Employing Sentence Structure: Syntax Trees as Prosody Generators

Sarah Hoffmann, Beat Pfister

Speech Processing Group, ETH Zürich, Switzerland
{hoffmann,pfister}@tik.ee.ethz.ch

Abstract

In this paper, we describe a prosody generation system for speech synthesis that makes direct use of syntax trees to obtain duration and pitch. Instead of transforming the tree through special rules or extracting isolated features from the tree, we make use of the tree structure itself to construct a superpositional model that is able to learn the relation between syntax and prosody. We implemented the system in our SVOX text-to-speech system and evaluated it against the existing rule-based system. Informal listening tests showed that structural information from the tree is carried over to the prosody.
Index Terms: speech synthesis, prosody, syntax anaysis

1. Introduction

Prosody is an essential part of speech synthesis but notoriously difficult to handle. The variety of functions that are expressed in prosody makes it difficult to analyse or reproduce specific effects in isolation. Prosodic cues are ambiguous and therefore often enforced by other speech cues. Mistakes leave the listener with mixed signals that make speech less comfortable to listen to. A text-to-speech (TTS) system cannot sound natural without appropriate prosody. Different solutions have been proposed over time.

Unit selection largely circumvents the prosody generation issue by using the prosody from the underlying speech corpus with little modification. TTS systems that are based on diphone or HMM synthesis need to deal with prosody directly because they assemble speech on a segmental level. Each segment needs to be modified in terms of its duration and pitch contour.

There is a large gap between the functional purpose of prosody and the final realisation of duration and pitch. Linguistic theory tries to narrow this gap by using a more abstract description of prosody realisation expressing prosodic events in terms of phrases and accents. Rule systems have been developed to derive this abstract prosodic transcription from the text through syntax analysis and TTS systems have made successful use of these rules to derive natural sounding prosody, e.g. [1] and [2]. However, rule systems are language-specific and incomplete in the sense that they rely on heuristics about how prosody is usually expressed.

HMM synthesis systems [3] use a purely statistical approach, learning to generate prosody from features which are directly derived from the input text. They use methods like part-of-speech tagging to describe the context of the segment to generate. No explicit knowledge of the relation between text and realisation is required, thus making the method language-independent. However, the context these systems can take into account is very restricted for practical reasons, making it difficult to learn relationships that span over larger sentences.

The advantages of statistical prosody generators can be combined with systems that use syntax analysis when the output of the syntax analysis is used to learn prosodic features like pitch and duration directly. Syntax trees can provide a wider context and also carry information about the types of constructs and their relationships. The learning step ensures language-independence of the system. To this end, [4] describes a method to extract phrasing and pitch accents from syntactic dependency trees. Yet little work has been done to exploit the tree structure to extract more subtle prosodic cues.

In this paper, we propose a statistical approach to prosody generation that allows to learn duration and pitch directly from an unmodified syntax tree. The system was implemented as a prototype in the SVOX text-to-speech system, a concatenative TTS system that already uses syntax analysis to improve correctness of the speech output.

2. Prosody contours

A TTS system requires information about duration and pitch on a segmental level. Both features are highly variable even within a syllable because they are influenced to a large extent by the properties of the segments themselves. A syntax tree, however, mainly contains information about the sentence structure which influences prosody on a much wider scope. To take that into account, the prosody generation process is split into two stages. In the first stage, the prosody contour on a syllable level is generated from the syntax tree. In a second stage, these prosody contours are further processed to yield the concrete prosody features that are processed by the speech synthesis. The second stage will be discussed briefly in section 5. The main focus of this paper is on the first stage, the generation of syllable prosody contours from syntax trees.

The syllable prosody contours produced by the syntax trees are still a description of the concrete prosody realisation, only the segmental influence needs to be removed. We used a vector with the following five elements to describe duration, pitch and pause: Duration is abstracted as the lengthening/shortening of the syllable before and after the center of the syllable, where the average length of a syllable is computed as the sum of average durations of each segment of the syllable. Two duration values are used to account for the effect that the duration of syllables is often asymmetrically changed. Pitch is described by the mean fundamental frequency (F0) and the extent of movement within the syllable, i.e. the difference between maximum and minimum F0. Both are normalized with respect to the F0 range used by the speaker. There are studies that reported that pitch behaviour within the syllable is a relevant feature, e.g. [5] used F0 tilt for prominence detection. We found that the influence of segmental properties on the F0 contour is much greater and therefore have excluded it from the prosody contour description of the syllable. The length of the pause after the syllable constitutes the final element of the contour. If there is no pause, it is set to zero.

