Assigning Phrase Breaks from Part-of-Speech Sequences

Paul Taylor and Alan W Black

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

Abstract

This paper presents an algorithm for automatically assigning phrase breaks to unrestricted text for use in a text-to-speech synthesizer. Text is first converted into a sequence of part-of-speech tags. Next a Markov model is used to give the most likely sequence of phrase breaks for the input part-of-speech tags. In the Markov model, states represent types of phrase break and the transitions between states represent the likelihoods of sequences of phrase types occurring. The paper reports a variety of experiments investigating part-of-speech tagsets, Markov model structure and smoothing. The best setup correctly identifies 79% of breaks in the test corpus.

1 Introduction

An important problem in text-to-speech (TTS) synthesis is to find suitable places in the text for the placement of prosodic phrase breaks. In a typical TTS system, phrase breaks are used by a number of modules, including:

1. **Fundamental frequency contour generation:** Major phrase boundaries delimit intonation phrases and are the only position where boundary tones can occur. Correct phrasing facilitates suitable accentuation as the last accent in a phrase is treated as the nuclear accent.

2. **Duration:** The duration module lengthens segments which occur immediately prior to a phrase boundary.

3. **Pause Insertion:** Pauses can be inserted in the middle of a sentence. The main deciding factor in this is whether a major phrase break has just occurred.

The performance of these modules is heavily dependent on the ability of the phrase break component to place its boundaries in appropriate places.

Past reviews (Ostendorf and Veilleux, 1994), (Wang and Hirschberg, 1992) describe two approaches. The first makes use of the fact that prosodic structure and syntactic structure are related, and uses some sort of syntactic information to predict prosodic boundaries (often in the form of heuristic rules). This approach has several disadvantages which make its use unattractive for real TTS systems. Rule-driven parsers are notoriously unreliable and can provide poor input to the syntax-to-prosody module. In addition, a rule-driven syntax-to-prosody module suffers from the same disadvantages as all rule driven systems: they are often difficult to write, modify, maintain and adapt to new domains and languages.

In light of these shortcomings, some researchers have tried a second approach whereby prosodic structure is derived from robust, if crude, features of the input text. The simplest of these is based on the content word/function word rule (e.g. Silverman (1987)) whereby a phrase break is placed before every function word that follows a content word. Despite its simplicity, such an approach can sometimes produce reasonable results. A number of other proposals based on either rule driven or statistical superficial analysis of the text have also been proposed (Wang and Hirschberg, 1992), (Hirschberg and Preito, 1994), (Ostendorf and Veilleux, 1994), (Veilleux et al., 1990).

This paper describes an algorithm of the second type which assigns phrase breaks using global optimisation techniques on sequences of part-of-speech (POS) tags.

2 Overview of the Algorithm

We define our problem as follows: the input text consists of a sequence of words and between each pair of words is a *word juncture*. There is a set of T word juncture *types* and it is the task of a phrase-break algorithm to assign the most appropriate type to each juncture. Most experiments in this paper use two types of juncture, *break* and *non-break*. In principle any number of types is possible, for example splitting the break type into minor and major gives 3 types, or following the ToBI scheme gives 5 types (Silverman et al., 1992). Here we give an overview of the standard version of our algorithm which assigns breaks and non-breaks to arbitrary input text.

The algorithm is trained and tested on a database of spoken English in which the original text has been hand annotated with a break label between words which are perceived as being phrase-breaks. The text is tagged with a hidden Markov model (HMM) part-of-speech (POS) tagger which replaces each word by its POS tag. The tags, c are chosen from a tagset $V v_1, ..., v_k$ of size K. The juncture between every pair of words is then marked as one of the word juncture types: in the standard case this is either break or non-break.

The algorithm uses a Markov model in which states represent juncture types, and the transitions between states represent the likelihood of particular sequences of breaks and non-breaks occurring. Each state has an observation probability distribution giving how likely that state is to have produced a sequence of part of speech tags. The state observation probabilities are called the *POS sequence model* and the set of transition probabilities is called the *phrase break model*. Bayes equation is used to relate the two, and the most likely juncture sequence for a given input can be found by searching through the model and picking the most likely path.

Training the model involves estimating the POS sequence model (the observation probabilities) and the phrase break model (the transition probabilities).

2.1 POS Sequence Model

The POS sequence model is trained by searching the training data for each juncture type and counting the number of distinct sequences of POS tags before and after the juncture. Generally, the POS sequence is a window of L tags around a juncture j_i, M tags preceding j_i and $L - M$ tags following j_i. In our standard system there are 2 tags before and 1 after the juncture ($L = 3, M = 2$). These counts are converted into probabilities by dividing each count by the total number of occurrences of that juncture type in the data. This gives an estimate of the probability of a POS sequence given a juncture type.

Let us denote a POS sequence $c_{i-M}, .., c_i, .., c_{i+L-M}$ as C and the number of times this occurs in the training set as $count(C)$. The number of times a juncture type occurs is given by $count(j)$. Thus an estimation of the probability is given by:

$$\tilde{P}(C|j) = \frac{count(C|j)}{count(j)} \tag{1}$$

which in expanded form is:

$$\tilde{P}(c_{k-M}, ..., c_{i-1}, c_i, c_{i+1}, ..., c_{i+L-M}|j_i) = \frac{count(c_{i-M}, ..., c_{i-1}, c_i, c_{i+1}, ..., c_{i+L-M}|j_i)}{count(j_i)} \tag{2}$$

2.2 The Phrase Break Model

The phrase break model is trained by examining the database again, but this time ignoring the POS information and only examining junctures. A n-gram of order N is constructed which represents the probability of different sequences of junctures. Using J^N_{i-1} to represent the the previous sequence of N junctures, we have:

$$P(j_i|J^N_{i-1}) = P(j_i|j_{i-1}, j_{i-2}, j_{i-3}, ..., j_{i-N+1}) \tag{3}$$

2.3 Combining the Models

A network of T^{N-1} nodes and T^N arcs is constructed ($N = 1$ is a special case and has the same topology as $N = 2$ - see figure 1) . Each node represents a juncture type, and when $N > 2$ the nodes represent a juncture in the context of previous junctures. The POS sequence probabilities do not take account of context, and so for a given juncture type are the same no matter where the node occurs in the network. For example, if $N = 3$, we will have 2 break nodes, one for when the previous juncture was a break and one for when the previous juncture was a non-break. These nodes have the same observation probabilities. Figure 1 shows networks for $N = 1$, $N = 2$ and $N = 3$.
Under this formulation we have the likelihood $P(C_i|j_i)$ (the POS sequence model) representing the relationship between tags and juncture types, and $P(j_i|j_{i-1}, ..., j_{i-N+1})$ (the n-gram phrase break model) which represents the *a priori* probability of a sequence of juncture types occurring. This is used to give a basic regularity to the phrase break placement, enforcing the notion that phrase breaks are not simply a consequence of local word information.

