Available online at www.sciencedirect.com
ScienceDirect
Computer Speech and Language 21 (2007) 519–542
COMPUTER SPEECH AND LANGUAGE
www.elsevier.com/locate/csl

Stochastic and syntactic techniques for predicting phrase breaks

Ian Read *, Stephen Cox

School of Computing Sciences, University of East Anglia, Norwich NR4 7TJ, UK

Received 31 March 2006; received in revised form 17 August 2006; accepted 22 September 2006
Available online 26 October 2006

Abstract

Determining the position of breaks in a sentence is a key task for a text-to-speech 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. In addition, the position of breaks must be known before other components of the sentence's prosodic structure can be determined. We consider here some methods for phrase break prediction in which the whole sentence is analysed, in contrast to most previous work which has focused on analysing an area around an individual juncture. One of the main features we use is part-of-speech tags. First, we report an algorithm that reduces the number of tags in the tagset whilst improving break prediction accuracy. We then describe three approaches to break prediction: by analogy, in which we find the best-matching sentence in our training data to the unseen sentence; by phrase modelling, in which we build stochastic models of phrases and use these, together with a "phrase grammar", to segment the unseen sentence; and finally, using features derived from a syntactic parse of the sentence. All techniques achieve well above our baseline performance, which used punctuation symbols to determine break positions, and performance increased with each successive technique. Our best result, obtained on the MARSEC corpus and using a combination of parse tree derived features and a local feature, gave an F score of 81.6%, which we believe to be the highest published on this dataset.

Keywords: Text-to-speech; Prosody; Phrase breaks

1. Introduction

The goal of a universal text-to-speech (TTS) synthesizer is to be able to take any given passage of text, and automatically generate speech that is indistinguishable from that of a human reading the passage. Although considerable progress has been made towards this goal, synthesizers are still poor at the supra-segmental features of speech, including intonation, stress, rhythm and phrasing. This should not be surprising, as these features are typically determined by the semantics as well as the syntax of the text and hence are difficult to

* Corresponding author. Tel.: +44 1603 593220.
E-mail addresses: ihr@cmp.uea.ac.uk, ianharveyread@gmail.com (I. Read), sjc@cmp.uea.ac.uk (S. Cox).

0885-2308/$ - see front matter © 2006 Published by Elsevier Ltd.
doi:10.1016/j.csl.2006.09.004

estimate. If speech synthesizers wish to mimic human speech, they must be able to produce natural sounding prosody. In this work we focus on the prediction of one of the most fundamental aspects of prosody: the position of phrase breaks within a sentence. Natural sounding synthesis also requires accurate intonation (Hirschberg, 1993) and durations (Campbell and Isard, 1991; van Santen, 1998); as algorithms for predicting these features commonly use phrase breaks as input features, errors introduced in phrase break prediction can adversely effect the whole prosody prediction phase. Because of this, we have focused on the crucial problem of identifying breaks in a sentence before investigating the problem of intonation.

When a sentence is being read, some words naturally group together to form phrases. This leads to the theory that a sentence can be described as a hierarchical structure of *prosodic phrases* (Pierrehumbert, 1980). Previous techniques for predicting the prosodic phrasing of a sentence have mainly focused on using a set of features derived from a window centred on a juncture between two words (Taylor and Black, 1998; Busser et al., 2001; Hirschberg and Prieto, 1995; Wang and Hirschberg, 1992). As prosody applies to a whole sentence, we argue that when making predictions, we need to consider the sentence as a complete unit. To illustrate this point, consider the phrase "a thousand people were led to safety" in the context of the following sentences:

a. A thousand people were led to safety after being trapped by a fire in the London Underground last night.
b. A senior fire brigade officer estimated that as many as a thousand people were led to safety from the trains and from platforms.

Both these examples come from the prosodically annotated MARSEC corpus (see Section 4). The sequence "a thousand people were led to safety" has a different prosodic phrase structure in the two sentences. In the first example, the sequence of words does not include any prosodic phrases. However, in the second example, there is a phrase break between "people" and "were". Numerous syntactic features contribute to differences between these two examples; in the second example, the break after "people" corresponds to the end of a six word noun phrase, whereas in the first example the same juncture marks the end of a noun phrase of three words. The position of the phrases within the sentence and the surrounding phrases also affects the different prosodic rendering. These two sentences are examples that can be differentiated using syntactic analysis. However, consider examples *c* and *d*

c. John doesn't play cards because he's bored.
d. John doesn't play cards because he's bored – he plays them because he is an addict.

Most readers would divide the sentence *c* into two phrases, with a break between "cards" and "because", as the implication of the sentence is that John is bored, and as a result he does not play cards. In example *d*, most readers would not place a break between "cards" and "because", but would pause after "bored". The implication here is that John does play cards, not due to boredom, but because he has an addiction to them.

These two sentences are examples where it is necessary to perform a semantic analysis of the sentence before a correct prosodic rendering of it can be made. The pair are also unusual in that the addition of the second phrase to the first sentence alters its semantics in such a way that its prosody is also altered.

Such sentences present a formidable challenge to language processing, but informal analysis of the data used in the experiments reported here (transcripts of news broadcasts) showed that very few of them required either semantic or pragmatic considerations to enable correct prosodic phrasing. Instead we have focused on using syntactic features that operate over the whole sentence so we can model how human speakers plan the phrasing of a sentence as a whole unit. Sentences (c) and (d) above demonstrate that it is unlikely that techniques focusing on individual junctures can succeed completely.

This paper describes some techniques for break prediction that are based on using the whole sentence rather than features gathered from a local area around a word juncture. Section 2 begins with an overview of the theory of prosodic phrasing, followed in Section 3 by a review of previous research on predicting prosody. The different corpora and criteria used for evaluating our algorithms are presented in Section 4. Section 5 explores optimizing part-of-speech (POS) tags for use in phrase break prediction algorithms. A dynamic programming model is presented in Section 6 that makes predictions for unseen sentences by aligning the breaks from the most similar sentence in a set of annotated examples. Section 7 proposes a method using Hidden

Markov Models (HMMs) to segment sentences into phrases. Finally, we present research using features extracted solely from a syntactic parse tree to predict prosodic phrase breaks in Section 8.

2. Prosodic phrasing

A spoken utterance can be broken down into a hierarchical structure of prosodic phrases. An example of the prosodic phrase structure of the sentence "Today eighteen people asked Judge John Owen to excuse them from the jury pool." is shown in Fig. 1. The words in this sentence divide into four main phrases. These are known as intonational phrases (Beckman and Pierrehumbert, 1986). When read aloud, there are noticeable pauses between words on the boundaries of two intonational phrases, for example between "Today" and "eighteen", "Owen" and "to", and "them" and "from". Lower down the prosodic phrase hierarchy, intonational phrases can in turn be divided into sequences of smaller units – intermediate phrases – that have fewer salient pauses between them. In this example, the second intonational phrase comprises two smaller phrases that are demarcated by a brief pause between "people" and "asked".

2.1. Annotating Prosody – ToBI

ToBI (*To*nes and *B*reak *I*ndices) (Silverman et al., 1992) is one of the most widely adopted systems for annotating prosody. The ToBI recommendations specify four levels of annotations; orthographic, miscellaneous, tones and break indicies. Fig. 1 shows the ToBI transcription of the sentence's breaks on the line beneath the orthographic words.

Boundaries between prosodic phrases are marked as break indices between two orthographic words. The level of the break index represents the degree of juncture between two words. A juncture of zero denotes no boundary, with evidence of cliticization – such as the boundary between "I" and "'m" in "I'm". A break level of one indicates no break between two words. At the other end of the scale, the boundary between two intonational phrases is noted by the presence of a level four break. Level three breaks specify an intermediate phrase boundary. Both level three and level four boundaries occur alongside distinct events on the tone level. Level two breaks are used to denote breaks with an apparent disjuncture between two words, but they are not accompanied by the tonal discontinuities that characterize level three and four breaks. The "p" diacritic is used to denote cases where the speaker appears not to be pausing on purpose, but still sounds disfluent. This is common in hesitations caused by struggling to find the next word to say. In order to distinguish such cases from fluent level two breaks, these are annotated as "2p".

When there is a degree of uncertainty regarding the break index of a particular juncture, a negative is placed after the annotated break index. For example, "4-" is used to denote a break that probably lies somewhere between level three and level four.

In text-to-speech synthesis, it is common to consider just two levels of classification; break or non-break (Taylor and Black, 1998; Busser et al., 2001; Marsi et al., 2003). Intonational and intermediate phrases (corresponding to ToBI values of three and four) are considered to denote phrase boundaries; we adopted this convention for the research presented here.

Different analysts may disagree in their positioning of breaks in a text and different speakers may position breaks slightly differently when reading a sentence aloud. However, some experiments have investigated the

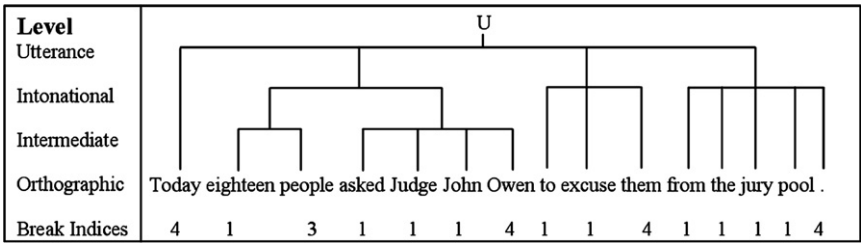

Fig. 1. Prosodic phrasing of a sentence taken from the Boston corpus.

level of agreement between trained annotators when they transcribe spoken sentences with their ToBI phrase break values (Pitrelli et al., 1994) and have found that there are high levels of agreement. Ignoring the p diacritic and the use of the "-" symbol, they reported an agreement in break marking measured over all pairs of transcribers of 70.4%. When agreement was relaxed to be within ± 1 the actual boundary value (i.e. values 2, 3 and 4 being valid when the actual juncture is 3), this figure was raised to 92%.

