ASSIGNING PHRASE BREAKS FROM PART-OF-SPEECH SEQUENCES

Alan W Black and Paul Taylor

Centre for Speech Technology Research, University of Edinburgh,
80, South Bridge, Edinburgh, U.K. EH1 1HN
http://www.cstr.ed.ac.uk
email: awb@cstr.ed.ac.uk, Paul.Taylor@ed.ac.uk

1. BACKGROUND

One of the important stages in the process of turning unmarked text into speech is the assignment of appropriate phrase break boundaries. Phrase break boundaries are important to later modules including accent assignment, duration control and pause insertion.

A number of different algorithms have been proposed for such a task, ranging from the simple to the complex. These different algorithms require different information such as part of speech tags, syntax and even semantic understanding of the text. Obviously these requirements come at differing costs and it is important to trade off difficulty in finding particular input features versus accuracy of the model.

The simplest models are deterministic rules. A model simply inserting phrase breaks after punctuation is rarely wrong in assignment, but massively underpredicts as it will allow overly long phrases when the text contains no punctuation. More complex rule-driven models such as [1] involve much more detailed rules and require the input text to be parsed. On the other hand statistically based models offer the advantages of automatic training which make movement to a new domain or language much easier. Simple direct CART models using features such as punctuation, part of speech, accent positions etc. can produce reasonable results [5]. Other more complex stochastic methods optimising assignment over whole utterances (e.g. [8]) have also been developed.

An important restriction that sometimes is ignored in these algorithms is that the inputs to the phrase break assignment algorithm have to be available at phrase break assignment time, and themselves be predictable from raw text. For example, some algorithms require accent assignment information but we believe accent assignment can only take place after prosodic boundaries are identified. A second example is the requirement of syntactic parsing of the input without providing a syntactic parser to achieve this. Thus we have ensured that both our phrase break assignment algorithm is properly placed within a full text to speech system and that the prediction of any required inputs is included in our tests.

A second requirement for our algorithm was introduced by our observation that many phrase break assignment algorithms attempt to estimate the probability of a break at some point based only on local information. However, what may locally appear as a reasonable position for a break may in fact be less suitable than the position after the next word. That is, assignment should not be locally optimised but globally optimised over the whole utterance. For example in the sentence

I wanted to go for a drive in the country.

a potential good place for assignment may locally appear to be between “drive” and “in” based on part of speech information. However in the sentence

I wanted to go to a drive in.

such a position is unsuitable. Another example is a uniform list of nouns. Breaks between nouns are unusual but given a long list of nouns (e.g. the numbers 1 to 10) it then becomes reasonable to insert a phrase break.

Thus we wish our model to have reasonable input requirements, use predicted values for the inputs as part of the test and consider global optimisation of phrase break assignment over the whole utterance.

2. BASIC MODEL

We formally define the problem as follows. Between every pair of words is a *juncture*, which can take one of a number of *juncture types*. In the simplest case the set of juncture types consists of break and non-break (the only case discussed here), but in principle any number is possible. The task of the algorithm is to decide the best sequence of juncture types for each sentence.

Our algorithm uses a Markov model where each state represents one of the juncture types and emits probabilities of part-of-speech (POS) sequences occurring. We usually take two words before and one after the juncture in questions to represent the POS sequence:

$$P(c_{k-1}, c_k, c_{k+1} | j_k)$$

where j is the juncture in question and c_k is the POS tag immediately before it.

In the simplest case there are only two states in the model, one for break and the other for non-break.

The transition probabilities are determined by the prior probability of each juncture type occurring. Bayes' rule is used to combine the emission probability of each state ($P(C|j)$) with the transition probability ($P(j)$), to find the probability we are interested in, $P(j|C)$. This case represents local information only, i.e. the basic probability of each juncture type at each point. To take more context into account we use an n-gram of juncture sequences. The ngram gives the probability of a juncture type given the previous N junctures, i.e.

$$P(j_k|j_{k-1}, j_{k-2}, \ldots, j_{k-N+1})$$

For an ngram of size N, a Markov model is built with 2^{N-1} nodes, each corresponding to a particular unique sequence of juncture types. The transition probabilities are taken from the ngram and give the likelihood of particular sequences occurring. All states of a particular juncture type are tied, i.e. the emission probabilities of all break states are equal regardless of position in the network.

Training the network is straightforward: the emission probabilities are calculated by collecting all the examples of breaks in the training data. For all the possible unique POS sequences, C, counts are made from the data of how many times each occurs. These are converted to the probability $P(C|break)$ by dividing by the total number of breaks. $P(C|non-break)$ is calculated the same way. The ngrams are calculated by counting the number of times each unique sequence of length N occurs in the training data.