The probability we are interested in is $P(j_i)$ given the previous sequence of junctures and the POS sequence at that point. This probability can be rewritten as follows:

$$P(j_i|C_i, J_{i-1}^N) = P((j_i|J_{i-1}^N)|C_i) \tag{4}$$

and using Bayes equation

$$P(j_i|J_{i-1}^N|C_i) \propto P(j_i|J_{i-1}^N).P(C_i|(j_i|J_{i-1}^N)) \tag{5}$$

We make the assumption that the probabilities of all states of a particular juncture type are equal (e.g. $P(C_i|break, non-break) = P(C_i|break, break)$), so

$$P(C_i|(j_i|J_{i-1}^N)) = P(C_i|j_i) \tag{6}$$

and from equation 5, the probability of a juncture type given the preceding types and POS sequence becomes

$$P(j_i) \propto P(j_i|J_{i-1}^N).P(C_i|j_i) \tag{7}$$

3 Data and Evaluation

We have used the Spoken English Corpus (Arnfield, 1994) for all the experiments described here. This is a database of spoken British English recorded from BBC Radio 4. It is mostly speech read from scripts in the form of news stories, weather reports etc. This corpus has been labelled with POS information and has been hand labelled with major and minor prosodic phrase breaks. The section of the the corpus that we use has 39369 words and contains 7750 breaks (both major and minor). The corpus comprises 40 separate stories, of which 30 were used for training and 10 for testing. This resulted in a training set consisting of 31,707 words and 6346 breaks, and a test set of 7662 words and 1404 breaks.

3.1 Performance Criteria

Performance is assessed with reference to N, the total number of junctures in the test set, and B, the total number of junctures which are breaks. A *deletion* error (D) occurs when a break is marked in the reference sentence but not in the test sentence. An *insertion* error (I) occurs when a break is marked in the test sentence but is not in the reference. A *substitution* error (S) occurs when a break occurs in

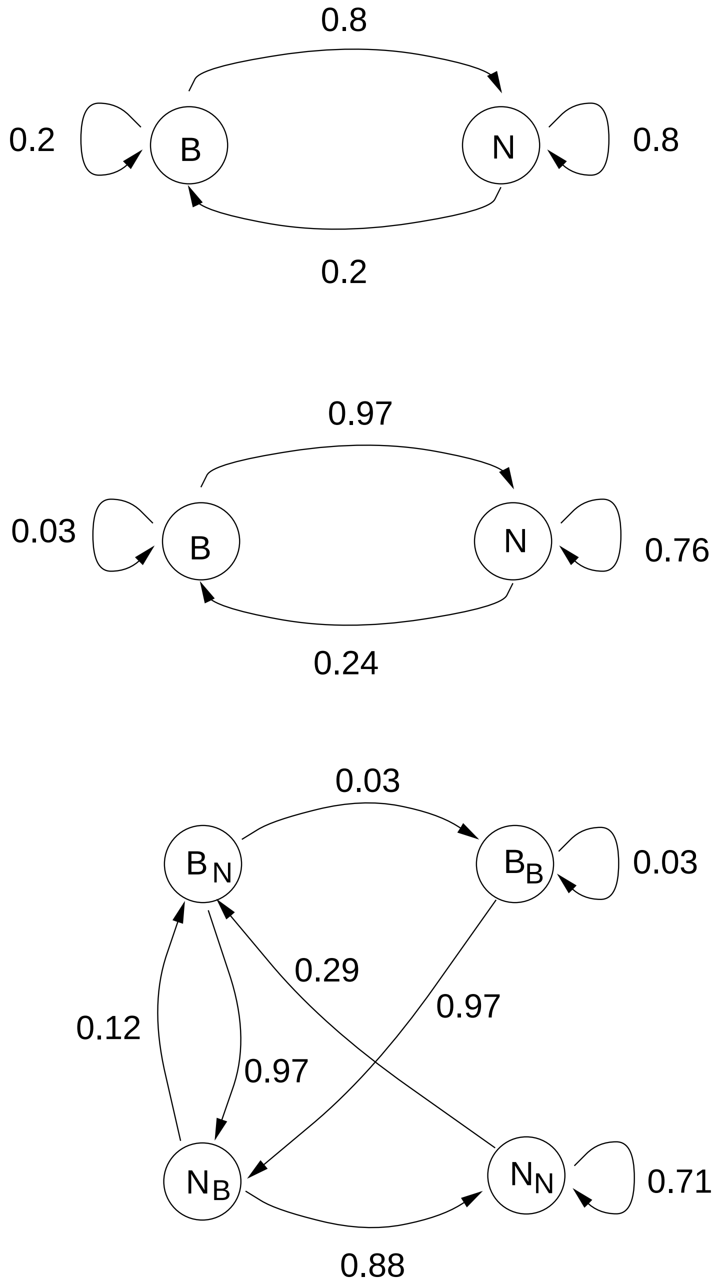

Figure 1: Models for N=1, N=2 and N=3, showing actual transition probabilities calculated from the training data. The states marked B are for breaks and those marked N are for non-breaks. Subscripts in state names indicate the juncture type of the previous state. In the N=1 case the transition probabilities are just the context independent probabilities of the juncture types occurring, i.e the transition probabilities to a state don't depend on the previous state. In the N=2 case, the transition probabilities take into account the previous juncture. Thus in this model it is very unlikely that a break will follow a break (0.03), while in the N=1 case this would still have a relatively high probability (0.2). Looking at the probabilities of sequences of non-breaks, we see differences in the probability of a non-break following two previous non-breaks. As N increases, we see that the probability of long sequences of non-breaks decreases ($P(N_i|P_{i-1}, P_{i-2})$ = 0.8 for N=1, 0.76 for N=2 and 0.71 as N=3). Thus a higher order ngram helps prevent unrealistically long sequences of just non-breaks or just breaks. The POS sequence model probabilities (not shown here) are associated with each state. All states of the same basic type are the same and so the probability distributions for state B_B (a break following a break) are the same as for state B_N (a break following a non-break).

the right place but is of the wrong type. This type of error is only relevant when more than one type of break is being recognised. There is no single best way to measure the performance of a phrase break assignment algorithm, and a variety of approaches have been proposed in the literature. We explain these performance measures below.

