USING STATISTICAL MODELS TO PREDICT PHRASE BOUNDARIES FOR SPEECH SYNTHESIS

Eric Sanders[1] and Paul Taylor[2]

(1) Nijmegen University, Netherlands.
email: eric@challenge1.let.kun.nl
(2) Centre for Speech Technology Research, University of Edinburgh, Edinburgh, U.K.
email: Paul.Taylor@ed.ac.uk
http://www.cstr.ed.ac.uk

ABSTRACT

This paper describes a variety of methods for inserting phrase boundaries in text. The methods work by examining the likelihood of a phrase break occurring in a sequence of three part-of-speech tags. The paper explains this basic technique and desribes more sophisticaed variations using distance probabilities.

1. INTRODUCTION

The derivation of the prosodic phrase structure of a sentence is an important part of the text analysis module of any text-to-speech (TTS) system. The derived prosodic structure is used in a number of subsequent modules including the duration module, where the location of phrase boundaries is needed for the correct prediction of phrase-final lengthening effects, and the intonation module where the location of the phrase boundaries helps in deciding where nuclear accents should be placed.

It is well known that there is a relationship between the prosodic and syntactic structures of a sentence. It is not surprising then that many TTS systems derive the prosodic structure by first performing a syntactic parse, and then manipulating this structure to produce a prosodic parse. However, syntactic parsing modules often have large space and time requirements and nevertheless make substantial numbers of errors on unrestricted text. These considerations have lead to simpler approaches whereby the prosodic structure is derived directly from the text [4], [5].

This paper describes a series of experiments designed to extract prosodic phrase structure directly from text.

2. DATA

The basic idea of the experiments was to use part-of-speech (POS) tags to predict phrase boundaries. We used the Spoken English Corpus (SEC) [3] and [1] which is a compilation of spoken British English (taken mainly from BBC Radio 4 broadcasts). The text is annotated with POS tags, parse trees and prosodic boundaries. The annotations were produced manually with the exception of the parsing, which was done semi-automatically. The prosodic transcription was done by two expert transcribers. The part of speech tags assigned to the words in the SEC are from the CLAWS tagset.

3. WORD SEQUENCES

All the experiments are based on overlapping sequences of three POS tags (i.e. *trigrams).* A preliminary experiment showed that the best use of these trigram sequences was in the prediction of a phrase break between the second and third words in the sequence.

The algorithms described below make use of the probability of a break occurring within a trigram. This probability is calculated from the training data by finding all the sequences of a particular trigram, finding out how many breaks are marked, and dividing the number of breaks by the total number of occurrences of the sequence.

4. PART OF SPEECH TAGGING

There are about 200 different tags in the original CLAWS tagset, which is too many for direct training (this would result in $200^3 = 8,000,000$ possible trigrams, whereas there are only 18131 words in the training set). The tags that were used were *adjective, adverb, noun, determiner, subjunction & conjunction, preposition, pronoun, auxiliary verb, main verb* and *other*.

For the experiments described here, we used the hand marked POS tags in the SEC. For speech synthesis use, the POS tagging must be done automatically and this can be accomplished using algorithms such as that described by Brill [2].

5. PARSING STRATEGIES

5.1. Method 1

This method makes use of the trigram probabilities alone. It considers every trigram in a sentence and places a break when the probability is over a certain threshold T. T can vary from 0 (always a break) to 1.0 (only a break when all occurrences in the training data had a break).

When the probability of a certain trigram is 0.5 and T is assigned 0.5, then errors will occur in about 50% of the cases where that trigram appears. The closer to 0 or 1 a probability is, the fewer errors occur.

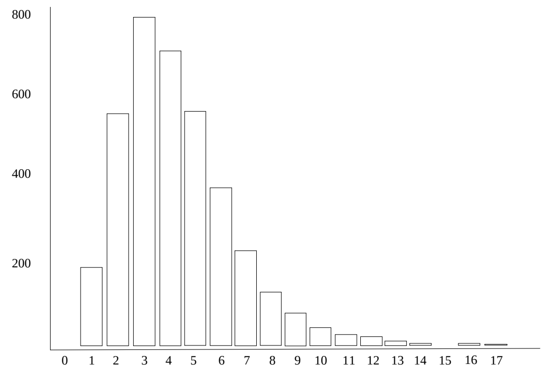

Figure 1. This plot shows the distribution of phrase lengths. The x-axis gives the length of the phrase in words and the y-axis gives the number of phrases of that length in the database.

