Dynamic Computational Networks: Syllabification and Accent

Brandon Rhodes

June 2, 2019

1 Synopsis

My dissertation proposal can be summarized by the following points:

1. Determining syllable structure and assigning stress to these structures are key components of natural language phonology, as they interact to describe rhythmic structure in observed data.

2. The approach of using a Dynamic Computational Network attempts to pare down both the representations **and** the rules used in modeling these phenomena.

    - Phonemes are represented as nodes in a neural network for syllabification; syllables are represented as nodes in a neural network for stress assignment. Adjacent nodes have weighted connections, and the activation value of a node is determined by linear functions of these weighted connections and other exogenous factors (such as position).
    - Syllable structure and stress assignment amounts to identifying local and global maxima among these nodes
    - The two networks (syllabification and accent) have the same basic architecture: this means we can propose a 'novel' architecture, which consists of the two networks where they are connected in some way. Connecting the networks may be one way to address the problem of quantity-sensitive systems.

3. My contributions will ideally be the following:

    - Develop an architecture which combines syllable and stress networks
    - Apply this architecture to quantity-sensitive stress systems – English at first
    - Modify and/or develop learning algorithms for this network
    - Describe important mathematical properties of the architecture

2 Introduction

This proposed dissertation will be a study of accent and syllabification, primarily from the standpoint of Dynamic Computational Networks (DCNs). As I will outline below, DCNs are neural networks which do not make many assumptions, and it is for this reason that I find them most interesting in trying to account for syllabification, accent and their interaction in quantity-sensitive accent systems. The basic structures of the networks and the constraints on how they perform computations are simple and explicit, but with them we have a great deal of flexibility, which gives rise to the possibility of accounting for both the straightforward and more complex patterns we see in natural language. Although this possibility is exciting, what is more exciting (and definitely more important) is the fact that DCNs allow us to ask specific, concrete questions about what they can and can not account for; in addition, they are amenable to being learned — i.e. we can use data to find values for the parameters that will optimize some criterion of well-formedness. With this, my dissertation will ask the following questions, which fall into a few broad categories: theory, empirical coverage, learnability and methodology.

1. Theory and empirical coverage

   - What inventory of parameters is necessary for the neural network? Goldsmith and Larson have proposed a handful for quantity-insensitive systems, but are these sufficient?
   - How can we model syllabification and accent at the lexical level in quantity-sensitive systems? Specifically, how can we do this for American English? How should we connect the neural network which derives syllable structure and the network which derives accent pattern? And, how does information flow between the two systems when they are connected?
   - Can, and should, this model be generalized to a continuous wave-based theory?

2. Learnability

   - Is simulated annealing the only viable way to learn parameters in the model? This assumes no probability distribution over the data, but if we can assume some distribution over the data, could we then use (a) version(s) of stochastic gradient descent to learn parameters?
   - Can we learn the inherent sonority of phonological segments from the data?

3. Methodology

   - How much data, and what type of data, is necessary for adequate learning of the parameters?
   - How should we train, test and assess the model? Are paradigms such as cross-validation a meaningful way of doing this when we care about potentially make claims of this model's psychological/cognitive plausibility?

These questions will be answered in the dissertation, but initial thoughts and hypotheses will be discussed below. I will first introduce the basic theory behind DCNs; then, I will discuss how to extend this theory to have some account for quantity-sensitive systems; this will lead to a brief discussion on learning of the parameters in the model; last, we will discuss some methodological matters and show briefly some of the basic work already done.

3 Dynamic Computational Networks (Goldsmith and Larson, early 1990s)

3.1 Overview

Goldsmith and Larson use dynamic computational networks as a theory of accent and syllabification, and, at its core, that is exactly what it is; however, it is much more. On the level of the data, dynamic computational networks (DCNs) offer an architecture which appears to make strides in capturing much of the subtlety and language-specific departures from empirically strong principles such as the Sonority Sequencing Principle (Generalization) and Onset Maximization. On a deeper level, dynamic computational networks are a brilliant effort to integrate two notions which have continually come in and out of fashion in phonological theory – rules and representations – with the innovative geometrical structures from influential theories (such as Metrical Theory) and notions of rules and constraints from others (Derivational Phonology) being brought together, where in the past it seemed to be that either one or the other was emphasized. Popping up even another level, this framework hopes to be a starting point to one of the most interesting – yet, in my opinion, down-played – questions pertaining to natural language in general: 'What kind of [. . . neural . . .] hardware would be good at performing that [metrical systems] kind of computation?' (Goldsmith, 2016). Despite its non-canonical and mathematical nature with regard to other phonological theories of accent and syllabification, this does not imply it is complicated; in fact, in many ways it involves much less. The following material in this section will be concerned with explaining the structure, assumptions and consequences of the model and then showing the model in (some) action.

3.2 The model

The view of syllabification and accent described above assumes that these two phenomena have the same organizing principles, which means they can be modeled with the same structure albeit with different settings of the parameters in the structure (Goldsmith, 1992). The structure of the network for both of them is as follows: a sequence of units, which are symbolic of segments in the process of syllabification and which are symbolic of syllables in the process of accent; and a pair of parameters α, β, which are weighted connections each unit has to its right and left neighbor respectively. This is seen in figure 1. In addition to this structure, we have a set of parameters, all of which can be learned and/or stipulated on a language-to-language basis: the connection weights α, β; internal activations x_i for the unit in position i; positional activations p_i for position i (independent of unit); and biases b_i for the unit in position i.

$$\begin{tikzcd}
x_0 + b_0 \arrow[r, bend left, "\beta"] & x_1 + b_1 \arrow[l, bend left, "\alpha"] \arrow[r, bend left, "\beta"] & x_2 + b_2 \arrow[l, bend left, "\alpha"] \arrow[r, bend left, "\beta"] & x_3 + b_3 \arrow[l, bend left, "\alpha"] \arrow[r, bend left, "\beta"] & x_4 + b_4 \arrow[l, bend left, "\alpha"] \arrow[r, bend left, "\beta"] & x_5 + b_5 \arrow[l, bend left, "\alpha"] \\
p_0 \arrow[u] & \cdots & \cdots & \cdots & \cdots & p_5 \arrow[u]
\end{tikzcd}$$

Figure 1: The dynamic computational network used in syllabification and accent. The nodes represent a phonological segment or syllable; the $x_i + b_i$'s represent a particular node's inherent activation and bias (which is determined by phonetic/phonological properties of the segment); α, β are left-and-righward arrows from the nodes; arrows into the nodes represent activation associated with a particular position in the sequence.

We can summarize the model as such:

(1) • Units of activation	PHONOLOGICAL SEGMENTS OR SYLLABLES
• Parameters:	
(i) α, β	RIGHT-LEFTWARD EXCITATION/INHIBITION
(ii) x_i	UNIT INHERENT ACTIVATION
(iii) b_i	UNIT BIAS
(iv) p_i	POSITIONAL ACTIVATION/BIAS

Now, we will go into more detail of each aspect.

3.2.1 Units of activation

The units of activation are simply a sequence of nodes. Traditionally, the sequence of nodes for the network responsible for syllabification corresponds to the skeletal tier, consisting of phonological segments, and the sequence of nodes for the accent network corresponds to the timing (metrical) tier, consisting of syllables for which accent can be assigned. Each one of these nodes is represented by a real-valued number, called its activation value, which we will later see is determined by a dynamic process involving several other objects in the model. The activation value has different, yet familiar, interpretations for each network. For the syllabification network, we think of these values as the 'derived' sonority of a segment, or we can say it is the current sonority of a segment, as we will see that sonority is not a rigid notion in this framework. In the accent network, we can think of the activation values as the 'derived' weight of the node (and in some cases a single node may correspond to a syllable). The activation values will ultimately determine the syllable and accent contours assigned to a sequence of phonological segments.

v_0
v_1
v_2
v_3
v_4
v_5

Figure 2: Units of activation in a dynamic computational network. Each $v_i \in \mathbb{R}$

3.2.2 Inherent activation and bias

The inherent activations x_i and biases b_i are real-valued numbers which factor into every computation of a unit's activation; in other words, we can think of the inherent activation x_i and bias b_i as input to a function which calculates the activation value of a unit i. Traditionally, the inherent activation of a unit has a close connection to a segment's sonority, as conceived of in previous linguistic literature; however, the bias of a unit has no clear connection to any particular theoretical construct in previous literature. The bias is an important aspect, though, when we start to think about quantity-sensitive stress systems. Although we will separate them here, we can think of the positional activations p_i (described below) and the biases b_i as being related and possibly even equivalent in some circumstances.

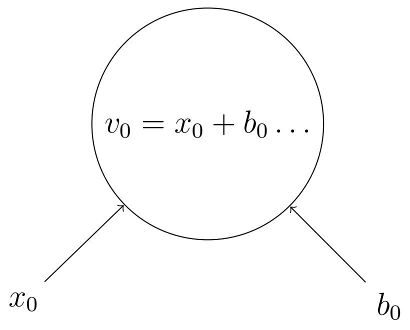


Figure 3: Inherent activation and bias of a node. The activation value of a node is partially determined by the inherent activation and bias of a segment or syllable which occupy the position.

3.2.3 Positional activation

Positional activations p_i are exactly as they are named: they are real-valued numbers which contribute position specific activation to a unit's total activation value. These p_i are independent of the linguistic content in position i; this is different than an inherent activation x_i and bias b_i, which are functions of the linguistic content to which the node in position i corresponds.

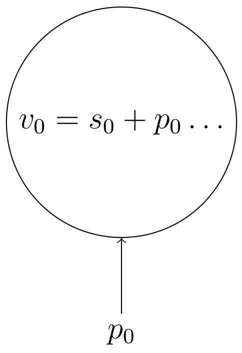


Figure 4: Positional activation of a node. This value is a position specific activation contributed to a node's activation value. $s_0 = x_0 + b_0$.

3.2.4 Lateral activation/inhibition, α, β

The objects which mediate the local – and, to some extent, determine the global – interactions of the units are the α and β parameters, which assign a real-valued weight in the interval $[0, 1]$ to a node's connection to its right and left neighbor respectively. We will see that these parameters bear a lot of responsibility in capturing language to language differences and certain contextual phenomena in accent and syllabification. For example, a positive value for α will indicate that the rightward neighbors of a unit will have an excitatory/positive contribution to the activation value of that node; a negative value for β would mean that the left neighbors would have an inhibitory/negative contribution to a unit's activation value.