At run time, this network is efficiently searched using the Viterbi algorithm to find the most likely sequence of junctures given the input POS tags for a sentence.

3. PART OF SPEECH TAGGING

The POS tags for our training and test data are determined using a fully automatic POS tagger. Part of speech tagging has become a quite mature field and we simply follow the known technology. We use a standard HMM-based tagger (as in [4]) which estimates the probability of a part of speech tag sequence given a sequence of words. For training our POS tagger we use the WSJ corpus in the Penn Treebank [7], which consists of around one million words. Their basic tagset (after some simple reduction) consists of 37 tags. We treat each punctuation symbol as a word with the tag *punc*. Using a tri-gram model we achieved 94.03% accuracy on a held out test set. Given our experiments in tagset size described below, we also investigated the accuracy of POS taggers using a reduced tagset. We discovered that reducing the tagset, then building a model gives better results (96.18%) than using the full 37 tags. Even better results were achieved by using the full tagset to tag the data and then reducing to the smaller set (97.04%). Hence we use a tri-gram model built using the full 37 tagset and reduce it as required.

4. TRAINING DATA

The MARSEC database of spoken British English [9] is used for training and testing. It consists of stories recorded from BBC Radio 4 including extracts from talk shows, news and weather reports, and continuity announcements. The corpus is labelled with part of speech tags and two levels of break. For the following tests we split the database into a training set of 30 stories consisting of 31,707 words containing 6,346 breaks and a test set of 10 stories consisting of 7,662 words and 1,404 breaks.

Although this database has its own tagset, we did not use it in our tests. The MARSEC tagset is different from the WSJ Penn Treebank one and the mapping appears non-trivial. Also there is not enough data in the MARSEC database itself to train a POS tagger. The MARSEC data was therefore re-tagged using our tagger with the WSJ tagset.

5. PERFORMANCE CRITERIA

Unfortunately, it is not easily to judge the success of a phrase break assignment algorithm. As there are typically more non-breaks than breaks (in our data about 4:1), failure to predict a break can be judged better than over-predicting a break if a simple percentage overall correct score is used. Counting just the correct breaks is useful but only if some measure of over-prediction is included (if you massively over predict, the percentage breaks correct score will be high).

Another more serious problem is that there can be different but valid ways for a speaker to phrase an utterance. As the assigned results are compared against actual examples they may differ in acceptable ways, as well as unacceptable ways, and there is no easy way to find out the type of error. Ostendorf and Veilleux [8] deal with this problem by having five different speakers read each test utterance. Assignment is considered correct if the whole utterance matches any of the five samples. Unfortunately, we did not have the resources to re-record our database examples and hence could only do a direct match to one example. However, the results in [8] indicate that the best results when comparing with a single speaker are likely to still be the best when compared with multiple examples, even though some assignments are judged incorrect by the measurement.

Here we present results with three figures, percentage breaks correct, overall (breaks and non-breaks) correct and percentage non-breaks incorrect (a measure of break over prediction).

6. EXPERIMENTS AND RESULTS

This section reports results from some simple algorithms and goes on to show experiments on our model, investigating how variations in POS tagset size, ngram size and smoothing affect performance.

6.1. Some simple algorithms

As with all models, there are trade-offs between complexity, both in time and space, and ease of implementation. The table below gives results from some simple algorithms tested on our data. The first inserts a phrase break deterministically after all punctuation while the second inserts a phrase break after all content words that are succeeded by a function word (e.g. as suggested by [10]).

Model	B correct	Overall	NB incorrect
punc	54.27%	90.76%	0.85%
c/f	84.40%	71.29%	31.73%

We can see that the punctuation-model conservatively assigns breaks at positions that are almost always correct, but misses many others. The content/function model gets many more correct but at the cost of massive over insertion.

Within our basic model there are a number variables to investigate, including POS tagset size, size of POS window for POS sequence model, and size of n-gram for phrase break model.

6.2. Tagset size

The full tagset of 37 is too large to esimate all models reliably, so we investigated using smaller tagsets. To find the optimal tagset size we tested a progression of tagset sizes starting from 37 down to 2. We used a greedy algorithm finding the best tag combination at each stage. We found that a tagset size of 23 (formed by collapsing the sub-categories of the four major categories in the original) gave the best results. The following results show the results comparing the original, the 23 size set and sets of size 3 and 2. ts_s2 only distinguishes words from punctuation, and ts_3 distinguishes content words, function words and punctuation. A ngram of length 6 was used throughout (see below).

