Parsing Phonetic Segments into Syllables

by

Georg Dorffner, Stan C. Kwasny,
and Robert F. Port

Computer Science Department
Indiana University
Bloomington, Indiana 47405

TECHNICAL REPORT NO. 217

Parsing Phonetic Segments into Syllables

by

G. Dorffner, S.C. Kwasny, & R.F. Port

March, 1987

Parsing Phonetic Segments into Syllables[†]

Georg Dorffner[‡]

Stan C. Kwasny

Computer Science Department
Indiana University
Bloomington, Indiana 47405

Robert F. Port

Linguistics Department
Indiana University
Bloomington, Indiana 47405

Keywords:
Speech Analysis
(Deterministic) Parsing

ABSTRACT

Speech recognition poses many unique problems for Computational Linguistics. We are examining an approach to parsing phonetic segments into syllable structures using a deterministic parser. Our goals are to study the role of timing information in speech recognition and to design structures appropriate for the task of parsing segments into syllables and syllables into words.

† This work was supported in part by the National Science Foundation under grant No. DCR-8518725.

‡ Current Address: Department of Medical Cybernetics & Artificial Intelligence, University of Vienna, Vienna, Austria.

1. Motivation

Our goal in this research is to examine the potential of automatic speech recognition using symbolic phonetic descriptions of the input signal. We are assuming the existence of a low-level system component that can detect segments in a speech input stream and label them using relatively simple, but reliable classes of labels. The phonetic segments are to be assembled into words in order to make the recognition process complete. We believe that successful speech recognition begins with the ability to accurately identify syllables and words in the signal and thus are not yet considering syntactic or other levels of parsing in this project.

A simple solution to the problem would be to provide a lexicon where each entry contains a phonetic transcription (i.e., a string of phonetic symbols) of a word. The process of putting the symbols of the input string together to form words would then be a vast search and pattern matching, which would quickly become too costly and inefficient if the lexicon is to contain a large number of entries.

Therefore we decided to try parsing the string of phonetic symbols into larger units, making the lexical access easier. One type of unit which seems to be very appropriate for this purpose is the syllable. In most cases, words can be clearly separated into syllable-like segments. Another reason for choosing syllables is because of features like stress which are mostly bound to syllabic structures. These factors, when detected, provide additional valuable information about word identity.

As a result of these considerations we have built a parser which in a limited way scans the input string of phonetic symbols and puts them together into syllables. We have found that a deterministic parser similar to that of Marcus (1980) is most appropriate for this purpose. Indeed, human recognition seems to happen without extensive backtracking, therefore suggesting that recognition is possible in a deterministic framework.

In the following sections we describe the problems we have encountered while taking a deterministic approach to our problem. Furthermore, we introduce several extensions to Marcus' original parser that we developed and that have helped us achieve success in the task of efficiently finding words from a selected vocabulary in a stream of symbols.

2. The Peculiarities of Speech

Parsing has been known as a tool of analyzing symbolic strings and has been applied as such mainly to "language" inputs, that is, inputs that contain words of a (human or formal) language as symbols. Trying to use parsing techniques to analyze symbols which represent a continuous speech signal is essentially analogous to that. However, there are several differences in parsing speech that must be faced.

First of all, detecting and labeling segments in a speech signal cannot always be done reliably. That means, that once one chooses an alphabet of symbols which is detailed enough for the desired purposes it cannot always be guaranteed that all incoming speech segments can be classified accurately according to that alphabet. This is not only due to noise that inevitably accompanies speech signals but also to inconsistencies and errors in the production of the utterance. We therefore had to find a way of dealing with the input string even when we cannot be sure that all symbols can be described with the same degree of detail.

What goes along with the first observation is the fact that not every single segment in a symbolic stream representing speech is necessarily semantically significant and important for the outcome of the recognition task. The process can very well be successful without knowing about all the details of every symbol, as the signal is redundant enough to allow compensation. For this reason, the parser should not have to wait until all information comes in, but rather be able to continue analysis with whatever information there is.