3. Extracting prosody from syntax trees

The syntax trees are generated with the SVOX syntax analysis [6] which is based on a chart-parser that uses sentence- and word-level definite clause grammars and a lexicon that contains orthographic and phonetic transcriptions. For the new prosody generation presented here, the syntax tree is simplified as follows: The nodes are stripped of any additional attributes, thus a node in the tree is described only by the constituent of the underlying grammar rule. The word subtrees are collapsed into a single node such that words make up the leaves of the simplified tree. Only the phonetic transcription together with information about lexical stress is retained. In addition, the transcription is processed by a syllabification unit which adds syllable boundaries making syllable information available to the prosody generator as well.

The syllable prosody module needs to generate a prosody contour for each syllable as described in the previous section. Essentially, the hierarchical structure of the tree needs to be mapped onto a linear sequence of contours. The hierarchical structure of the trees is conceptually similar to superpositional prosody models like [7] and [8]. These models build on the assumption that the final realization of prosody in an utterance is the result of the superposition of multiple base contours. In most models, the base contours are related to prosodic units: syllable, foot, phrase. The prosodic units form a hierarchy much like the syntax tree. The main difference is that the structure is motivated by prosodic events. An approach where the units are much closer to the syntactic structure is the skeleton-flesh proposed in [9]. The model makes a similar distinction between syllable and segment prosody generation as described in section 2. The final hierarchy for the syllable model consists of a compressed version of a syntax tree. This preprocessing of the tree is still rule-based. The aim of our approach is to omit any rule-based processing of the syntax trees because even the smallest modification of the syntax grammar makes it necessary to manually adapt a rule-based postprocessing of the trees.

Each node in the tree is considered a potential prosodic unit that may contribute to the final prosody of each syllable that is part of the subtree headed by that node. To that end, each node is assigned a *mutator function*. The function describes the local contribution of the node to each syllable's prosody contour. Consequently, the prosody contour of a single syllable is the result of the superposition of the mutator functions of all nodes on the path between the root of the tree and the leaf node the syllable is attached to.

There are two important differences with respect to the traditional superpositional models: first, syntax trees do not have a fixed number of levels. Second, not every node in the syntax tree may be relevant for the final prosody contour. Both problems will be discussed in the remainder of this section.

Superpositional models assume independence between the prosody contours generated on each level. The contours depend solely on the context features for the specific unit. This assumption clearly cannot hold for syntax trees because they can be nested arbitrarily deeply. If prosody contours are simply added then the final structure could become arbitrarily pronounced. There is a clear dependency on the modifications done by other mutators. To take that into account, the elements of the prosody contour generated so far are used as additional input features for the mutator functions.

Thus, the complete mutator function can be defined as

$$M_C : F_{M_C}, P \rightarrow P \tag{1}$$

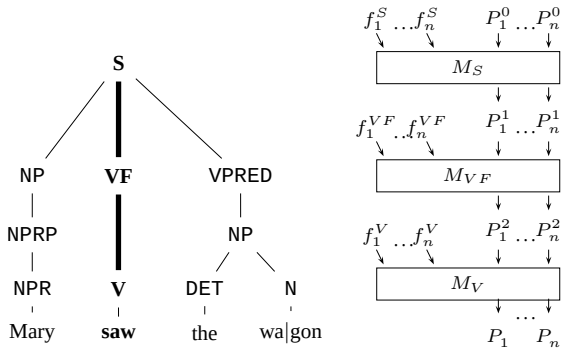

Figure 1: *Application of mutators for syllable 'saw'. P_x^0 is the initial neutral prosody contour, P_x the final contour and f_x^C are the input features of the mutator belonging to constituent C.*

where F_{M_C} is the feature vector for the mutator function M_C and P constitutes the vector describing the prosody contour of a syllable. Let $C_1, \ldots, C_n$ be the sequence of constituents on the path from root constituent C_1 to the leaf constituent C_n. Then the prosody contour of a single syllable is computed recursively as

$$P_k = M_{C_k}(F_{M_C}, P_{k-1}) \tag{2}$$

where P_n is the final contour and P_0 the neutral contour.