Numerous prosodic features are clearly related; however, accurately defining these relationships for use in TTS has been a problematic task and one that is as yet unsolved. This problem affects the entire prosody prediction phase in TTS, as the optimal order for predicting features such as intonation, timing and phrasing is unknown. In most TTS systems, phrase breaks are predicted first and then used to aid predicting the location of pauses, the intonation (Hirschberg, 1993) of the utterance and its duration (Campbell and Isard, 1991), (van Santen, 1998). Therefore, errors introduced in phrase prediction will also affect subsequent prosodic predictions. In an intonation pattern, the sequence of tones is dependent on their location within a prosodic phrase; an obvious example of this is that major phrase boundary tones can only occur at the boundaries between intonational phrases. In human speech, the duration of units is directly related to the phrasing of the utterance, as segments tend to be lengthened immediately before phrase boundaries, hence the use of phrase breaks when predicting durations (Campbell and Isard, 1991; van Santen, 1998). In Marsi et al. (2003) an algorithm was presented for simultaneously predicting phrase breaks and pitch accents. They achieved this by concatenating the break symbol with the pitch accent symbol for each feature vector. This makes the assumption that the features used for predicting phrase breaks are equally effective for predicting pitch accents. This is not necessarily the case, as certain features may be highly relevant for predicting pitch accents (for example lexical stress), but have limited correlation with phrase breaks (or visa versa), thus introducing inaccuracies into the algorithm. Using this joint prediction approach, Marsi et al. (2003) reported results similar to those obtained when the tasks were performed separately using the same features and data set.

Intuitively, there is a connection between the prosodic and syntactic phrasing. Although much research has been conducted into prosodic and syntactic phrasing, a direct mapping between these structures has yet to be defined. Methods to determine the prosodic phrasing of a sentence, given only its syntactic phrase structure have produced promising results (Fach, 1999; Koehn et al., 2000). These suggest that a model combining previous research using local features with sentence-level structures could improve an algorithm on this task. We take up this point again in Section 8.

3. Previous prosody prediction techniques

We will first examine the merits and limitations of previous techniques that have been used for prediction the prosodic phrasing of some given text. Previous work on this problem has been divided into two broad approaches – rule-based methods and data driven methods.

3.1. Rule-based methods

Early work on predicting prosody was related to extracting features from performance structures (Gee and Grosjean, 1983). At the time, there were discrepancies between linguistic phrase structure theories and the actual performance structures that humans produce. This research aimed at reproducing the prosodic phrasing by making adjustments to the syntactic parse of the sentence (Allen et al., 1979; Dommergues and Grosjean, 1981; Grosjean et al., 1979).

Bachenko and Fitzpatrick (1990) extended this work and used information about syntactic adjacency to a verb and constituent length to determine prosodic phrasing for synthetic speech. Their goal was to determine "discourse neutral" prosodic phrasing, resulting in a sentence level phrasing that is independent of a sentence's semantics.

An obvious approach to predicting prosodic phrasing is to work on the assumption that boundaries are marked by appropriate punctuation (such as full stops, commas, hyphens, colons and brackets). Using this method, Taylor and Black (1998) demonstrated that punctuation correctly identifies approximately half of the actual breaks. Unfortunately in informal writing (e.g. emails), punctuation can be idiosyncratic, incorrect

or non-existent, so it cannot be relied upon. However, it does provide a useful baseline and we give results using punctuation for break prediction in Section 4.2.

A simple rule that places a break between content words that are followed by function words has been investigated by Silverman (1987) and Sorin et al. (1987). Taylor and Black (1998) report an overall accuracy of 71%, with 84% of breaks correct using this technique. However, the technique over-predicts, giving insertion rates as high as 32%. But as noted in Busser et al. (2001), the results given in this paper do not translate correctly into precision/recall, with recall calculated as 116.46%.

3.2. Data driven techniques

The increased availability of prosodically annotated corpora, coupled with the success of probablistic techniques in related fields has led to data driven techniques emerging as the most successful method for predicting prosodic phrase structures. In general, most of these approaches regard the prediction of prosodic phrasing as the task of making a decision at each word juncture regarding its most likely type, as opposed to predicting the sentence's phrasing as a whole.

Classification and regression trees (CARTs) (Breiman et al., 1984) have been used to process such features as part-of-speech (POS) tags, syntactic constituents, and other prosodic information (e.g. previous boundary location, pitch accent location and phrase duration) (Ostendorf and Veilleux, 1994; Wang and Hirschberg, 1992). Given a set of feature vectors, each with an associated category (in this case break or non-break), a CART is automatically constructed for classifying new feature vectors and hence word junctures into the most likely class.

Busser et al. (2001) presented results using a similar technique called memory-based learning (MBL) (Daelemans et al., 2004). MBL works by keeping all the training examples in memory, and predicting the class of new feature vectors based on the most similar training example(s) in memory, an approach which is similar to the technique of k-nearest neighbour in pattern classification. When evaluated on the MARSEC corpus, the best F score (see Section 4.1) produced was 74.4%.

Taylor and Black (1998) used a "window" of a few words (POS tags) around each juncture in an n-gram model of phrase break sequences to compute the most probable sequence of breaks and non-breaks. Using a window of two words before and one after, the algorithm achieved an F score of 78.3%. We believe that this is currently the best published result on the MARSEC corpus.

3.3. Break prediction using the whole sentence

Although partially successful, n-gram models of POS sequences do not provide enough discrimination for successful classification of junctures, because the context window is too small to capture the long range aspects of prosodic cues, and probably also because POS features on their own are not sufficiently powerful for this task.

As prosody applies to a whole sentence, it seems natural to consider the sentence as a complete unit when making prosodic predictions. The prosodic structure of a sentence has two different levels – words and phrases – and therefore models can be created for making predictions at both levels of this hierarchy. By combining models for words *and* phrases, it is possible to make predictions based on the sentence as a single unit. There are several ways to approach this. In Section 6 we investigate the idea of matching unseen sentences to example sentences and predicting breaks using the break structure in the best matching sentence. Section 7 introduces the idea of modelling groups of phrases and using stochastic matching techniques to find the best sequence of models that fit unseen sentences. Finally, Section 8 re-examines the idea of using syntactic parsing to predict breaks, combining features from the parse with other features in a decision tree classifier.

4. Corpora

For the purposes of these experiments, two separate data sets were used. The first was the Boston Radio News Corpus (Ostendorf et al., 1995) annotated to the full ToBI specification (Silverman et al., 1992). This

was divided into a training set of 13,754 words (3437 breaks), with an additional 15,333 words (3894 breaks) reserved for testing. The Machine Readable Spoken English Corpus (MARSEC) (Roach et al., 1993) consists of transcripts from news stories, talk shows and weather reports from BBC Radio 4. Phrase break annotations covered only levels 1, 3 and 4 of the full ToBI recommendations, but this was more than adequate for these experiments. The MARSEC data was split into a training set of 31,936 words (6345 breaks) and a test set of 7710 words (1404 breaks). The radio announcers recorded in these data sets are professionally trained and speak in a highly consistent style, which is clearly important in this work.

As described in Section 2, for the purpose of phrase break prediction within TTS, it is common to flatten the prosodic phrase hierarchy so that a juncture is considered to be either a break or a non-break. In these experiments, word junctures of level 3 or above were considered to be breaks, and any juncture with a lower level was treated as a non-break.

Both data sets use hand-corrected POS tags; however they originated from different tagsets. Hence, the data was first automatically retagged using the Brill tagger (Brill, 1995), which uses the Penn Treebank tagset (Santorini, 1990). This has the added advantage of making our experiments more realistic as a TTS application would not use hand-corrected tags.

In accordance with the Penn Treebank guidelines (Santorini, 1990), the words were normalized before tagging. Normalization involves separating punctuation from adjoining words, as well as splitting verb contractions and the Anglo-Saxon genitive of nouns so that each morpheme can be tagged separately. For example, ''won't'' maps to ''wo'' and ''n't''; similarly ''gonna'' becomes ''go'' and ''na''. A phrase break will never occur in the middle of a split word and hence we use rules for preventing such a break. However splitting is essential for tagging, and intelligently separating punctuation gives strong indicators for the location of phrase breaks, as already discussed in 3.1.

4.1. Evaluation criteria

Taylor and Black (1998) measured the percentage of breaks correct and non-breaks correct. For this method, they recorded N as the number of junctures in the test set, and B the number of them that are breaks. A deletion error (D) occurs when a break is marked in the test data but not predicted. Conversely, an insertion error (I) occurs when a break is predicted that does not occur in the test data. When considering multiple break levels, substitution errors (S) occur when a break is predicted, but is not of the correct type. The overall accuracy of the algorithm is measured using the following indicators:

$$\text{BC} = \text{Breaks correct} = \frac{B - D - S}{B} \times 100\% \tag{1}$$

$$\text{NC} = \text{Non-breaks correct} = \frac{N - I - S}{N} \times 100\% \tag{2}$$

$$\text{JC} = \text{Junctures correct} = \frac{N - D - S - I}{N} \times 100\% \tag{3}$$

$$\text{JI} = \text{False insertions} = \frac{I}{N} \times 100\% \tag{4}$$

$$\text{BI} = \text{False insertions w.r.t. breaks} = \frac{I}{B} \times 100\% \tag{5}$$

The overall number of breaks correctly predicted is only significant when considered with non-breaks correctly predicted. In the data used here non-breaks account for 70–80% of all junctures. Hence an algorithm that marks every juncture as a non-break will apparently be predicting approximately 70–80% of all the junctures correctly.