Parsing speech involves dealing with many more ambiguities than parsing language, especially ambiguities in syllable boundaries. Reasons for this include the smaller depth of the syntactic structure (we can only distinguish maybe 3 or 4 levels, like syllable, onset/coda, consonant/vowel, etc.) and the fact that the symbols used are not as distinct in function as are words due to their syntactic category. For example, almost every consonant segment can begin either an onset or a coda of a syllable, therefore not giving as much information as, say, a determiner does about the substructure (NP) it belongs to.

Finally, there are a variety of other sources of information besides just the symbols which can help in making the recognition task easier. One of the most important sources, as we have shown in many experiments (Port, et al., 1986; Dorffner, et al, 1986; Reilly & Port, 1985), is timing, that is the relation of durations of different segments. Stress, although strongly related to timing, would be another one. A parser for speech should make use of this additional information.

3. Extensions to the Deterministic Parser

3.1. Multiple Structure Levels, Label Hierarchies and Monotonistic Structure Building

In an earlier paper (Dorffner, Kwasny, Port, 1986) we have introduced two major expansions of Marcus's original wait-and-see parser (WASP). We have augmented the parser to allow for parallel construction of several structures inherent to the input, like syntax, semantics, intonation, referential structure, etc. Applied to speech the parser is able to account for the multiple structural levels inherent in speech. The parallel building process can be utilized to coordinate information sources in successfully completing a parse occassionally even with incomplete input.

To achieve the flexibility of the parser toward incomplete input strings (of the sort described above) we introduced a hierarchy of labels rather than one fixed alphabet of symbols. One possible hierarchy is illustrated in Figure 1.

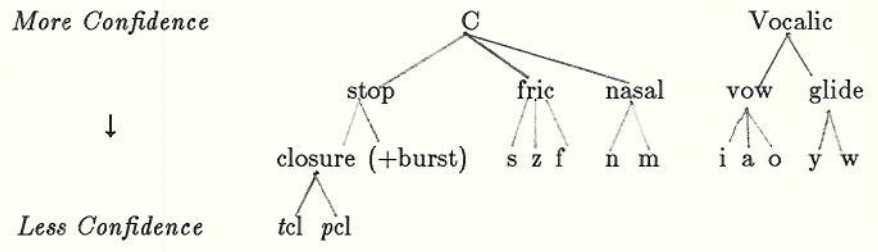

Figure 1: Hierarchy of Labels

The parser is able to accept a label at any position in the hierarchy. This depends of course on the degree of detail confidently known about the symbol. Additional detail may be added if it should be possible at a later point in the parse. This means that only labels known with high enough confidence get attached to nodes (syllables) thus never requiring the parser to detach an element or throw away any node. This is consistent with the deterministic nature of the parser. Furthermore, as an expansion to the original WASP, our parser may add information, thereby only increasing information on the nodestack but never changing it. This is what we call "monotonistic structure building". More details about these features can be found in the earlier paper.

3.2. Making Use of Timing Information

As experiments we have conducted in our lab have shown (Port, et al., 1986), there is a great deal of information about word and syllable content in segmental durations and their relations to each other. These relations are not simple ones, as earlier attempts of describing them seemed to suggest (Port, 1981). In other words, computing simple ratios of, for example, vowel to consonant length, is not sufficient to extract all the information contained in timing measures.

This lead us to the idea of using a statistical classification technique, called discriminant analysis (DA), applied to a set of durational measures to predict certain properties of the speech signal (Klecka, 1980). In our task of syllable parsing this technique can particularly be helpful in predicting syllable boundaries. To prove this hypothesis we designed a database, called (after one of its word entries) the "Aztec" database. It contains tokens of sequences that all match one general pattern of labels. We assembled the set of sequences so that it contained all possible combinations of phonetic labels, syllable boundaries and stress structure. The phonetic template matched by all sequences is:

$$(1) \quad [\text{vow}] \underset{B_1}{|} [\text{fric}] \underset{B_2}{|} [\text{stop}] \underset{B_3}{|} [\text{vow}]$$

This is one of many possible templates a parser might be faced with while trying to extract syllables out of a string of symbols. That is, the parser has already parsed everything which comes before the first vowel (which is recognized as being the center of a syllable) and now has to decide where to place the boundary between the current syllable and the following one.