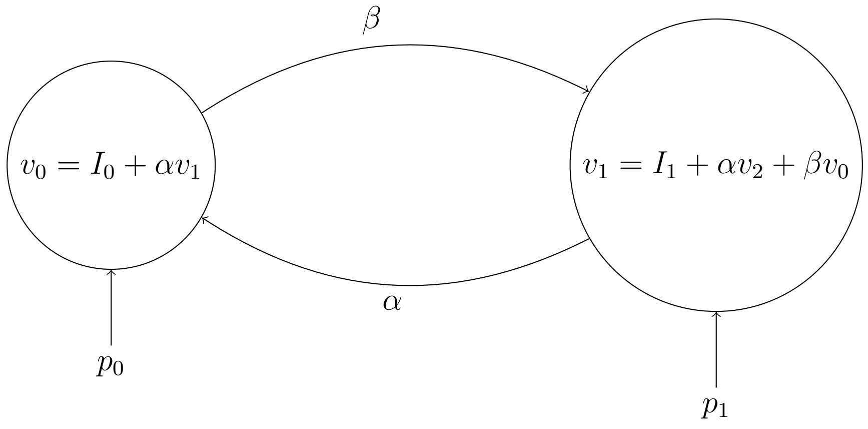


Figure 5: Lateral excitation and inhibition mediated by α and β. The α parameter determines the proportion of the right neighbor's activation to contribute to the calculation of the current node; the β parameter does the same for the left neighbor. Note that the first node does not receive left activation and the last node does not receive right activation. $I_i = x_i + b_i + p_i$.

3.3 Constraints and (further) assumptions

At a high level, the syllable structure and accent pattern will be obtained by a series of computations, each changing the activation values of the sequence of units, and these computations will stop when there is no significant improvement in the well-formedness of the sequence of units from that sequence of units at the previous step. This is known as harmonic application (Larson and Goldsmith, 1992). In Larson (1993), the criterion for convergence was that if a single node's activation value at time step t was not greater than its activation value at time step $t-1$ for some prespecified value $\delta > 0$, then computations would stop, as the system would be considered in equilibrium; in other words, if $|v_t - v_{t-1}| < \delta$, then the system has converged and stop computation. I will follow the same protocol in this proposal. The predicted surface structure is then determined by identifying local maxima and minima among the sequence of nodes, possibly subject to some type of thresholding. Thus, we have a gradient computation of the candidate surface structure, and we have a categorical determination of it: this idea is not far from tradition. Let us now turn to the details of the computations.

3.3.1 Computation of activation values

The computation of the activation values is a central part to the framework; fortunately, the assumptions made on this front are straightforward. First, activation values are real numbers; second, the calculations are recursive; and last, at any given step (iteration), the calculations are linear in the α and β parameters. The activation value of a unit v_i at a given point in time t is the sum of the following values: the unit's inherent activation and bias, the positional activation, the activation of unit v_{i+1} at the previous step $t-1$ multiplied by α and the activation of unit v_{i-1} at the previous step $t-1$ multiplied by β. With our notation, this is $v_i^t = x_i + b_i + p_i + \alpha \cdot v_{i+1}^{t-1} + \beta \cdot v_{i-1}^{t-1}$. From this, we see that the values change over time, meaning there is a notion of initial and derived activation values.[1] In the context of syllabification, this implies two notions of sonority: *inherent* and *derived* sonority. We see this below in example (2). Additionally, with the syllabification example, activations from the segment's position and segment's neighbors contribute to contextual effects on a segment's sonority.

(2) a. Computation in notation: $v_i^t = x_i + b_i + p_i + \alpha \cdot v_{i+1}^{t-1} + \beta \cdot v_{i-1}^{t-1}$

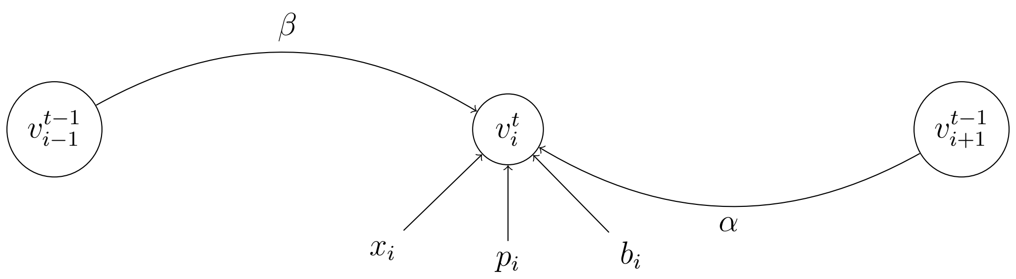
b. Computation in words:

node i current activation = inherent activations + positional bias + excitation/inhibition from left and right neighbors at previous step

c. Linguistic intuition (for syllabification):

derived sonority = inherent sonority + contextual influence due to specific position in sequence + local influence due to immediate neighbors

[1]However, note that the inherent activation, bias and positional bias do not change over time. We assume these are fixed upon entering computation.



$$v_i^t = x_i + b_i + p_i + \alpha v_{i+1}^{t-1} + \beta v_{i-1}^{t-1}$$

Figure 6: Computation of the unit i at time t. Arrows going into the node denote the values involved in the computation of v_i^t.

3.3.2 Determination of syllable structure and accent pattern

Determining the syllable structure of a sequence of segments and the accent pattern for a sequence of syllables is not involved: we identify local minima and maxima. For syllabification, a local maximum will correspond to a syllable nucleus while local minima will correspond to syllable onsets. Any segment in between a syllable nucleus and a syllable onset will imply it is part of a complex onset, and any segment between the nucleus of its syllable and the onset of a following syllable will correspond to what we have traditionally thought as the coda. For accent, a local maximum will correspond to a stressed syllable, and the prominence of the assigned stress is related to the derived activation value. Larson and Goldsmith (1992) found it useful to consider having a threshold in some cases: for example, a segment is only pronounced or stress is only assigned when the derived activation is greater than some threshold T. We will not use a threshold in determining syllable structure and accent contours in this proposal.

4 Quantity-sensitive systems

Quantity-sensitive stress systems present the interesting challenge of connecting the accounts of syllabification and accent in a language. In the terms of dynamic computational networks, this question becomes 'What is the relationship between the syllabification and accent networks?' There are a few (salient) options: (i) they exist separately and there are bidirectional weighted connections between them; (ii) they exist separately with weighted connections which are uni-directional; or (iii) the output for the syllabification network directly determines the basic architecture of the accent network. Each of these options will have different implications not only for the nature of the surface forms generated but also for the algorithm used to learn the values of the parameters. I will briefly sketch out each one of these options in turn.

4.1 Networks exist independently and have bidirectional connections

The option of having the syllabification and accent networks existing separately and with bidirectional connections imposes the least restrictions on the architecture of the combined network. This will mean that we can think of there being two layers of nodes where there are not only connections between nodes on the same layer (as we have seen above) but also connections between nodes in the syllable network and nodes in the accent network. A general snapshot of such a network would be that seen in figure 7.

accent network

$$\begin{tikzcd}
w_0 \arrow[r, leftrightarrow] \arrow[d, dotted, leftrightarrow] \arrow[dr, dotted, leftrightarrow] & w_1 \arrow[r, leftrightarrow] \arrow[d, dotted, leftrightarrow] \arrow[dr, dotted, leftrightarrow] \arrow[dl, dotted, leftrightarrow] & w_2 \arrow[r, leftrightarrow] \arrow[d, dotted, leftrightarrow] \arrow[dr, dotted, leftrightarrow] \arrow[dl, dotted, leftrightarrow] & w_3 \arrow[r, leftrightarrow] \arrow[d, dotted, leftrightarrow] \arrow[dr, dotted, leftrightarrow] \arrow[dl, dotted, leftrightarrow] & w_4 \arrow[r, leftrightarrow] \arrow[d, dotted, leftrightarrow] \arrow[dr, dotted, leftrightarrow] \arrow[dl, dotted, leftrightarrow] & w_5 \arrow[d, dotted, leftrightarrow] \arrow[dl, dotted, leftrightarrow] \\
v_0 \arrow[r, leftrightarrow] & v_1 \arrow[r, leftrightarrow] & v_2 \arrow[r, leftrightarrow] & v_3 \arrow[r, leftrightarrow] & v_4 \arrow[r, leftrightarrow] & v_5
\end{tikzcd}$$

Syllabification network

Figure 7: Two layer dynamic computational layer with bidirectional weights between the layers. This is an illustration of a connection pattern where the two layers are with bidirectional connections: the dotted lines indicate connections between the accent layer $w_i's$ and the syllabification layer v_i's. The solid lines indicate the within layer weighted connections described above. All possible connections between layers are not shown.

With this architecture, there are a few things which are not clear. It is first not clear how we should interpret the weighted connections of nodes between the layers. Although an intuitive interpretation may not be paramount in modeling these phenomena, one is preferable. Second, it is not clear how much locality between layers we should permit our architecture to have. With the layers, we have assumed a node has connections only with its immediate neighbors, which is a natural starting point, but I am not sure of any such 'natural' locality assumptions for nodes on different layers. The diagram above shows bidirectional connections from one node on a given layer to three nodes on the other layer (or to just two nodes for the case of edge nodes); however, this is just an illustration and not to be suggestive of the architecture which should be adopted. Last, given many current accounts of accent, it appears that accent is a property of syllables, and this seems to suggest that the structure of the accent network is dependent upon the output of the syllabification network

– i.e. the number of nodes in the accent network should be the number of syllables derived by the syllabification network.

The computations activations would change as well. An activation value v_i for a given unit i at a step t would not only depend on the activation of its neighbors $i-1$ and $i+1$ at time t (in addition to its inherent activation, bias and positional activation), it would also depend on the activation from (a) node(s) on the other layer. For example, we would have the following equation for a node i on the syllabification network: $v_i^t = x_i + b_i + p_i + \alpha \cdot v_{i+1}^{t-1} + \beta \cdot v_{i-1}^{t+1} + \sum_j \gamma_j w_j^{t-1}$, where the summation over w_j indicates the neighbors on the accent network to which the node i is connected. We can write the set of these neighbors of node i as $N(i)$.

(3) a. <u>Computation in notation</u>:

$$v_i^t = x_i + b_i + p_i + \alpha \cdot v_{i+1}^{t-1} + \beta \cdot v_{i-1}^{t-1} + \sum_{j \in N(i)} \gamma_j \cdot w_j^{t-1}$$

$$w_i^t = x_i + b_i + p_i + \alpha \cdot w_{i+1}^{t-1} + \beta \cdot w_{i-1}^{t-1} + \sum_{j \in N(i)} \gamma_j \cdot v_j^{t-1}$$

b. <u>Computation in words</u>:

node i current activation = inherent activations + positional bias + excitation/inhibition from left and right neighbors at previous step + excitation/inhibition from nodes on other layer

c. <u>Linguistic intuition (for syllabification)</u>:

derived sonority = inherent sonority + contextual influence due to specific position in sequence + local influence due to immediate neighbors + interaction with accent system

4.2 Networks exist independently and have unidirectional connections

The fundamental assumption underlying this option is the same as section 4.1: the syllabification and accent layers exist independently of one another. The difference is simple: instead of having connections between the layers which go in both directions, the connections are (consistently) directed in only one way. Loosely speaking, this would mean phonological information from the different layers flows in only one direction – either from the accent system to the syllabification system or vice versa. The latter flow of information is shown below in figure 8.