This problem is addressed by calculating results in terms of precision and recall, and using an F score for combining these two measures into a single number that is indicative of the performance of the algorithm (van Rijsbergen, 1975). Precision is used to measure how many of the phrase breaks predicted by the algorithm are correct, and recall measures what percentage of the phrase breaks in the test data are correctly predicted. The F score gives a balanced measure of overall quality by harmonizing precision and recall.

$$P = \frac{\text{Number of breaks correct}}{\text{Number of breaks predicted}} \tag{6}$$

$$R = \frac{\text{Number of breaks correct}}{\text{Number of breaks in the test set}} \tag{7}$$

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \tag{8}$$

When $\beta = 1$, F_β gives an even balance between precision and recall, and Eq. 8 simplifies to:

$$F = \frac{2PR}{P + R} \tag{9}$$

If an algorithm does not identify any breaks, precision is undefined. In our tables of results, we have used "-" to indicate when this happens.

4.2. Baseline

As discussed in Section 3.1, punctuation can indicate around half of the phrase breaks. To accommodate this in our experiments, punctuation is treated as a single unit, detached from any surrounding words. When using syntactic parsers that follow the Penn Treebank guidelines (Santorini, 1990), it is essential to identify punctuation in this way prior to parsing. To establish a baseline for our experiments, we labelled any juncture which was preceded by either brackets, colons and semi colons, commas, quotation marks and exclamation marks. The use of periods can be problematic for this task, because as well as marking sentence boundaries, they are also used for acronyms and initials (e.g. F.B.I., J. Smith). Regular expressions were developed to distinguish periods that demarcated sentences with periods used for other purposes. These expressions handled all instances of periods correctly.

Table 1 gives the precision (P), recall (R) and F scores (F) achieved on the two corpora by this punctuation model. As expected, the results are of high precision (implying most punctuation indicates a break) but low recall (as there are many breaks that do not have any punctuation marks associated with them).

5. Reducing the part-of-speech tagset

Part-of-speech (POS) tags have been shown to be a good feature for predicting phrase breaks (Taylor and Black, 1998), (Busser et al., 2001) and the accuracy of POS taggers is in excess of 96% (Daelemans et al., 1996). Taylor and Black (1998) noted that automatically tagged data "performs as well as (if not better)" than hand-annotated data. Most POS taggers use at least 40 different tags. However, these tagsets were developed for descriptive use, and it is not clear that such a large set of tags provides useful information for the task of classifying junctures as breaks or non-breaks. It is well known that adding non-information-bearing features to a classifier introduces noise that degrades performance. Hence prior to formulating algorithms for our task, we attempted to first find a smaller set of POS tags that would be suitable for the task of prosodic phrase break prediction. A correctly chosen reduced set should give higher accuracy on the task, and also the benefit of reduced processing and storage space. Furthermore, prosody prediction algorithms that use POS tags as features do not consider similarities between tags when making predictions. Grouping similar tags takes advantage of these similarities.

The algorithm introduced here gives a method for taking any given POS tagset, and then grouping the tags into a smaller number of tag classes that are useful in the task of predicting breaks. Initially our data was auto-

Table 1
Baseline results from using punctuation to predict breaks

Boston			MARSEC			Mean		
P (%)	R (%)	F (%)	P (%)	R (%)	F (%)	P (%)	R (%)	F (%)
95.8	35.4	51.7	81.6	45.9	58.7	88.7	40.6	*55.2*

matically tagged using the Brill tagger (Brill, 1995), which uses the Penn Treebank tag set (Santorini, 1990) of approximately 40 symbols. Section 4 contains more information on the data sets used for these experiments.

5.1. Best first search

Given all the possible groupings of POS tags, our goal is to find the one that yields the highest overall accuracy. The brute-force approach of generating all the possible arrangements of POS tags and evaluating the performance of each is computationally infeasible. However, we can approximate this by considering each grouping of POS tags as a unique "state" and searching over a large number of states. We use the accuracy on the break prediction task of a particular tag grouping as a measure – or heuristic – of its quality. Starting from an initial state, the heuristic sensibly guides the search by only exploring states that suggest the highest potential.

"Best first search" (Pearl, 1984) is an algorithm for searching through a large state space by considering all unexpanded states and then expanding those states with the highest likelihood of leading to a goal (the nature of goals and expanding is discussed later in Section 5.2). The structure of the search algorithm is as follows:

```
1: open ← expand(initialState)
2: while open ≠ { } do
3:    top ← findBestNode(open)
4:    if isGoalState(top) then
5:       return top
6:    else
7:       close ← close ∪ top
8:       open ← open − top
9:       open ← open ∪ expand(top)
10:   end if
11: end while
```

5.2. Expanding states

The initial state has all POS tags grouped into a single class, called `bag`. Given a state, the `expand` function works by going through every element of `bag`, and moving it to one of the other available classes and then a new class on its own. This is demonstrated in Fig. 2 using the three tags adjective, noun and verb as an example. As this is the initial state, the only possible expansions are to iteratively place the elements of `bag` into new classes.

Fig. 3 demonstrates a more realistic example, where elements are extracted from `bag`, and either grouped with another class, or individually. Given that the fourth expansion is the same as the second, there is no need to evaluate the fourth state. These new states, along with their accuracy, are recorded in the *open* list. Each cycle of the algorithm expands the state in *open* that gives the highest overall performance. Once a state has been expanded, it is added to the *close* list. The algorithm checks *open* and *close* to prevent repetitions when new states are being created.

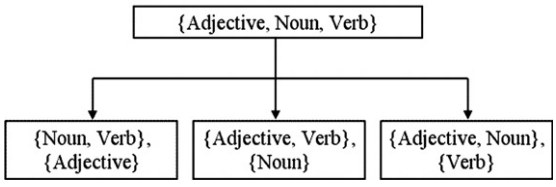

Fig. 2. Expanding the initial state of {adjective, noun, verb}. A simple tagset has been used for illustrative purposes.

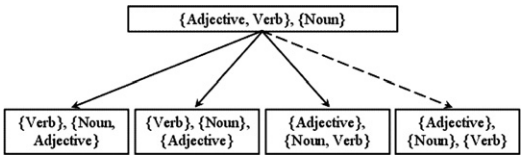

Fig. 3. Expanding a state where adjective is grouped with verb, and noun is separate. The dashed line denotes a state that has already been visited.

One of the problems when using this algorithm is defining a suitable heuristic, h, and goal(s). In our case, the most appropriate candidate for a heuristic is the F score achieved when evaluating using the given tagset. The optimal goal state is the state that gives the highest accuracy on the task, but we are only certain to find this by an exhaustive search. Hence we halt the search when the performance converges after a certain number of iterations. Although finding a high quality set of POS classes takes considerable computation time, this is an offline process that only needs running once.

5.3. Heuristic

For the heuristic, we used the performance (as measured by the F score) of an n-gram model of POS tags. This model, described in Taylor and Black (1998), makes predictions based on the POS tags surrounding a juncture. They reported that using two POS tags preceding the juncture and one following gave the highest performance in their system, and we used these values in our own experiments.

For each unique trigram of tags, S, found in the training-set, we estimate the probability that the juncture associated with S is a break as:

$$\Pr(j = \text{Break}|S) = \frac{\text{count}(\text{Break},S)}{\text{count}(S)} \tag{10}$$

Clearly, $\Pr(j = \text{Non-Break}|S) = 1 - \Pr(j = \text{Break}|S)$.

Classification of a juncture from the test-set characterized by a sequence of tags S' is made by assigning to class $j*$ where

$$j^* = \underset{j}{\text{argmax}}\{\Pr(j = \text{Break}|S'), \quad \Pr(j = \text{Non-Break}|S')\} \tag{11}$$

In their full system, Taylor and Black combined the n-gram POS sequence model with another model that predicts based on the preceding phrase break sequence. This reduces the likelihood of predicting a long sequence of breaks, and also favours predicting a break after a long sequence of non-breaks. As we were focusing here on the performance of different tagsets themselves in break prediction, we did not include this second model.

Data sparseness is a common problem in n-gram modelling – it is possible that the trigram S' will have been seen only once, or never, in the training-data. In statistical language modelling, this problem is addressed by using smoothing techniques to assign part of the "probability mass" of the model to unseen events. We compared the performance of a number of tagsets with and without Good Turing smoothing (Church and Gale, 1991) and found that smoothing affected the F scores only very slightly, by typically $\pm 0.05\%$. As the reduced tagsets found by the algorithm consisted of only 7–14 tags, it seems that there was sufficient training data to make good estimates of n-grams and hence smoothing was not required.

5.4. Results

During the tagset reduction experiments, the algorithm tended to favour tagsets that gave good results on the testing data of a particular corpus. Therefore, the quality of an arrangement of tags was evaluated on sections of both the Boston and MARSEC data, taking the mean F score as the indicator of quality. Furthermore, only the training data sets were used for the tagset reduction tests. The training data from each of

the corpora was split into two new data sets; a development training set and a development testing set. By using a development training and testing set, we are more likely to find an unbiased dataset.

Table 2 summarises the results of the n-gram model using the standard Penn Treebank tagset.

After running the POS reduction algorithm for a large number of iterations using the development training and testing data sets, the 50 best performing tagsets were then evaluated on the full data sets. All of these tagsets groupings were very similar, typically consisting of 7 or 8 tag classes. The tagset that yielded the highest performance when evaluated on the full data set is shown in Table 3.

Tables 2 and 3 show that the reduced when compared with a full tagset achieved an increase in recall, with precision staying roughly the same for the full and reduced sets. Furthermore, performance is comparable across the two corpora.

