Stochastic and Syntactic Techniques for Predicting Phrase Breaks

Ian Read and Stephen Cox

School of Computing Sciences, University of East Anglia, Norwich, UK
{ihr, sjc}@cmp.uea.ac.uk

Abstract

Determining the position of breaks in a sentence is a key task for a text-to-speech (TTS) system. We describe some methods for phrase break prediction in which the whole sentence is considered, in contrast to most previous work which has focused on using local features. Three approaches are described: by analogy, where the breaks from the best-matching sentence in our training data is used for the unseen sentence; by phrase modelling, in which we build stochastic models of phrases to segment unseen sentences; and finally, using features derived from a syntactic parse tree. Our best result, obtained on the MARSEC corpus and using a combination of parse derived features and a local feature, gave an F score of 81.6%, which we believe to be the highest published on this dataset.

1. Introduction

Our overall goal is to find algorithms for predicting the location of prosodic phrase breaks for an utterance to be spoken by a text-to-speech (TTS) system. A synthesized sentence containing incorrect breaks at best requires increased listening effort, and at worst, may have lower intelligibility and different semantics from a correctly phrased sentence.

Previous techniques for predicting breaks mainly use features derived from a window centred on a juncture between two words [1], [2], [3]. As prosody applies to the whole sentence, we argue that predictions need to consider the sentence as a complete unit. To illustrate this point, consider these sentences:

a. John doesn't play cards because he's bored.

b. John doesn't play cards because he's bored—he plays them because he is an addict.

Most readers would divide the first sentence into two phrases, breaking between "cards" and "because", with the implication that John is bored, and as a result he does not play cards. In the second sentence, most readers would not break between "cards" and "because", but pause after "bored", implying that John does play cards, not due to boredom, but because he has an addiction. The second sentence shows we need to understand it fully before a correct prosodic rendering can be made. Although we do not consider semantic processing in this paper, this example illustrates how considerations that operate over the whole sentence need to be made when a speaker plans the break positions.

2. Corpora

These experiments used two data sets; the Boston Radio News Corpus [4] consisting of a training set of 13,754 words (3,437 breaks), with another 15,333 words (3,894 breaks) for testing. The Machine Readable Spoken English Corpus (MARSEC) [5] (transcripts from BBC Radio 4) was used as a training set of 31,936 words (6,345 breaks) and a test set of 7,710 words (1,404 breaks). As is common in TTS, we consider two levels of juncture; break or non-break [1], [2]. The data sets were normalized [6] and tagged using the Penn Treebank part-of-speech (POS) tags with the Brill tagger [7] for consistency.

2.1. Evaluation Criteria

Performance is calculated in terms of precision (P), recall (R) and an overall F score (F):

$$P = \frac{number\ of\ breaks\ correct}{number\ of\ breaks\ predicted}$$
$$R = \frac{number\ of\ breaks\ correct}{number\ of\ breaks\ in\ the\ test\ set}$$
$$F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P+R}$$

For this study, $\beta = 1$ was used allowing F_β to give an even balance between precision and recall, thus presenting a single measure of an algorithm's overall quality. Evaluations were performed using both the MARSEC and Boston corpora.

2.2. Baseline

To establish a baseline result, we labelled any juncture within a sentence as a break when preceded by punctuation (i.e. brackets, commas, colons, semi colons, quotation marks and exclamation mark). The results in Table 1 indicate that punctuation is a strong indicator of breaks; however, the low recall shows that many breaks are not associated with punctuation.

3. Prediction-By-Example

Given a sufficiently large training set of annotated examples, it may be possible to predict the breaks in an unseen sentence by analogy i.e. by finding the most similar sentence in the training set, and using the break positions from this sentence to mark the breaks in the new sentence. For this approach, given a new sentence, we measure the Levenshtein distance between this new sentence's sequence of part-of-speech (POS) tags, and the POS sequences of all the training samples. Once the most similar sentence has been found (i.e. the one with the shortest Levenshtein distance from the unseen sentence), the phrase break structure from this matching sentence is then aligned with the new sentence using dynamic programming.

3.1. Results

The prediction-by-example technique was first evaluated using the full Penn Treebank tagset on both corpora. As shown in Table 1, this technique performs worse than the rule-based punctuation algorithm in section 2.2. The full Penn Treebank tagset was then replaced by a reduced tagset derived using techniques described in [8]. Table 1 shows this raised the F score by 9.39% (a result that out-performs the punctuation baseline).

