OPEN ACCESS

Vocabulary Learning and Instruction

Volume 12, Number 1 (2023)
https://doi.org/10.29140/vli.v12n1.1076

Castledown

Vocabulary Networks Workshop 3: Activating Words in a Network (II)

Paul Meara
Swansea University, UK
p.m.meara@gmail.com

Imma Miralpeix
Universitat de Barcelona, Spain
imiralpeix@gmail.com

Abstract

This paper is part 3 of a series of workshops that examine the properties of some simple models vocabulary networks. This Workshop focusses on how the vocabulary network responds when words become easier to activate. The Workshop is linked to an on-line practice room where readers can explore these processes for themselves.

Introduction

In Part 2 of this Workshop, we looked at how a model vocabulary network reacts when inactive words are briefly activated. These simulations led us to conclude that networks with the characteristics described in Part 2 were surprisingly resistant to change. Turning ON even very large numbers of words generally does not appear to have a lasting effect on the overall activation level of the networks. One implication of this is that we need to rethink some common approaches to vocabulary acquisition in second languages. Specifically, exercises which just activate for a short time words that learners already know are not likely to make a lasting improvement to their active vocabulary. This conclusion is a more than a bit counter-intuitive, and it illustrates the power of simulations to make us question some widely held assumptions about the way vocabularies work.

In this Section, we look at a different way of stimulating activity in a vocabulary network.

Data Availability Statement: All relevant data are within this paper.

Another approach to activating a vocabulary network

You will recall that each of the words in our simple vocabulary networks has two inputs from other words in the network, and words can respond to these inputs in different ways. In the simulations in Part 1, we identified two different word types: AND words are difficult to activate – they only become activated if **both** of their inputs are already activated; OR words are easier to activate – they will become activated if **either one or both** of their inputs are already activated. In the simulations in Part 2, this characteristic was a fixed one that was determined at the time when the network was set up, and it was not affected by any subsequent events where we turned words ON. However, there is no intrinsic reason why this property should be a fixed one. Indeed, we could argue that making words easier to turn ON is a very plausible feature of real-world vocabulary networks. This suggests that it would be worth us looking at networks where the way a word responds to its inputs is not a fixed characteristic, but one that can be allowed to vary. Specifically, we can ask what happens to a network when some of its AND words are allowed to become OR words. Our initial guess would be that making some words easier to activate would improve the overall level of activation in the vocabulary network, but it is difficult to see exactly how this process would work in practice. This makes it a good candidate for a simulation approach.

We want to build a program to examine how making a number of words easier to activate changes the overall level of activation in a vocabulary network, but there are many ways we could do this, so first of all we need to be very specific about how our program is going to work. Figure 1 illustrates three approaches to this question: we have called them Methods A, B and C.

Method A: the obvious line of attack would be to use the same general approach that we used in the previous Part, while changing the nature of the **events** that the network experiences. So, instead of merely turning a word ON, each event in our simulation will choose a random word, turn it ON if necessary, change its Type to OR if necessary, update the network to take account of this change, and then report back (see Method A in Figure 1). This approach is very easy to implement, but it has two main disadvantages. The first problem is that it is a bit lacking in motivation: what circumstances might cause a word to change its Type? Why would this happen randomly? It is not entirely convincing to say that these changes "just happen". A second problem is that Method A is sensitive to the number of ON words in the

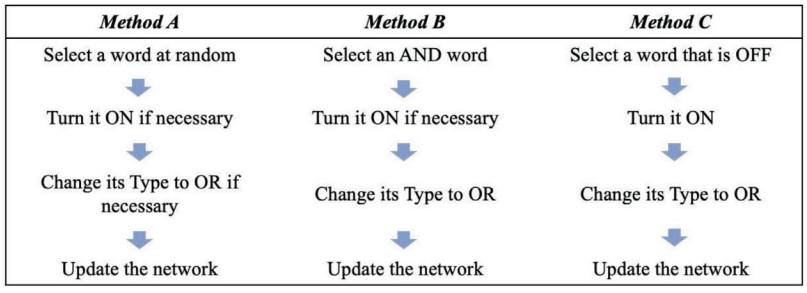

Figure 1 *Three ways to increase the number of OR words in a network.*

network: it will become less effective as the number of ON words in the network increases, and when the number of ON words in the network is large. Furthermore, if a word is ON already, then changing its status from AND to OR is not going to make any difference to its current state.