The best reduced set of POS tag classes is shown in Table 4 (an explanation of the meaning of the tags shown in Table 4 can be found in Santorini (1990)). All the high performing arrangements of tagsets typically had two main classes that corresponded to the content/function words mentioned in Section 3 (Silverman, 1987; Sorin et al., 1987). However, using only these two main classes leads to over-prediction of breaks, as observed in Taylor and Black (1998). When these two classes are supplemented with a few much smaller classes, of typically just 1 or 2 tags, performance is greatly increased. With some groupings of tags, the particle tag tended to form a group on its own. Punctuation symbols such as comma, full stop and colon tended to form their own class, often grouping with proper plural nouns. These symbols generally have a very high correlation with breaks and so their grouping together seems plausible. Opening quotation marks and genitive markers formed their own group. Existential "there" usually formed a group of its own. In the test data, this tag is almost exclusively followed by non-breaks, therefore it is probably isolated for this reason.

In these experiments, we employed an n-gram model for juncture classification as a convenient way of developing a reduced tagset. Our later results demonstrate that reduced sets are also more effective when other longer-range classification methods were used. A more detailed discussion of this technique is presented in Read and Cox (2004).

Table 2
Performance of the phrase break prediction algorithm using the standard Penn Treebank tagset

Boston			MARSEC			Mean		
P (%)	*R* (%)	*F* (%)	*P* (%)	*R* (%)	*F* (%)	*P* (%)	*R* (%)	*F* (%)
80.5	64.1	71.4	80.1	63.3	70.7	80.3	63.7	*71.0*

Table 3
Results of using the highest performing tagset

Boston			MARSEC			Mean		
P (%)	*R* (%)	*F* (%)	*P* (%)	*R* (%)	*F* (%)	*P* (%)	*R* (%)	*F* (%)
80.6	73.1	76.7	80.1	70.7	75.1	80.4	71.9	*75.9*

Table 4
The best set of POS classes found by the reduction algorithm

Tags
'', (,)
,, ., :, NNPS
CC, WDT
CD, FW, JJ, JJR, JJS, NN, NNP, NNS, UH
DT, IN, MD, PDT, PRP, PRP$, RB, RBR, RBS, TO,
VB, VBD, VBG, VBN, VBP, VBZ, WP, WP$, WRB
EX
POS, ``
RP

6. Prediction-by-example

Our contention is that, in contrast to techniques that use features local to each word juncture to predict breaks, the whole sentence should be analysed to plan the position of the breaks. A simple way of performing such an analysis is by comparison with sentences in the training set. 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. Of course such an approach can only be viable when the training set is large and when the unseen sentences are similar in terms of their structure and content to the training set sentences. But this is a situation fairly common in TTS, where it is often required to read formulaic language such as stock market quotes or travel directions.

The main problem that needs resolving for this approach is finding a suitable distance measure for measuring the similarity between two sentences. Such a distance measure needs to be based on features that are significant cues to prosody. As the amount of training data is small, the likelihood of exact word sequences being repeated is very low, and so it is infeasible to use word identities as features. Instead, we reduce words to their POS tags. We also need a way of estimating a "distance" between sentences of unequal length, and of sentences that have a similar syntactical structure but differ in some superficial ways. The Levenshtein distance (Levenshtein, 1966) provides the optimal alignment between two strings, and hence the Levenshtein distance between each sentence's POS sequence is used here. Dynamic programming is used to find the Levenshtein distance.

The algorithm works by ranking the sentences in the training data based on their distance from the new sentence. As calculating the distance for all the training data would incur a very large computation cost, the data is initially pruned to exclude sentences that do not contain at least one matching tag. Finally, the phrase break structure from the closest example is aligned with the new sentence using dynamic programming (Bellman, 1962) to find the best mapping from the old sentence to the new one.

6.1. Islands of certainty

As sequences increase in length, the likelihood of finding an exact match decreases. One way of tackling this problem is to reduce the sentences to smaller sequences by splitting at points where there is a high probability of a phrase break and then using these smaller sequences on this algorithm. We term the breaks found this way "islands of certainty". We used two methods for locating islands of certainty. As shown in Section 4.2, punctuation is very precise at locating phrase breaks, and hence is a good model for finding the initial boundaries for dividing the sentences. Another approach is to use the POS reduction method of Section 5 to find a tag set that gives high precision, then to use this tagset with an n-gram model to deduce the location of highly likely phrase breaks. This makes the algorithm predict junctures in two passes; the first pass splits the sentence using punctuation or a high precision n-gram model, the second pass uses dynamic programming for detailed matching.

6.2. k-Nearest neighbour

The approach of matching the input sentence to the closest sentence in the training data may be thought of as a "one nearest neighbour" approach. We experimented with identifying the k-nearest neighbours (Cover and Hart, 1967) to the input sentence and checking the consistency of the breaks found in these sentences.

For each juncture of the input sentence, we label the juncture as the most frequently occurring juncture type in the aligned k sentences.

6.3. Results

The prediction-by-example technique was first evaluated using the full Penn Treebank tagset on both data sets. The results in Table 5 (row one) show that this technique performs worse than the rule-based punctuation algorithm in Table 1. The full Penn Treebank tagset was then replaced by the best-performing reduced POS tagset discussed in Section 5. These results are also shown in Table 5 (row two).

Table 5
Results of prediction-by-example

	Boston			MARSEC			Mean		
	P (%)	*R* (%)	*F* (%)	*P* (%)	*R* (%)	*F* (%)	*P* (%)	*R* (%)	*F* (%)
Full tagset	64.3	46.7	54.1	54.0	38.6	45.0	59.1	42.7	*49.6*
Reduced tagset	70.5	54.0	61.2	59.7	54.1	56.8	65.1	54.0	*59.0*
n-Gram splitting	70.9	55.3	62.1	61.9	55.4	58.5	66.4	55.4	*60.3*
Punctuation splitting	72.7	57.1	63.9	64.4	63.3	63.8	68.5	60.2	*63.9*
n-Gram & punctuation splitting	71.6	57.4	63.7	63.6	64.0	63.8	67.6	60.7	*63.8*
7-NN & splitting	83.3	60.8	70.3	73.5	66.5	69.8	78.4	63.6	*70.0*

All the experiments from Reduced tagset downwards all use the reduced tagset from Section 5.

Using the reduced tagset raised the *F* score from 49.57% to 58.96%, so that this model now out-performs the baseline punctuation prediction rules.

6.3.1. Islands of certainty

The results of our experiments with using two different algorithms for making the initial splits in the data set are shown in row three, four and five of Table 5. The algorithm benefit from both of the initial splitting approaches with the most significant increase in performance when punctuation was used to split the sentences on a first pass.

6.3.2. k-Nearest neighbour

The *k*-NN approach gave further improvements by making more informed predictions based on the multiple examples, as shown in row size of Table 5. We experimented with a range of values for *k* and found the best results were achieved when predictions considered the seven most similar examples from the training corpus. Fig. 4 plots the mean *F* scores obtained when varying the value of *k* between 1 and 50.

6.4. Analysis of errors

The main source of errors in the prediction-by-example algorithm is misalignment of breaks. A bad alignment occurs between two sentences when the algorithm finds a plausible match, yet it fails to correctly align 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 Fig. 5, where the break is misaligned so that it is placed after the WDT tag, when it should have been before it.

The fundamental problem with this approach is the small amount of labelled data available. When the unseen sentence is short, the probability of finding analogous sentences in the training data is high, but with long sequences, the likelihood of finding a suitable match in a limited amount of training data is low. This problem is still present when "islands of certainty" are used. In other words, this model of a

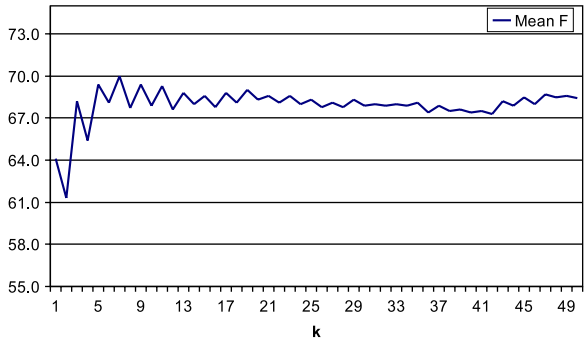

Fig. 4. Results of varying *k*.

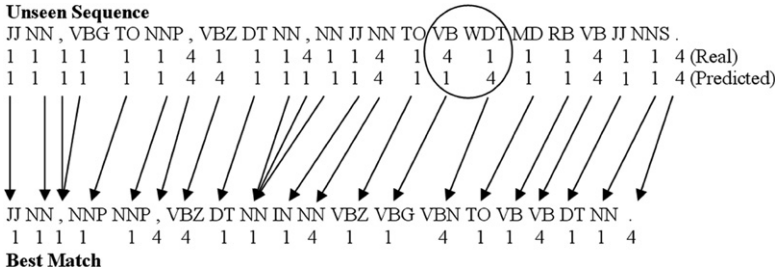

Fig. 5. Example of prediction-by-example realigning breaks.

whole sentence is too specific to be useful. In addition, the algorithm is very slow as it requires the comparison of many sentences. However, our experimentation with the prediction-by-example technique showed us that whole phrases within sentences often matched well, even when the phrases were somewhat dissimilar. This led to the idea that instead of representing a sentence as a single entity made up of a sequence of words, we should try to model individual phrases within the sentence. Hidden Markov models (HMMs) are well suited to this task.