Excluding phonological restrictions like aspiration of a stop in syllable-initial position, there are three possibilities for the syllable boundary in this template, depicted as B_1, B_2 and B_3. We recorded all the sequences using several speakers and hand-measured the durations of the segments (plus some additional measures like duration of adjacent consonants, voicing stretches, etc.). Then we analyzed the timing data using discriminant analysis in an attempt to predict the syllable boundary (B_1, B_2 or B_3). We used some of the tokens for training, i.e. computing so-called discriminant functions, and all of the tokens for the prediction.

The results show that in many cases predictions can be made with very high confidence, therefore suggesting that a parser can make good use of such timing information. That means that the parser can have rules that call the discriminant function corresponding to the label template under consideration and use the results for a decision about which symbols to attach to the coda of one syllable and which to the onset of the next. These rules most appropriately have a condition that asks if the prediction of one of the boundaries is above a certain threshold (e.g. 90 or 95%). If so, then the rule fires, causing one syllabic structure to be completed and another to be initiated.

However, this will work only in some of the cases. Very often, the probabilities will not reach the confidence threshold. To avoid too large an error rate, we do not want the parser committing to a hypothesis with a low confidence level. Therefore, the parser attempts to increase the confidence levels returned by the discriminant function.

In the example using (1) we assumed that there exists a discriminant function that has been trained on a large corpus of data matching a general template. This function takes the temporal properties of the elements in the template and predicts the position of the syllable boundary. However, by doing this we have not made use of information about other features of the signal, like the identity of the elements. One way of incorporating that information would be to provide a discriminant function for more specific templates, as well. For example, with the following more specific template pattern:

(2) [vow] [s] [stop] [vow]

one would expect the confidences of predictions using DA to rise, because the variation due to the label that had been specified was ruled out. The results of the "Aztec" experiments have shown that this is in fact the case, but the probabilities in this particular case improve only slightly.

We have also taken another approach in cases where the confidences were not above the threshold. After discovering that the confidences p(B_1), p(B_2) and p(B_3) for a sequence like (1) are not high enough, we make an assumption that (1) had a particular syllable boundary, for example B_1. We next use another discriminant function that has been trained on only B_1-cases to predict something else, for example the fricative identity. If there are only two possible fricatives (as in the "Aztec" database - [s] and [z]), we get the two conditional probabilities, p(s ˜|˜ B_1) and p(z ˜|˜ B_1). Similarly, we can get p(s ˜|˜ B_2), p(s ˜|˜ B_3), p(z ˜|˜ B_2) and p(z ˜|˜ B_3). Analogously in the next step, we suppose that (1) does have an [s] (independently from what we might know about (1) in reality). We then use a discriminant function that was trained only on templates that contain an [s], this time predicting the syllable boundary again. Thus, we get more conditional probabilities: p(B_1 ˜|˜ s), p(B_2 ˜|˜ s), etc. Using Bayesian probability theory, we can then combine all the probabilities, according to the formula:

$$\begin{aligned}\hat{p}(B_1) &= p(B_1 \mid s)p(s)+p(B_1 \mid z)p(z) \\ &= p(B_1 \mid s)[p(s \mid B_1)p(B_1)+p(s \mid B_2)p(B_2)+p(s \mid B_3)p(B_3)]+ \\ &\quad p(B_1 \mid z)[p(z \mid B_1)p(B_1)+p(z \mid B_2)p(B_2)+p(z \mid B_3)p(B_3)]\end{aligned}$$

analogously for p(B_2) and p(B_3). The probabilities p(B_1), p(B_2), p(B_3) on the right hand side are the original ones, which we had after the first step of the run. Those on the left side ($\hat{p}$ (B_1), etc.) will be new values, which often are higher for the correct prediction and lower for the other ones.

By doing this, we have tested if certain predictions (here: syllable boundary and fricative identity) are consistent with each other. The chain of reasoning, summarized, goes something like this: If the boundary is B_1 (or B_2 or B_3), then it should predict the fricative identity with a high probability, and if the fric is [s] (or [z]), it should in turn predict the syllable boundary with a high probability. Thus, the overall confidence should go up. If, on the other hand, the boundary is not B_1 (or B_2 or B_3), then the predictions of the fricative identity should be much worse, therefore decreasing the confidence.