Method B: seems slightly more plausible. In this method, the program first finds an AND word, turns it ON if necessary, and changes its Type from AND to OR. Method B should be more effective than Method A because, provided that there are suitable candidates, each event should result in a word being upgraded. However, this approach has the disadvantage that the first step will repeat until the program finds a suitable candidate for an upgrade, and this could take a very long time if the number of OR words is already high. In an extreme case, where all the words are already OR words, the program will fail to find a candidate. In less extreme situations, the program may still run slowly when the number of AND words in the network is small, simply because it has to search harder to find an AND word to upgrade.

Method C: looks first for a candidate word that is OFF already, turns it ON, and then changes its Type to OR. This approach deliberately targets OFF words, so it should work faster when the overall activation level in the network is low. However, Method C also has its own weaknesses, as it will sometimes select an OFF word that is already an OR word, and in that case the event will not produce a change in the network. Again, these events with no effect will be most likely to occur when the number of OR words in the network is high.

These three methods are superficially very similar, but they will all work in subtly different ways.

It is easy to think of other methods that could be used to turn AND words into OR words. For instance, the three methods above all make the assumption that words change their Type one at a time – a single word at each event. But we could ask: how will a vocabulary network respond to an event where twenty or thirty words change their Type in a single event? Or we can ask: how would a network respond if we devised a method that does not rely on randomly selected words? For example, most networks have some words that provide inputs for lots of other words, while some words do not provide inputs to any other words: what would happen if we devised a method that prioritises the first type, rather than the second? The important idea here is how thinking in this way about the details of the programming forces us to question the assumptions that we are building into the simulations. No one method is intrinsically better than the others, but they do introduce different biases, and we need to be aware that the choices we make when we build a simulation are not entirely neutral. This idea will be a recurring theme throughout this Workshop.

In the next section, we will be working with a program that implements Method A in Figure 1, despite our earlier reservations. The rationale for making this choice is that we are deliberately making our simulations as simple as possible, and Method A seems to be simpler than the other Methods that we have outlined. However, you should bear in mind that other approaches will be worth investigating too.

Program-4: More OR words in the network

Program-4 is the basic simulation set that we will explore in this Part of the Workshop. To run these simulations go to the workshop Home Page: ***https://www.lognostics.***

co.uk/Workshop/index.htm and select Program-4. The program should give you a data input screen that looks like Figure 2.

This screen contains the usual two panels. This time, you have in the top panel four parameters that give you some control over the networks you create and the events that affect them. As usual, the network parameter, **NTWK**, determines the initial set up of your vocabulary network, the way the words in your network are linked to other words, and the way that the words in your network respond to inputs from other words. The initialisation parameter, **INIT**, randomly determines whether words are ON or OFF when the network is initialised. The number of events parameter, **nEv**, determines how many events will take place in your simulation, and the randomise events parameter, **rEv**, determines which words are randomly affected by these events.

When the program starts, it randomly initialises the network, turning about half of the words ON. It then allows the network to update itself until it finds a stable attractor state. This will usually happen after around 50 updates. Events are scheduled to take place at random intervals, starting at update 100, and they take place before the regular update procedure is called. Each **event** selects a single word, turns it ON, and sets its Response Type to OR. Words of this type go ON if either one or both of their inputs are ON. If the program selects a word that it is currently OFF, then it temporarily turns the word ON. If the program selects a word that is currently an AND word, then it permanently changes its Response Type to OR. No other changes are made by the program. The question we are interested in is how these minimal changes affect the overall level of activation in the network.

