1. Given a corpus of \( m \) unique word types, out of which a subset \( n \) is chosen as feature words (typically \( n << m \))

   a \( m \times 2n \) co-occurrence matrix \( C \) is computed where:
   - \( C_{ij} \) is the count of number of times the \( j \)th word type occurs with the \( i \)th feature word as left hand neighbor (left context)
   - \( C_{i+j+n} \) is the count of number of times the same \( i \)th word type occurs with the same \( j \)th feature word as right hand neighbor (right context)

2. This \( m \times 2n \) matrix is sparse and each vector has a dimension of \( 2n \). A transformation is applied to this raw co-occurrence matrix to project it into a dense lower dimensional space. A typical transformation is a PCA type projection using SVD.

   \[
   C_{m\times2n} = U_{m\times r} D_{r\times r} V_{r\times2n}^\top, \quad \text{where } D \text{ is a } r \times r \text{ diagonal matrix whose diagonal entries are the best } r \text{ singular values of } C, \text{ and the corresponding left and right singular vectors are the columns of } U \text{ and } V \text{ respectively.}
   \]

3. The matrix \( U \) is a dense, lower dimensional representation of the co-occurrence matrix \( C \). Each row of matrix \( U \) is a \( r \) dimensional feature vector representing the corresponding word.

At the end of this process each word type in the corpus is represented by a \( r \) dimensional continuous valued vector in the latent space.

In [15], the latent space is again quantized, by means of clustering the points representing the word types into a pre-specified number of clusters based on the cosine similarity between vectors. However this step is omitted in [4], where the latent space representation is directly used in a regression tree (CART) framework for phrase break prediction. We follow the same approach in our work.

One issue with using LSA for deriving continuous valued word features arises from the fact that choosing only a small subset \( n \) of the total unique word types \( m (n << m) \) as feature words for computing co-occurrences in some sense constrains the context, resulting in only some information of the distributional characteristics being captured. A naïve solution to this issue would be to choose all \( m \) unique word types as feature words. However, this would lead to a \( m \times 2nm \) matrix which would lead to word vectors with a high level of sparsity and dimensionality. Another issue is that the LSA technique generates features which then have to be used in a separate machine learning framework (like CART or ANN), resulting in a two stage process.

As a solution to the above mentioned issues, we propose a neural network dictionary learning architecture where we induce task specific word features, and use them for phrase break prediction in the same framework, resulting in a one stage process. We also do not impose any constraint on the context while inducing the features, resulting in a greater amount of task specific information about word characteristics being captured. This approach is on similar lines to the work in [17].

3. Task specific continuous word features using neural network dictionary learning

We propose a neural network dictionary learning architecture to induce task specific word representations, i.e., we would like to derive word representations specific to phrase break prediction. Our proposed architecture uses a multilayer perceptron (MLP) setup as a discriminative classifier. This enables us to induce word features that are specific to the phrase break classification task. Figure 1 shows our proposed architecture.

![Figure 1: Neural network based dictionary learning architecture to induce task specific word representations](image)

The first layer of our proposed architecture extracts features for each word in an input sentence. The remaining layers of our architecture use a MLP setup as a discriminative classifier. We describe each layer of our architecture in detail below.

3.1. Transforming words into vectors

The first layer of our proposed architecture extracts features from words. As the input to our architecture is raw words and not numerical features, we first map words into real valued vectors for processing by subsequent layers. To do so we consider the words as indices in a finite dictionary \( D \). That is, each word in the corpus is represented by an index taken from a dictionary \( D \) of real valued vectors \( \{\vec{g}_1, \cdots, \vec{g}_N\} \), where \( N \) is the number of unique word tokens in the database, and all vectors \( \vec{g}_i, i = 1, \cdots, N \) are of the same dimension.

Each word \( w_i \) is embedded into a \( d \)-dimensional space using a mapping function \( g(\cdot) \):

\[
g(w_i) = \vec{y}_i.
\]
where $\vec{y}_i \in \mathbb{R}^d$ is the $i^{th}$ row of the dictionary $D$, and $d$ is the dimension of the word vector.

An input sentence $[w_1, w_2, \ldots, w_n]$ of $n$ words is thus transformed by the first layer of our architecture into a series of $n$ vectors $[g(w_1), g(w_2), \ldots, g(w_n)]$ each of dimension $d$, by applying the lookup table operation.

The dictionary $D$ can be generated in a number of ways. Typically a randomly generated dictionary is used, however there is no restriction on using a dictionary generated in some other fashion.