These ideas closely resemble the mechanism of mutual excitation (or inhibition) of partial hypotheses that are consistent (or inconsistent, respectively) with each other, as is done in many connectionist networks. For example, in a recognition network for visual representation of words (like McClelland & Rumelhart, 1981) a letter unit excites all the word units that contain the letter, while, in turn, all word units excite all the letters they contain. Similarly, in the case of applying discriminant analysis, a hypothesis of a syllable boundary might "excite" other properties, like the identity of the fricative, which in turn improves the likelihood of the boundary (if they are consistent).

3.3. Lexical Access

As mentioned earlier, the system is most efficient if lexical access can be postponed until the end of the parse when all the syllables are known. However, in many cases the parse cannot be done merely by using techniques just described. If we restrict it to a few trials, lexical access can also be helpful in finding syllable boundaries. The restriction we have to make is that we use lexical access only while considering one of the consonant clusters between two vowels (like in the template above), because then the parser only has to make guesses about how long the coda of the current syllable is, and lookups in the lexicon can be done with what has already been parsed plus a short sequence from the buffer. If we wanted to apply lexical access stretching over sequences beyond the next vowel, we would have to make guesses about two boundaries, which very quickly increases the number of trials.

As a result of that assumption, lexical access is only of great help during syllable parsing if the consonant cluster that contains the ambiguity contains a word boundary, rather than only a syllable boundary. That does not mean we never allow lexical access beyond the limit just discussed, but we want to postpone it until as late as possible (in fact at some point, if all other attempts fail, the parser will have to do it).

3.4. Local Parallelism

If all the techniques described above fail, there might still be a more efficient way to continue parsing before attempting a less constrained lexical access. Normally, the deterministic scheme of a WASP does not allow for any backtracking process. That means that everything that is attached to the nodestack has to stay there. There is no mechanism for a temporary solution which might be revised later. All the facilities and techniques we added to the parser and that we have described so far stay within this framework.

However, in parsing speech it might seem appropriate to incorporate backtracking to a certain, very restricted extent. It is not very likely that human recognition takes place deterministically down to the microscopic level, i.e., to the detail of recognizing phonetic elements. It is very much conceivable that - speaking in the symbolic metaphor of mind - temporarily, over a short period of time, several hypotheses can co-exist, from which, however, one gets sorted out very soon.

Applied to a deterministic parser this could mean the introduction of a backtracking scheme that is restricted to be acting only locally, without allowing many branching points of hypotheses or long stretches over which backtracking can take place. We refer to this idea as "local parallelism" (or "local backtracking"[1]), as globally the parser would still act deterministically in Marcus's sense. Therefore, this idea is not counter Marcus' original motivation. Marcus himself tried to solve local indeterminism by providing the buffer which constitutes a restricted lookahead. In our case, additional facilities - like local parallelism - seem to be necessary to avoid using the buffer ad absurdum, that is, as an unrestricted lookahead.

There are several ways of achieving local parallelism:

- Allowing only one branching point per time. If another branching point is about to come up, the decision about the old one already must have been made.
- Restricting multiple hypotheses to one or two nodes in the stack. This is not very restrictive during syllable parsing, because, at least in our implementation, there are rarely more than two nodes on the stack at any one time (on syllable level).
- Building a data structure between buffer and node stack for parallelism, which, like the buffer itself, is restricted in size.

We have implemented the first idea. That is, we allow backtracking, but only once. At a point of multiple hypotheses about syllable boundaries the current status of the parse (i.e. nodestack, buffer, input string) is stored under a global variable. Also stored is a rule packet with which the parse is to be continued in case the current status is recovered. In such cases, the first of the hypotheses is taken to be true and the parse is continued. If at one of the next syllable boundaries lexical access with the part that has been parsed already is successful then the hypothesis was the correct one and a global flag is set to indicate that another branching point may be handled. If the parse fails at one of the next potential syllable boundaries then it is assumed that the wrong hypothesis has been taken. The old status is recovered and the parse continued with the stored rule packet (which, in the normal case, pursues the second hypothesis), and so forth.

[1] Backtracking can be viewed as pseudo-parallelism of competing hypotheses.