3.2. Analysis of Errors

The main source of errors in this algorithm is misalignment of breaks. A misalignment between two sentences occurs when a plausible match is found, yet it fails to align correctly the junctures, thus causing the breaks to be either a few junctures before or after where they should be located. An example of a misalignment is circled in Figure 1, where the break is misaligned after the WDT tag, when it should be before it.

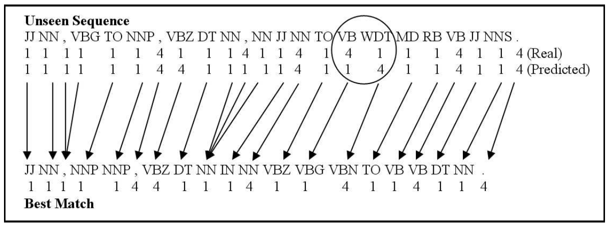
Unseen Sequence
JJ NN , VBG TO NNP , VBZ DT NN , NN JJ NN TO VB WDT MD RB VB JJ NNS .
1 1 1 1 1 1 4 1 1 1 4 1 1 4 1 4 1 1 1 4 1 1 4 (Real)
1 1 1 1 1 1 4 4 1 1 1 1 1 4 1 1 4 1 1 4 1 1 4 (Predicted)
JJ NN , NNP NNP , VBZ DT NN IN NN VBZ VBG VBN TO VB VB DT NN .
1 1 1 1 1 4 4 1 1 1 4 1 1 4 1 1 4 1 1 4
Best Match

Figure 1: Example of prediction-by-example aligning breaks.

Because of the limited amount of available training data, modelling a whole sentence is too specific to be useful. As sentences get larger, the probability of finding a close match becomes very low. In addition, the many comparisons make this technique very slow. However, whole phrases within sentences often matched well, even when the rest of the sentence was dissimilar. Thus suggested we should try to model individual phrases within the sentence. Hidden Markov Models (HMMs) are well suited to this task.

4. Prediction by Phrase Modelling

A sentence can be modelled at two levels within the prosodic hierarchy; as a sequence of words and a sequence of phrases. Hence we can construct sets of models for both of these levels, and combine them to make break predictions on the sentence as a whole. This is achieved by estimating the best sequence of phrase models that "explains" an unseen sentence. Breaks are postulated at the junctures of the phrase models. Hidden Markov Models (HMMs) [9] are well suited to modelling sequences of symbols that display systematic variation. Furthermore, HMMs allows us to use powerful n-gram language modelling from speech recognition to form a probabilistic "grammar" of phrase sequences. This grammar is used to guide the decoding of the phrases, in the same way statistical language models aid the decoding of an acoustic signal in speech recognition.

Syntactically similar phrases such as "the car", "the fast car", "the blue cup" and "the hot cup" are clustered into a phrase model. (For the purposes of these experiments, we define a phrase to be a sequence of words (POS tags) between two phrase breaks.) Phrases were grouped together using k-means clustering, with the initial k cluster centres selected at random from the training data. The distance between two phrases is calculated using the Levenshtein distance between each phrase's POS sequence, in the same way that sentences are compared for prediction-by-example in section 3.

After clustering, each sentence in the training data can then be represented as a sequence of phrase models. From this representation we estimate a bigram "language model" for phrase models, which consists of a set of probabilities $\Pr(phrase_{t+1} = c_j | phrase_t = c_i)$. Unseen sequences are handled with back-off and Good-Turing smoothing [10].

A discrete HMM is built to model each cluster. Each HMM models the way in which the particular set of syntactically similar phrases can be generated by a finite-state machine. Each state of the HMM has an associated set of P probabilities (where P is the number of POS tags), of observing each POS tag in that state. Baum-Welch re-estimation [9] is used to train these probabilities. The HMM Toolkit (HTK) [9] is used for training and decoding the models.

A number of different HMM topologies were evaluated, as shown in Figure 2. All are left to right, modelling word order, but use different possibilities for the state transition matrix a_{ij}, which represents the probability of making a transition from state s_i to s_j. With topology A, all left to right transitions are allowed i.e. $a_{ij} > 0 \; \forall \; j \geq i$, giving a high degree of freedom. Topology B allows $a_{ij} > 0$ for $j = i$ or $j = i + 1$, limiting the transition from one state to itself or the next state. Topology C relaxes topology B by adding transitions to the final state. The optimal number of states was determined by experimenting with a range of values. As well as these uniform values, we also experimented using the mean phrase length of the training samples used to build the respective HMM, giving a variable number of states from phrase model to phrase model.