7. 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 by combining them, make break predictions on the sentence as a single unit. This can be accomplished by estimating the best sequence of phrase models that "explains" an unseen sentence. Breaks are then postulated as occurring at the junctions of these phrase models. Hidden Markov models (HMMs) (Young et al., 2002) are well suited to the task of modelling sequences of symbols that display systematic variation. Furthermore, using HMMs allows us to adapt the powerful techniques developed for n-gram language modelling in speech recognition to form a probabilistic "grammar" of phrase sequences. This grammar can then be used to guide the decoding of the phrases, in exactly the same way that a statistical language model guides the decoding of an acoustic signal in speech recognition.

7.1. Phrase clustering

For the purposes of the experiments presented in this section, we define a phrase to be a sequence of words between two phrase breaks. In order to form models of phrases, it is required to first cluster together similar phrases. For instance, syntactically similar phrases like "the car", "the fast car", "the fast red car", "the blue cup" and "the hot cup" can be clustered together in a model of a single phrase. Each word in a phrase is replaced with its corresponding POS tag prior to clustering.

Phrases were grouped together using k-means clustering (Duda et al., 2000). Initially, the first k cluster centres are chosen randomly as phrases from the training data. The remaining phrases are then assigned to the "nearest" cluster centre. The distance between two phrases is calculated using the Levenshtein distance between each phrase's POS sequence, in the same way as the distance between sentences was estimated for prediction-by-example. From the resultant clusters, the cluster centres are calculated. A cluster centre is defined to be the "most central" phrase within the cluster, where "most central" means the phrase whose maximum distance to all the other phrases is the lowest of all the phrases in the cluster.

This process is repeated until no changes of cluster membership are encountered, typically taking between 3 and 6 iterations. A problem with this method of clustering is defining a suitable value for k. We tested a range of different values, and then analysed the effect they had on the overall accuracy.

Once all the phrases have been assigned to clusters, each cluster is given a unique index, the ith cluster being labelled as c_i. Each sentence in the training data can then be represented as a sequence of phrases. From this representation we estimate a bigram "language model" for phrases, which consists of a set of probabilities

$\Pr(\text{phrase}_{t+1} = c_j | \text{phrase}_t = c_i)$. When unseen sequences are encountered, a back-off scheme (Katz, 1987) is used in tandem with Good-Turing smoothing (Church and Gale, 1991; Good, 1953).

7.2. Training the phrase HMMs

Each cluster made from the training data is used to build a corresponding discrete HMM. One of these HMMs models the way in which the particular set of syntactically similar phrases used to train the model can be generated by a finite-state machine. Each state of the HMM has a set of P probabilities associated with it (where P is the number of POS symbols used), which are the probabilities of observing each POS tag in each state. Baum-Welch re-estimation (Young et al., 2002) is used to train these probabilities. All training and decoding of the models is performed with the HMM Toolkit (HTK) (Young et al., 2002).

7.3. Number of states

The HMMs are configured to model the prosodic phrases by setting the number of states in each HMM to the mean length of the phrases in the training data. When all the training examples in a particular cluster are shorter than the mean phrase length, the number of states for that model is reduced to the length of the largest training observation in that cluster. This is to ensure that all the states will have observations associated with them. Experiments were carried out in which the number of states in the model was set to the mean phrase length based on all the observations in just that cluster.

7.4. State transitions

All topologies in the HMMS are left to right in order to model word order. However, there are various possibilities for specifying the state transition matrix a_{ij}, which represents the probability of making a transition from state s_i to s_j. The most flexible topology (topology A), when four states are used, is shown in Fig. 6. Here any left to right transition is allowed i.e. $a_{ij} > 0 \ \forall \ j \geqslant i$, allowing a high degree of freedom in modelling.

Allowing $a_{ij} > 0$ for $j = i$ or $j = i + 1$ only limits the transition from one state to itself or the next state (topology B, Fig. 7). A slightly less restrictive variation of this is to also allow a transition to the final state, allowing for the insertion or deletion of words at the end of phrases (topology C, Fig. 8).

7.5. Decoding

Viterbi decoding (Forney, 1973) was used to segment a new input sentence into the most likely sequence of HMMs. The resulting transcription from the Viterbi decoder represents the most likely sequence of prosodic phrases, given the input POS observations, and the breaks are postulated at the junctures of these HMMs.

As well as the trained HMMs, the Viterbi decoder also utilized the phrase sequence bigram model. The "grammar scale factor" used in the HTK Viterbi decoder allows the contribution of the bigram model to be weighted against the contribution of the POS phrase HMMs. By experimenting with a range of different weight values, an optimum scale factor can be found.

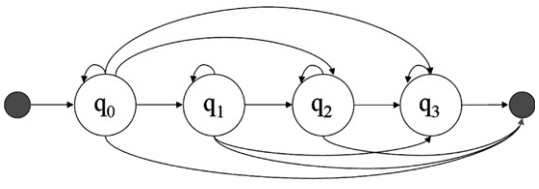

Fig. 6. HMM topology A – allowing transitions between s_i and s_j such that $j \geqslant i$.

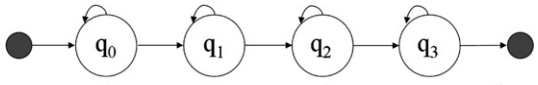

Fig. 7. HMM topology B – allowing transitions between s_i and s_j such that $j = i$ or $j = i + 1$.

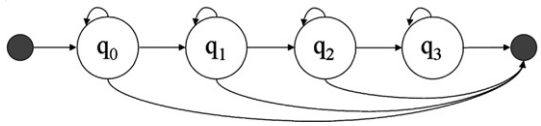

Fig. 8. HMM topology C – allowing transitions between s_i and s_j such that $j = i$, $j = i + 1$ or $j = FinalState$.

7.6. Optimization of the models

As hidden Markov models 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 first set of experiments used topology C shown in Fig. 8. The number of clustered phrases (HMMs) was set to 100 and the grammar scale factor was set to 1. We experimented with a range of different values for each parameter to find the optimal configuration. Once all of these values had been optimized individually, the best value for each parameter was used in a final model.

7.6.1. Number of phrases

HMMs were constructed for a range of number of phrases at intervals of 5, between 1 and 300. Fig. 9 shows the mean F score when evaluated on the Boston and MARSEC corpora.

Fig. 9 shows that the best range for the number of clusters is in the range 40–80 and that there is little difference in performance when the number of phrases was over about 100. The best result from these experiments was when 45 phrases were used, which yielded a mean F score of 56.66%.

7.6.2. States

As the number of states is directly related to the state transition values, it is important to optimize these two parameter sets jointly. For the three topologies outlined in 7.4, we varied the number of emitting states in each cluster between 1 and 25. Also, we experimented with setting the number of states to the mean phrase length of the training samples used to build the respective HMM so that the clusters have a variable number of states. Table 6 shows this approach almost always out-performed using a uniform number of states.

Table 6 shows the overall best result was achieved using seven emitting states with topology A (allowing any left to right transition, that is to allow $a_{ij} > 0 \; \forall \; j \geqslant i$). The fact that allowing a high degree of freedom

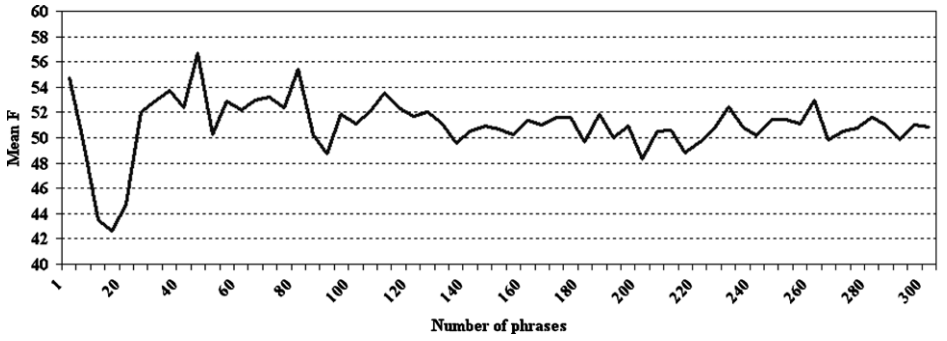

Fig. 9. The mean F score achieved when varying the number of phrases.

Table 6
Mean F scores using different topologies

Number of emitting states	Topology A	Topology B	Topology C
	Mean F (%)	Mean F (%)	Mean F (%)
1	34.0	34.0	34.0
2	37.5	37.5	38.5
3	51.1	51.1	52.3
4	57.2	53.7	56.5
5	59.9	52.1	56.3
6	61.0	51.4	52.6
7	*61.7*	49.2	50.0
8	57.7	47.6	47.5
9	58.7	46.8	46.3
10	56.4	44.8	45.2
15	53.9	42.4	46.2
20	54.6	42.7	46.9
25	54.6	42.7	46.9
Variable (mean phrase length)	60.2	*57.1*	*61.3*

in the state transitions gives best performance is probably due to the wide variability in length and composition of phrase sequences.

7.6.3. Decoding

Table 7 contains a summary of the results achieved with a range of different grammar scale factors.

Precision rises as the grammar scale factor increases, but at the expense of reduced recall. The best results were achieved when the grammar scale was set to 1, i.e. when the phrase sequence model and the phrase HMMs themselves contributed equally to the overall likelihoods. The mean gain in F score obtained from using the phrase sequence model is 2.3%

7.7. Results

Table 8 reports the results obtained when combining the best parameters from the initial set of experiments. The final model consisted of 45 HMMs, each with seven emitting states using topology A, and with a grammar scale factor of 1. The final mean F score of 63.70% presented in Table 8 clearly shows that combining the optimized parameters gives the best result achieved using this algorithm.