Model	B correct	Overall	NB incorrect
ts_{37}	74.22%	90.26%	6.04
ts_{23}	79.48%	91.60%	5.57
ts_3	68.30%	89.36%	5.91
ts_2	58.55%	88.01%	5.38

In general our experiments showed that the optimal tagset size is between 15 and 25. Our standard tagset of 23 could be reduced slightly with a small improvement by combining rare tags (e.g. **fw**, foreign word) into the major categories.

6.3. Ngram model size

Next we investigated various ngram models, given the likelihood of all sequences of break and non-break up to length N. The following table shows the effect on performance of varying N.

n-gram	B correct	Overall	NB incorrect
1	68.38%	91.46%	3.23
2	77.42%	91.18%	5.65
3	78.13%	90.97%	6.08
4	79.63%	91.34%	5.96
5	79.49%	91.34%	5.93
6	79.27%	91.60%	5.57
7	78.49%	91.53%	5.47
8	78.42%	91.44%	5.57

It appears that from these experiments, the value of N is not critical so long as it is above 2, i.e. so long a some context is used. We used a variety of standard ngram smoothing techniques but none had any significant effect on performance.

6.4. Smoothing the models

Because our training does not contain all possible combinations, our models are especially poor when there are only zero or one occurrences of a POS sequence. To combat this we experimented with smoothing the POS sequence frequencies. We used two forms of smoothing. First Good-Turing smoothing [3] to ensure there were no contexts with zero occurrences, and then a form of back-off smoothing [6], i.e. using progressively smaller contexts to estimate the frequency of a context when only a few actually existed in the training set.

The results of smoothing slightly improved our results. Again given tagset 23, and a 6-gram, and POS sequence model of two before and one following gives results of:

Model	B correct	Overall	NB incorrect
unsmoothed	77.07%	91.49%	5.27
smoothed	79.27%	91.60%	5.57

7. COMPARISONS WITH OTHERS

As a final test we applied our best model to the test set given in [8] and got better results than they got with their own model. Our model gives 72.72% Bs correct, 4.27% NB incorrect, while their best gives 70% Bs correct, with 5% NB incorrect. To be fair we cannot claim that this improvement is due solely to our model as our model was trained on a significantly larger training set, but it does show that our model is not specific to our particular test set.

8. DISCUSSION

We feel our model is adequate as a phrase break assignment algorithm as it is simple, performs well and does not have unrealistic input requirements. Both the model and the POS tagger have been fully implemented and are used as the standard phrase break assignment model in the distributed version of the Festival Speech Synthesis System [2].

Although we feel we have thoroughly tested this model and investigated the varying of its basic parameters to find the (near) optimum values, we feel that there is ultimate limit to how well such a model can perform with only part of speech information. It has been shown that some decisions about phrase break assignment can only reasonably be made with additional syntactic information. For example, identification of the use of prepositions as verb particles is an identifiable mistake in our (and other's) algorithms. Also the verb balancing rule as described in [1] is a valid phenomena which our current algorithm cannot capture without the addition of some form of syntactic information. There is of course a trade-off between time to run a reliable parser and the possible improvement in results, but we still feel such investigation is worthwhile, if only to find out the benefits it would give in improving on our current results.

9. ACKNOWLEDGEMENTS

We gratefully acknowledge the support of the UK Engineering and Physical Science Research Council (EPSRC grant GR/K54229).

REFERENCES

[1] J. Bachenko and E. Fitzpatrick. A computational grammar of discourse-neutral prosodic phrasing in English. *Computational Linguistics*, 16(3):155–170, 1990.

[2] A. W. Black and P. Taylor. The Festival Speech Synthesis System: system documentation. Technical Report HCRC/TR-83, Human Communciation Research Centre, University of Edinburgh, Scotland, UK, January 1997. Avaliable at http://www.cstr.ed.ac.uk/projects/festival.html.

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

[4] S. DeRose. Gramatical category disambiguation by statistical optimization. *Computational Linguistics*, 14:31–39, 1988.

[5] J. Hirschberg and P. Prieto. Training intonation phrase rules automatically for English and Spanish text-to-speech. In *Proc. ESCA Workshop on Speech Synthesis*, pages 159–163, Mohonk, NY., 1994.

[6] S. Katz. Estimation probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35:400–401, 1987.

[7] M. Marcus, B. Santorini, and M. Marcinkiewicz. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19:313–330, 1993.

[8] M. Ostendorf and N. Veilleux. A hierarchical stochastic model for automatic prediction of prosodic boundary location. *Computational Linguistics*, 20(1):27–55, 1994.

[9] P. Roach, G. Knowles, T. Varadi, and S. Arnfield. Marsec: A machine-readable spoken english corpus. *Journal of the International Phonetic Association*, 23(1):47–53, 1993.

[10] K. Silverman. *The structure and processing of fundamental frequency contours*. PhD thesis, University of Cambridge, 1987.