3.2. Using context

In order to include contextual information, we use a window approach, where we consider a window of size $z$ around the word under consideration. In this approach, each word in the window is first passed through the lookup table operation in 3.1 producing a $d$ dimensional vector for each word. There are $z$ such vectors, which are concatenated together producing a fixed length $z \times d$ vector which is now used as input to the MLP.

3.3. Neural network training and dictionary update

The elements of the initial dictionary $D$ are automatically updated during MLP training. The MLP is set up as a discriminative classifier for the task under question (phrase break prediction in this case), and is trained using backpropagation. During each iteration of training, along with the updating of weights in each layer, each element of the dictionary $D$ is updated. When the training is complete, each element of the final updated dictionary $D_{\text{final}}$ represents a task specific representation of the corresponding word.

During testing, the features for the test sentences are extracted in the same manner as in 3.1 and 3.2 using the final updated dictionary $D_{\text{final}}$. These features are then used as input to the MLP to generate the classification decision.

4. Experiments

4.1. Data used

We use speech and audio data taken from the following three audiobooks: Emma by Jane Austen (EMMA), Sense and Sensibility by Jane Austen (SAS) and Walden by Henry David Thoreau (WALDEN), in the LibriVox database. Each book was recorded by a volunteer and the style of the corpus is “audio book”.

For each audiobook dataset, the speech and the corresponding text were segmented using the INTERSLICE tool, and a CLUSTERGEN voice was built within the Festival framework.

4.1.1. Annotation of phrase breaks

As we do not have a dataset with hand annotated phrase breaks, we derive the location of phrase breaks from the data. Using the Festival utterance structures generated during the CLUSTERGEN voice building process, we extract the location of the breaks introduced by the speaker. We mark all break relations labeled B and BB as breaks (PB) and all other break relations as non-breaks (NPB). At the end of the process each word in the text corpus is labeled as a break or non-break depending upon whether or not a break occurs after that word.

4.2. Phrase break prediction using LSA based continuous word features

We applied the procedure described in Section 2 to the text of EMMA, SAS and WALDEN audiobooks. For each corpus a train set and a test set was created by removing 10% of the data to be used as a held out test set while the remaining 90% was used as a train set.

For each train set, the list of unique word types was compiled, giving us the value of $m$ in each case ($m = 7137$ for EMMA, $m = 6726$ for SAS and $m = 10866$ for WALDEN). For each corpus, the 250 most frequently occurring words were used as feature words ($n = 250$) and the value of $r$ was set to 50. A $m \times 500$ raw co-occurrence matrix $C$ was computed for each corpus and reduced to a $m \times 50$ matrix $U$, using SVD.

An important issue here is how unseen words are handled at test time. We follow a similar approach to that in [22], where we take a portion of the train set (we use 100 words which occur only once) and rewrite them using a special unseen token. Features are then computed for all words in the train set (including the special unseen token) using the procedure described in 2. At test time, all unseen words are mapped to this special token and are represented by the corresponding feature vector.

4.3. Phrase break prediction using continuous features derived from neural network dictionary learning

We applied the procedure described in Section 3 to EMMA, SAS and WALDEN audiobooks. As in 4.2 a train set and a test set was created for each corpus by removing 10% of the data as a held-out test set while the remaining 90% was used as a train set. We use the list of unique word types computed in 4.2, and a $m \times d$ matrix is used as the initial dictionary in each case ($m = 7137$ for EMMA, $m = 6726$ for SAS, $m = 10866$ for WALDEN), where $d = 50$. As a result, applying the window technique described in 3.2 the dimensions of the input vectors becomes 250. The MLP architecture by used here is 250L 500N 50N 500N 2S, where L represents “linear” activation, N represents “tangential (tanh)”) activation and S represents “sigmoidal” activation. To handle unseen words at test time we use the same method described in 4.2.

We also explored combining these unsupervised continu-ous valued features with additional linguistic features (like POS tags) from other sources. We hypothesize that these additional linguistic features add complementary information to what is captured during the dictionary update.

4.4. Systems built

1. System CL : LSA word features used for phrase break prediction in a CART framework System CL is a regression tree (CART) model, where the predictee is the phrase break index (PB/NPB) of the word and the predictors are the features associated with that word. In order to include context, we concatenate the features of the previous two words and the next two words with the feature of the word under question, giving us a 250 dimensional feature vector in each case.