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

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

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

$$\text{False insertions w.r.t junctures} = \frac{I}{N} \times 100\%$$

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

The difference between breaks-correct and junctures-correct lies in whether non-breaks are included in the calculation. The junctures-correct score gives credit when both the test and reference sentences have a non-break at the same juncture, while the breaks correct score only looks at junctures with breaks. In our data, the number of non-breaks outnumbers the number of breaks by a ratio of about 4:1, and hence an algorithm which marks everything as non-break will score about 80% junctures-correct, but 0% breaks-correct. (Wang and Hirschberg (1992) gives nearly identical figures for the relative number of non-breaks to breaks.) Because the breaks-correct score is not dependent on the relative distributions of breaks and non-breaks, we regard this as a better indicator of algorithm performance. The assessment of insertions is more troublesome: one can either calculate them as a percentage of the number of breaks in the test set or of the number of junctues, as in Ostendorf and Veilleux (1994). For reasons of readability and succinctness we use only the breaks-correct, jucntures-correct and juncture-insertion scores in this paper.

3.2 Testing Methodology

While some algorithms in this field are portrayed as multi-purpose systems which could be used in speech synthesis, speech recognition or automatic database tagging (Ostendorf and Veilleux, 1994), (Wang and Hirschberg, 1992), our system is designed and optimised solely for use in a unconstrained text-to-speech system. To that end we test the system on the input format it can expect in real use: a continuous stream of ascii characters with no division into words, sentences or other units. The division of the text into such units is performed automatically within the initial text processing modules of the text-to-speech system. The POS tagging is performed automatically, and no acoustic features are used. These testing conditions are as close as one can get to how the system would operate in real use: it is important to bear this in mind when comparing these results to other systems.

Experiment	Breaks-correct	Junctures-correct	Juncture-insertions
Det. P	54.274	90.758	0.852
Det. PCF	84.402	71.288	31.728
Prob P 1-gram	54.986	91.099	0.799
Prob P 6-gram	58.547	88.006	5.385
Prob PCF 1-gram	54.886	91.099	0.799
Prob PCF 6-gram	68.305	89.393	5.849

Table 1: Comparison of basic models.

4 Part-of-Speech Tagging

We use a standard HMM-based tagging framework as is commonly found in a number of systems (e.g. (DeRose, 1988)). This model consists of two parts: an n-gram model for part of speech sequences and a likelihood distribution model of part of speech tags for words. These parts are combined using Bayes Theorem and the Viterbi algorithm is used to find the most probable part of speech sequence given a set of words.

Although the Spoken English Corpus is marked with POS tags, this corpus has too few words to train a HMM POS tagger. Instead we used the Penn Treebank (Marcus et al., 1993) which consists of around 1.2 million words from the the Wall Street Journal (WSJ). Apart from size, we do not think that the two corpora are significantly different with respect to POS behaviour. The words in the WSJ data are tagged automatically with subsequent hand correction. Punctuation is reduced to a single tag giving us a base tagset of $K = 37$. A generic, unknown word POS distribution is made from the POS distributions of a set of less frequent words and there is a special distribution for words containing just digits. The tagger correctly tagged 94.4% of the words of an independent test set of 113,000 words.

The parameters in our POS sequence model are calculated from POS tag occurrences and it is clear that while the full tagset may potentially be the most discriminative, it also leads to sparse data problems. A series of experiments found that a tagset of size 23 was the overall best. The reduction in size can be carried out in two ways, either by mapping the output of the tagger onto a smaller tagset, or by training the tagger on the smaller set. We found that when we post-mapped the tagset the performance was 97.0%, while training and testing on the reduced set gave a worse figure of 96.2%. Hence we always use the full tagset for POS tagging purposes and reduce the size of the set afterwards.

5 Part-of-Speech Sequence Models

This section describes experiments which investigate the effect of varying the parameters relating to equation 2, namely the size of the POS sequence window, L; the number of tags before the juncture, M and the size, K and composition, V of the tageset. In these experiments, for compactness we use two phrase break models representing n-grams of order 1 and 6. A 1-gram represents the simplest case, and as explained later the 6-gram is the best performing phrase-break model.

5.1 Punctuation, Content Words and Function Words

First, we report results from two deterministic algorithms, one that uses punctuation only and another which uses this plus the content/function word distinction. No training is necessary in these cases. Row 1 (*Det. P*) in Table 1 shows the results from the deterministic punctuation-only algorithm which simply places a phrase break after every punctuation mark in the text. As one might expect, this shows