The order of application of the mutator functions is important because of the recursive nature of this definition. By applying the mutators top-down from root to leaf, they will be applied in decreasing order of scope. The base contour is a constant prosody contour P_0. The exact value of the contour does not matter because the grammar is structured in a way that each sentence has a root constituent S. The output of its tree mutator represents the neutral sentence prosody. Figure 1 shows an example how the prosody contour of the syllable 'saw' is applied for the sentence 'Mary saw the wagon'.

While there is a relationship between the hierarchy of prosodic units and the syntax tree, syntax trees contain more detail. Nodes in the syntax tree primarily express syntactic relationships, not all of them have an equivalent in prosody.

The recursive definition of the mutator functions also allows to take into account nodes that do not contribute to the final prosody. Those nodes are assigned zero mutator functions, a mutator that does not modify prosody, i.e. $P_k = P_{k-1}$. The classification whether or not a node is relevant then becomes part of the learning process of the mutator functions.

4. Tree mutators

So far, the tree mutators were defined only in terms of a generic feature set. The choice of appropriate features is important because it constrains the relationships that can be expressed through prosody. The constituent type alone is clearly not sufficient to define a unique contour. In enumerations, for example, each member creates a different prosodic structure even though all members have the same constituent and potentially exactly the same structure of their subtree. The immediate context of a node is therefore important. Because the mutator function computes the prosody of a single syllable, the position of the syllable within the subtree spanned by the nodes needs to be taken into account as well. The complete list of context features used for the mutators is as follows:

- parent constituent

- constituent of right/left neighbour
- constituents of siblings to the right/left
- punctuation before/after node
- number of syllables from right/left boundary
- lexical stress of syllable

Note that the position of the syllable within the subtree proved not to be relevant. This is consistent with the observation made by [9] that relevant changes happen predominantly on the boundaries of prosodic units.

The constituent features are binary features. There is one feature for each constituent type that describes its presence or absence. The way our grammar is structured entails that the set of relevant parent or neighbour constituents is much smaller than the full set of constituents used in the grammar. The feature space can therefore be greatly reduced by constructing one tree mutator function per constituent. The feature set for each mutator function can be automatically constructed from either the grammar or from the training set. The latter has the advantage that only features will be taken into account for which actual training material is available. Similarly, constituent types that do not appear sufficiently frequently in the training material can be excluded and assigned zero mutators instead.

While in principle any regression learner is usable as a mutator function, CART [10] proved most suitable with respect to the properties outlined above. In particular, they are able to cope with heavily correlated and irrelevant features from the automatic feature selection process.

Training the tree mutator functions requires the intermediate output of the application steps. Because all intermediate outputs are initially unknown we rely on an analysis-synthesis learning approach. Initially all mutator functions are set to be zero mutator functions. The mutator functions are applied and the intermediate prosody contours computed. The intermediate contours are then used to compute new mutator functions. These steps are repeated iteratively until the mutator functions are sufficiently trained.

In order to account for zero mutators, the CART are trained on the change in the prosody contour and the output functions of the CART models have been slightly modified to filter out those nodes where the underlying examples do not sufficiently support the proposed prosody modification. More formally, let $S = (s_1, \ldots, s_n)$ be the set of underlying samples that support a leaf. For one sample s_i, let $p^I_{s_i}$ be an element of the input prosody contour and $p^O_{s_i}$ the expected output. Then the set of samples that support the trend set by the node can be defined as

$$P = \left\{ s \in S \middle| sgn(p^O_s - p^I_s) = sgn\left(\frac{\sum p^O_{s_i} - p^I_{s_i}}{N}\right) \right\} \quad (3)$$

The resulting prosody difference δp of the leaf is then computed as

$$\delta p = \begin{cases} \frac{\sum p^O_{s_i} - p^I_{s_i}}{N} & \text{if } |P| > \theta N \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where θ defines the degree to which the samples need to support the prosody modification. The higher a value is chosen, the more pronounced a prosodic event needs to be actually taken into account by the mutator. We found that a value of 0.8 produced a balanced output.

Apart from these modifications, the trees are constructed using standard algorithms. One CART model is trained for each element of the prosody contour. However, as each tree uses the complete contour vector in its input, dependencies between the different elements can be learned as well.