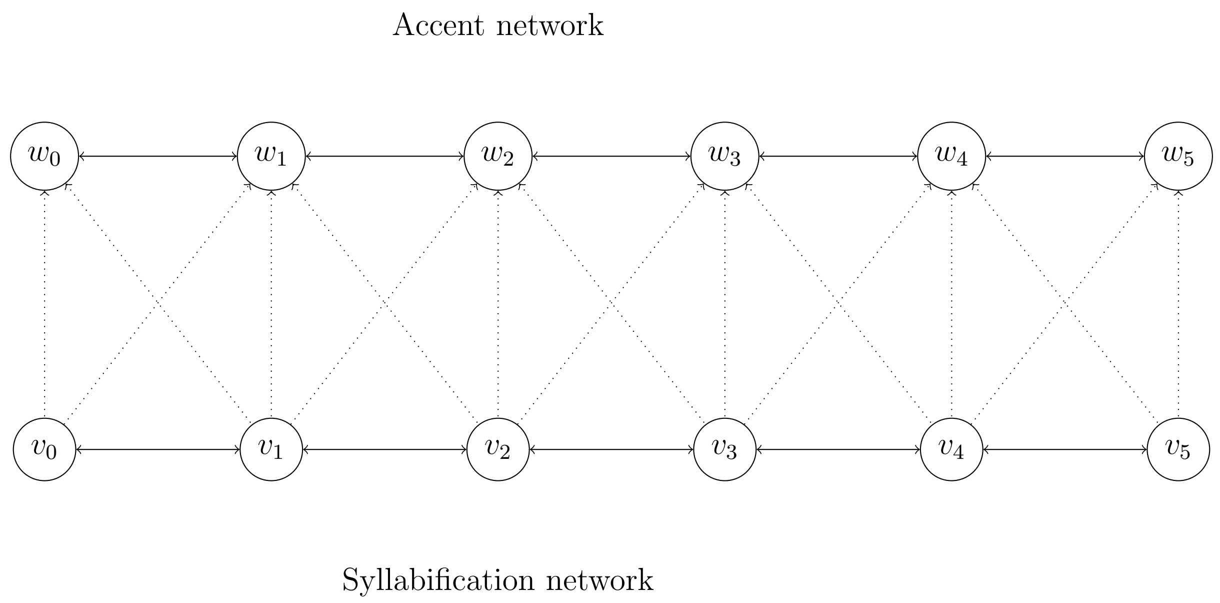


Figure 8: Two layer dynamic computational layer with unidirectional weights between the layers. This illustrates an architecture where activation values from the syllabification network is used in the computation of the activation values in the accent network. This is not, however, representative of all the possible connections between the layers.

The computations of the activation values for a node only change slightly from those mentioned above in (3). As an example, we will take the case where there are connections from the syllabification network to the accent network. Nodes in the accent network would not contribute any to the computation of the activation value for a node in the syllabification network, implying that the computation would just the be same as the original formulation of the network; however, activation values for nodes on the accent network would have a contribution from (a) node(s) on the syllabification network, as seen in the previous section. For this example, we would have the computations shown in (4).

(4) Computation in notation:

$$v_i^t = x_i + b_i + p_i + \alpha \cdot v_{i+1}^{t-1} + \beta \cdot v_{i-1}^{t-1}$$
$$w_i^t = x_i + b_i + p_i + \alpha \cdot w_{i+1}^{t-1} + \beta \cdot w_{i-1}^{t-1} + \sum_{j \in N(i)} \gamma_i \cdot v_j^{t-1}$$

One interesting note about this option is that it has some resemblance of a feedforward neural network. Although the layers have recursive computations involved, there is a defined direction between the layers. Thinking ahead, this could lend itself to learning the between layer parameters by common algorithms such as gradient descent in backpropagation.

4.3 Output of syllabification determines structure of accent layer

Perhaps the most intuitive linguistically speaking is the option where the output of the syllabification network directly determines the structure and some properties of the accent network. Under this view, the syllabification process will result in a sequence of segments that are syllabified, and the derived amount of syllables will determine the amount of nodes in the accent layer. We could take it a step further and say that the inherent activation values of the nodes on the accent network, which now represent syllables, are a function of the derived activation values, or ‘derived sonority’, of the segments in the syllabification layer which comprise the syllable. This additional assumption makes this approach similar to the one just above in section 4.2. Schematically, this option looks like figure 9. Green nodes correspond to segments in the first syllable and orange nodes correspond to those in the second syllable, and the colored arrows indicate any contribution they may have to the activation value of a node on the accent network.

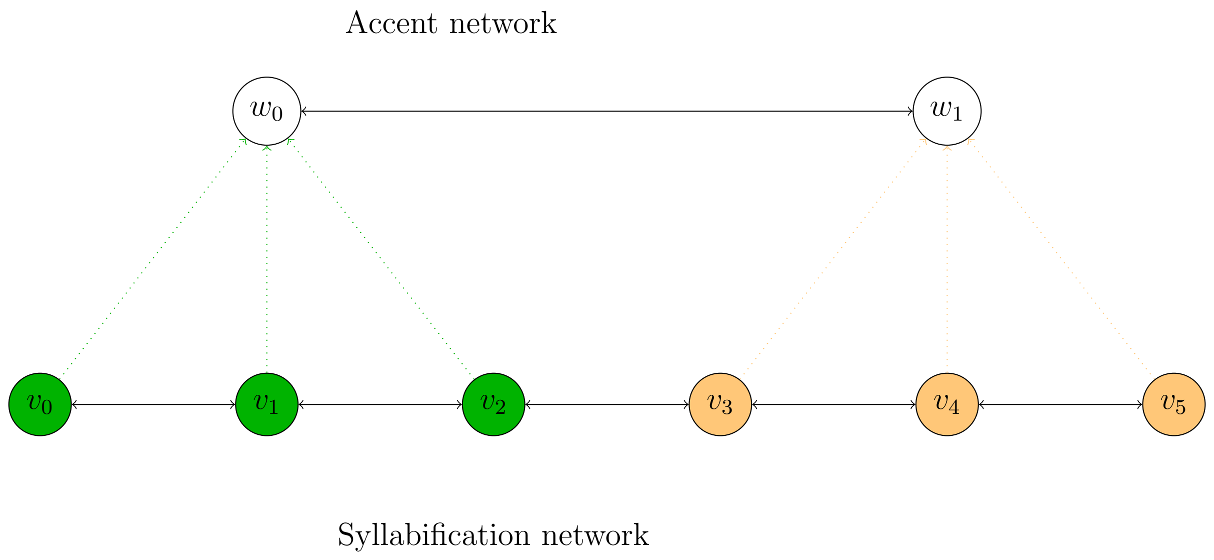


Figure 9: Syllabification network determines the structure and some properties of the accent network. The syllabification network first determines the syllable structure, which determines the amount of nodes in the accent network and possibly determines part of the activation values of the nodes on the accent layer.

Although the picture may not seem significantly different than the option discussed above in section 4.2, these options are not the same. First, we see in the option above that the syllabification network has no influence on the structure of the accent layer; in other words, we must make some assumption about the number of nodes on that layer in the previous option whereas the number here is ‘derived’ and is the number of syllable nuclei in the sequence of segments. Second, the computation for the activation values for nodes in the accent network is not the same: if these values depend on activation values from the syllabification network at all, they only depend on the value at the last step. Given that there is some contribution from the syllabification network to the accent network, this means a step in the computation of the activation value on the accent network consists of the inherent activation (which is the activation from the syllabification network at the last step), the bias, positional activation and excitatory/inhibitory activation from its neighbors.

The equation is then $w_i^t = x_i + b_i + p_i + \alpha \cdot w_{i-1}^{t-1} + \beta \cdot w_{i+1}^{t-1}$ which is equal to $w_i^t = \big(\sum_j \gamma_j \cdot v_j^{final}\big) + b_i + p_i + \alpha \cdot w_{i-1}^{t-1} + \beta \cdot w_{i+1}^{t-1}$, since $x_i = \sum_j \gamma_j \cdot v_j^{final}$.

(5) a. <u>Computation in notation</u>: $w_i^t = \left(\sum_{j \in N(i)} \gamma_j \cdot v_j^{final}\right) + b_i + p_i + \alpha \cdot w_{i+1}^{t-1} + \beta \cdot w_{i-1}^{t-1}$

b. <u>Computation in words</u>:

accent node i current activation = final activation from nodes on syllabification layer + positional bias + excitation/inhibition from left and right neighbors at previous step

c. <u>Linguistic intuition (for accent)</u>:

prominence = prominence of syllable (syllable weight) + contextual influence due to specific position in sequence + local influence due to immediate neighbors

This option is suggestive of a more modular architecture, suggesting that we can 'unhook' one component from the other; so, we can think of the syllabification network as generating the input for the accent network, and we are not committed to positing any complex interaction between the two layers. Modeling syllabification and accent this way has implications for how we learn the parameters. Under this this option, we can first learn the parameters for the syllabification network to generate input for the accent network, and then we can learn the parameters for the accent network. The organization of the network exhibited here, where there is clear separation of the components, is possibly the most similar to traditional approaches in the field.

5 Learning of the parameters

5.1 Modified simulated annealing in Larson (1993)

Learning the connection weights α, β in the model described in section 3 had been done by a modified version of the simulated annealing algorithm in Larson (1993). The basic idea behind simulated annealing is that in order to learn the parameters, you take a random walk through a space which the parameters α, β span until you find a point in that space, call it $(\hat{\alpha}, \hat{\beta})$, that optimizes the well-formedness of the output of the network. Since there are only two parameters α, β in this case, the space is two-dimensional.[2] This walk through the two-dimensional space of potential α's and β's is modulated by a temperature τ: the higher the temperature the more likely the next step in the random walk will be larger. The key to the algorithm is how the temperature τ relates to well-formedness: the algorithm produces an increase in temperature when the network derives output which is not well-formed, and it produces a decrease in temperature when the network derives output which is well-formed. We will look at an example for concreteness. Consider a network with just one layer that derives the syllabification of a sequence of phonological segments. The version of the simulated annealing algorithm in Larson (1993) goes as follows: first, present the network

[2]It turns out to be a subset of the plane, actually.

with a sequence of phonological segments and a label for the correct syllabification; if the network derives a syllabification which matches that of the label, lower the temperature by some constant $\Delta < 1$, such that $\tau_{new} = \Delta\tau_{old}$; if the network derives a syllabification which does not match that of the label, add some random value ε to α_{old} such that $\alpha_{new} = \alpha_{old} + \varepsilon$, do the same for β and increase the temperature τ by the magnitude of the change in α and β, so $\tau_{new} = \tau_{old} + \sqrt{(\alpha_{old} - \alpha_{new})^2 + (\beta_{old} - \beta_{new})^2}$. When the temperature 'cools' to a sufficient temperature T, the system is considered at equilibrium and changes in α, β stop and are said to have been learned. The algorithm is shown below in example (6).