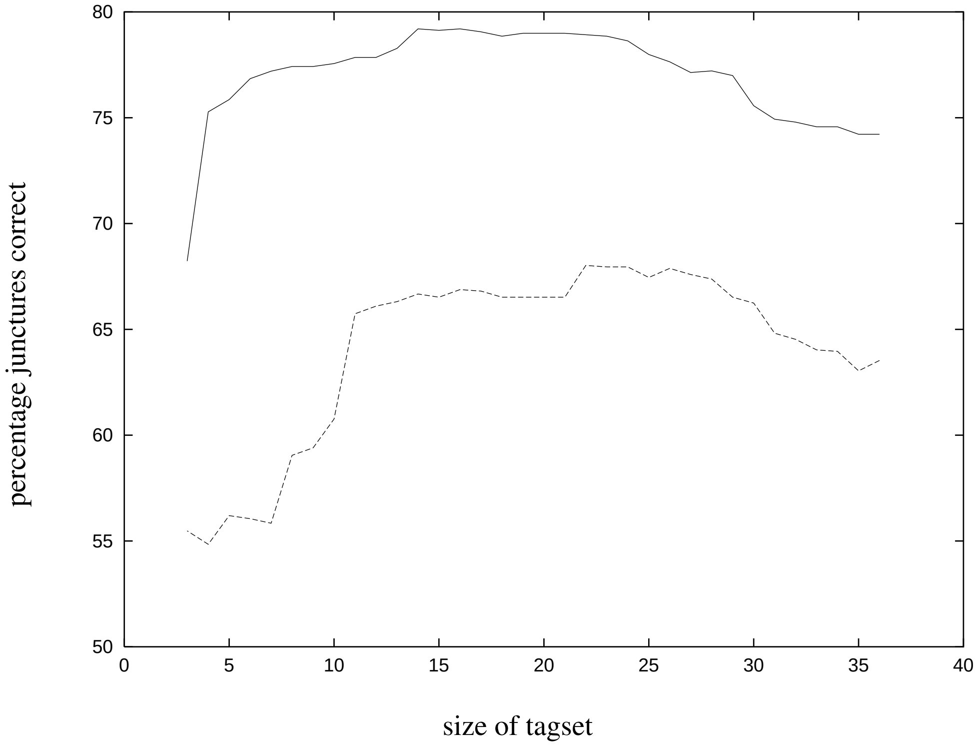

Figure 2: Plot of junctures-correct for a 6-gram (upper line) and 1-gram (lower line) against K, the size of the tagset.

that punctuation is a very reliable indicator of phrase-break presence. The insertion rate is very low at 0.852% showing that punctuation nearly always implies a phrase break. This algorithm correctly finds about half the breaks and inserts very few false ones.

Row 2 (*Det. PCF*) shows the results for when breaks are placed after every punctuation mark and before every function word following a content word. The number of breaks correct increases considerably, but only because of a massive over prediction of break placement. The junctures-correct score and the insertion score are the worst for any of the experiments described here.

Rows 3 (*Prob P 1-gram*) and 4 (*Prob P 6-gram*) show the results from our algorithm using a POS sequence model with one tag following and one preceding the juncture ($L = 2, M = 1, V = \{non - punctuation - word, punctuation\}$). Row 3 shows that the probabilistic punctuation-only algorithm produces very similar results to its deterministic counterpart when a 1-gram phrase break model is used, but that break prediction accuracy increases when a higher order n-gram phrase break model ($N = 6$) is used (row 4). Row 5 (*Prob. PCF 1-gram*) shows that dividing the non-punctuation class into content words and function words has no significant effect on performance ($L = 2, M = 1, V = \{function, content, punctuation\}$) compared to row 3. If we look at the relevant POS sequence frequency counts in the training data, we find a that there are 1751 non-break instances and 1002 break instances for the $content - function$ sequence. Thus a non-break is always more probable, and given a 1-gram phrase-break model, breaks will never be inserted. However, if we use a higher order phrase-break model, the combined probability will get high enough to make breaks more probable if the distance from the last break is more than a few words. The figures in row 6 (*Prob PCF 6-gram*) show this effect.

In summary, the deterministic punctuation algorithm is “safe” but massively under-predicts, while the deterministic punctuation/content/function word algorithm massively over-predicts. The probabilistic counterparts only perform acceptably if a high order phrase break model is used.

5.2 Larger POS Tagsets

The content/function word rule was motivated by sentences such as "The tall man | was walking | down the street" (f c c f c c f c) where a break occurs after the first noun phrase. While many verb constructions in English contain auxiliaries (function words), constructions such as the simple past do not, and in sentences such as "The tall man walked down the street" (f c c c c f c) this rule cannot place a break in the same place. Such cases can only be tackled by using a larger tagset.

To ensure good performance, the POS sequence models must be trained on a sufficient number of examples. There is an inevitable trade-off between robustness, achieved by having a large number of each word sequence in the training data, and discriminative ability, achieved by allowing a large tagset that can model individual effects. While a tagset of $K = 3, V = \{function, content, punctuation\}$) is too small, very large tagsets prevent accurate estimation as the number of possible types of word sequence is K^L (i.e. it grows exponentially on the POS sequence window size). Finding the best possible tagset (one which genuinely optimises the performance) is a difficult problem and we have not attempted a full solution. Rather we approached the problem empirically and report results from a series of experiments using various tagsets.

All the experiments here were performed on data tagged with the HMM tagger. It might have been interesting to perform a comparison between the system trained on these tags and the original ones in the corpus which were tagged by hand. Unfortunately the MARSEC corpus has a different tagset to that of the POS tagger training data and so there was no easy way to perform a rigorous comparison without introducing artifacts from the mapping of one tagset to the other. However, informal experiments seem to show that the automatic algorithm performs as well (if not better) on automatically tagged data as on hand tagged data.

Figure 2 shows results of an experiment measuring how performance varies as a function of tagset size. New tagsets were formed by using a greedy algorithm to collapse categories in the original tagset. For each stage in the process, we found which combination of two current clusters gave the best performance and chose that as the tagset for the next stage. We cannot claim that the scores for each tagset are the best possible for a tagset of that size as there are many other possible ways to make tagset groups from the original set. However, from evidence from this and other experiments conducted on tagset reduction, where clustering was more linguistically directed, we believe this experiment shows the general trend of performance against tagset size. Again, no single measure can be taken as the sole indicator of performance, but the best results seem to be when the tagset size is somewhere between 15 and 25. From more detailed analysis involving all the types of phrase models, we found that a tagset of size 23 gave the best overall performance. This tagset is linguistically the most easy to describe: the distinctions between subtypes of the four major categories (nouns, verbs, adjectives and adverbs) were ignored, combining the four basic noun tags into a single category and likewise for the 6 verbial, 3 adjectival, and 4 adverbial tags. All punctuation was grouped as one tag. This tagset of 23 was used in most of the subsequent experiments.

