Unsupervised continuous-valued word features for phrase-break prediction without a part-of-speech tagger

Oliver Watts, Junichi Yamagishi, Simon King

Centre for Speech Technology Research, University of Edinburgh, UK

O.S.Watts@sms.ed.ac.uk jyamagis@inf.ed.ac.uk Simon.King@ed.ac.uk

Abstract

Part of speech (POS) tags are foremost among the features conventionally used to predict intonational phrase-breaks for text to speech (TTS) conversion. The construction of such systems therefore presupposes the availability of a POS tagger for the relevant language, or of a corpus manually tagged with POS. However, such tools and resources are not available in the majority of the world's languages, and manually labelling text with POS tags is an expensive and time-consuming process. We therefore propose the use of continuous-valued features that summarise the distributional characteristics of word types as surrogates for POS features. Importantly, such features are obtained in an unsupervised manner from an untagged text corpus. We present results on the phrase-break prediction task, where use of the features closes the gap in performance between a baseline system (using only basic punctuation-related features) and a topline system (incorporating a state-of-the-art POS tagger).

Index Terms: phrase-break prediction, part of speech tagging, unsupervised learning, Vector Space Model, Latent Semantic Analysis

1. Overview

Conventional data-driven text-to-speech (TTS) conversion systems bridge the gap between text and speech by using an intermediate representation that will here be called a *linguistic specification*. This specification is typically given in terms of linguistic features such as phonemes, syllables, intonational phrases, shallow syntactic information, and so on. A set of classifiers is typically used in a cascade to derive the specification from text input. The features on which this paper will focus are *part of speech* (POS) and *intonational phrase-breaks* (PBs). POS, or syntactic category, is useful in predicting acoustic values in its own right; it also typically provides input to other classifiers downstream in the TTS 'cascade', such as a PB predictor. Correct phrasing in turn is important for predicting acoustics, and can also be used to feed other predictors (of e.g. pitch accents).

This conventional methodology can be problematic for several reasons. Not least, the training of classifiers assumes the availability of hand-labelled training data in large quantities. Manual annotation is expensive and time-consuming, and might therefore prohibit the training from data of classifiers in languages where the necessary resources are not already available.

The unsupervised induction of word-classes from text is an important problem, both from an engineering and a scientific point of view, and research has been motivated variously by pragmatism [1] and a desire to model aspects of human acquisition of syntactic category [2]. The method we use in the present work is based closely on the Vector Space Model (VSM) approach used in [3]. Briefly, in [3], members of a discrete set (words) are mapped to points in a continuous space defined by the rows of a word co-occurrence matrix; transformations are applied to this space so that characteristics of words' distributional behaviour are captured in a few dimensions of the transformed space. Finally in [3], the transformed space is quantised once more, resulting in another discrete set (induced POS-like categories).

In the work presented here, we produce a transformed space as in [3], but do not partition the space directly into a manually specified number of clusters. Rather, we use the coordinates that specify words' locations in the space as continuous-valued features that are in-put into the next module in our front-end cascade – in this case, a PB predictor. We perform supervised prediction of PBs using classification trees; as predictor features, however, we use the VSM features that have been acquired in an unsupervised fashion. We leave it up to the tree-inducer to partition the discovered space in a way that best serves the purpose of the PB predictor. Thus we do not need to determine a suitable number of POS-like classes. By delaying a hard decision about number of categories in our cascade system, we avoid making premature decisions about where the boundaries of the relevant classes are. We note that this approach has things in common with sequential Multitask Learning [4].

Besides the decision about the number of classes to induce, another problem commonly presented by methods designed to induce POS-like categories is evaluation: mapping discovered clusters to gold-standard ones is problematic both in implementation (there are various ways to compute the overlap of discovered and reference classes) and conception (how can we be sure that what we take as the 'gold standard' ought to be considered as such?). This problem is overcome in the present work by treating the discovery procedure as a module in a bigger system. The word-feature finding method we use is unsupervised, but the end goal of the task adopted here (PB prediction) is – in the current set-up – supervised. This allows computation of measures such as precision and recall on the supervised task in the usual way. We note that [5] uses a similar methodology, testing the effectiveness of discovered word classes on a number of language processing tasks.