(6) Modified simulated annealing algorithm

```
For every lexical item $w$ in training data:
    1. Present network with sequence of phonological segments $w$ to syllabify and a
       label for the correct syllabification
    2. Check if derived syllabification matches label for correct syllabification
    3. • If correct, $\tau_{new} = \Delta\tau_{old}$.
       • Else:
         $\alpha_{new} = \alpha_{old} + \varepsilon$
         $\beta_{new} = \beta_{old} + \varepsilon'$
         $\tau_{new} = \tau_{old} + \sqrt{(\alpha_{old} - \alpha_{new})^2 + (\beta_{old} - \beta_{new})^2}$
         where $\varepsilon, \varepsilon' \sim N(0, \tau_{old}^2) \times c$ and $c$ is a hyper-parameter to be tuned
    4. If $\tau_{new} < T$, stop; else, go back to step 1.
```

5.2 Beyond simulated annealing

The simulated annealing algorithm above does not assume any joint distribution $p(X, Y)$ over the data X (sequences of phonological segments) and labels Y, and this means that when learning the parameters, we are left with a lot of information that is not used: for example, in a sequence of phonological segments $x_1\, x_2\, x_3 \ldots x_n$, we know that the activation of x_1 will have a greater effect on the derived activation of x_2 and the dependence decreases for segments farther away from x_1. If we could reasonably assign a joint probability distribution to the data and labels, then we could estimate the parameters of the model by methods such as maximum likelihood estimation (MLE) or maximum a posteriori (MAP); additionally, we can possibly choose a model where estimation turns out to be a convex optimization problem. One way of assigning a probability distribution would be to model the two layers which correspond to syllabification and accent as a Markov Random Field (MRF). Other Markov models, such as Hidden Markov Models, are ubiquitous in natural language processing, and this generalization to undirected graphs, the type of object we will most likely be working with in this dissertation, is a promising starting point. They are especially attractive given the locality of the dependence we have so far assumed. Work on DCNs has not yet explored models of this kind, but the previous questions addressed above have correlates in the language of Markov Random Fields. For example, the question on the nature of the structure of the connections within and between layers are questions about conditional independence in MRFs. Work in this area should be exciting, and it will most likely be more telling with regard to the capability of the model, which is one of the main questions this dissertation will answer.

5.3 Learning inherent sonority: leveraging the linearity of the computations

One fundamental aspect of DCNs is the nature of their input, as the input has a large effect on providing the basic shape of the derived forms. Ignoring bias and positional activation for now, we **assume** the input to each node to be the 'inherent sonority' of the phonological segment; however, we could try to **learn** inherent sonorities in addition to the parameters α, β, etc. This comes from recognizing the fact that at every step in the computation of the derived activations, the resulting expression is linear in the input x_i . To see this, consider a three-node, single-layer network with connection weights α, β and inherent activations x_1, x_2 and x_3, again ignoring bias and positional activations for the sake of simplicity. Let v_i^t represent the derived activation of node i at step t in the computation. In the absence of a formal proof, we can be convinced of the fact that the derived activations will be linear in the x_i after observing just three steps in the computation — i.e. there will be no terms involving powers greater than 1, such as $x_i^2, x_i^3, \ldots$, at any step in the computation.

(7) Step t = 0:

$v_1^0 = x_1$
$v_2^0 = x_2$
$v_3^0 = x_3$

Step t = 1:

$v_1^1 = v_1^0 + \alpha v_2^0 = x_1 + \alpha x_2$
$v_2^1 = v_2^0 + \alpha v_3^0 + \beta v_1^0 = x_2 + \alpha x_1 + \beta x_3$
$v_3^1 = v_3^0 + \beta v_2^0 = x_3 + \beta x_2$

Step t = 2:

$v_1^2 = v_1^1 + \alpha v_2^1 = x_1 + \alpha(x_2 + \alpha x_1 + \beta x_3) = (\alpha^2 + 1)x_1 + \alpha x_2 + \alpha\beta x_3$
$v_2^2 = v_2^1 + \alpha v_3^1 + \beta v_1^1 = x_2 + \alpha(x_3 + \beta x_2) + \beta(x_1 + \alpha x_2) = (2\alpha\beta + 1)x_2 + \alpha x_3 + \beta x_1$
$v_3^2 = v_3^1 + \beta v_2^1 = x_3 + \beta(x_2 + \alpha x_1 + \beta x_3) = (\beta^2 + 1)x_3 + \beta x_2 + \alpha\beta x_1$

Step t = 3:

$\vdots$

Observing that at any step in the derivation the derived activations will be linear in the x_i, we can start to think about using one of the many methods for learning a model which is linear in its parameters. That is, we can now consider the situation where α, β are known, and estimate values of the parameters $x_1, x_2, \ldots, x_n$. However, we are stuck: we need to know the x_i to estimate α, β, and we need to know α, β to estimate the x_i, but in reality, we do not know α, β nor the x_i a priori. This problem is reminiscent of the one which is present in situations which involve using the Expectation-Maximization algorithm to estimate the parameters, and I think that a similar concept could be used here. Namely, we can first make initial guesses on the values of α, β, and then assume these parameters are known to learn the x_i; then, we can assume the newly learned x_i are known and learn new, better values for α, β and iterate this process until convergence. This could be one way to learn inherent sonority from the data. It is summarized below in (8).

(8) Step 1: Initialize α, β and assume known

Step 2: Learn inherent sonorities x_i, with α, β

Step 3: Assume learned values for inherent sonorities x_i are known, and learn values for α, β

Repeat until convergence

6 Methodology and current work

6.1 Methods and potential bottlenecks

The typical pipeline for work on this project will be as follows: (i) implement software to do the computations of Dynamic Computational Networks; (ii) collect labelled data for training of the model; (iii) process data; (iv) train the model on a subset of the labelled data and test the remainder; (v) assess the model's performance by test error on held-out data. Currently, I have developed the crucial computational scaffolding for the beginning stages of the project: specifically, I have implemented the main algorithms Larson (1993) has in his dissertation, and then I have implemented the quantity-sensitive architecture described in section 4.1, where the accent and stress layer exist independently of one another and they have bidirectional connections between them. I will show some of the output below.[3] The remaining work will consist of implementing the several alternative learning algorithms I have discussed above in section 5. Step (ii) is the most important, practically speaking, and will be the largest bottleneck for this project. Collecting labelled data for both syllabification and accent is not easy for many languages: it is for this reason that I will focus mostly on English when it comes to development of the theory and software. Other languages I may look at are Russian and Arabic, where natural language processing data resources may be more readily available. Again, steps (iii) – (v) have the computational scaffolding in place, and preliminary results have already been obtained. We will now turn to look at each one of these steps (iii) – (v) in more detail with respect to the current work I have done.

6.2 Data characteristics and processing

The dataset I am currently working with is the syllabified version of the CMU Pronouncing Dictionary version 0.6 provided by Bartlett et al. (2009).[4] They used a structured support

[3]I have not implemented Larson (1993)'s version of simulated annealing where he learns feature values for phonological segments as well. In the future, I could do this alongside what I have discussed above in section 5.3 and can compare the efficacy.

[4]Available at `https://webdocs.cs.ualberta.ca/~kondrak/cmudict.html`.

vector machine (SVM) approach to syllabify the CELEX dictionary with 98% accuracy, and they applied their algorithm to the CMU Pronouncing Dictionary to provide syllable boundaries, but did not provide labels for onset and rime or onset, nucleus and coda. Each dictionary entry consists of a sequence of phonemes in ARPAbet format (codes found in the Appendix) where the hyphens represent syllable boundaries. An example of an entry looks like the following, where the dictionary will sometimes acknowledge variants in pronunciation.

```
(9)  WARRIORS       W AO1 - R IY0 - ER0 Z
     WARRIORS(2)    W AO1 R - Y ER0 Z
```

To process this data, I made the simplification to consider all vowels to be nuclei of syllables and any phonological segment to the left as part of the onset and those segments to the right as part of the coda. This assumption holds for a majority of the lexical items in English, with the exceptions being syllabic consonants, such as the syllabic nasal in *button*; however, the CMU Pronouncing Dictionary appears to always transcribe syllabic consonants as a schwa + consonant combination. We should keep this in mind when doing a more detailed evaluation of the model, but for now it is an assumption which allows me to work with a dataset of considerable size. The assumption made here would mean for the two examples in (9) and the word *button*, I would provide the following syllabification:

```
(10) WARRIORS       W AO1 - R IY0 - ER0 Z     ONONNC
     WARRIORS(2)    W AO1 R - Y ER0 Z         ONCONC
     BUTTON         B AH1 - T AH0 N           ONONC
```

The labels I use for syllabification are those based on Larson (1993): namely, we label onset segments as U (left/increasing) and segments part of the rime as D (right/decreasing). He shows in his dissertation that this is equivalent to the onset-nuclues-coda (ONC) scheme as well as other potential schemes, which treat the complex onsets and codas in a more theoretically agnostic way.[5] The labels I use for accent are based on the annotations from the CMU Pronouncing Dictionary: only vowels are assumed to be stressed. Although the dictionary has levels 0, 1, and 2 for degrees of stress, I do not use these degrees in the labels and assume vowels of degree 1 and 2 are stressed while those of degree 0 are unstressed, as an initial simplification. This means each accent label will be a binary vector; however, this does not mean that the network can not account for secondary stress. It will hopefully be derived as a consequence of network computations. Last, as input to the syllabification network, I use a rough sketch of a sonority hierarchy for English, and as input to the accent network I use a vector of zeros. Both of these assumptions are subject to change in the future, but for now they give us preliminary results, allowing us to observe broad-level behavior of the network. The hierarchy for sonority I assume is in (11) and a typical data structure I have for a lexical item is in (12).

(11) vowels = 7 > semi-vowels = 6 > liquids = 5 > nasals = 4 > fricatives = 3 > affricates = 2 > stops = 1

[5]Specifically, those phonological segments which are not peaks (H) or troughs (L) of sonority would be considered as O (other), given that troughs are always considered part of the onset (an assumption consistent with the Maximal Onset Principle in previous literature): for example, in *basketball* we would have the label LHOLHOLHO.

(12)
```
raw data: WARRIORS(2), W AO1 R - Y ER0 Z
syllable structure: ONCONC
syllable label: UDDUDD = (0,1,1,0,1,1)
accent label: (0,1,0,0,1,0)
inherent sonority: (6,7,5,6,7,3)
inherent accent : (0,0,0,0,0,0)
```

6.3 Training the model and evaluation of performance

The routine for training the model to learn α and β is slightly different than that found in Larson (1993): I train the model and assess its performance by K-fold cross-validation. This paradigm for learning parameters of a model consists of the following steps: (i) partition the dataset (assumed to be a random sample) into K sets; (ii) for each $k = 1, \ldots, K$, train the model on data from sets other than k, so train on sets for $k' = 1, \ldots, K$ except for $k' = k$, to estimate the parameters α and β, and then test the model on data from set k and store the error obtained; (iii) average the errors on each test set to get an estimate for the model's performance. The routine is more succinctly described below in (13).