5.3 Smoothing POS Sequence Models

The standard model uses 23 POS tags in a POS sequence model of length 3. This gives 12167 different possible observations. With a training set of 31,707 words it is clear that there will be a large number of POS sequences which never occur or occur only once. In the basic model, sequences with zero counts are assigned a small fixed floor probability. These cases are not particularly important as the chances of breaks and non-breaks being inserted is now governed by the phrase model. More worrying are single

Experiment	Phrase Break Model	Breaks-Correct	Junctures-Correct	Juncture-Insertions
unsmoothed	1-gram	69.940	91.56	3.600
smoothed	1-gram	68.376	91.46	3.227
unsmoothed	6-gram	77.070	91.49	5.270
smoothed	6-gram	79.274	91.60	5.569

Table 2: Effect of smoothing on accuracy

Experiment	Phrase break model	Breaks-correct	Junctures-correct	Juncture-insertions
$L = 2, M = 1$	1-gram	61.040	91.424	1.589
$L = 3, M = 2$	1-gram	68.376	91.464	3.227
$L = 4, M = 3$	1-gram	61.895	90.145	3.358
$L = 4, M = 2$	1-gram	62.037	90.478	2.981
$L = 2, M = 1$	6-gram	78.134	91.104	5.913
$L = 3, M = 2$	6-gram	79.274	91.597	5.569
$L = 4, M = 3$	6-gram	73.148	90.025	6.093
$L = 4, M = 2$	6-gram	71.937	89.786	6.110

Table 3: Results varying the number of words in the POS sequence window and how many are before and after the juncture

occurrences. If a POS sequence is observed only once and with a break at the juncture, this will be assigned the same probability as when a large number of breaks and zero non-breaks are observed for a POS sequence. Clearly the second case is a better indicator that the POS sequence in question really does carry a high likelihood of a break.

To counter this problem we employ a smoothing technique which adjusts the frequency counts of rare and non-occurring POS sequences. First Good-Turing (explained in Church and Gale (1991)) smoothing is used to adjust the frequency counts of all occurrences for the break and non-break model. This effectively gives zero counts a small value and reduces the counts of rare cases. Next a form of backing-off is applied whereby a juncture likelihood $P(c_{k-1}, c_k, c_{k+1}|j_k)$ is discarded if its adjusted frequency count falls below a threshold, and the estimate $P(c_k, c_{k+1}|j_k)$ is used instead. A threshold of 3 usually gave the best results. Table 2 gives the results comparing the V_{23} tagset under smoothing and no smoothing. The smoothed POS sequence models with the 6-gram phrase break model are significantly better than the unsmoothed equivalents with both word and break accuracy increasing at only a slight word insertion decrease.

The table shows that smoothing significantly increases performance when used with a high order n-gram phrase break model.

5.4 Varying POS sequence length

Equation 2 shows the general POS sequence formula which is expressed in terms of a window of L tags with M of these tags before the juncture and $L - M$ tags after. We can expect longer sequences to be potentially more discriminative, but more prone to sparse data problems. Table 3 shows results from experiments which varied L and M. These were performed on the 23 POS tagset, using smoothing and a 1-gram and 6-gram phrase break model. For both phrase model conditions the $L = 3, M = 2$ condition outperforms the others.

Phrase break Model	Minor correct	Major correct	Junctures-correct	Junctures-insertions
1-gram	60.125	56.415	90.128	2.035
6-gram	68.703	59.502	90.211	3.923

Table 4: Minor and major phrase breaks

5.5 Minor and Major

Most of our work has centred around a model comprising two types of juncture, namely “break” and “non-break”, but the algorithm can easily be applied to an arbitrary number of juncture types. The MARSEC corpus is in fact labelled with two types of break, “major” and “minor” and so it was straightforward to build a system that predicted these types. The method is as before, but now three POS sequence models are required and the phrase break model has a vocabulary of 3 (implying higher perplexities and more sparse data problems). Note that when more than two types of break are used, substitution errors need to be counted for when major breaks are placed where minor breaks occur and vice versa. Table 4 reports the results.

We have not yet performed as comprehensive an analysis on multiple break types as with the single type, but have noted that the best tagsets for the each case are different. While the V_{23} tagset was the best for the single break case, larger tagsets gave better results for the multiple case, despite the more severe sparse data problems that occur. This indicates that finer distinctions are needed for discrimination of major and minor types.

6 Using Distance Information: the Phrase Break Model

It has been previously noted (Gee and Grosjean, 1983), (Bachenko and Fitzpatrick, 1990), (Wang and Hirschberg, 1992) (Sanders and Taylor, 1995) that phrases occur at somewhat regular intervals and that the likelihood of a break occurring increases with the distance since the last break. The distribution of phrase lengths in words is shown in figure 6. It is clear that the majority of phrases are between 3 and 6 words long.

Previous work has attempted to use this information (Wang and Hirschberg, 1992) (Sanders and Taylor, 1995) but has been hampered by the problem that in a real situation we cannot be certain where the previous break has occurred, and hence it is impossible to reliably compute the distance since the last break. One could simply work in a left to right fashion, placing breaks where they seem most likely and then using that as a definite point to measure the distance to the next hypothesised break. Although simple, this approach is not recommended as a single error cannot be recovered from and may cause errors in all the subsequent decisions. We adopt a different solution, made possible by the use of probabilities throughout the system. Although we can’t say when the previous break to juncture j_i occurred, we can estimate the *probability* that a break occurred at any of the previous junctures $j_{i-1}, j_{i-2}, j_{i-3}$ and thus try to examine all the possibilities.

Instead of explicitly modelling the distance from the last break, we use a n-gram model which gives the probability of all sequences of juncture types in a window of size N. (Note this window is completely separate from the window used in the POS model). N-Grams are the most popular choice of language model in speech recognition systems. The number of possible n-grams is given by T^N, where T is the size of the vocabulary. Because of this, speech recognition language models are usually bigrams or trigrams. Here T is the number of juncture types and in our standard system we have a vocabulary of $T = 2$ (break and non-break) and so it is possible to model n-grams as high as 6 and 7 robustly.