The experimental set-up used in the work presented here is designed to serve as a testing-ground for wider use of these techniques in HMM-based speech synthesis. Our aim is to use these features directly in acoustic model context clustering. We use a classification task in the present work for ease of evaluation: credible evaluation of the acoustic models that would result from application of these features to acoustic model clustering can only be carried out through subjective listening tests. This makes the gradual testing and refinement of the features prohibitively expensive and time-consuming. Our assumption is that features found useful for the PB prediction task will also be useful for the acoustic model clustering task.

2. Previous Work on PB Prediction

The availability of a part of speech (POS) tagger is a central requirement in the majority of work on PB prediction. Many different machine learning techniques have been applied to this task; for example, decision trees have been used [6], n-gram models [7], finite-state transducers [8], and memory-based learning [9]. The input to whatever classifier is used, however, has consistently included POS tags as features of central importance. Previous work therefore suggests that reducing the set of surface-forms to a smaller set of symbols on knowledge-based, distributional grounds (i.e. via POS tagging) is a necessary first step for PB prediction. This motivates the present work, where we examine the induction of features that can stand in as 'surrogates' for POS tags in this task.

Note that various researchers have shown that a pre-defined set of tags (such as the Penn Treebank tag-set) is often not optimal for the PB prediction task. For example, mapping the full Penn Treebank set to a smaller, coarser set gives improved performance in [7] (where the mapping is manually specified) and in [10] (where is it learned through an optimisation procedure). Conversely, producing a finer set of classes for certain words is tried both in [11] (successfully) and in [9] (with less success). In both cases, systems using surface forms of words are compared with systems using POS tags and a 'mixed mapping', where POS tags are used except for some small groups of words (e.g. function words) where surface forms are used. It can be seen, then, that a knowledge-based POS tag-set provides a good – but not optimal – set of word classes for PB prediction. This suggests the potential for unsupervised induction of classes for this task to improve on the use of generic knowledge-based classes.

3. Unsupervised Word-Type Features

3.1. Previous Work on POS Induction

The procedure for word-feature discovery that forms the basis of our system is taken from the first of the three POS induction methods described in [3] (where it is called 'induction based on word type only'). The idea that motivates this model is that a word type can be characterised distributionally: in terms of the words with which it co-occurs in a body of text. Therefore, a context vector is assembled for each of the m word types in a large body of text; the number n of words with which co-occurrence is to be tallied (which we will call *feature words*) is determined, in the range $[1, m]$. [3] uses the Brown corpus, giving 47,025 word types (m) and sets n at 250. An m by $2n$ co-occurrence matrix, $\boldsymbol{C}$, is assembled such that $\boldsymbol{C}_{ij}$ records a count of the number of times the i^{th} word type occurs with the j^{th} feature word as its left-hand neighbour, and $\boldsymbol{C}_{ij+n}$ records a count of the number of times the same (i^{th}) word type has the same (j^{th}) feature word as its right-hand neighbour. The raw co-occurrence matrix is then decomposed by a Singular Value Decomposition:

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T \tag{1}$$

The diagonal elements of $\boldsymbol{D}$ are conventionally sorted in descending order of magnitude: taking the first r columns of $\boldsymbol{U}$ and $\boldsymbol{D}$ and the first r rows of $\boldsymbol{V}^T$ gives matrices whose product is the best rank r approximation of $\boldsymbol{C}$. We will denote the matrix derived from U by taking its first r columns as $\boldsymbol{U}_r$, an m by r matrix whose rows are vectors summarising the interactions of the m word types of the corpus with their neighbours. [3] uses a value of 50 for r. The discrete symbols of word types from the corpus data are thus converted into continuous values giving the coordinates of points in a r-dimensional space. In the last stage of the method, the space is re-discretised, by means of clustering the points representing word-types into a prespecified number of clusters according to cosine similarity between vectors.