(13) K-fold cross-validation

- (i) Partition data set X into K parts (called 'folds'); $X = X_1 \cup \ldots \cup X_K$
- (ii) For k in $1, \ldots, K$:
  1. Train model on data $X_{-k} = X - X_k$ to get estimates $\hat{\alpha}_k, \hat{\beta}_k$ at fold k for α and β
  2. Test model using estimates $\hat{\alpha}_k, \hat{\beta}_k$ on data X_k
  3. Compute the error, call it $error_k$, on this training set
  4. Store $error_k$ and parameter estimates $\hat{\alpha}_k, \hat{\beta}_k$
- (iii) Compute the average error: $\overline{error} = \frac{1}{K}\sum_{k=1}^{K} error_k$

Now, I will explicitly outline how the training and evaluation was done, as these are topics which are flexible and amenable for change. To train the model, I followed Larson (1993) and used the modified version of simulated annealing mentioned above in section 5.1, so this means I did the following for each lexical item w in X_{-k} during fold k: (i) present w to a DCN with current parameters $\alpha_{current}$ and $\beta_{current}$ and run network to get predicted syllabification $\hat{S}(w)$; (ii) send the predicted syllabification $\hat{S}(w)$ and the actual syllabification $S(w)$ of w to a simulated annealing function, which consists of steps 2, 3 above in example (6); (iii) check to see if the temperature $\tau_{current}$ is below some threshold T, and if it is, stop the learning, making $\alpha_{current} = \hat{\alpha}_k$ and $\beta_{current} = \hat{\beta}_k$. Notice step (i) here is step 1, step (ii) here is steps 2 and 3, and step (iii) is step 4 in example (6) describing simulated annealing. The training described just above is summarized below.

(14) Training for fold k: for each w in X_{-k}

(i) Present w to DCN with parameters $\alpha_{current}$ and $\beta_{current}$ to get predicted syllabification $\hat{S}(w)$

(ii) $\alpha_{new}, \beta_{new}, \tau_{new} = \text{simulated\_annealing}(\hat{S}(w), S(w), \alpha_{current}, \beta_{current}, \tau_{current})$

(iii) If $\tau_{new} > T$, then go back to (i); else stop training and $\hat{\alpha}_k = \alpha_{new}$ and $\hat{\beta}_k = \beta_{new}$

Evaluating the errors made on the test set X_k follows a similar procedure except that the estimated (learned) parameters $\hat{\alpha}_k$ and $\hat{\beta}_k$ from training are used to get the predicted syllabification $\hat{S}(w)$, as we are testing the model based on these learned values of the parameters. The loss function I use a 0/1 loss, which means that if the system predicts **completely** the correct syllabification, it is counted as correct and otherwise it is counted as incorrect — i.e. there is no partial credit. It is common in many machine learning paradigms to relax this assumption to allow for a less harsh (and more easily optimizable) loss function, but I will not do so in this proposal. I will consider other loss functions in the future, but the preliminary results are those based on a basic version of the implementation in Larson (1993). Using a 0/1 loss function means that evaluating the error for fold k goes as follows for each word w in X_k: (i) present w to the DCN with parameters $\hat{\alpha}_k$ and $\hat{\beta}_k$ to get a predicted syllabification $\hat{S}(w)$ of w; (ii) compare predicted syllabification $\hat{S}(w)$ with the actual syllabification $S(w)$; (iii) if correct, move on to next lexical item and if incorrect, count it as an error.

(15) Testing for fold k: for each w in X_k

(i) Present w to DCN with parameters $\hat{\alpha}_k$ and $\hat{\beta}_k$ to get predicted syllabification $\hat{S}(w)$

(ii) Compare predicted syllabification $\hat{S}(w)$ to true syllabification $S(w)$

(iii) If $\hat{S}(w) = S(w)$, then correct; otherwise, count as error and move on to next w in X_k

Training and evaluation is done analogously for the case of a quantity-sensitive systems where you have two layers and are predicting syllabification **and** accent. The only difference is that we have the parameters γ_i which mediate the connections between layers, and these parameters are also learned by simulated annealing. We still work with a 0/1 loss function, in the spirit of Larson (1993), where the system must get **both** and the syllabification and accent correct or else the prediction will be considered a mistake. Here, using a 0/1 loss becomes even harsher, as we will see below in section 6.4.

6.4 Preliminary results

I have done basic runs of the DCN on a randomly chosen 3000 word subset of the CMU Pronouncing Dictionary for syllabification alone and then for syllabification together with accent; in other words, I used Goldsmith and Larson's network discussed in section 3.2 for syllabification and a version of the model I describe in section 4.1 for syllabification and accent jointly. I used the modified version of simulated annealing in Larson (1993), discussed in sections 5 and 6, to learn the parameters of the models, and I performed 5-fold cross validation to evaluate the accuracy of the models. Modeling just the syllabification of English, we see an average test error of 30% and for modeling syllabification and accent jointly we see an average test error of 80%. Now, I will discuss in more detail each scenario: predicting syllabification only and predicting jointly syllabification and accent.[6]

6.4.1 Predicting syllabification

For this experiment, I assumed no positional activation or inherent bias, so the only nonzero known parameters were the inherent activations, given by the sonority hierarchy in (11). Consistent with Larson (1993), I observed on average learned values of α and β to be around 0.20: they were in the intervals of $[0.11, 0.23]$ and $[0.11, 0.24]$ respectively. This resulted in an average of 30% error on each test set. Both α and β positive implies that the derived sonorities will always be greater than the inherent sonorities; α and β similar implies that respecting the inherent sonority wave, for the most part, is the optimal thing to do in order to syllabify as many words correctly as possible, given the current model.

(16) Relevant points for syllabification experiment:

x_i	INHERENT SONORITY GIVEN BY HIERARCHY
p_i	ASSUMED TO BE ZERO
b_i	ASSUMED TO BE ZERO
30%	AVERAGE TEST ERROR

Range of estimates:

$[0.11, 0.23]$	α
$[0.11, 0.24]$	β

The main problems this system had was with consonant clusters: specifically, it struggled with fricative-stop and stop-fricative clusters in onsets and codas, and it struggled with word-internal consonant clusters, where the network would frequently classify the coda of the previous syllable as part of the onset of the following. Although they may seem like separate problems on the surface, we can note that they come from the same underlying problem: a disruption of sonority sequencing. Example mistakes were *reminds*, *scraped*, *unlock* and *orlene*. Since it gets these words wrong, we should expect that it gets words with clusters where the sonority sequence is not violated correct. This is what we see, as words such as *limb* and *canvas* are correct. The derived (red) and inherent (blue) sonority waves for *reminds*, *unlock*, *limb* and *canvas* are seen below in the figures. We may hypothesize that giving negative positional activation to the word edges would remedy this problem, but it does not improve much. It corrects mistakes on the edges, such as *scraped* and *shocked*, but word-internal clusters remain incorrect, suggesting that a solution to address the more general underlying problem of violating sonority sequencing may be preferred. I will leave this to future work: these are just preliminary results.

[6]However, I will not discuss in great detail tuning of the hyper-parameters. Changing hyper-parameters such as the decay rate Δ for τ, threshold for convergence in simulated annealing T and threshold for the DCN itself δ, and the constant c which multiples the random noise ε in simulated annealing did not substantially change the accuracy for the purposes of this proposal: they just changed if, and how quickly, the program converged.

```
          Word:
          REMINDS

          Phon string:
          R,IY0,M,AY1,N,D,Z

          Syllabification:
          ONONCCC

          Inherent sonority (blue):
          [5. 7. 4. 7. 4. 1. 3.]

          Derived sonority (red):
          [6.70970277 7.97863005 5.70970277 7.97863005 4.24755733 1.73493914
           3.01611782]

          Actual syllable:
          [0. 1. 0. 1. 1. 1. 1.]

          Predicted syllable:
          [0. 1. 0. 1. 1. 0. 1.]

          (alpha, beta):
          (0.21,0.214)

Out[192]: array([5., 7., 4., 7., 4., 1., 3.])
```

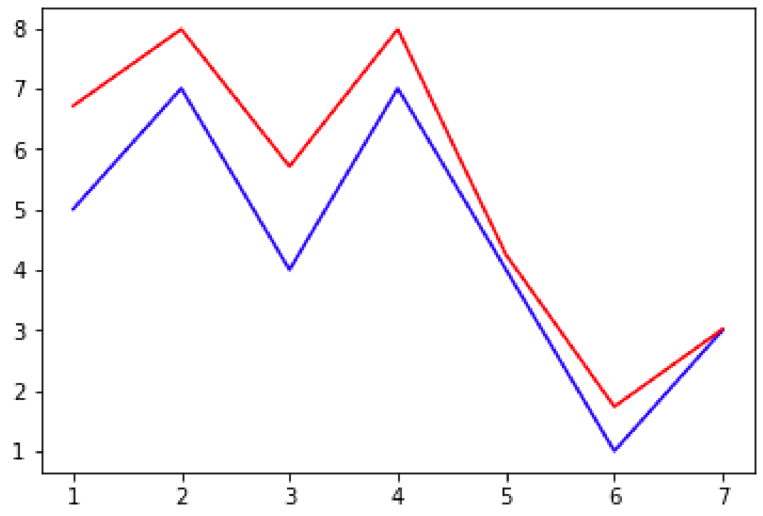


Figure 10: Derived (red) against inherent (blue) sonority for the word *reminds*. We see that the sonority of the word-final [z] should be lower than that of the [d] preceding it.

```
Word:
UNLOCK

Phon string:
AH0,N,L,AA1,K

Syllabification:
NCONC

Inherent sonority (blue):
[7. 4. 5. 7. 1.]

Derived sonority (red):
[8.37322106 5.6488526  6.77183204 7.27171301 1.11535939]

Actual syllable:
[1. 1. 0. 1. 1.]

Predicted syllable:
[1. 0. 0. 1. 1.]

(alpha, beta):
(0.21,0.214)

Out[194]: array([7., 4., 5., 7., 1.])
```

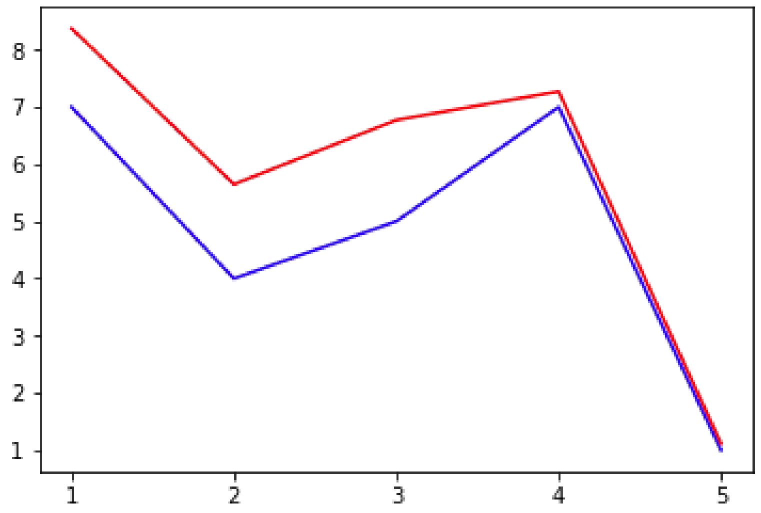


Figure 11: Derived (red) against inherent (blue) sonority for the word *unlock*. We see that the sonority of the [n] should be higher than that of the [l] following it.

```
Word:
LIMB

Phon string:
L,IH1,M

Syllabification:
ONC

Inherent sonority (blue):
[5. 7. 4.]

Derived sonority (red):
[6.94998781 8.00561236 4.12695865]

Actual syllable:
[0. 1. 1.]

Predicted syllable:
[0. 1. 1.]

(alpha, beta):
(0.21,0.214)

Out[202]: array([5., 7., 4.])
```