The overall goal of the algorithm is to find the most likely sequence of juncture types for the input

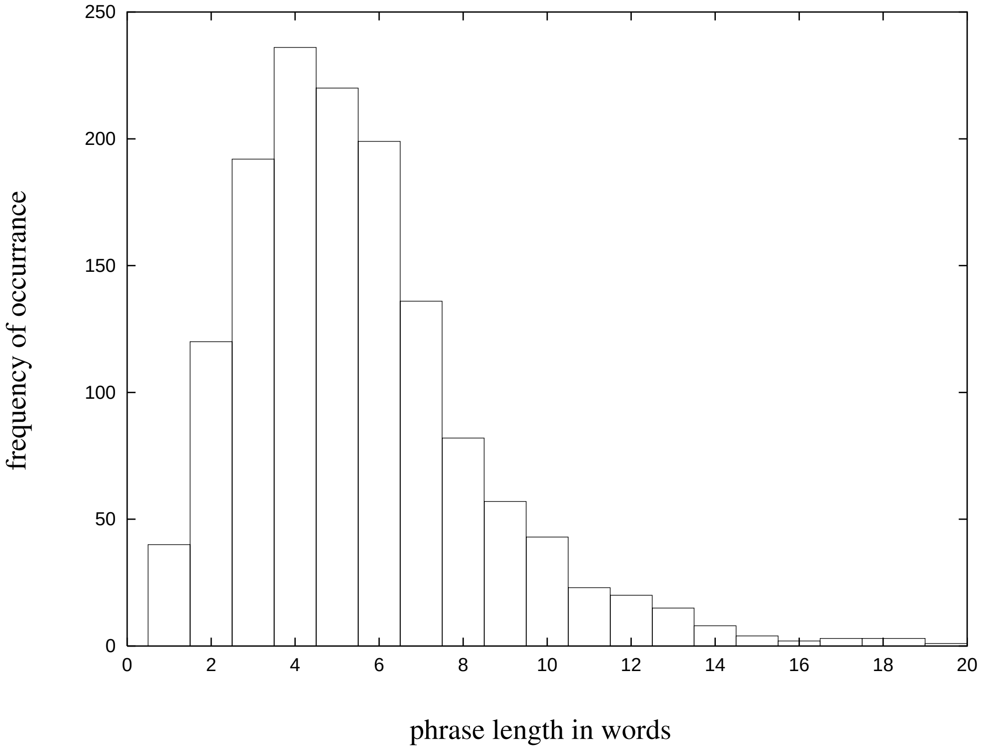

Figure 3: Plot of frequency of occurrence of phrases of different lengths in words

POS sequence. The probability of a sequence of juncture types ($P(J|C)$) is calculated using Bayes' Rule by multiplying the POS sequence likelihood by the phrase probability. In principle, the probability of all possible sequences of juncture types is calculated and the highest is chosen. If calculated directly this entails ST^S possibilities where S is the number of words in the sentence. Even for the case when the juncture types are simply break and non-break ($T = 2$), the number of paths can be very large for long sentences (e.g. over a million for a 20 word sentence) and if the T is larger the number of paths becomes intractable. Instead we use the Viterbi algorithm which finds the most likely path in about T^2N calculations. For a detailed explanation see Rabiner and Juang (1994).

6.1 Varying the Order of the N-gram

Perplexity is a measure of average branching factor and can be used to measure how well an n-gram predicts the next juncture type in the test set. If N is the order of the n-gram and Q is the number of junctures in the test set, the perplexity B can be calculated from the entropy H by:

$$B = 2^H \tag{8}$$

where

$$H = -\frac{1}{Q}\sum_{i=1}^{Q} logP(j_i|j_{i-1}, j_{i-2}, ..., ji - N + 1) \tag{9}$$

N-grams can be estimated from simple frequency counts of the data. Figure 6.1 shows how perplexity of a phrase break model of juncture types break and non-break varies as a function of n-gram order. The differences between the various phrase-break models are not large, but it can be seen that the 6-gram has