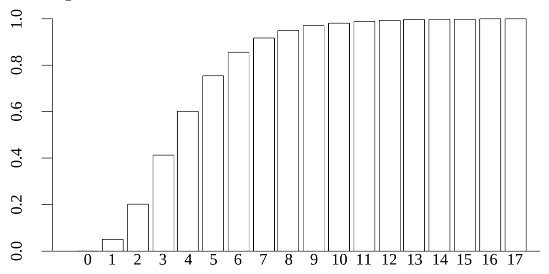

Figure 2. This plot shows the phrase distance probability distribution.

5.2. Method 2

A basic failing of the simple trigram approach is that it doesn't take into account the fact that the location of a prosodic phrase break is dependent on where the last break occurred. It is unlikely that three phrase breaks will occur after consecutive words; it is also unlikely that a sequence of, say, 30 words will be spoken without a break.

To rectify this failing in the basic trigram approach we developed an additional measure termed the *distance probability*, which is calculated from the phrase length distribution characteristics of the corpus. Figure 1 shows the phrase-length distribution.

The longest phrase in the corpus had 17 words. To calculate the phrase-length characteristics, the number of phrases of length 1, 2, 3 up to 17 are counted and divided by the total number of phrases. This gives a probability of a phrase being exactly n words long. The *distance probability* is the probability that there is a break after exactly n words given that there hasn't been a break before this word, and is calculated by summing all the phrase-length probabilities from 1 to n. Figure 2 shows the distance probability distribution.

Thus, the greater the distance from the previous break, the higher the probability of a break being inserted.

Method 2 combines the trigram probabilities with the distance probability by keeping a counter indicating the distance from the last break. The algorithm calculates the probability of aa break by multiplying the trigram probability and the distance probability. If the total probability is over the threshold T, a break will be inserted. As soon as a break is inserted, the phrase-length counter is reset to 0.

5.3. Method 3

This method works by using a lookahead of L trigrams. All L trigrams are examined by considering the normal trigram break probability and the distance probability. The trigram which has the highest probability is tested against the threshold T and if this is exceeded a break is inserted. Once a break has been inserted, the algorithm starts again by examining the next L trigrams after this break.

The advantage of this method is that it picks a single best trigram in a relatively long sequence of words, thus ensuring that breaks are sparsely placed and when they are placed they are placed in the best location. The method will however make errors in the few cases where breaks should be placed close together.

5.4. Method 4

This method uses an exhaustive search for the best combination of breaks in a sentence using the trigram probabilities. It examines every possible combination of boundaries between words in a sentence and computes the probability of that combination. After all the combinations have been examined, it picks the one with the highest probability and inserts the breaks accordingly.

If a sentence has N words, then there are 2^{N-2} combinations possible (because trigrams are used, the place after the second word is the first to be considered for a break). A sentence with 4 words has the following possibilities:

word word word word word
word word word word | *word*
word word word | *word word*
word word word | *word* | *word*
word word | *word word word*
word word | *word word* | *word*
word word | *word* | *word word*
word word | *word* | *word* | *word*

All possible combinations of breaks and trigram probabilities are calculated and the most probable pattern is chosen as being correct. This method obviously incurs heavily computational cost because of its exhaustive search. A sentence with a length of 22 words, which is not uncommon in the corpus, will cause over a million combinations to be examined.

5.5. Method 5

This method employs a limited exhaustive search by using a two pass approach. The first pass places phrase breaks at every punctuation mark, and the second pass performs an exhaustive search between these breaks. This drastically reduces the search space, but there are still some sequences in the corpus which have too many

Method	% Correct breaks	% Inserted breaks	% Word-pair correct	Adjusted Word-pair score
1	70	22	89	0.5
2	68.5	27	87	0.45
3	70	16	89	0.5
4	72	14	90	0.54
5	69	26	87	0.44

Table 1. Results.

words in a sequence without any punctuation, so a further limit on maximum sequence length was imposed.