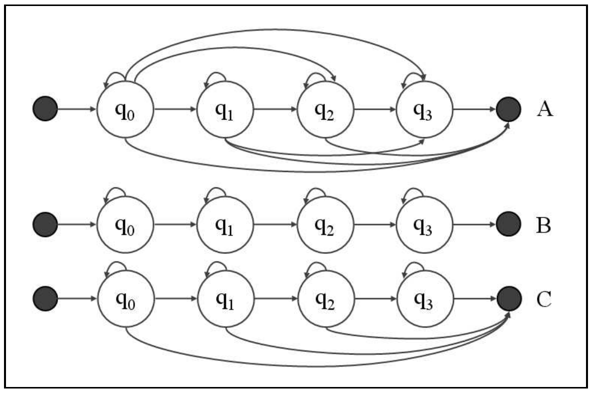

Figure 2: Example of HMM topologies A, B and C, using 4 states.

Viterbi decoding was used to segment new sentences into the most likely sequence of HMMs, with phrase breaks postulated where the HMMs join. The Viterbi decoder facilitates the combination of probabilities from the phase HMMs and from the bigram phrase sequence model. A "grammar scale factor" is used to weight the significance of the bigram model against the POS phrase HMMs.

4.1. Optimization of the Models

As HMMs have a number of parameters, it is important to experiment with a range of different values during optimization. Initially, we made some assumptions about the starting values of these parameters. This consisted of topology C with three emitting states, 100 phrase clusters (HMMs), and grammar scale factor of 1. We experimented with a range of different values for each parameter to find the optimal configuration. Once all of these values have been optimized individually, the best values are combined for a final model.

The number of phrase clusters was varied between 1 and 300 at intervals of 5; the best mean F score of 56.7% was obtained when 45 were used. The number of states and the state transition values were optimized jointly. For the three topologies outlined above, the number of emitting states was varied between 1 and 25, as well as the mean phrase length. The overall best F score of 61.7% was achieved using 7 emitting states with topology A. Varying the grammar scale factor between 0 and 30, the best result was achieved with a factor of 1 (i.e. when the phrase sequence model and the phrase HMMs contribute equally to the overall likelihood), giving an F score of 51.1%. The mean gain in F score obtained from using the phrase sequence model is 2.3%.

4.2. Results

Table 1 reports a best overall mean F score of 63.7%, obtained from using the optimized parameters, consisting of 45 HMMs, each with topology A using 7 emitting states, and a grammar scale factor of 1.

4.3. Analysis of Errors

By far the most common error observed with this approach was to place a break one word prematurely. Increasing the grammar scale factor reduces this error, but leads to significantly fewer breaks being correctly predicted. We concluded that although the HMMs do a good job at modelling phrases, there is probably not enough information in POS sequences for the task of break prediction. In the next section, we describe experiments using syntactic parse features.

5. Syntactic Parsing

There is clearly a strong relationship between prosodic phrase structure and syntactical structure. Consider this sentence taken from the MARSEC corpus: "The little girl and the lion went into the classroom just as the teacher was calling the register." Using the Collins parser [11], the automatically derived syntactic parse of this sentence is shown in Figure 3. In this case, the phrase breaks occur at exactly the junctures between the major syntactic phrases. Although this sentence is unusual in the exact correspondence of breaks and syntactic phrases, it does highlight the potential of such features.

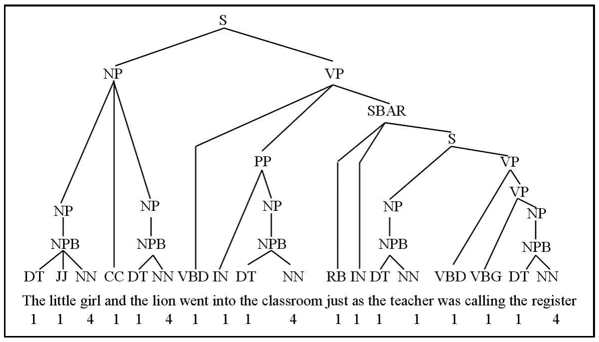
S
NP
VP
SBAR
S
PP
VP
VP
NP
NP
NP
NP
NP
NPB
NPB
NPB
NPB
NPB
DT JJ NN CC DT NN VBD IN DT NN RB IN DT NN VBD VBG DT NN
The little girl and the lion went into the classroom just as the teacher was calling the register
1 1 4 1 1 4 1 1 1 4 1 1 1 1 1 1 1 4

Figure 3: Syntactic parse and prosodic phrase break for the sentence "The little girl and the lion went into the classroom just as the teacher was calling the register."