Think about how you would expect the simulations in this set to work before you run them. Specifically, think about whether you would expect the vocabulary network to absorb any changes, as the networks in program-2 and program-3 did. If not, how quickly would you expect the activation levels in the vocabulary networks to grow, and what factors will affect this growth? Now set the value of the NTWK and the

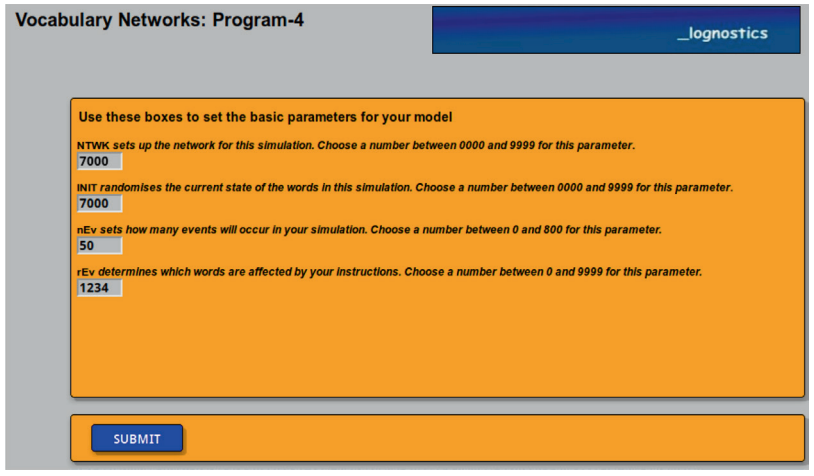

Figure 2 *The data input page, Program–4.*

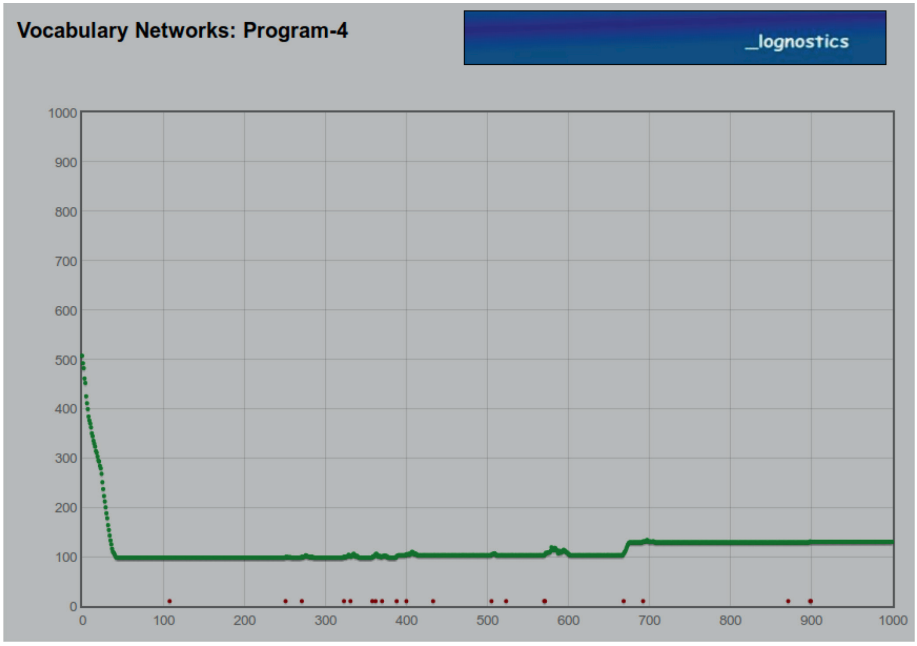

Figure 3 *The report page for Program-4 when NTWK: 7000; INIT: 7000; nEv: 20; rEv: 1234.*

INIT parameter to 7000. Set the value of nEv to 20: this tells the program to schedule 20 events. Set the value of rEv to 1234 and click the SUBMIT button. The program should return a report that looks like Figure 3. Figure 3 shows that Network 7000 is a network with an attractor state where only a few words are activated.

You read this chart in exactly the same way as you read the charts in Part 2 of the Workshop. The chart shows you that you have a network consisting of 1000 words, and the green line shows you the number of words which are in an activated state after each update of the network. The red dots at the foot of the chart show you where an event has taken place. In Figure 3 there are twenty events. At each event, a single word is selected, turned ON and upgraded to an OR word. The network starts out with about 500 words activated at random. Over the first 50 updates, it gradually sinks to an attractor state where 100 words are ON. The first event takes place about update 105, but it appears to have no effect (why not?). The events that take place between update 250 and update 405 cause a few small ripples in the activity level of the network, but they do not have a lasting effect. Likewise, there is a small ripple of activity around update 580. The two updates that occur just before update 700 introduce a slightly higher level of activity in the network, raising the number of ON items to about 125 words. More importantly, this increment appears to be permanent: it does not dissipate as more updates affect the network. Clearly, some kind of qualitative change has taken place in the network as a result of the increase in the number of OR words it contains.