5. Realisation in the SVOX TTS

The SVOX text-to-speech system is a concatenation-based system that uses linguistic information to improve synthesis output. The prosody generation is done in two steps, a speaker-independent and a speaker-dependent part [11]. In the first part, an abstract phonological description is created from the syntax tree. This step is entirely rule-based. In the second step, the concrete prosody is generated using neural networks (ANN). The network for duration has exactly one output that describes the length of one segment. The F0 network outputs a sequence of F0 values that form the concrete F0 contour of a syllable. The input features for the ANN are generated from the phonological description. Thus, they are comprised of segmental properties from the phone sequence and of prosodic properties about accentuation and phrasing. The duration ANN has 80 inputs and 40 nodes in the layer. The F0 ANN is recurrent with 54 inputs, 25 nodes in the hidden layer and 10 feedback nodes.

The neural networks already proved to be able to produce natural-sounding prosody. Thus, they are well suited for the generation of the segment-level prosody as outlined in section 2. The CART-based prosody generation can be incorporated into the existing system by removing the rule-based generation of abstract prosody and replacing prosody-related features in the segmental ANN with the prosody contour vector. The segment-related input features of the ANN remain unchanged. The resulting ANNs are significantly smaller. The duration ANN has 37 inputs and 20 hidden nodes and the F0 ANN contains 21 input, 15 hidden and 10 feedback nodes.

For the training of the CART models, a speech corpus is used for which only a textual transcription is required. The syllable prosody contours and the prosody parameters for the segments are extracted from the speech signal in a completely automatic fashion. In a first step, the syntax tree for each sentence is computed from the orthographic transcription using the same syntax module that is used during synthesis. The phonetic transcription contained in the syntax tree is used to obtain a phonetic segmentation of the speech signal according to the algorithm described in [12]. Finally, the F0 values are extracted using the method described in [13].

The segmental ANNs are trained on the actual output of the syllable prosody generators so they can learn to compensate for systematic errors in the CART tree output.

6. Discussion

For evaluation, we compared the statistical CART generation approach with the rule-based prosody generation as previously used in the SVOX TTS system. For this, prosody models were generated form a German speech corpus for both approaches. To evaluate the influence of the syllable prosody, we also trained a baseline model using only the ANN models of the segmental stage. It was trained using only the segmental features that are shared between the CART-based and rule-based models.

For training we used a German speech corpus consisting of 1573 sentences spoken by one professional female speaker. The corpus contains statement sentences comprising phenomena like address, numbers and enumerations. The sentences were spoken in isolation, the speaking style was rather lively.

The baseline model showed a duration error of 23.6 ms and an error on F0 of 32.0 Hz (based on a vocal range of 200 Hz)

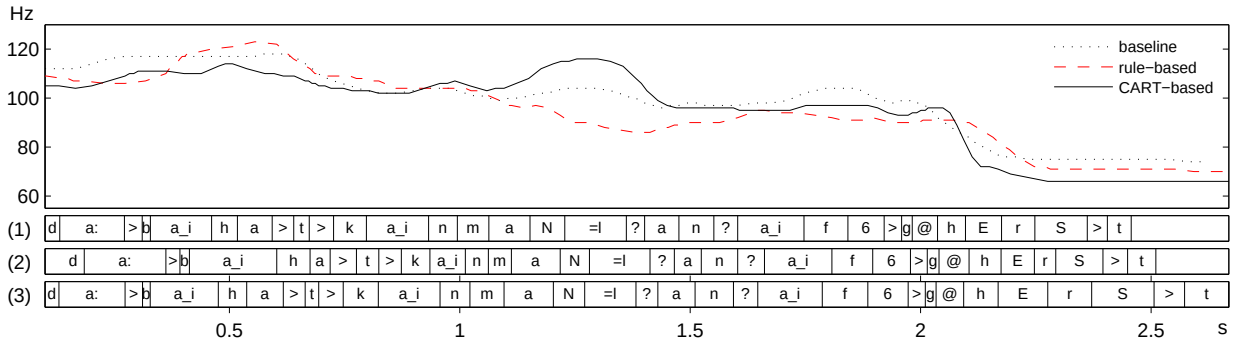

Figure 2: *Prosody generated for the sentence 'Dabei hat kein Mangel an Eifer geherrscht.' The upper part shows the generated F0 contours, below the segmentation generated by the baseline model (1), the rule-based model (2) and the CART-based model (3).*

after 200 training iterations. The rule-based system improved the error to 20.7 ms and 29.7 Hz respectively. The CART-based model was able to learn duration equally well with a training error of 20.7 ms. It did slightly worse for F0 with an error of 30.5 Hz, most likely because the ANN models can make use of their recurrent structure for the sentence prosody as well.