Given an automatic parse, we need to determine features that are cues for prosodic breaks. Koehn et al. [12] presented a model extending [3], by adding the following features in their CART classifier:

- K0 - The size of the longest phrase ending at the current word.
- K1 - A binary flag denoting whether the phrase is a major phrase (i.e. NP, VP, PP, ADJP or ADVP).
- K2 - The size of the next phrase on the same level of the tree.
- K3 - A binary flag denoting if the phrase is an SBAR.

We used the above features and added three extra features:

- K0A - The number of nodes dominated by the longest phrase ending at the current word.
- K2A - The number of nodes dominated by the next phrase on the same level of the tree.
- LPS - The phrase type of the biggest phrase ending at the current word.

Treating the parse tree as a graph, we consider the number of nodes that are traversed in moving from one word to the next word in the sentence. This led to the following features:

- PD - Parse Depth - the distance from the top node to the current node.
- PDD - Parse Depth Difference - the difference in parse depth of the i^{th} and $(i+1)^{th}$ word. $PDD_i = depth_{i+1} - depth_i$.
- DNW - Distance to Next Word - the shortest path from the i^{th} word to the $(i+1)^{th}$ word.

We used the POS trigram model described in [8] to add the (local) probability of a phrase break at each word juncture (referred to as NGPB).

5.1. Results

It would theoretically be possible to incorporate the features mentioned above into our HMM models, but the disadvantage of this approach is that these features are heterogeneous and are discrete rather than continuously valued. This makes it difficult to represent them as a multivariate Gaussian distribution in the way that is done with, for instance, the set of front-end features typically used in speech recognition. Since use of the parse information now incorporates long-range effects within a sentence into the features themselves, for these experiments, we have used a decision tree classifier rather than an HMM segmenter. The C4.5 decision tree classifier was used to build models that combine these multiple features. Decision tree classifiers have been shown to be useful for the break prediction task [3]. We constructed and evaluated models using all the possible combinations of these features. Table 1 presents the best results from this approach, which uses a combinations of the features K0A, NGPB, PDD and DNW.

The single best F score of 81.56% was achieved using K0, K1 and NGPB, when evaluated on the MARSEC corpus. However, on the Boston data, this same feature set gave 77.91%, indicating that some features are sensitive to the material used.

5.2. Analysis of Errors

Below are some examples of the errors made by the algorithm. Correctly predicted phrase breaks are denoted by $||$, insertions are marked by Δ and Ω signifies deleted breaks.

1. Barry Cane $||$ of Hyannis $||$ is accused Ω of trying Ω to import Ω more than five kilos $||$ of cocaine.
2. The increase in the Ω earnings limit Ω for pensioners $||$ is also worth a comment.
3. Craig Mactenel $||$ is legislative liaison $||$ for the Fisheries Δ and Wildlife Ω Department.

Algorithm	Boston			MARSEC			Mean		
	P	R	F	P	R	F	P	R	F
Punctuation Model (baseline)	95.8%	35.4%	51.7%	81.6%	45.9%	58.7%	88.7%	40.6%	55.2%
Prediction-By-Example (full tagset)	64.3%	46.7%	54.1%	54.0%	38.6%	45.0%	59.1%	42.7%	49.6%
Prediction-By-Example (reduced tagset)	70.5%	54.0%	61.2%	59.7%	54.1%	56.8%	65.1%	54.0%	59.0%
Prediction-by-phrase-modelling (HMMs)	66.8%	57.2%	61.6%	64.4%	67.3%	65.8%	65.6%	62.2%	63.7%
Syntactic Features (K0A, NGPB, PDD, DNW)	82.1%	77.0%	79.5%	85.9%	76.4%	80.9%	84.0%	76.7%	**80.2%**

Table 1: Comparison of the algorithms presented in this paper.