Also, the flag could be set automatically after three or four consonant clusters, restricting backtracking to the length of one or two words.

4. The Parsing Process

The process of parsing a continuous string of phonetic symbols into syllables can be described as follows: The center of syllables are vowels or vocalic elements. The consonant clusters between those vowels have to be parsed into the coda of the first syllable (if any) and the onset of the next one (if any). Therefore the main problem of the parsing mechanism is to find the boundary between coda and onset in the consonant clusters.

Among all the ways of providing information that helps in decision making about syllable boundaries, optimally there is a certain order which reflects itself in priorities and order of how rules are applied:

1. Phonological Rules
2. Discriminant Analysis
    a. Direct
    b. Increasing Confidence
3. Lexical Access Rules
4. Local Parallelism
5. Extensive Lexical Access

In many cases phonological rules alone help to decide. After that we try using timing information, provided there is a discriminant function for the template. Then restricted lexical access should take place, because in many cases that can help a lot. Finally, before extensive lexical access has to be employed, the mechanism of local parallelism tries to find a solution.

5. Details about the Parser

5.1. Extensions to a Wait-And-See Parser (WASP)

The parser we have implemented includes a series of extensions to the original WASP in order to accomplish all the facilities we have been describing in this paper.

To include the durations of the segments in the input string, we added a second buffer (as well as a second input string) that contains timing measures. Both buffers are acted upon by the interpreter in parallel so that at any point one position in the symbol buffer corresponds to one position in the timing buffer (its duration). The introduction of the second buffer is necessary because the durations of the segments are independent of their symbolic description and thus cannot be simply a feature attached to the symbol.

As already described in Dorffner, Kwasny, Port (1986) we extended the rules by dividing the left hand sides (conditions) into two parts. The conditions in the first parts all correspond to a buffer position (in the right order) and may trigger attention shifting rules. The conditions in the second part can be arbitrary lisp functions (preferably one of the provided standard functions), as long as they return a truth value. This introduces the possibility of having additional conditions like sending a component to a frame representing one of the multiple structures (and checking if it can be attached).

Nodes in our parser cannot only bear features but can also have additional data like a duration or a result from discriminant analysis (which has to be stored in order to be evaluated by a rule) associated to them.

Finally, a variety of standard actions have been added (to "old" standard actions like attaching, creating nodes, etc.) to facilitate tools like discriminant analysis, lexical access and local parallelism.

5.2. Attention Shifting Rules

Our original idea was to capture the levels of the syntactic structure (syllable - onset/coda - consonant/vowel) with attention shifting (AS) rules. While this works very well with parsing consonants (like 'stop') as part of smaller segments (like 'closure' and 'burst'), implementing 'onset' and 'coda' with the help of attention shifting rules bears some problems. In Marcus's original paradigm, attention shifting (AS) rules trigger deterministically and it is assumed that once the interpreter steps down a level by firing an AS rule, the parse at the lower level succeeds and deposits its result into the buffer. For example, if the presence of a determiner like "the" causes an AS rule to fire in a syntactic parser looking for an NP, then the parse can only be successful if the determiner is really the beginning of an NP, otherwise the parser halts. This goes along with the determinism assumption that once the parser has built some substructure, it can never tear it down again.

However, when parsing syllables it is not as clear what is the beginning of an onset or a coda, because nearly any consonant label can start either of them. Therefore a rule of the form

```
((feature 'stop 1)
 (feature 'onset 2)
        →
 (attach-to-node 1))
```

will assume that the parse of the onset in the second buffer position is successful. Otherwise we would have to backtrack and to detach some elements we already attached to the onset (this, of course, is counter the WASP idea).

We could get around this problem by attaching all elements of an onset or a coda at once. Then we would have only two options: Parsing of the substructure succeeds and we attach the elements, or parsing fails, we do not attach anything and therefore can throw away the node that we temporarily created.