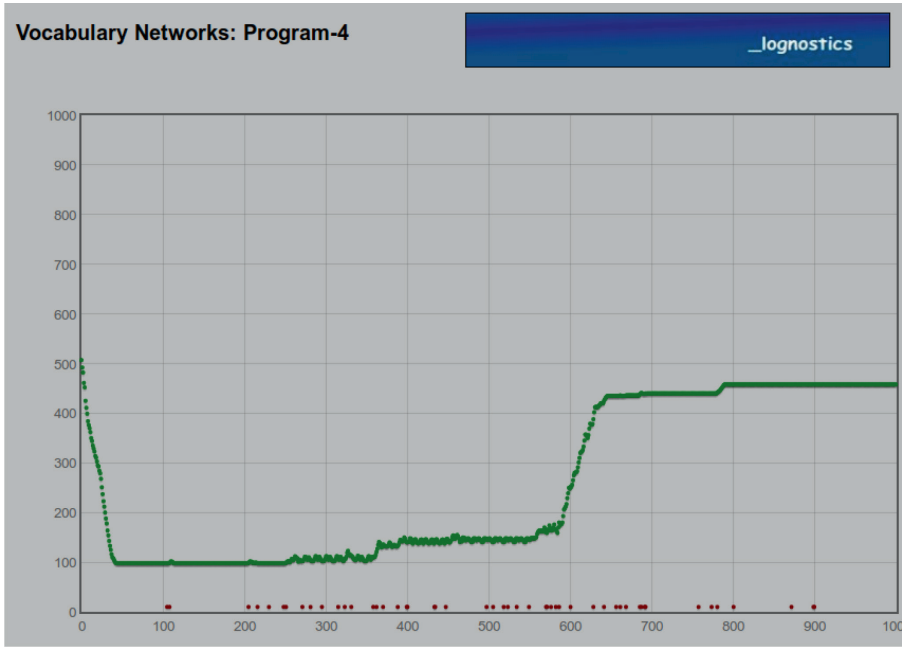

Figure 4 *The report page for Program–4 when NTWK: 7000; INIT: 7000; nEv: 50; rEv: 1234.*

Now set the value of the nEv parameter to 50 and run the simulation again. You should get a report that looks like Figure 4. This report shows that initially Network 7000 is not strongly affected by the events that it encounters. Around update 250, however, the network becomes slightly unstable – the wavy line at this point indicates that a number of words are flipping between the ON state and the OFF state. Around update 375 we get a small uplift in the overall activation level of the network, and this increase persists until update 550 when there is another small rise. At update 590 the network experiences a sudden, very large increase in the number of active words, levelling out at about 450 words. Another small increase occurs at update 785, and when the events stop at update 900, the network has found what appears to be a new stable attractor state.

This pattern of responding is very different from the patterns we found in Part 2 of the Workshop. It is obvious from Figure 4 that making words easier to activate – changing their response type from an AND word to an OR word – can have a big impact on the overall level of activation of the vocabulary network. We did not find any effects as large as this in Part 2 of the Workshop. The figure suggests that a relatively small number of events has a marked effect on Network 7000. We get our first permanent uplift in activity after about 15 events, and a massive uplift in activity after about 30 events. On the face of it, this is very surprising. In Part 2 of the Workshop we found that even massive events where a couple of hundred words were turned ON repeatedly failed to have a permanent effect on the overall activity level of a network. Here, in

contrast, as few as thirty events that change the status of just a single word seem to be giving us a permanent uplift in the network's overall activity level. Figure 4 suggests that network 7000 has increased its activity level by 450% by the time it reaches update 800. To put it mildly, this is astonishing: just 30 update events are enough to make a lasting impact on nearly 50% of the words in the network.