4. Kassler Δ says,|| unlike the Federal Ω Supreme Court, || there's no litmus test || on particular issues || that Massachusetts high court nominees || must pass.

In the above examples—notably 1 and 2—it seems many reference breaks could be omitted without any noticeable loss in quality. They reflect the "semantically dense" nature of the radio news broadcasts, where newsreaders tend to read slowly, pausing to emphasize semantically important words. The error in sentence 3 is due to the algorithm's ignorance of "Fisheries and Wildlife" being a collocation. In example 4, there is a break after "federal" to contrast the two different court systems. This demonstrates how semantic and long-range considerations affect a sentence's prosody. These concepts are clearly beyond what is implemented in this classifier.

6. Summary and Discussion

This paper has examined some different techniques using the whole sentence for the prediction of prosodic phrase breaks. Table 1 summaries the performance of all these algorithms. Although prediction-by-example and the stochastic models of phrases performs well above the punctuation baseline, they are clearly not general enough for practical use, indicating POS features do not carry enough information for making accurate segmentations. Using a decision tree classifier based on a localized n-gram probability and long range features extracted from a syntactic parse gave the best results. On the MARSEC corpus, our single best result (F = 81.56%) outperforms the results reported by Busser et al. [1] (F = 74.4%) and Taylor & Black [2] (F = 78.3%) on the same corpus. On a different data set, Koehn et al. [12], obtained an F score of 82.0%. However, their feature set included hand-annotated pitch accents, which prevents a direct comparison. They achieved 70.8% using just syntactic features.

As the majority of the prediction errors were caused by ignorance of the sentences' semantics, a complete solution to this problem undoubtedly demands semantic and pragmatic processing. But it is not clear how much syntax-based features (either local or long range) can be taken. Further improvements in syntactic parsers will benefit the techniques presented here, as could alternative machine learning techniques such as maximum entropy [13] and support vector machines. Our future work will include adding more features to the classifier used for the syntactic parse experiments [3], and incorporating multiple features into the HMM experiments using vector quantization [9].

However, it may be more interesting to focus on measures of how subjectively disturbing the effect of misplaced breaks is on the semantics or intelligibility of a synthesized sentence. By quantifying this effect, it may be possible to devise an algorithm that minimizes the Bayes risk.

7. References

[1] G. Busser, W. Daelemans, and A. van den Bosch, "Predicting phrase breaks with memory-based learning," in *4th ISCA Tutorial and Research Workshop on Speech Synthesis*, Perthshire Scotland, Aug. 2001, pp. 29–34.

[2] A. Black and P. Taylor, "Assigning phrase breaks from part-of-speech sequences," *Computer Speech and Language*, vol. 12, pp. 99–117, 1998.

[3] J. Hirschberg and P. Prieto, "Training Intonational Phrasing Rules Automatically for English and Spanish text-to-speech," *Speech Communication*, vol. 18, pp. 281–290, 1995.

[4] M. Ostendorf, P. J. Price, and S. Shattuck-Hufnagel, "The Boston University radio news corpus," Boston University, Tech. Rep. ECS-95-001, 1995.

[5] 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. 47–53, 1993.

[6] B. Santorini, "Part-of-Speech Tagging Guidelines for the Penn Treebank Project," University of Pennsylvania, Tech. Rep., 1990.

[7] E. Brill, "Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging," *Computational Linguistics*, vol. 21, no. 4, pp. 543–565, 1995.

[8] I. H. Read and S. J. Cox, "Using Part-of-Speech for Predicting Phrase Breaks," in *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, Seoul, South Korea, October 2004.

[9] S. Young, G. Evermann, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book (for HTK Version 3.2.1)*, 3rd ed. Cambridge University Engineering Department, 2002.

[10] D. Jurafsky and J. H. Martin, *Speech and Language Processing*. Prentice-Hall, 2000.

[11] M. Collins, "Head-Driven Statistical Models for Natural Language Parsing," Ph.D. dissertation, University of Pennsylvania, PA, 1999.

[12] P. Koehn, S. Abney, J. Hirschberg, and M. Collins, "Improving Intonational Phrasing with Syntactic Information," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2000.

[13] S. Cox, "Generalised Probabilistic Descent Applied To Phrase Break Modelling," in *Eurospeech*, Lisbon, Portugal, 2005, *To appear*.