Manual inspection showed that the models largely agreed on the realisation of primary phrase boundaries but differed on the realisation of the secondary ones. Figure 2 shows such an example sentence from the training corpus. Note that the overall trend of F0 is already present in the baseline. The modifications introduced by the syllable models are purely related to accentuation and phrasing. While the rule-based system did not add any phrase boundaries, the CART-based system was able to take advantage of information from the syntax tree: it learned to place the main accent to the first noun phrase (*kein Mangel an Eifer*) and add an audible minor phrase boundary before the propositional phrase. Modifications like these mean also that the CART-based model is much more sensitive to errors in the syntax analysis stage. For example, in some ambiguous sentences the analysis may split a noun phrase into two noun phrases. While the rule-based system is written in a way to not add a phrase boundary in such cases and thus ignore the error in terms of prosody, the CART-based model had learned to put an audible break making the analysis error obvious to the listener.

The largest problem for the CART-based model was the limited training material. While the corpus size was sufficient to train the ANN-models in the rule-based model, there was much less material available for the training of the mutator functions. A larger and more varied corpus might allow the CART-based system to generate a much richer prosody.

7. Conclusion

In this paper, we introduced a method to generate prosody contours directly from a syntax tree without utilizing an intermediate abstract prosody representation. We showed that the approach is able to produce a viable prosody contour and that it is able to incorporate structural information from the syntax tree.

The grammar used in the prototype was not specially tuned to take structures into account that are prosodically relevant. Phenomena like negation and contrast have a strong influence on prosody and can be easily formulated in the context of a DCG grammar framework. Thus, our approach allows to formulate specific prosodic effects from a purely functional point of view and leave the concrete realisation to be learned by the system, provided that sufficient training material is available.

8. References

[1] E. Fitzpatrick and J. Bachenko, "Parsing for prosody: what a text-to-speech system needs from syntax," in *Proceedings of Annual AI Systems in Government Conference*, 1989.

[2] F. Malfrère, T. Dutoit, and P. Mertens, "Fully automatic prosody generator for text-to-speech," in *5th International Conference on Spoken Language Processing*, 1998.

[3] T. Yoshimura, K. Tokuda *et al.*, "Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis," in *Proceedings of Eurospeech*, 1999.

[4] J. Hirschberg and O. Rambow, "Learning prosodic features using a tree representation," in *Proceedings of Eurospeech*, 2001.

[5] G. Abete, F. Cutugno *et al.*, "Pitch behavior detection for automatic prominence recognition," in *Proceedings of SpeechProsody*, 2010.

[6] C. Traber, "SVOX: The implementation of a text-to-speech system for German," Ph.D. dissertation, Computer Engineering and Networks Laboratory, ETH Zurich, 1995.

[7] H. Fujisaki, "A note on the physiological and physical basis for the phrase and accent components in the voice fundamental frequency contour," *Vocal Fold Physiology: Voice Production, Mechanisms and Functions*, 1988.

[8] J. van Santen, A. Kain *et al.*, "Synthesis of prosody using multi-level unit sequences," *Speech Communication*, vol. 46, 2005.

[9] G. Bailly and B. Holm, "SFC: A trainable prosodic model," *Speech Communication*, vol. 46, 2005.

[10] L. Breiman, J. Friedman *et al.*, *Classification and Regression Trees*. Wadsworth, 1984.

[11] H. Romsdorfer, B. Pfister, and R. Beutler, "A mixed-lingual phonological component which drives the statistical prosody control of a polyglot TTS synthesis system," *Machine Learning for Multimodal Interaction*, 2005.

[12] S. Hoffmann and B. Pfister, "Fully automatic segmentation for prosodic speech corpora," in *Proceedings of Interspeech*, 2010.

[13] T. Ewender, S. Hoffmann, and B. Pfister, "Nearly perfect detection of continuous F0 contour and frame classification for TTS synthesis," in *Proceedings of Interspeech*, 2009.