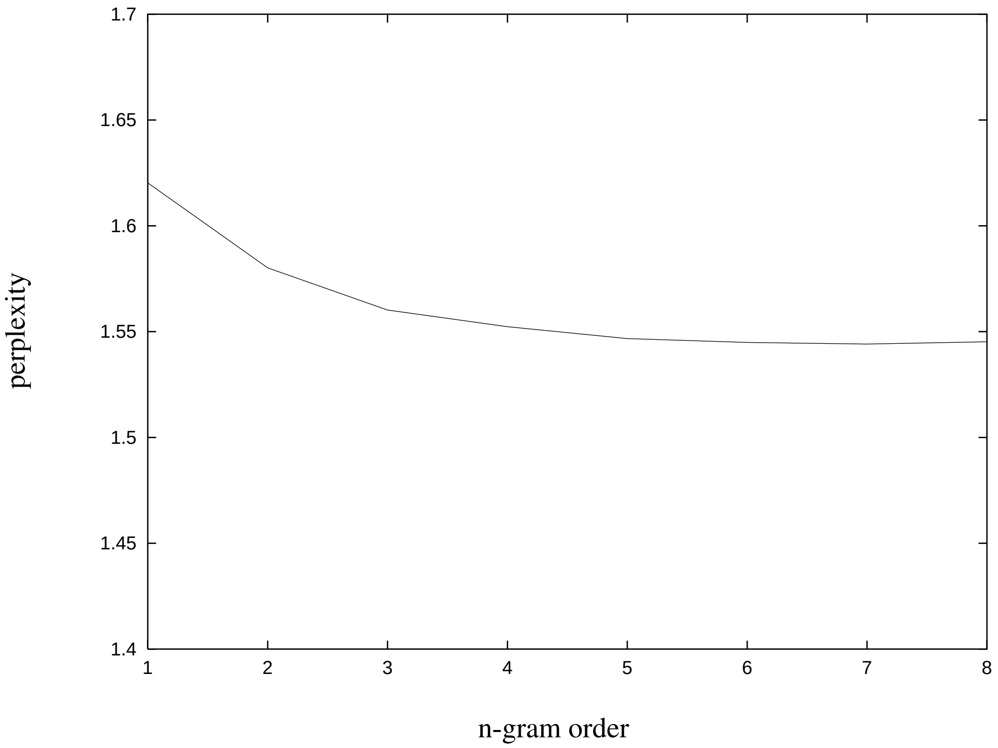

Figure 4: Plot of perplexity against order of n-gram

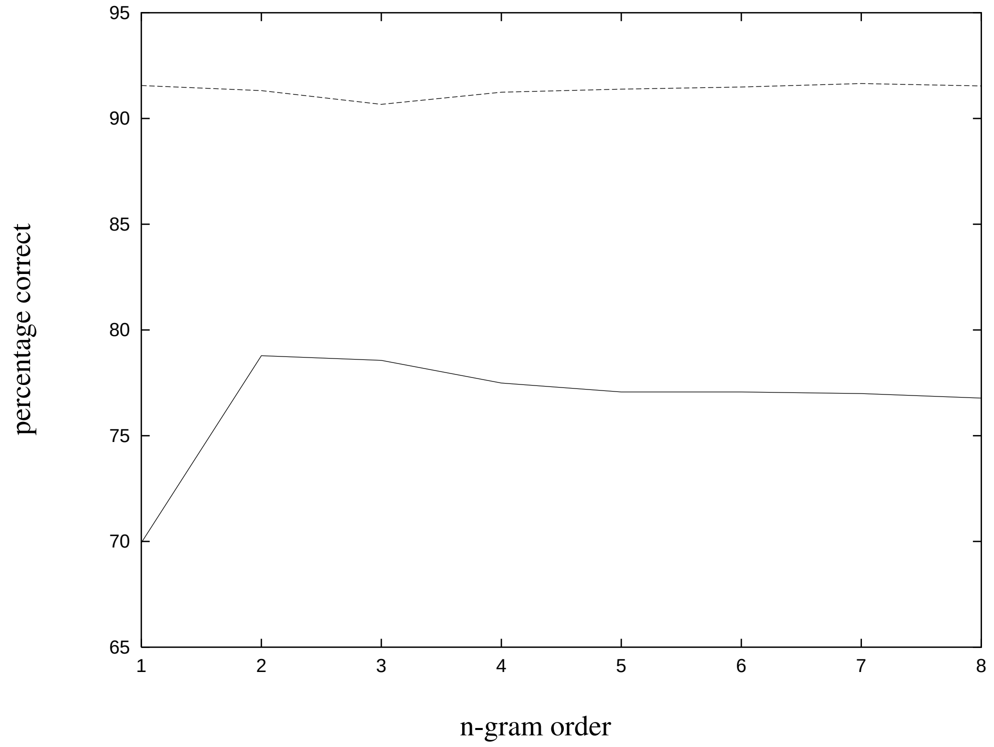

Figure 5: Plot of junctures-correct (upper line) and breaks-correct (lower line) against order of n-gram

N-gram order	Breaks-correct	Junctures-correct	Juncture-insertions
1	69.94	91.56	3.60
2	78.78	91.32	5.86
3	78.56	90.67	6.62
4	77.49	91.24	5.67
5	77.07	91.39	5.40
6	77.07	91.49	5.27
7	76.99	91.65	5.07
8	76.78	91.54	5.14

Table 5: Comparison of different order phrase break models

a perplexity of 1.54 compared to the unigram case of about 1.62. It is common in language modelling to use smoothing to account for rare and unseen cases. We recalculated our phrase-break model parameters using 3 types of smoothing: a fixed floor for unseen cases; Good-Turing smoothing, which alters the probabilities of rarely seen cases as well as unseen cases; and back-off smoothing whereby the values for rare n-grams are computed from the n-1-grams equivalents. None of the types of smoothing had a significant effect on the perplexity or indeed the overall results. In practice we use the simplest type of smoothing where unseen n-grams are given a frequency count of 1 during training.

Figure 6.1 and table 5 show how the order of the n-gram affects overall performance. The most noticeable effect is the big increase in performance between the unigram phrase-break model and the rest, which are fairly similar. This result is due to the phrase-break model assigning a very low probability (0.03) for a break given a preceding break compared with a probability of 0.2 for the same sequence from the unigram. The higher order n-grams perform slightly better than the bigram in terms of junctures-correct and juncture-insertions. In table 5 the 7-gram performs the best. In most of our experiments n-grams of order 6 and 7 had consistently better results than the other n-grams, but the difference was often slight. Figure 6.1 also shows that the perplexity of n-grams of about this order is slightly lower than for the others.

7 Discussion

It is worth mentioning a few practical aspects of our algorithm. As the training procedure is fully automatic, it is simple to retrain the system on a new prosodically marked database. Training is quick, requiring a single pass over the data for the POS sequence model and the phrase break model. At run-time, the Viterbi decoder requires only a few calculations per input sentence (several orders of magnitude less than the signal processing component of the TTS system, for instance). The framework also allows for some flexibility at run time. As is common with speech recognition Viterbi decoders, our system allows a grammar scaling factor which controls the relative importance of the POS sequence and phrase break models. With some experimentation, this can be effectively used to control the relative insertion and deletion ratios.

The algorithm has been implemented in the Festival speech synthesis system (Black and Taylor, 1997). The setup in Festival uses a POS sequence model based on 23 tags, with two tags before the juncture and one after. Smoothing for the POS sequence model is in the form of Good-Turing smoothing followed by back-off smoothing of threshold 3. An unsmoothed 6-gram phrase break model is used.

We have not yet attempted to use our system on any languages other than English. We expect that the algorithm will work with any language that has a phrase structure which can be related to superficial syntactic information such as POS tags.

7.1 Interpreting results

Throughout we have given at least three performance figures (junctures-correct, breaks-correct and juncture-insertions) for each experiment. While these give a reasonably representative indicator of true performance, and have been invaluable in system development, they must be treated with caution. First of all, it is difficult to judge the relative importance of insertion and deletion errors. Our best performing systems often increase the breaks-correct figure to the detriment of the juncture-insertion figure. From listening to the output, we believe that this is an acceptable tradeoff, and the best systems from a perceptual point of view are the ones which have the highest breaks-correct while maintaining a high junctures-correct score.

Although formal listening tests might give a better indication of the relative importance of different systems, they are extremely costly compared to the automatic scoring techniques used here. However, it should be noted that a certain amount of "perceptual tuning" can be carried out by scaling the phrase break and POS sequence model individually. Thus the relative importance of the two components in equation 7 can be controlled. The scaling affects the relative numbers of insertions and deletions, and hence the amount of scaling can be set according to which type of error is deemed most significant.

The scoring technique compares the system's placement of phrase breaks with that of a human labeller and so it is important to ask how consistently humans label phrase breaks. Unfortunately, no consistency figures are available for our data, but we think it is safe to assume that the labelling consistency is about the same as that measured by Pitrelli et al (1994), who obtained a breaks-correct figure of 92.5% for a similar task. This effectively defines the upper limit one can expect from an automatic system.

It is also important to note that not all errors are of equal importance. Partly this is due to the fact that speakers don't always place breaks in the same place: some junctures can take either breaks or non-breaks without sounding odd, while other junctures must always be of the same type. Ostendorf and Veilleux (1994) proposed a solution to this by having the text in the test set spoken by several different speakers. Sometimes all the speakers agreed, sometimes not. By comparing the output of their system with each instance of the test sentences, it was possible to assess if an error under the usual criterion was actually in a potentially acceptable place. Unfortunately such a comparison measure is not available to us, as we only have a single version of each sentence in the test set.