3.2. A Continuous Space Word Model for PB Prediction

We here applied the procedure described in Section 3.1 to the text of the Wall Street Journal section of the Penn Treebank, *with the exception of the final clustering stage*. The text consists of 1,173,766 tokens and 43,507 types, which gives the value of m in our experiment. The tokenisation of the corpus was used as provided (punctuation tokens were retained), but all other information was discarded (POS tags, sentence boundaries, capitalisation). Values of n and r used were the same as those used in [3], i.e. 250 and 50. A 43,507 by 500 matrix $\boldsymbol{C}$ was constructed and reduced to the 43,507 by 50 matrix $\boldsymbol{U}_r$ as in Section 3.1.

At this point our procedure diverges from the one outlined in Section 3.1. The final clustering used by [3] is omitted: we choose instead to work with the undiscretised space spanned by the rows of $\boldsymbol{U}_r$. Continuous values from $\boldsymbol{U}_r$ were left to be queried directly during decision-tree building. We did this because we consider the final rediscretisation stage as the weak point of the technique outlined in Section 3.1; it was necessary in [3] as the evaluation presented there is based on the overlap between induced and reference categories, a type of evaluation not used here. Working with continuous-valued features has several advantages (theoretical and practical) over conventional discrete POS-like sets. For example, it allows us to partition the space which has been discovered in a way that is optimal for the supervised task we are tackling. Also, omitting the clustering step avoids assigning word types deterministically to a single discovered class (which may be undesirable, given the ambiguity of the token–POS relationship). Although not allowing the disambiguation of specific tokens, distributing the representation of words types across several dimensions means that in theory different dimensions of the space could characterise a type's 'nounness' and 'verbness', without needing to make a hard decision about which class an ambiguous type belongs to.

[3] does not state explicitly how unseen words are dealt with at tagging time in its System 1. The simple method we use here is as follows: a portion of the training corpus (we used 1% of the corpus in the form of consecutive words) is held out while a list of seen types is compiled from the remainder of the corpus. Tokens in the held-out set that are not in this list of 'seen' word types are rewritten in the corpus with a special symbol for unseen words. Co-occurrence statistics are then gathered as in Section 3.1. This allows co-occurrence counts to be collected which characterise the distributional behaviour of the class of words likely to be unseen. At tagging time, unseen words are similarly rewritten with the unseen symbol and tagging proceeds by look-up from a lexicon-like list, associating word-types with their corresponding feature vectors.

4. Experiments

4.1. Data

The data used for the present work was obtained combining data from the SEC database and its extension, MARSEC [12]. The corpus consists of material from a variety of speakers and genres, broadcast on BBC radio programmes during the 1980s. A subset of 39 stories was used, the main exclusions consisting of material from the *Poetry* and *Dialogue* genres.

MARSEC and SEC were automatically merged to obtain punctuated, plain orthography (although tokenised) text (SEC) with PB annotation (MARSEC). All tokens were converted to lower-case, and punctuation marks and PBs were associated with the token that precedes them as a feature of that token.

There are 34,824 tokens in the subset of the corpus prepared (not counting punctuation marks). 11% of these are from the 'overlap' section (annotated prosodically by both the corpus's transcribers). We set these aside as our test set; this choice of test set enables easy replication of our training/test division and is balanced with respect to speaker and genre (consisting of whole sentences from across the different stories and genres of the corpus). 50% of the test set tokens are chosen at random, and transcriber GOK's annotations are used for these; transcriber BJW's annotation is used for the remaining tokens. 10% of the remaining 89% was set aside as a development set for system tuning. The points for this development set were picked randomly from across the non-overlap section of the data (on a word-by-word basis). This results in a training–development–test division of 80%–9%–11%.