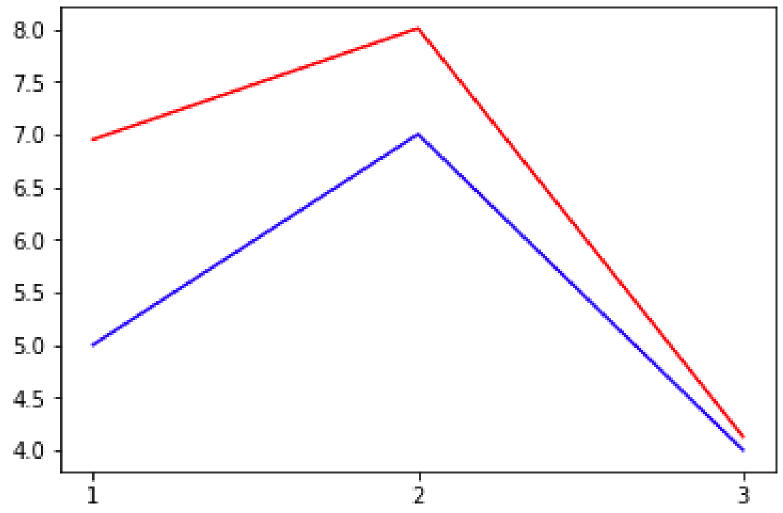


Figure 12: Derived (red) against inherent (blue) sonority for the word *limb*. We see the values of α and β respect the sonority sequencing.

```
         Word:
         CANVAS

         Phon string:
         K,AE1,N,V,AH0,S

         Syllabification:
         ONCONC

         Inherent sonority (blue):
         [1. 7. 4. 3. 7. 3.]

         Derived sonority (red):
         [2.98702193 8.25445054 5.19041408 4.89060255 7.76097925 3.12309223]

         Actual syllable:
         [0. 1. 1. 0. 1. 1.]

         Predicted syllable:
         [0. 1. 1. 0. 1. 1.]

         (alpha, beta):
         (0.21,0.214)
Out[204]: array([1., 7., 4., 3., 7., 3.])
```

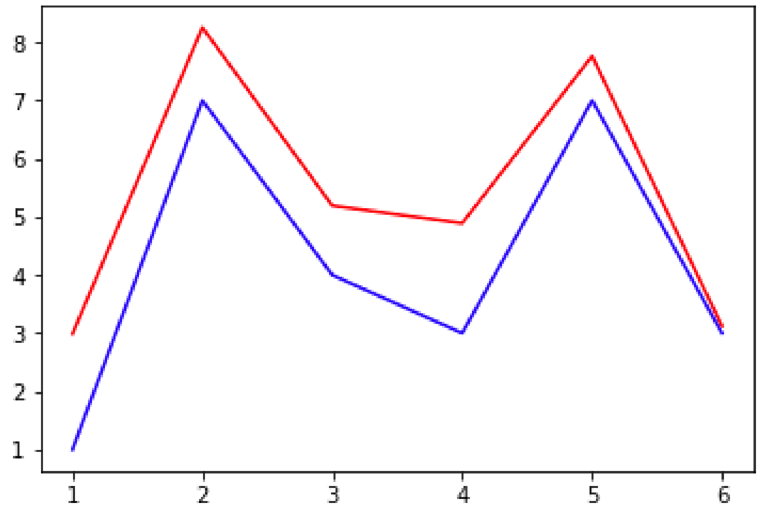


Figure 13: Derived (red) against inherent (blue) sonority for the word *canvas*. We see the values of α and β respect the sonority sequencing.

6.4.2 Predicting syllabification and accent jointly

For this experiment, I assumed the version of the network that has independent layers for syllabification and accent with bidirectional connections such that a node on a given layer is connected to the node on the other layer which is directly above/below it and to those nodes on the other layer which are the right and left neighbors of the node directly above it: the weights of these connections will be called γ_{direct} and $\gamma_{neighbors}$ below and in the output. I have also allowed for the flexibility of each layer having their own α and β values. As with the previous experiment, I assumed zero positional activation and zero bias, and I assumed the input to the accent layer was also zero: the only nonzero known values a priori were the inherent sonorities for the syllabification layer, which were given by the sonority hierarchy mentioned in (11). The architecture I just described is shown below in figure 14.

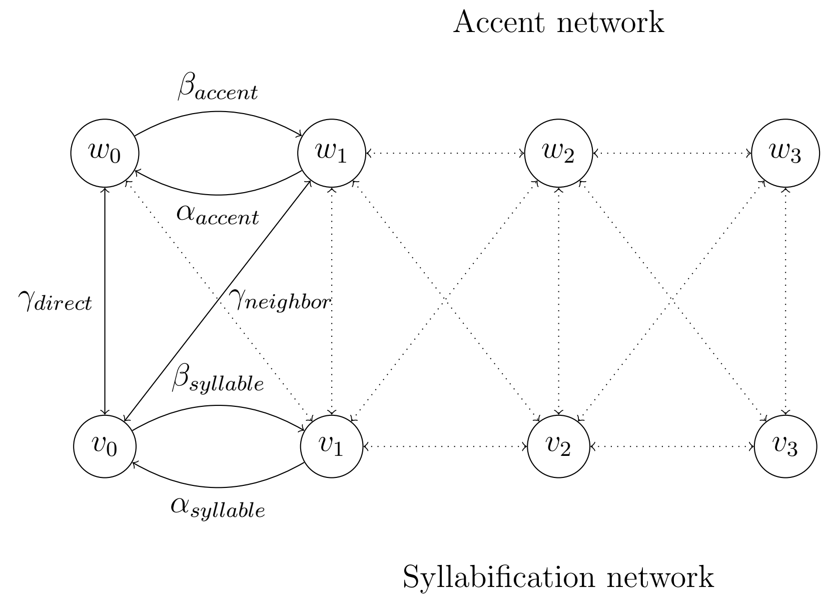


Figure 14: Two layer dynamic computational layer with bidirectional weights between the layers used in the experiment. Each layer will have its own value for α and β, but the layers will share the same connection weights γ_{direct} and $\gamma_{neighbors}$ which connect them. The bolded lines are just there to emphasize what each parameter is, but the pattern follows for the rest of the network.

A modified version of the simulated annealing discussed above in sections 5 and 6 was used in this experiment to incorporate the new parameters γ_{direct} and $\gamma_{neighbors}$ which connect the layers. I made the following changes to the algorithm: (i) it adds random value $\varepsilon_{direct}, \varepsilon_{neighbors} \sim N(0, \tau_{old}^2)$ to the parameters $\gamma_{direct}, \gamma_{neighbors}$ when a mistake is made on **either** the syllable or accent layer; (ii) it adds a random value to $\alpha_{syllable}$ and $\beta_{syllable}$ **only when** there is a mistake on the syllabification layer; and (iii) it adds a random value to α_{accent} and β_{accent} **only when** there is a mistake on the accent layer. The modified simulated annealing algorithm is discussed below in (17). All random values which are added to the parameters are normally distributed with mean 0 and variance τ_{old}^2 – i.e. $\varepsilon, \varepsilon' \sim N(0, \tau_{old}^2)$.

(17) Modified simulated annealing algorithm

```
For every lexical item $w$ in training data:
  1. Present network with sequence of phonological segments $w$ to syllabify and
     provide accent, along with a label for the correct syllabification and correct
     accent pattern
  2. Check if derived syllabification matches label for correct syllabification
  3. • If correct, continue.
     • Else:
       $\alpha^{syllable}_{new} = \alpha^{syllable}_{old} + \varepsilon$
       $\beta^{syllable}_{new} = \beta^{syllable}_{old} + \varepsilon'$
       $\Delta_{syllable} = \sqrt{(\alpha^{syllable}_{old} - \alpha^{syllable}_{new})^2 + (\beta^{syllable}_{old} - \beta^{syllable}_{new})^2}$
  4. Check if derived accent matches label for correct accent
  5. • If correct, continue.
     • Else:
       $\alpha^{accent}_{new} = \alpha^{accent}_{old} + \varepsilon$
       $\beta^{accent}_{new} = \beta^{accent}_{old} + \varepsilon'$
       $\Delta_{accent} = \sqrt{(\alpha^{accent}_{old} - \alpha^{accent}_{new})^2 + (\beta^{accent}_{old} - \beta^{accent}_{new})^2}$
  6. • If syllable and accent both correct, $\tau_{new} = \Delta\tau_{old}$
     • Else:
       $\gamma^{direct}_{new} = \gamma^{direct}_{old} + \varepsilon$
       $\gamma^{neighbors}_{new} = \gamma^{neighbors}_{new} + \varepsilon'$
       $\Delta_{between} = \sqrt{(\gamma^{direct}_{old} - \gamma^{direct}_{new})^2 + (\gamma^{neighbors}_{old} - \gamma^{neighbors}_{new})^2}$
       $\tau_{new} = \tau_{old} + \frac{1}{3}\left(\Delta_{between} + \Delta_{accent} + \Delta_{syllable}\right)$
  7. If $\tau_{new} < T$, stop; else, go back to step 1.
```

When using 0/1 loss, these assumptions do not lead to good performance with an average test error of 80%; however, often times the mistakes are not egregious. We see that the average predicted values for all parameters are mostly positive, implying that inhibition tended to provide worse predictions during training. The ranges for the values of α and β change, becoming a bit wider, as the computations here are less stable and have higher variance. The range for $\alpha_{syllable}$ is similar to that as before with it being [0.07, 0.23], but the range for $\beta_{syllable}$ seems to have shifted to the left a bit with it being [-0.04, 0.19]. The parameter γ_{direct} was greater than the parameter for $\gamma_{neighbors}$ on average, implying that connections which could be thought of as 'closer' in some sense are more important for these computations. This information is summarized below.