6. TESTING AND RESULTS

The training data consisted of 18131 words and 3526 breaks. The test set consisted of 6420 words and 1376 breaks. Because of the exhaustive search conducted for method 4, it was impractical to train and test this on the entire data and smaller amounts of data were used.

6.1. Method of computing the scores

It should be clear that there isn't a single best method for calculating the accuracy of placing breaks. Here we employ two methods.

The first method is a simple percentage score of whether the breaks that an algorithm places are the same as those in the database. The total number of correct breaks and the total number of spurious inserted breaks are calculated are divided by the total number of breaks to give percentage scores.

The second method examines every word pair in the test set. If there is a break and the algorithm places a break, this is correct. If there is no break and the algorithm does not place a break, this is also correct. Otherwise there is an error. The total score is the number of correct word pairs divided by the total number of word pairs. A disadvantage of this technique is that it doesn't take into account the natural skew in the data regarding the number of breaks to the number of non-breaks. For example, if only 10% of the word pairs contained a break, an algorithm could score 90% by not placing any breaks at all. Therefore an additional measure was employed that took this into account: if the score using the normal method is S and the proportion of non-breaks to the number of word-pairs is B then the adjusted score is $\frac{S-B}{1-B}$.

In the SEC, 78% of the inter-word spaces are non-breaks ($B = 0.78$). If, a method scores 90% of the inter-word spaces correct ($S = 0.9$) then the adjusted score is $\frac{0.9-0.78}{1-0.78} = 0.55$. In this method a score of below 0 indicates the algorithm is doing no better than chance.

Table 1 gives the results of the five methods.

7. DISCUSSION

All the algorithms reported here have achieved high accuracy in finding phrase breaks. It is somewhat surprising how similar all the results are, with the word-pair scores ranging from 87% to 90%. It is surprising that the results for the simple trigram technique perform slightly better than methods 2 and 3 which use phrase-length information. There are many ways in which the trigram and distance probabilities can be combined, and the multiplication method adopted here is non-optimal. Future work will consider more sophisticated schemes for combining the two sources of information.

A much bigger difference is observed in the numbers of insertions for each method, with methods 3 and 4 producing far fewer insertions than the other methods.

The results for method 4 are also interesting. Using method 4 in an actual synthesis system is clearly impractical, but the results are useful in that they give an upper limit of performance using the trigram approach. As the results for the other techniques are close to this figure, it is clear these techniques are close to their optimal performance. Different techniques will have to be used to increase performance past the 90% mark.

Most of the techniques rely on a threshold value which has been chosen to keep the numbers of insertions and deletions equal. Depending on the exact purpose to which the system is put, insertion errors may be deemed more important than deletions and vice-versa. The threshold value can be varied to achieve a different balance between insertions and deletions.

It is also important to note that the algorithms are attempting to mimic the patterns of the phrase breaks in the database rather than to produce phrase breaks in acceptable places explicitly. While this makes for easy training and testing, there are some difficulties with this approach. Speakers are not necessarily consistent in the placing of phrase breaks in a sentence. For a given text, there may be many possible phrase break placings which are deemed acceptable. The technique described here does not take this into account and therefore a break which is incorrect in our scoring method may still be acceptable to listeners.

The techniques have a number of practical advantages which make them suitable for use in a real-time text-to-speech system. Except for the exhaustive search methods, the algorithms are fast and efficient. Furthermore, they can easily be re-trained on any suitable corpus.

REFERENCES

[1] Simon Arnfield. *Prosody and Syntax in Corpus Based Analysis of Spoken English*. PhD thesis, University of Leeds, 1994.

[2] Eric Brill. *A Corpus-Based Approach to Language Learning*. PhD thesis, University of Pennsylvania, 1993.

[3] G. Knowles and L. Taylor. Manual of information to accompany the SEC corpus. Technical report, UCREL:The University of Lancaster, 1988.

[4] Kim Silverman. *The Sructure and Processing of Fundamental Frequency Contours*. PhD thesis, University of Cambridge, 1987.

[5] N.M. Veilleux, M .Ostendorf, P. J. Price, and S. Shattuck Hufnagel. Markov modelling of prosodic phrase structure. In *International Conference on Speech and Signal Processing*. IEEE, 1990.