4.2. CART

We use CART [13] as the classification method in this work. We expect the use of CART to yield respectable predictors for this work as its suitability for the task has been established previously (Section 2). This choice is also motivated by similarities between classification trees and the regression-type trees commonly used for acoustic model clustering in HMM-based speech synthesis systems. We here use the classification task – with its manageable-sized corpora and standardised evaluation criteria – as a testing ground for developing feature-extraction methods that can be generalised to other tasks (the acoustic clustering task among them). The following procedure was used to build trees for all experiments: classification trees were fully grown using the Gini diversity index as impurity measure and then pruned using minimal cost-complexity pruning; the complexity parameter used was determined by 10-fold cross-validation [13].[1]

4.3. Features Extracted and Systems Built

All Systems The feature to be predicted for each word by all systems was the level of PB associated with it. All three break types in the original annotation (major break, minor break, disfluent pause) were mapped to a single symbol for 'break', *B*, and words with none of these associated were given a symbol for 'no break', *NB*.

B: Baseline System All systems built had access to features relating to punctuation and tokens' position in utterance. For each token in the corpus, the following 5 basic features were used:

- 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, till} the beginning/end of the utterance;

Baseline system B was built with access to only these features.

G: Baseline System with Guessed POS Full POS tagging can be approximated deterministically by compiling a list of the closed set of function words (or possibly, several sub-lists giving distinctions such as *pronoun*, *modal verb*, *etc.*), and to tag these words by look-up in the list(s), tagging all out-of-list words as content words. We tried such an approach in the present work, using 9 lists of different types of function words modelled as closely on those distributed with [15] as differences in tokenisation allow.

The tags obtained in this way for a token and its right-hand neighbour are included as features of that token. The CART algorithm searches all binary partitions of this small tag-set, and thus the simple content–function distinction is modelled implicitly. System G was constructed using these features in addition to the basic features.

T: Topline System with Full POS System T is our topline, and besides the basic features of System B, incorporates features obtained from a high-quality POS tagger that had already been trained on approximately 300,000 words of manually-tagged data from the Wall Street Journal [16]. Collapsing some fine distinctions made by the full Penn Treebank tag-set has previously been found to give improvements on this task [7, 10]. Early trials on the development set showed that the 23-tag system of [7] (grouping verbs, nouns, adjectives and adverbs into single classes) gives improved results also in the present set-up. We therefore use this 23-tag set for all of the POS features in the current work. For each token, the tag of that token is included as a feature as well as the tag of its right-hand neighbour. Contrary to the results with decision trees in e.g. [6], early trials on the development set suggested that a wider tag window does not improve results, and so a 2-word window was used in all of the current experiments (in effect, tags either side of a juncture are used to predict break type at that juncture).

U: System Using Unsupervised Word Features System U is our experimental system, and besides the basic features of System B, incorporates features obtained from the untagged text of the WSJ text as described in Section 3.2. As with System T, the obtained features of a token and its right-hand neighbour were associated with that token as features; in the case of System U, however, these features amounted to 100 continuous features (50 for current and following words) rather than 2 POS tags.

5. Results

Results of evaluation of the systems on development and test sets are given in Table 1, together with sizes of trees build (in number of leaf nodes). Plots of the top parts of trees for systems T and U are shown in Figure 1.

Baseline system B achieves an *F* measure on breaks in the test-set of 69.3%. The 'guessed POS' (GPOS) features that were incorporated into system G increased this to 75.8%, although performance of this system is less stable across test and development sets than that of any other system; on the development set, the increase in performance between B and G is much less (*F* scores of 69.2% and 70.9% respectively).

Topline System T gains an *F* measure on breaks in the test-set of 79.8%, thus representing a reasonable level of performance (cf. *F* measures of 74.4%, 78.3%, and 81.6% in [9, 7, 10] respectively on MARSEC data[2]). Examination of the binary partitions of the tag-set that are chosen for this tree suggests one reason why GPOS features provide a smaller improvement in performance than POS features: in the case of both splits in the upper portion of the tree for system T shown in Figure 1, the partition separates different classes of *content* words, and distinction between content word classes is obviously not enabled by the GPOS features.