(18) <u>Relevant points for joint syllabification and accent experiment:</u>

x_i	INHERENT SONORITY GIVEN BY HIERARCHY
p_i	ASSUMED TO BE ZERO ON BOTH LAYERS
b_i	ASSUMED TO BE ZERO ON BOTH LAYERS
80%	AVERAGE TEST ERROR

<u>Range of estimates:</u>

$[0.07, 0.23]$	$\alpha_{syllable}$
$[-0.04, 0.19]$	$\beta_{syllable}$
$[0.03, 0.24]$	α_{accent}
$[0.02, 0.23]$	β_{acent}
$[0.14, 0.24]$	γ_{direct}
$[0.07, 0.25]$	$\gamma_{neighbors}$

Although the class of mistakes is much more diverse here, there are a few trends: the network still has trouble with syllabifying correctly consonant clusters; often times a peak for an accent will be off by one position; and words with multiple accent are hard to correctly assign a pattern to. As we will see below in the figures, some of this occurs as a consequence of the network architecture: since edge nodes have one less connection than interior nodes do between layers (as they do not have either a left or right neighbor), the accent typically accumulates towards the middle of a word, and it can lead to a formation of just one accent peak, failing to form a secondary one. For the correct words, I have found a couple small trends: the network seems to account for multiple accent in compound words better and it still performs reasonably well when sonority sequencing is respected. Examples of incorrect predictions are *shocks*, *diabetic* and *scorpio* while a couple of correct predictions are *sunbathe* and *Salvador*. Figures of these are seen below. I have omitted the printing of α's and β's.

```
Word:
SHOCKS

Phon string:
SH,AA1,K,S

Syllabification:
ONCC

Inherent sonority (blue):
[3. 7. 1. 3.]

Derived sonority (red):
[ 6.24907728 10.54890657  3.94304416  4.2356511 ]

Actual syllable:
[0. 1. 1. 1.]

Predicted syllable:
[0. 1. 0. 1.]

Derived accent (yellow):
[4.7972519  5.45228141 4.41294836 1.85557991]

Actual accent maxima:
[0. 1. 0. 0.]

Predicted accent maxima:
[0. 1. 0. 0.]

(gamma_direct, gamma_neighbors):
(0.222,0.212)

Out[349]: array([3., 7., 1., 3.])
```

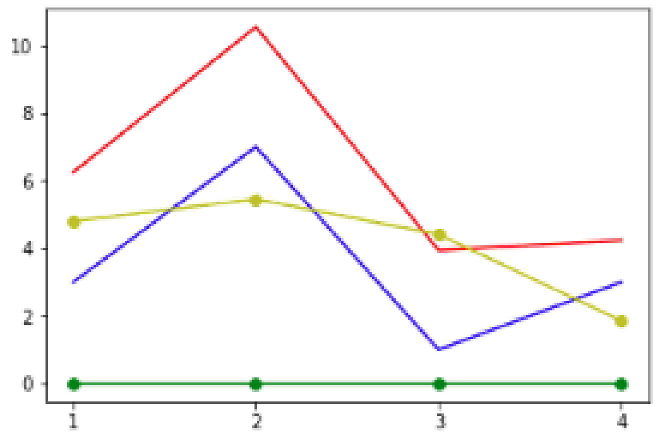
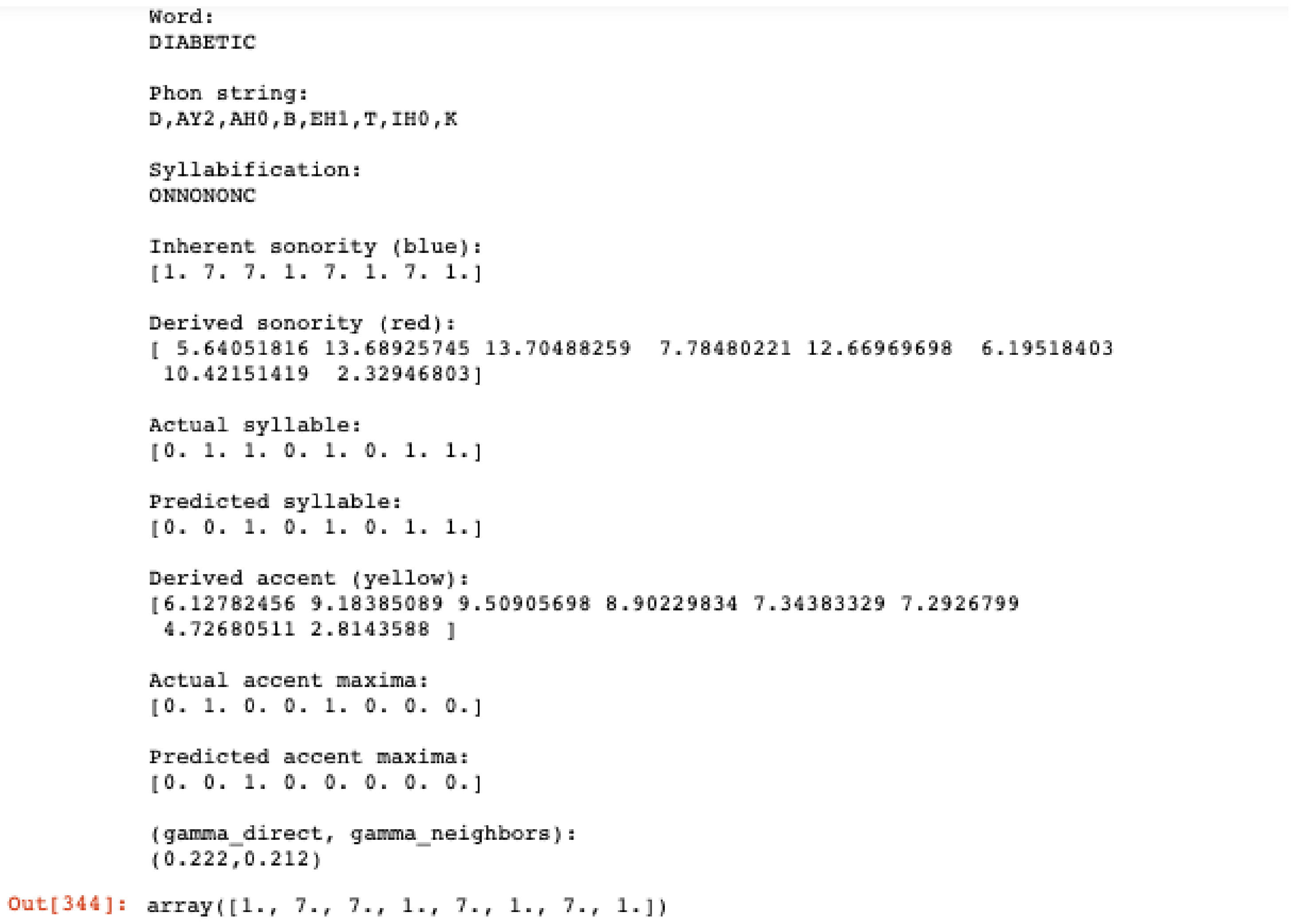


Figure 15: Derived (red) against inherent (blue) sonority and inherent (green) against derived (yellow) accent for the word *shocks*. The complex coda is not predicted correctly: the [s] should have lower derived sonority than the [k].



```
Word:
DIABETIC

Phon string:
D,AY2,AH0,B,EH1,T,IH0,K

Syllabification:
ONNONONC

Inherent sonority (blue):
[1. 7. 7. 1. 7. 1. 7. 1.]

Derived sonority (red):
[ 5.64051816 13.68925745 13.70488259  7.78480221 12.66969698  6.19518403
 10.42151419  2.32946803]

Actual syllable:
[0. 1. 1. 0. 1. 0. 1. 1.]

Predicted syllable:
[0. 0. 1. 0. 1. 0. 1. 1.]

Derived accent (yellow):
[6.12782456 9.18385089 9.50905698 8.90229834 7.34383329 7.2926799
 4.72680511 2.8143588 ]

Actual accent maxima:
[0. 1. 0. 0. 1. 0. 0. 0.]

Predicted accent maxima:
[0. 0. 1. 0. 0. 0. 0. 0.]

(gamma_direct, gamma_neighbors):
(0.222,0.212)

Out[344]: array([1., 7., 7., 1., 7., 1., 7., 1.])
```

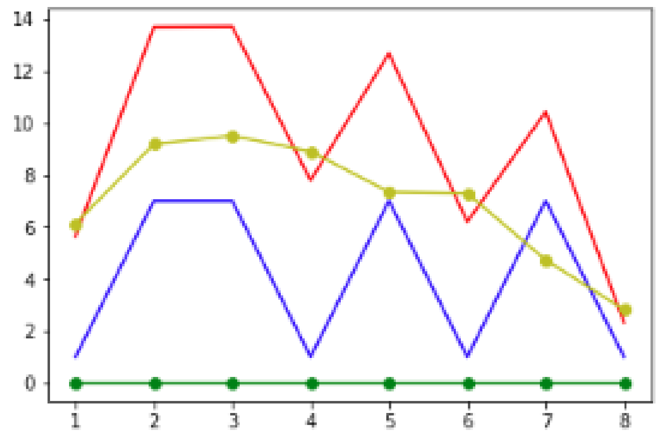


Figure 16: Derived (red) against inherent (blue) sonority and inherent (green) against derived (yellow) accent for the word *diabetic*. The primary accent was misplaced and the secondary accent was missed altogether.

```
Word:
SCORPIO

Phon string:
S,K,AO1,R,P,IY0,OW2

Syllabification:
OONCONN

Inherent sonority (blue):
[3. 1. 7. 5. 1. 7. 7.]

Derived sonority (red):
[ 6.24339251  6.76512219 13.45981312 11.22831222  7.4073845  12.27970459
  9.26066988]

Actual syllable:
[0. 0. 1. 1. 0. 1. 1.]

Predicted syllable:
[0. 0. 1. 1. 0. 1. 1.]

Derived accent (yellow):
[4.43911343 7.54370243 8.66920054 8.6821281  8.20147534 7.29399156
 4.79308236]

Actual accent maxima:
[0. 0. 1. 0. 0. 0. 1.]

Predicted accent maxima:
[0. 0. 0. 1. 0. 0. 0.]

(gamma_direct, gamma_neighbors):
(0.222,0.212)
```

```
Out[345]: array([3., 1., 7., 5., 1., 7., 7.])
```

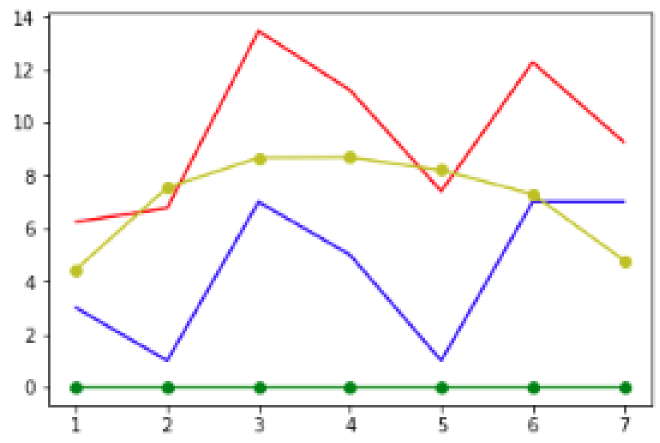


Figure 17: Derived (red) against inherent (blue) sonority and inherent (green) against derived (yellow) accent for the word *scorpio*. Here, primary accent was missed and secondary accent was misplaced. However, the complex onset was predicted correctly.

```
Word:
SUNBATHE

Phon string:
S,AH1,N,B,EY2,DH

Syllabification:
ONCONC

Inherent sonority (blue):
[3. 7. 4. 1. 7. 3.]

Derived sonority (red):
[4.13477078 7.96122107 4.6898168  2.19896718 7.71259599 2.95077403]

Actual syllable:
[0. 1. 1. 0. 1. 1.]

Predicted syllable:
[0. 1. 1. 0. 1. 1.]

Derived accent (yellow):
[1.51821332 2.02474549 1.64783587 1.50049845 1.65798838 1.27406044]

Actual accent maxima:
[0. 1. 0. 0. 1. 0.]

Predicted accent maxima:
[0. 1. 0. 0. 1. 0.]

(gamma_direct, gamma_neighbors):
(0.136,0.071)
```

```
Out[347]: array([3., 7., 4., 1., 7., 3.])
```