7.2 Comparison with other systems

Our results compare favourably with those of other systems. For example, our best score of breaks-correct is 79.27% compared to 70% in Ostendorf and Veilleux (1994). Wang and Hirschberg (1992) prefer a measure which is the average of the breaks correct and the non-breaks correct. Our best score using this measure is 86.6% which compares with their score of 81.7% on a text-to-speech task (as opposed to other tasks where they allow acoustic information also). When we ran our system on the test sentences used by Ostendorf and Veilleux, we achieved 72.72% breaks correct with 4.27% juncture insertions, compared with 70% and 5% reported in their paper. This shows that our system has some ability to transfer to completely unseen data in a different domain. Although our system is the best on the common test data, the improvement cannot solely be put down to differences in the techniques themselves. We trained our system on substantially more data and this may have been a large factor in the improvement. Aside from actual performance, we believe our system is somewhat simpler than the two others mentioned here and this may make it more attractive from an implementation point of view.

7.3 Future Improvements

In this paper we have presented a framework for phrase break assignment from POS information and have attempted a thorough investigation into what the optimal parameter settings in the framework should be. One area we feel still needs further investigation is that of tagset composition. The experiments in section 5.2 used a greedy algorithm to collapse categories in the original 37 tagset to form a series of smaller tagsets. While this is a sensible way to progress we feel that a more sophisticated technique could do better. It may be possible to choose a tagset based on its actual ability to discriminate juncture types. We have also considered a system where two parallel sets of tags are used, one for before the juncture and the other for after.

While we believe more investigation into tagset composition would help in reducing errors we also believe that there is only so far that superficial text analysis techniques like this can go. From examining the errors in the test set we believe that a more sophisticated analysis will be needed to correct some of the errors. Although we argued in the introduction against using syntactic parsers for phrase break assignment, our reasons stem from the basic inaccuracy of these parsers, not because syntactic parses themselves are unhelpful. Recently several stochastic parsers have been presented which are trained on hand parsed trees and employ statistical techniques during parsing, e.g. (Magerman, 1994). These have been shown to significantly outperform rule-driven parsers. It is possible that a statistical parser could provide reliable parses and hence facilitate phrase break assignment.

References

Arnfield, S. (1994). *Prosody and Syntax in Corpus Based Analysis of Spoken English*. PhD thesis, University of Leeds.

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

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

Church, K. W. and 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.

DeRose, S. (1988). Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14:31–39.

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

Hirschberg, J. and Preito, P. (1994). Training intonational phrasing rules automatically for English and Spanish text-to-speech. In *Second ESCA/IEEE workshop on speech synthesis*.

Magerman, D. (1994). *Natural Language Parsing as Statistical Pattern Recognition*. PhD thesis, Stanford University.

Marcus, M., Santorini, B., and Marcinkiewicz, M. (1993). Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19:313–330.

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

Pitrelli, J. F., Beckman, M. E., and Hirschberg, J. (1994). Evaluation of prosodic transcription labeling reliability in the ToBI framework. In *ICSLP94*, volume 1, pages 123–126.

Rabiner, L. R. and Juang, B.-H. (1994). *Fundamentals of Speech Recognition*. Prentice Hall.

Sanders, E. and Taylor, P. A. (1995). Using statistical models to predict phrase boundaries for speech synthesis. In *Proc. Eurospeech '95, Madrid*.

Silverman, K. (1987). *The Sructure and Processing of Fundamental Frequency Contours*. PhD thesis, University of Cambridge.

Silverman, K., Beckman, M., Pitrelli, J., Ostendorf, M., Wightman, C., Price, P., Pierrehumbert, J., and Hirschberg, J. (1992). ToBI: a standard for labelling English prosody. In *Proceedings of ICSLP92*, volume 2, pages 867–870.

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

Wang, M. Q. and Hirschberg, J. (1992). Automatic classification of intonational phrasing boundaries. *Computer Speech and Language*, 6(2):175–196.

Acknowledgements

We would like to thank Steve Isard for his useful comments offered on the draft paper. Special thanks go to Eric Sanders who did much of the early investigative work on POS phrase break detection that lead eventually to the statistical approach developed here. The research reported here was conducted within the Satissfy project, and we gratefully acknowlegde the support of the UK Engineering and Physical Science Research Council grant, GR/K45299.

Appendix: Tagsets

The full initial 37 tagset set found from the WSJ is given in table 6. The best tagset, shown in table 7, is formed by collapsing these tags into 23 tags. As *ex*, *fw* and *2* are typically unreliably predicted, and quite rare, they are not included in the implementation released with Festival, and POS tags of this type, if predicted, are treated as part of the *nn_nnp_nnps_nns* group. Although this marginally reduces the accuracy for our test set it reduces the size of models and hence seems worthwhile in a run-time system.

Tag	Function
CC	coordinating conjunction
CD	cardinal number
DT	determiner
EX	existential “there”
FW	foreign word
IN	preposition
JJ	adjective
JJR	adjective, comparative
JJS	adjective, superlative
MD	modal
NN	non-plural common noun
NNP	non-plural proper noun
NNPS	plural proper noun
NNS	plural common noun
of	the word “of”
PDT	pre-determiner
POS	posessive
PRP	pronoun
puncf	final punctuation (period, question mark and exclamation mark)
punc hline RB	other punction adverb
RBR	adverb, comparative
RBS	adverb, superlative
RP	particle
TO	the word “to”
UH	interjection
VB	verb, base form
VBD	verb, past tense
VBG	verb, gerund or present participle
VBN	verb, past participle
VBP	verb, non-3rd person
VBZ	verb, 3rd person
WDT	wh-determiner
WP	wh-pronoun
WRB	wh-adverb
sym	symbol
2	ambiguously labelled

Table 6: The original WSJ tagset

Tag
cc
cd
dt
ex
fw
in
jj_jjr_jjs
md
nn_nnp_nnps_nns
of
pdt
pos
prp
punc_puncf
rb_rbr_rbs_rp
to
uh
vb_vbd_vbg_vbn_vbp_vbz
wdt
wp
wrb
sym
2

Table 7: The best clusterd tagset. The names directly show the clustering of the original WSJ set.