Table 7
Summary of results from varying the grammar scale factor during decoding

Grammar scale factor	Mean		
	P (%)	R (%)	F (%)
0	42.5	58.2	48.8
1	51.4	51.0	*51.1*
2	60.3	36.7	45.5
3	68.6	30.7	42.0
4	76.1	27.6	40.1
5	80.0	26.1	38.9
10	89.5	21.8	34.9
15	95.3	20.6	33.9
20	98.2	20.3	33.7
25	98.7	20.1	33.4
30	99.3	20.0	33.3

Table 8
Final results from predicting phrase breaks by modelling phrases

Boston			MARSEC			Mean		
P (%)	*R* (%)	*F* (%)	*P* (%)	*R* (%)	*F* (%)	*P* (%)	*R* (%)	*F* (%)
66.8	57.2	61.6	64.4	67.3	65.8	65.6	62.2	*63.7*

POS Tag	IN	NN	IN	DT	NN	NN	IN	DT	JJ	IN	NNP	.
Actual	1	4	1	1	1	4	1	1	1	1	1	4
Predicted	1	4	1	1	4	1	1	1	1	1	1	4

Fig. 10. Extract from a sentence showing a common error from the prediction by phrase modelling technique.

7.8. Analysis of errors

By far the most common error observed with this approach was to break prematurely by one word. An example of this error is illustrated in Fig. 10. This error may be because there is insufficient information in the training phrases supplied to partition the second and third phrases. Our conclusion is that although the HMMs do a good job at modelling phrases, there is simply not enough information in POS sequences to determine breaks accurately. In the next section, we describe experiments where information from a syntactic parse of the sentence was added into our classifier.

8. Syntactic parsing

There is clearly a strong relationship between phrase breaks and syntactical structure. For instance, 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." The automatically derived syntactic parse of this sentence is shown in Fig. 11. The phrase breaks in this case occur at exactly the junctures between the major phrases of the parse, and this behaviour is typical of fairly simple sentences.

As already demonstrated in Section 1, syntax alone cannot predict phrase breaks, but it is clearly of considerable use in the problem. The success of the early algorithms that used parse features was poor, but this might be ascribed to the low accuracy of the parsers then available. Recent advances (Charniak, 2000; Collins,

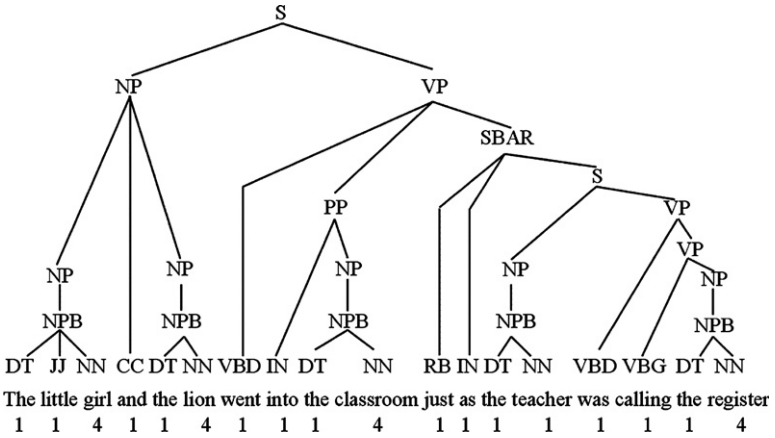

Fig. 11. Syntactic parse and prosodic phrase break for the sentence "The little girl and the lion went into the classroom just as the teacher was calling the register."

1999) have led to a number of highly accurate methods for parsing, which suggests that revisiting this technique may be advantageous. This was demonstrated by Koehn et al. (2000) who gained an increase in predicting prosodic phrasing when introducing parse features. In this section, we describe experiments that use features derived from a syntactical parse of a sentence to determine the sentence's break structure.

Building upon the work of Koehn et al. (2000), our motivation here is to find more powerful features for modelling the relationship between syntactic and prosodic tree structures. In addition to this, we wanted to combine this approach with benefits of the POS tag reduction algorithm reported in Section 5.

After a parse has been made of a sentence, we need to determine features from the parse that are likely to give information about the location of breaks. Koehn et al. (2000) presented a model that extended the work of Hirschberg and Prieto (1995), by adding the following parse features into their CART classifier:

- K0 – The size of the biggest phrase that ends in 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 parse tree.
- K3 – A binary flag denoting if the phrase is an SBAR, the assumption being that a break is common before sub-clauses.

Koehn et al. (2000) assumed that if a phrase was a single word (e.g. "and", "is") then it could be collapsed with the following phrase in the feature extraction process. We used the above features and added three extra features:

- K0A – The number of nodes dominated by the biggest phrase that ends in the current word.
- K2A – The number of nodes dominated by the next phrase on the same level of the parse tree.
- LPS – The phrase label assigned to the largest phrase ending at the current word. With the largest parent symbol, when the symbol is not a major phrase – such as NP, VP, PP, ADJP, ADVP – the default of S is used.

8.1. Parse depth

One of the features not considered in the work of Koehn et al. (2000) was that of the word's depth on the parse tree. By regarding the parse tree as a graph, we speculated that the number of nodes that are traversed in moving from a word to the next word in the sentence is a good predictor for prosodic phrase boundaries. Using this assumption, the following further features are extracted:

- PD – parse depth – the distance from the top of the tree to the current node.
- PDD – parse depth difference – the difference in parse depth of the ith and $(i+1)$th word. $\mathrm{PDD}_i = \mathrm{depth}_{i+1} - \mathrm{depth}_i$.
- DNW – distance to next word – the number of nodes that have to be ascended and descended in order to move from the ith word to the $(i+1)$th word.

The Collins parser (Collins, 1999) was used for automatically deriving syntactic parses of our data. With this parser, punctuation is not incorporated into the final parse, so where punctuation was not present in the tree, feature values were moved from the previous word.

Returning to the sentence shown in Fig. 11, the corresponding features for each juncture are shown in Table 9.

8.2. Classifier

It would theoretically be possible to incorporate the features mentioned above into our HMM models. The disadvantage of this approach is that these features are heterogeneous and take on a small number of discrete values rather than being continuously valued.

Table 9
Extract from the feature set corresponding to the sentence shown in Fig. 11

Word	K0	K1	K2	K3	K0A	K2A	PD	PDD	LPS	Break index
The	1	F	1	F	1	1	6	0	S	1
Little	1	F	1	F	1	1	6	0	S	1
Girl	3	T	1	F	7	1	6	−2	NP	4
And	1	F	2	F	1	5	4	2	S	1
The	1	F	1	F	1	1	6	0	S	1
Lion	6	T	12	F	16	35	6	−2	NP	4
Went	1	F	3	F	1	8	4	1	S	1
Into	1	F	2	F	1	5	5	2	S	1
The	1	F	1	F	1	1	7	0	S	1
Classroom	3	T	8	F	8	23	7	−2	PP	4

This type of feature is better-handled by a discriminative classifier such as a decision tree rather than a generative classifier such as an HMM. Furthermore, the use of the parse information incorporates automatically into features themselves the long-range effects within a sentence that the HMMs aimed to model. Hence, for these experiments, we have used a decision tree classifier rather than an HMM segmenter. Decision tree classifiers have been shown to be useful for the break prediction task (Hirschberg and Prieto, 1995; Ostendorf and Veilleux, 1994). The C4.5 classifier (quinlan, 1992) was used to build decision trees from the parse trees data. This classifier is essentially an extension of the ID3 algorithm that builds a decision tree from a set of feature vectors, each labelled with an appropriate class. C4.5 extends the ID3 algorithm by adding better handling of unseen attribute values with the ability to classify records that have unknown attribute values. The C4.5 algorithm has the added ability to work with numerical ranges, which is essential for dealing with the features we are utilizing.

8.3. Combining the parse features with reduced POS sequence model

In Section 5, it was shown that using a window of reduced POS tags around a juncture gave improved performance over using tags from a full set. As well as the aforementioned feature set, for each word we also used the POS trigram model as described in Section 5.3 to estimate the probability of there being a phrase break at each word juncture, and added this value to the feature vector. This probability is referred to as NGPB.

8.4. Results

In order to evaluate the predictive ability of the new features proposed above, models were constructed and evaluated using all the possible combinations of these features. As with the POS reduction experiments, there was a risk of selecting a feature set that happened to work well on the given test set, as opposed to giving good performance across different data sets. To gain a fair evaluation of the algorithm, evaluations were performed using both the MARSEC and Boston corpora, and then calculating the mean of these two results. Table 10 shows the performance when a model was built using each of the individual features.

Using the mean of the F scores from both data sets, Table 11 presents the 10 best combinations of features. From this table, we can see which features contribute most significantly to the prediction process. Based on the initial indications of Table 10, K0, K0A and DNW are shown to be efficient predictors for phrase breaks. The places where no result can be calculated indicate that some of the features lead to classifiers that label each juncture as a non-break. However, when these features are included in a larger feature vector, they do aid predictions, as shown in Table 11.

Next, the n-gram probability of a break (NGPB) was introduced into the feature set. Table 12 shows the results; the best mean F score rises from 74.98% to 80.19%.