2. System NL : LSA word features used for phrase break prediction in a ANN framework System NL is a neural network trained as a discriminative classifier. The system is trained using backpropagation and outputs a PB/NPB decision given the input features of a word. As in System NL, we associate the features of the previous two words and the next two words with the word under question, giving us an
Table 1: Performance (in terms of the F-Measure) of the various systems on the phrase break prediction task

<table>
<thead>
<tr>
<th>Audiobook</th>
<th>LSA Features</th>
<th>Proposed Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>System CL</td>
<td>System NL</td>
</tr>
<tr>
<td>EMMA</td>
<td>0.65</td>
<td>0.64</td>
</tr>
<tr>
<td>SAS</td>
<td>0.64</td>
<td>0.64</td>
</tr>
<tr>
<td>WALDEN</td>
<td>0.64</td>
<td>0.63</td>
</tr>
</tbody>
</table>

3. **System RE**: Neural network dictionary learning with a randomly generated dictionary

   System RE uses a finite randomly generated initial dictionary to embed the words in a 50 dimensional space in the first layer of our network. This random dictionary is then updated during the MLP training as described in 3.3. Once the training is complete the final updated dictionary is a task specific representation of the words in a 50 dimensional space. During test, this updated dictionary is used by the MLP to generate the PB/NPB decision for the test set.

4. **System LE**: Neural network dictionary learning with LSA based initial dictionary

   System LE uses the LSA representation generated in 4.2 as the initial dictionary to embed the words in a 50 dimensional space in the first layer of the network. During the training of the MLP this dictionary is updated as described in 3.3. At the end of the training the updated LSA dictionary is a task specific representation of the words in a 50 dimensional space. During test, the updated dictionary is used by the MLP to generate the PB/NPB decision for the test set.

5. **System REP**: Neural network dictionary learning with a randomly generated dictionary + POS

   System REP is the same as System RE except that in System REP the POS information of the word is also included in the feature vector. We represent each POS tag by a real valued 5 dimensional vector, which is then appended to the 50 dimensional randomly generated dictionary entry for that word. As a result, the first layer of our network embeds the words in a 55 dimensional space. This dictionary is then updated during the MLP training as described in 3.3. During test, the updated dictionary is used to generate the PB/NPB decision for the test set.

6. **System LEP**: Neural network dictionary learning with LSA dictionary + POS

   System LEP is the same as System LE with the addition of the POS information of the word being included in the dictionary. As in System REP, we represent each POS tag by a real valued 5 dimensional vector, which is appended to the 50 dimensional LSA vector generated for that word. As a result the words are embedded in a 55 dimensional space. This dictionary is then updated during the MLP training. During test, the updated dictionary serves as a task specific representation of the words.

5. **Results**

   Table 1 shows the performance of all six systems described in 4.4, on the phrase break prediction task. We report our results in terms of the F-Measure [23] which is defined as the harmonic mean of precision and recall. F-Measure values range from 0 to 1, with higher values indicating better performance. We analyze the results in detail below.

1. **System CL vs System NL**: System CL and System NL show similar performance on the phrase break prediction task. This shows us that the choice of machine learning technique does not have an effect on the final performance when using LSA based continuous valued word representations. We use System CL as our baseline system in all further analysis.

2. **System RE vs System CL**: An observation of the results obtained for System RE and System CL, shows that there is an improvement in the performance when using our proposed architecture as compared to the LSA based approach. This validates our hypothesis that the neural network dictionary learning architecture manages to capture more information relevant to phrase break prediction as compared to the LSA based method in Section 2.

3. **System LE vs System RE**: A comparison of the performance of System LE and System RE shows that System LE performs better than System RE for phrase break prediction. This performance improvement while using a LSA dictionary as compared to a random dictionary can be explained by the fact that the LSA dictionary already has some information captured as compared to a randomly generated dictionary. As a result when the LSA dictionary is used for initialization, the dictionary learning captures more relevant specific information as compared to a random initialization.

4. **System REP vs System RE**: System REP performs better than System RE on the phrase break prediction task. This validates our hypothesis that encoding additional linguistic features (like POS tags), adds complementary information to the information captured during the dictionary update of the random word embeddings.

5. **System LEP vs System CL**: An analysis of the performance of our final system, System LEP, shows an absolute increase of 0.07 (for EMMA and SAS) and 0.06 (for WALDEN) in the F-Measure values, as compared to the baseline.

6. **Conclusions**

   In this paper we describe continuous valued feature representations for words. We propose a neural network dictionary learning architecture to induce task specific continuous valued word representations. We show that these task specific word features are better word representations as compared to those derived using Latent Semantic Analysis, by means of improved F-Measure values for phrase break prediction.

   In the future, we wish to explore the use of our proposed neural network dictionary learning architecture for other aspects of prosody prediction such as predicting the pitch contour at the word and phrase level, as well as for predicting the spectral parameters for text-to-speech synthesis.
7. References