The system using unsupervised word features, System U, gains an *F* measure on breaks in the test-set of 77.7%, thus clos-

[1]CART implementation used here: R package *rpart* [14].

[2]Possible differences of data division and preparation mean this comparison needs to be made with caution.

Table 1: Precision (P), Recall (R) and *F* measure (F) for Breaks (B) and non-breaks (NB) on Development and Test Sets; and Tree Size (# Leaf Nodes)

ID	Development Set						Test Set						Size
	B: P	B: R	B: F	NB: P	NB: R	NB: F	B: P	B: R	B: F	NB: P	NB: R	NB: F	
B	94.0	54.7	69.2	88.4	99.0	93.4	95.2	54.5	69.3	87.4	99.1	92.9	25
G	85.0	60.8	70.9	89.6	96.9	93.1	87.3	66.9	75.8	90.3	97.0	93.5	75
T	81.5	74.9	78.1	93.0	95.1	94.0	85.3	74.9	79.8	92.4	96.0	94.1	39
U	83.5	71.1	76.8	92.1	96.0	94.0	83.0	73.0	77.7	91.8	95.3	93.5	177

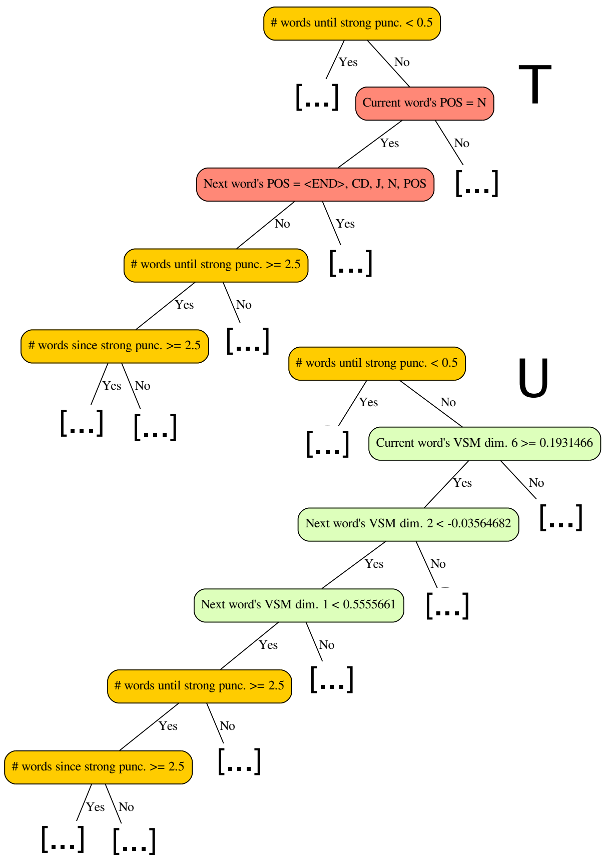
T
# words until strong punc. < 0.5
Yes
No
[...]
Current word's POS = N
Yes
No
[...]
Next word's POS = <END>, CD, J, N, POS
No
Yes
[...]
# words until strong punc. >= 2.5
Yes
No
[...]
# words since strong punc. >= 2.5
Yes
No
[...]
[...]
U
# words until strong punc. < 0.5
Yes
No
[...]
Current word's VSM dim. 6 >= 0.1931466
Yes
No
[...]
Next word's VSM dim. 2 < -0.03564682
Yes
No
[...]
Next word's VSM dim. 1 < 0.5555661
Yes
No
[...]
# words until strong punc. >= 2.5
Yes
No
[...]
# words since strong punc. >= 2.5
Yes
No
[...]
[...]