Table 10
Performance using just individual parse features

Feature	Boston			MARSEC			Mean		
	P (%)	*R* (%)	*F* (%)	*P* (%)	*R* (%)	*F* (%)	*P* (%)	*R* (%)	*F* (%)
K0	79.8	69.1	74.1	79.5	62.8	70.2	79.7	65.9	*72.1*
K1	–	0.0	–	–	0.0	–	–	0.0	–
K2	33.8	2.0	3.9	–	0.0	–	–	1.0	–
K3	55.8	2.1	4.1	–	0.0	–	–	1.1	–
K0A	83.3	62.7	71.5	78.8	67.0	72.4	81.0	64.8	*72.0*
K2A	–	0.0	–	–	0.0	–	–	0.0	–
PD	–	0.0	–	–	0.0	–	–	0.0	–
PDD	67.7	60.6	63.9	82.1	31.0	46.0	74.9	45.8	*54.5*
LPS	65.2	7.7	13.8	52.3	9.8	16.6	58.7	8.8	*15.2*
DNW	86.0	46.9	60.7	77.4	59.1	67.0	81.7	53.0	*63.9*

Table 11
Ten best results from using all possible combinations of features derived from the parse tree

Feature	Boston			MARSEC			Mean		
	P (%)	*R* (%)	*F* (%)	*P* (%)	*R* (%)	*F* (%)	*P* (%)	*R* (%)	*F* (%)
K0, K1, K0A, PDD, DNW	81.4	70.2	75.4	81.1	69.0	74.6	81.3	69.6	*75.0*
K0, LPS, K0A, DNW	81.8	70.3	75.6	80.9	68.8	74.4	81.4	69.5	*75.0*
K0, K1, K3, K0A, PDD, DNW	81.4	70.2	75.4	81.1	69.0	74.6	81.3	69.6	*75.0*
K1, K0A, PDD, DNW	81.5	70.3	75.5	81.4	68.5	74.4	81.5	69.4	*74.9*
K1, K3, K0A, PDD, DNW	81.5	70.4	75.5	81.4	68.4	74.3	81.4	69.4	*74.9*
K0, K1, LPS, K0A, DNW	81.8	70.2	75.6	80.8	68.7	74.3	81.3	69.5	*74.9*
K0, K1, K0A, PDD	81.3	70.4	75.5	81.3	68.5	74.4	81.3	69.5	*74.9*
K0, LPS, K0A, K3, DNW	81.8	70.2	75.6	80.9	68.5	74.2	81.4	69.4	*74.9*
K0, K1, K3, K0A, PDD	81.3	70.5	75.5	81.3	68.4	74.3	81.3	69.4	*74.9*
K0, K1, K0A, DNW	81.4	70.1	75.3	80.9	69.0	74.4	81.1	69.5	*74.9*

Table 12
Ten best results from using all different combinations of features derived from the parse tree combined with the n-gram probability model

Feature	Boston			MARSEC			Mean		
	P (%)	*R* (%)	*F* (%)	*P* (%)	*R* (%)	*F* (%)	*P* (%)	*R* (%)	*F* (%)
K3, K0A, NGPB, PDD, DNW	82.4	76.9	79.6	85.9	76.3	80.8	84.2	76.6	*80.2*
K0A, NGPB, PDD, DNW	82.1	77.0	79.5	85.9	76.4	80.9	84.0	76.7	*80.2*
K0, K0A, NGPB, PDD, DNW	82.8	76.7	79.6	86.5	75.7	80.7	84.6	76.2	*80.2*
K0, K3, K0A, NGPB, PDD, DNW	82.4	77.0	79.6	86.4	75.6	80.7	84.4	76.4	*80.2*
K0A, NGPB, DNW	82.2	76.6	79.3	86.1	76.5	81.0	84.1	76.5	*80.1*
K0, K3, K0A, NGPB, DNW	83.2	76.1	79.5	86.4	75.6	80.6	84.8	75.8	*80.1*
K3, K0A, NGPB, DNW	82.3	76.4	79.2	85.9	76.4	80.8	84.1	76.4	*80.0*
PD, K0, K3, K2A, NGPB, DNW	82.8	76.7	79.7	85.8	75.6	80.4	84.3	76.2	*80.0*
PD, K0, K2A, NGPB, DNW	83.0	76.9	79.8	85.6	75.5	80.2	84.3	76.2	*80.0*
K0, K3, K2A, NGPB, DNW	82.6	76.3	79.3	85.6	76.3	80.7	84.1	76.3	*80.0*

Table 13
Comparison of our the performance of our feature set with the performance of the Koehn feature set

Feature	Boston			MARSEC			Mean		
	P (%)	*R* (%)	*F* (%)	*P* (%)	*R* (%)	*F* (%)	*P* (%)	*R* (%)	*F* (%)
K0, K1, K2, K3	85.6	63.5	72.9	73.0	69.2	71.0	79.3	66.3	*72.0*
K0A, DNW	81.9	70.0	75.5	81.1	68.3	74.2	81.5	69.2	*74.8*
K0A, DNW, NGPB	82.2	76.6	79.3	86.1	76.5	81.0	84.1	76.5	*80.1*

As we stated in the beginning of this section, our primary research goal was to find more sophisticated features for modelling the relationship between syntactic and prosodic tree structures. Table 13 shows a comparison of the results of predicting prosodic phrase breaks using the features introduced in Koehn et al. (2000) compared with some of the new features introduced in this study. The results show a significant improvement from the introduction of our new features, with a substantial improvement when combined with the modified Taylor and Black n-gram model using the reduced POS tag (introduced in Section 5).

Although not present in these results, the single best F score of 81.56% was achieved by combining K0, K1 and NGPB, when evaluated on the MARSEC corpus. However, on the Boston data, this same feature set gave 77.91%. Similarly, with the Boston data, combining K1, K2, K2A, PDD and NGPB gives an F score of 79.93%, but when tested on the MARSEC corpus it achieves 78.82%. These results indicate that choice of features is sensitive to the material used. Using the Taylor and Black measures, our best configuration correctly identifies 93.4% of junctures correctly, getting 76% of breaks correct.

8.5. Analysis of errors

It was instructive to examine some of the junctures on which the algorithm had made errors. Shown below are some examples of errors made by the algorithm. Phrase breaks are denoted by ||. R is used to denote the reference annotation of the sentence, and P denotes the predicted break values.

(1) R: In nineteen seventy-six,|| Democratic Governor || Michael Dukakis || fulfilled a campaign promise || to de-politicize || judicial appointment.
P: In nineteen seventy-six,|| Democratic Governor Michael Dukakis || fulfilled a campaign promise || to de-politicize judicial appointment.
(2) R: Barry Cane || of Hyannis || is accused || of trying || to import || more than five kilos || of cocaine.
P: Barry Cane || of Hyannis || is accused of trying to import more than five kilos || of cocaine.
(3) R: The increase in the || earnings limit || for pensioners || is also worth a comment.
P: The increase in the earnings limit for pensioners || is also worth a comment.
(4) R: Craig Mactenel || is legislative liaison || for he Fisheries and Wildlife || Department.
P: Craig Mactenel || is legislative liaison || for the Fisheries || and Wildlife Department.
(5) R: The State|| public health department || must inspect everything || from milk plants || to hospitals || to police station lockups.
P: The State public health department || must inspect everything || from milk plants || to hospitals || to police station lockups.
(6) R: Kassler says,|| unlike the Federal || Supreme Court, || there's no litmus test || on particular issues || that Massachusetts high court nominees || must pass.
P: Kassler || says,|| unlike the Federal Supreme Court, || there's no litmus test || on particular issues || that Massachusetts high court nominees || must pass.

In the above examples – most notably 1, 2 and 3 – it seems that many reference breaks could be omitted without any noticeable loss in quality. They reflect the nature of the data, which was news reports broadcast over the radio. Because their scripts are "semantically dense", newsreaders tend to read slowly and pause before or after semantically important words to emphasize them. Although the classifier was trained on material of this nature, it may be that the resulting phrases are more closely correlated with naturally spoken phrases than syntactic parse constituents, therefore making them harder to identify.

The error in sentence 4 is due to the algorithm's ignorance of "Fisheries and Wildlife" as a collocation. Examples 5 and 6 demonstrate how semantic and long-range considerations are used by humans when planning a sentence's prosody. The break after "State" in example 5 may be to emphasize that it is the State rather than Federal public health department. Similarly, in example 6, there is a break after "Federal" to distinguish the Federal Supreme Court from the Massachusetts high court. Predicting breaks of this nature clearly require an understanding of the long range sentence structure and the semantics of contrasting concepts that is beyond what is implemented in this classifier.

Table 14
Comparison of the algorithms presented in this paper

Algorithm		Mean		
		P (%)	*R* (%)	*F* (%)
Baselines	Punctuation Model	88.7	40.6	*55.2*
	n-Grams (full tagset)	80.3	63.7	*71.0*
POS reduction	n-Grams & Reduced tagset	80.4	71.9	*75.9*
Prediction-by-example	Standard tagset	59.1	42.7	*49.6*
	Reduced tagset	65.1	54.0	*59.0*
	7-NN & splitting	78.4	63.6	*70.0*
Phrase modelling	Prediction-by-phrase-modelling (HMMs)	65.6	62.2	*63.7*
C4.5	Syntactic features	84.0	76.7	*80.2*

9. Summary of results

Table 14 summaries the performance of all the algorithms presented in this paper. The use of syntactic features has been shown to significantly outperform all the other algorithms.

10. Summary and discussion

This paper has examined some different techniques for the prediction of phrase breaks in sentences. In contrast to previous approaches that used information gathered locally from the neighbourhood of a juncture, our philosophy has been to develop techniques that consider the whole sentence when estimating the break positions. This led us to investigate first an approach based on using the similarity between the unseen sentence and stored sentences. Although results obtained on this technique are above the baseline established by using punctuation to demarcate breaks, this technique is clearly not general enough for practical use, and is also computationally slow. Our second approach was to build stochastic models of phrases and use these to segment an unseen sentence. These two techniques performed well, but our conclusion was that POS features did not carry enough information to enable a good segmentation. When long-range features derived from a syntactic parse of a sentence were combined with an n-gram probability derived from a localized window, and used these within a decision-tree classifier, we obtained our best result. All our techniques benefited from using a reduced POS tagset of about 7–8 symbols derived using a search procedure, rather than a full set of over 40 symbols. Our best result on the MARSEC corpus ($P = 86.3\%$, $R = 77.4\%$, $F = 81.6\%$) outperforms the results reported by Busser et al. (2001) ($F = 74.4\%$) and Taylor and Black (1998) ($F = 78.3\%$). Koehn et al. (2000), used a different data set and obtained an F score of 82.0%. However, their feature set included hand-annotated pitch accents, which prevents a direct comparison. When only syntactic features were used, their performance was 70.8%.