It is worth thinking about exactly what is going on in here. When the network is initialised, 50% of the words are categorised as AND words, and 50% are categorised as OR words. This is just a working assumption that we have built into the program. The events change these figures by increasing the number of OR words in the network. Since half the words are OR words when the program starts, and words are chosen randomly for upgrading, half of the events will probably select an OR word for upgrading. No change happens when this occurs, so only half of the events will actually upgrade a word from AND status to OR status. The implication of this is that upgrading words to OR status is even more effective than we first thought. A mere 15 upgrade events actually contribute to the massive uplift we get at update 590: a tiny number of events has radically changed the behaviour of the network.

By now, you should be asking how you can explore this idea further using Program-4. So here are some questions to guide your explorations.

- **What happens if we increase the number of events impacting on network 7000?**
  You can examine this by varying the value of the nEv parameter.
- **Does a different pattern of events produce a similar outcome on network 7000?**
  You can examine this by varying the value of the rEv parameter.
- **Do all networks behave like network 7000?**
  You can examine this by varying the value of the NTWK parameter.
- **Does a pattern of events that produces a surge in one network always produce a surge in other networks?**
  Check whether this is more likely if two networks have similar attractor states.
- **How big are the ripples caused by events?**
  Work with a low value of nEv and different values of sEv.
- **How many events have to occur before you get a massive surge in activation? Why?**
- **What is the minimum number of events needed for a surge to appear? What is the average number of events needed?**
- **Do clusters of events occurring together make a surge more likely? Why?**
- **What is the biggest surge that you can find?**
- **What conditions are necessary for a surge to occur?**
- **Can you predict when a surge is about to occur?**

There are no "correct" answers to these questions – your answers will depend to some extent on the values you choose for the different parameters, but you should nevertheless notice some patterns in the data that you collect. You may also notice some unexpected outcomes in the simulations that you run. Occasionally, upgrading a few words will result in **all** the words in the network becoming activated, and once

this happens the network is in a stable attractor state. A vocabulary network where all the words are ON does not seem to be a very probable model in real life. So, maybe we need to introduce another feature into our models which prevents this from happening? Ask yourself what would be a plausible way of doing this? What changes could we make to the model to prevent this outcome? What other effects would these changes make to the way the simulations work?

Generally speaking, however, you should find that you almost always get a result like the data shown in Figure 4: after a small number of upgrade events, the number of activated words in the vocabulary network rises suddenly and steeply and stays at that level. The really interesting thing here is how rapidly this shift occurs. In a vocabulary of 1000 words, it is not surprising to find that nearly all words are ON when most of the words are OR words, but it IS surprising that this effect appears so soon after we start implementing upgrade events.

Clearly, the rapid surge in the number of activated words looks like a very robust phenomenon. Clearly, too, it does not take very many upgrade events for this rapid rise to appear in the data. You may even have found some simulations where a mere handful of upgrade events is sufficient to activate the entire vocabulary network. This strongly suggests that "making words easier to activate" is a fundamental process in acquiring a vocabulary, and as such, it ought to be a fundamental feature of the way we teach vocabularies. However, it is not easy to think of any traditional exercise types that would directly emulate what we have been doing in this simulation. We need to ask: what kind of learning task would have the specific effect of making words you know only receptively permanently easier to activate? Once again, the simulations are making us ask what in real life would correspond to "making a word easier to activate".

We will follow this idea up in some of the later Parts of the Workshop, but first let us try to tease out what is going on in our vocabulary networks when the number of OR words increases.

Program-5: Controlling for the number of OR words with the ORWds parameter

You may have noticed that the surge in active vocabulary initiated by increasing the number of OR words in the network seems to appear when the number of OR words in the network approaches 600 words or so. As this is a recurrent finding, it probably suggests that the way our vocabulary networks have been set up are too simple and need to be reassessed. Basically, in the simulations that we have worked with so far, each word has an equal chance of being an AND word or an OR word when the networks are initialised. That results in about 50% of the total vocabulary falling into either one category or the other. But maybe this operating assumption is missing something important? Maybe a network where the number of OR words exceeds a certain threshold is fundamentally different from a vocabulary where most words are not readily activated? You can explore this idea with the simulation set in Program-5.