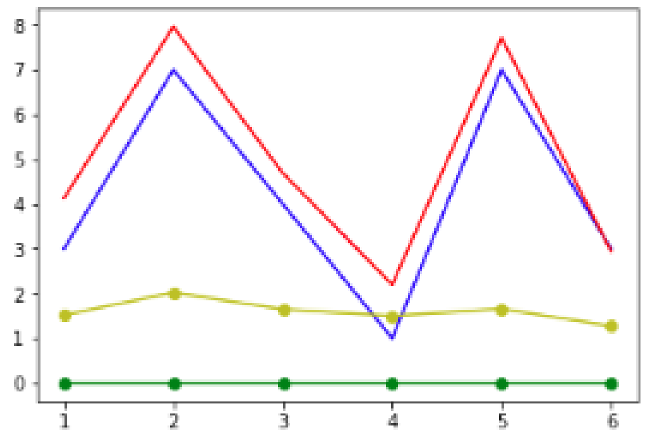
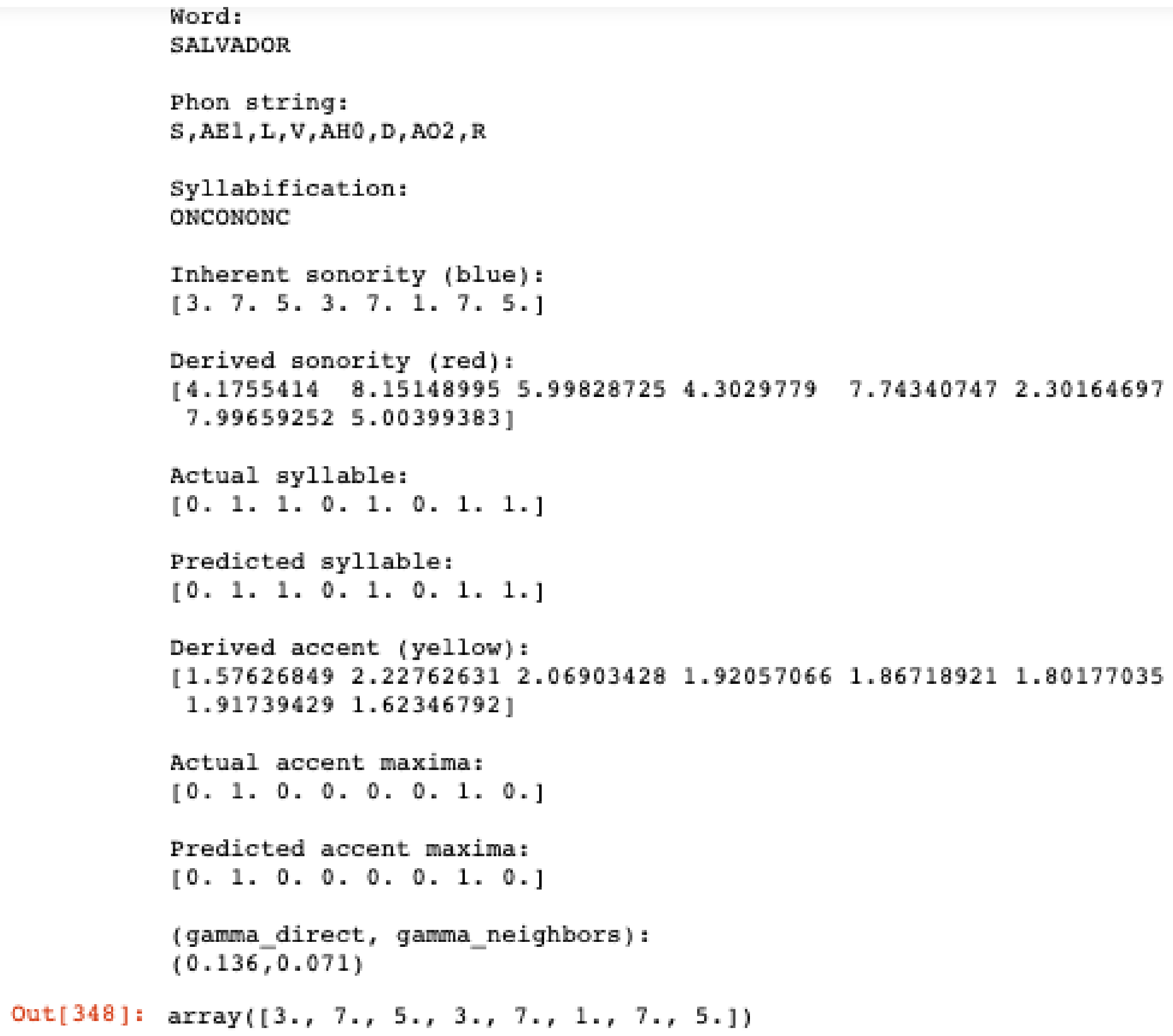


Figure 18: Derived (red) against inherent (blue) sonority and inherent (green) against derived (yellow) accent for the word *sunbathe*.



```
Word:
SALVADOR

Phon string:
S,AE1,L,V,AH0,D,AO2,R

Syllabification:
ONCONONC

Inherent sonority (blue):
[3. 7. 5. 3. 7. 1. 7. 5.]

Derived sonority (red):
[4.1755414  8.15148995 5.99828725 4.3029779  7.74340747 2.30164697
 7.99659252 5.00399383]

Actual syllable:
[0. 1. 1. 0. 1. 0. 1. 1.]

Predicted syllable:
[0. 1. 1. 0. 1. 0. 1. 1.]

Derived accent (yellow):
[1.57626849 2.22762631 2.06903428 1.92057066 1.86718921 1.80177035
 1.91739429 1.62346792]

Actual accent maxima:
[0. 1. 0. 0. 0. 0. 1. 0.]

Predicted accent maxima:
[0. 1. 0. 0. 0. 0. 1. 0.]

(gamma_direct, gamma_neighbors):
(0.136,0.071)
```

```
Out[348]: array([3., 7., 5., 3., 7., 1., 7., 5.])
```

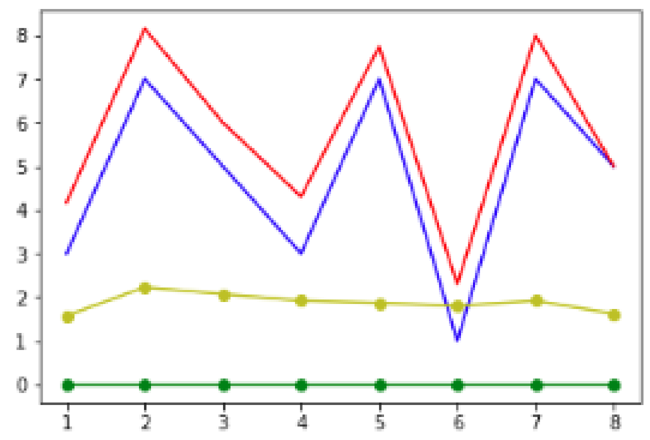


Figure 19: Derived (red) against inherent (blue) sonority and inherent (green) against derived (yellow) accent for the word *salvador*.

7 Discussion and conclusion

It is becoming clear that these basic versions of DCNs will not be sufficient models for English syllabification and for English syllabification and accent jointly; however, they do appear to be close, as the mistakes which are made are not egregious. Additionally, there are many open questions and next steps to take, and I will briefly discuss them here.

First, we can consider different architectures of the network. This would most likely entail adjusting our assumptions on the locality of the computations – i.e. which nodes are connected to which. For example, on the accent layer we could assume that nodes have connections to nodes which are two away from it, instead of just one. A change along these lines could help remedy the problem of the accent being off by one, as accent may be less likely to accumulate in the center of the word and more likely to alternate. As mentioned in section 5, we could also learn the values for positional activations and/or biases, which may help; or, we could stipulate these values. The former would be preferred, but second could more easily incorporate observations from previous analyses in the literature.

We could modify our learning algorithms as well. The 0/1 loss function used here and in Larson (1993) is unforgiving, and much of current machine learning literature makes use of approximations to 0/1 loss, as they are computationally easier to deal with. Also, if an appropriate loss function is chosen, such as optimizing the log-likelihood, it could lead to learning in a more intelligent way because it would change the values of parameters such as α, β and γ in a way that is not random. Currently, the parameters change by adding a random value to them, and this does not take much information into account. Last, we could consider not performing cross validation. Cross validation is a procedure which is useful for helping stave off any overfitting of the model: it helps the model generalize to new data. It is not clear that we want to ask our model to generalize to many new pieces of data. This would certainly be a great property to have, but it is not clear that it is a necessary property to have. Larson (1993) did not use cross-validation, and better results were obtained there.

Last, we could consider a probability model, such as that of a random Markov field (MRF). The model we are currently work with is deterministic, but we could consider a probabilistic model and then make predictions based on those configurations which maximize the probability. This has an interesting consequence for the interpretation. A model of this sort would not have the notion of derivation, which is a notion central to DCNs. However, it shares the interpretation of making predictions based on some optimal state: we can think of the nodes in DCNs now as reaching a point of equilibrium after being 'agitated' by inherent activations, and we could think of the nodes in a MRF network as being the optimal state of the system given the values of inherent activations.

References

Bartlett, S., Kondrak, G., and Cherry, C. (2009). On the syllabification of phonemes. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 308–316, Stroudsburg, PA, USA. Association for Computational Linguistics.

Goldsmith, J. (1991). A dynamic computational theory of accent systems. Lecture at the Conference on the Organization of Phonology at the University of Illinois Urbana. Available at `http://people.cs.uchicago.edu/~jagoldsm/DCN/Urbana.pdf`.

Goldsmith, J. (1992). Local modelling in phonology. In Davis, S., editor, *Connectionism: Theory and Practice*, chapter 7, pages 229–246. Oxford University Press.

Goldsmith, J. (1993). Harmonic phonology. In Goldsmith, J., editor, *The Last Phonological Rule: Reflections on Constraints and Derivations*, chapter 2, pages 21–60. University of Chicago Press.

Goldsmith, J. (2011). The syllable. In Goldsmith, J., Riggle, J., and Yu, A. C. L., editors, *The Handbook of Phonological Theory, Second Edition*, chapter 6. Blackwell Publishing Ltd.

Goldsmith, J. (2016). Dynamic computational networks. University Lecture. Available at `http://people.cs.uchicago.edu/~jagoldsm/slides/2016-dcn.pdf`.

Larson, G. (1993). *Dynamic Computational Networks and the Representation of Phonological Information*. University of Chicago, Department of Linguistics.

Larson, G. and Goldsmith, J. (1990). Local modeling and syllabification. In Ziolkowski, M., Noske, M., and Deaton, K., editors, *Papers from the 26th Annual Regional Meeting of the Chicago Linguistic Society: Parasession on the syllable in phonetics and phonology.*

Larson, G. and Goldsmith, J. (1992). Using networks in harmonic phonology. In Canakis, C., Chan, G., and Denton, J., editors, *Papers from the 28th Annual Meeting of the Chicago Linguistic Society.*

Larson, G. N. (1990). Local computational networks and the distribution of segments in the spanish syllable. In Ziolkowski, M., Noske, M., and Deaton, K., editors, *Papers from the 26th Annual Regional Meeting of the Chicago Linguistic Society: Parasession on the syllable in phonetics and phonology.*