As discussed in Section 8.5 the majority of the prediction errors were caused by ignorance of the semantics of the sentences. A complete solution to the break-placement problem undoubtedly demands semantic and pragmatic processing, but it is not clear how much further approaches based on deriving syntactically based features (either local to a juncture or across a whole sentence) can be taken. Future improvements in the accuracy of syntactic parsers should help increase performance of the techniques presented here.

The analysis of errors also demonstrates that the placement of some breaks in the data set were debatable. Koehn et al. (2000) conducted research into the use of optional breaks to annotate breaks where there was a level of uncertainty. During evaluation, errors associated with uncertain breaks were ignored, which led to a level of accuracy close to that achieved by a trained human annotator on the same task.

The Dutch data set used for Marsi et al. (2003) is significant as it was crafted specifically for the purposes of training a phrase break predictor to learn where phrases should ideally be placed, as opposed to the Boston and MARSEC corpora which marked where phrase breaks actually occurred in read speech. The analysis of errors in Section 8.5 showed that news readers tended to over emphasize their speech. Results obtained using classifiers trained on such speech may not transfer well to other applications. TTS ideally requires a neutral rendering of the speech which requires a corresponding annotated database.

The success of the C4.5 technique may be attributed to the fact that it is capable of utilizing and integrating a number of local and long-distance features, whereas the other techniques are more suited to using a single feature. There is still scope for research into extracting features from the syntactic parse tree. Possible candidate features include the position of the word within the sentence and within the syntactic phrase, the location of the word relative to the start and end of the sentence, the length of the word, the actual orthographic form of common words, etc.

Some recent work using machine learning techniques including maximum entropy (Cox, 2005) and support vector machines (Decoste and Schölkopf, 2002) suggests that further improvement may be possible. Investigation of these techniques will form part of our future work. However, from a text-to-speech application point of view, it may be more interesting to focus on how serious the effect of a misplaced break is. An error in the placement of a break may have a disastrous effect on the semantics or intelligibility of a synthesized sentence, or it may be relatively unimportant. If it were possible to quantify this effect, it should be possible to devise a break placement algorithm that minimized the Bayes risk and produced speech that was free of serious prosodic errors.

References

Allen, J., Hunnicutt, S., Carlson, R., Granstrom, B., 1979. MITalk-79: The 1979 MIT text-to-speech system. In: Wolf, Klatt (Eds.), Speech Communications Papers Presented at the 97th Meeting of the ASA. pp. 507–510.

Bachenko, J., Fitzpatrick, E., 1990. A computational grammar of discourse-neutral prosodic phrasing in English. Speech Communication 16 (3), 155–170.

Beckman, M., Pierrehumbert, J., 1986. Intonational Structures in English and Japanese. Phonology Yearbook 3, 255–310.

Bellman, R., 1962. Dynamic programming treatment of the travelling salesman problem. Journal ACM 9 (1), 61–63.

Breiman, L., Friedman, J., Olshen, R., Stone, C., 1984. Classification and Regression Trees. Chapman and Hall, New York.

Brill, E., 1995. Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. Computational Linguistics 21 (4), 543–565.

Busser, G., Daelemans, W., van den Bosch, A., 2001. Predicting phrase breaks with memory-based learning. In: Proceedings of 4th ISCA Tutorial and Research Workshop on Speech Synthesis. Perthshire Scotland, pp. 29–34.

Campbell, N.W., Isard, S.D., 1991. Segment durations in a syllable frame. Journal of Phonetics 19, 37–47.

Charniak, E., 2000. A maximum-entropy-inspired parser. In: Proceedings ANLP-NAACL, Seattle, Washington.

Church, K.W., Gale, W.A., 1991. A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. Computer Speech and Language 5, 19–54.

Collins, M., 1999. Head-driven statistical models for natural language parsing. Ph.D. thesis, University of Pennsylvania, PA.

Cover, T., Hart, P., 1967. Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13, 21–27.

Cox, S., 2005. Generalised probabilistic descent applied to phrase break modelling. In: Proceedings of Eurospeech. Lisbon, Portugal.

Daelemans, W., Zavrel, J., Berck, P., Gillis, S., 1996. MBT: A memory-based part of speech tagger-generator. In: Ejerhed, E., Dagan, I. (Eds.), Fourth Workshop on Very Large Corpora. Copenhagen, pp. 14–27.

Daelemans, W., Zavrel, J., van der Sloot, K., van den Bosch, A., 2004. TiMBL: Tilburg Memory Based Learner, version 5.1, Reference Guide. Tech. rep., Tilburg University, The Netherlands, iLK Research Group Technical Report Series No. 04-02.

Decoste, D., Schölkopf, B., 2002. Training invariant support vector machines. Machine Learning 46 (1–3), 161–190.

Dommergues, J., Grosjean, F., 1981. Performance structures in the recall of sentences. Memory and Cognition 9, 478–486.

Duda, R.O., Hart, P.E., Stork, D.G., 2000. Pattern Classification. Wiley.

Fach, M., September 1999. A comparison between syntactic and prosodic phrasing. In: Eurospeech. vol. 1 Budapest, pp. 527–530.

Forney Jr., G.D., 1973. The Viterbi Algorithm. Proceedings of the IEEE 61 (3), 268–278.

Gee, J.P., Grosjean, F., 1983. Performance structures: a psycholinguistic and linguistic appraisal. Cognitive Psychology 15, 411–458.

Good, I.J., 1953. The population frequencies of species and the estimation of population parameters. Biometrika 40, 16–264.

Grosjean, F., Grosjean, L., Lane, H., 1979. The patterns of silence: performance structures in sentence production. Cognitive Psychology 11, 58–81.

Hirschberg, J., 1993. Pitch accent in context: predicting intonational prominence from text. Artificial Intelligence 63 (1–2).

Hirschberg, J., Prieto, P., 1995. Training intonational phrasing rules automatically for english and spanish text-to-speech. Speech Communication 18, 281–290.

Katz, S.M., 1987. Estimation of probabilities from sparse data for the language model component of a speech recogniser. IEEE Transactions on Acoustics, Speech, and Signal Processing 35 (3), 400–401.

Koehn, P., Abney, S., Hirschberg, J., Collins, M., 2000. Improving intonational phrasing with syntactic information. In: Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP).

Levenshtein, V.I., 1966. Binary codes capable of correcting deletions, insertions and reversals. Cybernetics and Control Theory 10 (8), 707–710, originally in Doklady Akademii Nauk SSSR 163(4): 845–848 (1965)..

Marsi, E., Reynaert, M., van den Bosch, A., Daelemans, W., Hoste, V., 2003. Learning to Predict Pitch Accents and Prosodic Boundaries in Dutch. In: Hinrichs, E., Roth, D. (Eds.), Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics. pp. 489–496. Available from: <http://www.aclweb.org/anthology/P03-1062.pdf>.

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

Ostendorf, M., Veilleux, N., 1994. A hierarchical stochastic model for automatic prediction of prosodic boundary location. Computational Linguistics 20 (1), 27–54.

Pearl, J., 1984. Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley Longman Publishing Co., Inc..

Pierrehumbert, J.B., 1980. The Phonology and Phonetics of English Intonation. Ph.D. thesis, Massachusetts Institute of Technology.

Pitrelli, J.F., Beckman, M.E., Hirschberg, J., 1994. Evaluation of prosodic transcription labeling reliability in the ToBI framework. In: Proceedings of the International Conference on Spoken Language Processing (ICSLP). Yokohama, Japan, pp. 123–126.

Quinlan, J., 1992. C4.5: Programs for Machine Learning. Morgan Kaufmann.

Read, I.H., Cox, S.J., October 2004. Using Part-of-Speech for Predicting Phrase Breaks. In: Proceedings of the International Conference on Spoken Language Processing (ICSLP). Seoul, South Korea.

Roach, P., Knowles, G., Varadi, T., Arnfield, S., 1993. MARSEC: a machine-readable spoken English corpus. Journal of the International Phonetic Association 23 (2), 47–53.

Santorini, B., 1990. Part-of-Speech Tagging Guidelines for the Penn Treebank Project. Tech. rep., University of Pennsylvania.

Silverman, K., 1987. The Sructure and Processing of Fundamental Frequency Contours. Ph.D. thesis, University of Cambridge.

Silverman, K., Beckman, M.E., Pitrelli, J., Ostendorf, M., Wightman, C., Price, P., Pierrehumbert, J., Hirschberg, J., 1992. ToBI: a standard for labelling English prosody. In: Proceedings of the International Conference on Spoken Language Processing (ICSLP). vol. 2. Banff, Canada, pp. 867–870.

Sorin, C., Larreur, D., Llorca, R., 1987. A rhythm-based prosodic parser for text-to-speech systems in French. In: Proceedings of 11th International Congress of Phonetic Sciences vol. 1, pp. 125–128.

Taylor, P., Black, A.W., 1998. Assigning phrase breaks from part-of-speech sequences. Computer Speech and Language 12, 99–117.

van Rijsbergen, C.J., 1975. Information Retrieval. Butterworths, London.

van Santen, J., 1998. Timing. In: Sproat, R. (Ed.), Multilingual Text-To-Speech Synthesis: The Bell Labs Approach. Kluwer, Dordrecht, pp. 115–140.

Wang, M., Hirschberg, J., 1992. Automatic classification of intonational phrase boundaries. Computer Speech and Language 6.

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