But even if we solve the first problem, there are more. AS rules, as Marcus defined them, fire every time they are triggered and, after the parse on the lower level, return the interpreter to the previous level. There, all active rules are attempted again. That means, that if a rule asks for an onset in one of its conditions

```
((feature 'onset 1)
        →
      . . . )
```

and parsing of that onset is not successful (something that never occurs with NPs in Marcus's original parser, as mentioned above), then the rule will try to trigger an AS rule again, unless some sort of flag is set that this rule already lead to a parse that failed. In other words, the conditions of a rule do NOT depend on the success of an attention shifting process in the original implementation. We could get around this problem by making a condition dependent on the success of attention shifting and parsing at the lower level. By doing this, however, we would drastically change the original mechanism of the interpreter.

But there is yet another problem with using AS rules for onsets and codas. The deterministic mechanism would not allow us to detect ambiguities in syllable boundaries, because the first successful parse of a consonant cluster into coda and onset would be taken (if no other decision mechanism like lexical access is employed). Discriminant analysis as it is done for the "Aztec" data would never have a chance to help. For example,

```
((feature 'fric 1)
(feature 'onset 2)
→
. . . )
```

could be a rule for parsing a coda that consists of only a fricative. While parsing a phonetic representation of the utterance "the sport", for example, this rule can fire and parse the [s] as coda of the first syllable. This is due to the fact that according to phonological rules alone, there is more than one place for the syllable boundaries (this is what the "Aztec" database is all about). This ambiguity (according to phonological rules) is what we would like to detect so that the parser can employ techniques like discriminant analysis. The deterministic nature of the attention shifting mechanism, however, as was just illustrated, would leave the ambiguity unresolved.

For all these reasons, parsing of onsets and codas should be done on one level without the use of AS rules. This might be less elegant but seems to be the only way to avoid all the problems discussed above. The only places where parsing an onset or a coda with an AS rule can be kept in the system is at the beginning of the input string (first onset) and at the end thereof (last coda).

It should be mentioned that using AS rules to account for the fact that the input might contain labels of different detail and confidence (as part of monotonistic structure building, described earlier) can still be done without encountering similar problems. The only restriction is that "onset" and "coda" cannot be part of the label hierarchy.

6. Conclusions

We see potential in the method for parsing phonetic segments into syllables described here. To date, we have only experimented with data in the Aztec database, but more experimentation is planned. One recognized drawback to the method is the possibility that an enormous quantity of data will be required to provide sufficient information to allow us to generate good discriminant analysis equations. Our hope is that the equations will tend to cluster into groups and that several discriminations can be achieved by a single equation.

Acknowledgements

We are grateful to Sven Anderson for his dedicated work in setting up the Aztec database. We also appreciate contributions to this work by Daniel Maki.

References

Dorffner, Georg, Stan C. Kwasny, and Robert Port (1986) "Deterministic Parsing of Multiply-Structured Inputs," *Research in Phonetics and Computational Linguistics*, Report No.~5, 105-122, Department of Linguistics and Computer Science, Inidana University, Bloomington, IN, July, 1986.

Dorffner, Georg, Daniel Maki, William Reilly, and Robert Port (1986) "Timing as Speaker Independent Information for Speech Recognition," *Research in Phonetics and Computational Linguistics*, Report No.~5, 91-104, Department of Linguistics and Computer Science, Inidana University, Bloomington, IN, July, 1986.

Klecka, William (1980) *Discriminant Analysis*, Sage Publications, Beverley Hills, CA, 1980.

Marcus, M. P. (1980) *A Theory of Syntactic Recognition for Natural Language*, MIT Press, Cambridge, MA, 1980.

McClelland, J.L., and D.E. Rumelhart (1981) "An Interactive Activation Model of Context Effects in Letter Perception: Part 1, An Account of Basic Findings," *Psychological Review*, 88, 375-407.

Port, Robert, William Reilly, and Daniel Maki (1986), "Using Timing Information to Discriminate Words," *Research in Phonetics and Computational Linguistics*, Report No.~5, 73-90, Department of Linguistics and Computer Science, Inidana University, Bloomington, IN, July, 1986. Also appearing as: "Using Global Timing to Discriminate Words," (in press), *Journal of the Acoustic Society of America.*

Port, Robert F. (1981) "Linguistic Timing Factors in Combination," *Journal of the Acoustic Society of America 69*, 262-274.

Reilly W., and R. Port (1985) "Use of Timing to Discriminate Words," *Journal of Acoustic Society of America*, 78, S21, 1985.