Figure 1: *Top portions of trees built for System T (Basic and Brill tagger features) and U (Basic and Unsupervised VSM features); nodes querying basic features are coloured yellow, those querying supervised POS features are coloured pink, and those querying unsupervised VSM features are coloured green.*

ing most of the gap in performance between baseline system B and topline system T, and giving superior performance to system G. Note the generally similar topology of the top portions of the trees induced for systems T and U shown in Figure 1: T's question about the POS of the current word is mirrored in tree U by a question about dimension 6 of the current word's VSM features. Similarly, the following question about the POS of the following word in tree T is mirrored by 2 questions about dimensions 1 and 2 of the next word's VSM features in tree U. The final 2 questions in the tree fragments shown are identical.

6. Conclusions

Most of the performance improvement between our baseline and topline systems B and T, gained by incorporating knowledge-rich resources into the system via a state-of-the-art POS tagger, can alternatively be achieved by adding features extracted from plain text in an unsupervised manner, exemplified by our system U. Use of the unsupervised features also enables system U to outperform G, where full part of speech tagging is approximated by the use of manually constructed word lists.

Encouragingly, not only do the features extracted in an unsupervised manner of system U give a quantitatively similar increase in performance over system B compared with the knowledge-based features of system T, but they appear to be used in a qualitatively comparable way (the similarity of the trees' structures is mentioned in Section 5). It seems, however, that the features differ enough that there is some complementarity between them: experiments which cannot be outlined here in greater depth due to space restrictions show that combining a tagger's features with the unsupervised ones leads to improvements over using either set of features in isolation, something *not* achieved by directly combining the features from 2 conventional taggers, where performance degrades. These results encourage us to think that TTS front ends could be constructed with much less supervision than is commonly the case at present, with little or no degradation of state-of-the-art performance.

7. References

[1] R. Kneser and H. Ney, "Improved clustering techniques for class-based statistical language modelling," in *Proc. Eurospeech 1993*, Sep. 1993, pp. 973–976.

[2] M. Redington, N. Chater, and S. Finch, "Distributional information: A powerful cue for acquiring syntactic categories," *Cognitive Science*, vol. 22, no. 4, pp. 425–469, 1998.

[3] H. Schütze, "Distributional part-of-speech tagging," in *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics*, 1995, pp. 141–148.

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

[5] C. Biemann, C. Giuliano, and A. Gliozzo, "Unsupervised part-of-speech tagging supporting supervised methods," in *Proceedings of Recent Advances in Natural Language Processing*, 2007.

[6] M. Q. Wang and J. Hirschberg, "Automatic classification of intonational phrase boundaries," *Computer Speech & Language*, vol. 6, no. 2, pp. 175 – 196, 1992.

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

[8] A. Bonafonte and P. D. Agüero, "Phrase break prediction using a finite state transducer," in *11th International Workshop on Advances in Speech Technology*, 2004.

[9] B. Busser, W. Daelemans, and A. van den Bosch, "Predicting phrase breaks with memory-based learning," in *Proc. 4th ISCA Speech Synthesis Workshop*, 2001.

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

[11] A. Stolcke and E. Shriberg, "Automatic linguistic segmentation of conversational speech," in *Proc. ICSLP*, 1996, pp. 1005–1008.

[12] P. Roach, G. Knowles, T. Varadi, and S. Arnfield, "Marsec: A machine-readable spoken english corpus," *Journal of the International Phonetic Association*, vol. 23, no. 2, pp. 47–54, 1993.

[13] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Chapman and Hall, 1993.

[14] *Package 'rpart': Recursive partitioning and regression trees.* [Online]. Available: http://cran.r-project.org/web/packages/rpart/rpart.pdf

[15] A. Black, P. Taylor, and Caley, *The Festival Speech Synthesis System*, 1999.

[16] E. Brill, "A simple rule-based part of speech tagger," in *Proc. 3rd Conf. Applied Natural Language Processing*, 1992, pp. 152–155.