To run these simulations, go to the Workshop home page and select **Program-5.**

The data input page for Program-5 is similar to the data input page for Program-4, but this time you have an additional parameter that you can control: the **ORWds** parameter (see Figure 5). This parameter determines the number of OR items – words

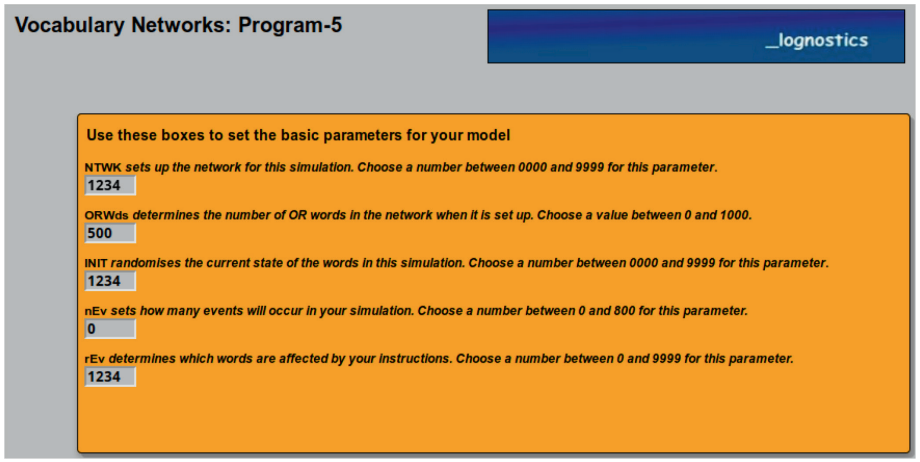

Figure 5 *The data input screen for Program–5.*

that are easier to activate – that appear in your vocabulary network when it is first initialised. If we set this parameter to 1000, then all the words in the vocabulary are easy to activate. If we set this parameter to 0, then none of the words in the network are easy to activate.

Set the nEv parameter to zero, and experiment with lots of different values of the ORWds parameter. With the nEv parameter set to zero you will get no events – your vocabulary networks should just initialise and stabilise themselves. Now Test ORWds = 0, ORWds = 100, ORWds = 200, and so on.

Most of these parameter settings will give you an output where the network settles into an attractor state with all its words OFF. However, somewhere between ORWds = 400 and ORWds = 600 there is a kind of 'magic window' where the outcomes are much less predictable. In these cases, the vocabulary networks nearly always find a stable level of activity where some words are ON and other words are OFF. There is a huge amount of variation as to what level the networks settle at, but you should find that there is a high correlation between the overall level of activity in the networks and the number of OR words they start off with.

Carry on raising the value of the ORWds parameter until you find a value for ORWds where all the words in the network are turned ON. You now have an upper bound where all words are ON, and a lower bound where all words are OFF. Check whether these bounds work for other networks by using a range of values of the NTWK parameter. You should find that most networks show the 'magic window' effect. But the size of the window, and its upper and lower bounds will vary from one network to another.

This result suggests that a critical feature of a vocabulary network is the number of OR words the network contains. However, the models we looked at in our earlier simulations did not take account of this. In all the earlier simulations, our networks were initialised with the AND parameter set at 50%, so most of the vocabulary networks that we have looked at so far will have had about half of their words set up

as AND words, and about half set as OR words. Clearly, with hindsight, this was a simplification too far. Fortunately, we can easily rectify this oversight.

Instead of setting the Response Type of each word randomly, we can specify how many words of each type we want our networks to contain when it is initialised. However, we could go a step further than this and say that all the words in our vocabulary networks start off as difficult to turn ON: i.e., we could initialise our vocabulary networks so that ALL the words start off as AND words. Think about how this would affect the way the networks behave when we make some words easier to activate. Initially the number of active words is likely to be very low – probably close to zero, but we could expect the number of active words to grow as we increase the number of OR words in the network. It is difficult to guess how many OR words we will need before a reasonable number of words will become activated, but we can find an answer to this question by running Program-5 again, this time allowing upgrade events to take place.

Go back to the input page for Program-5. Choose a new value for the NTWK parameter, set the ORWds parameter to 0, This means that all the words in your vocabulary network will be AND words, so difficult to turn ON, when the network is first initialised. Set the nEv parameter to 800. This means that an upgrade event will take place each time the network updates itself. In Program-5, each event makes a single word easier to activate by setting its Type to OR. Eventually, we will have a large number of OR words in the network, and we can expect some spontaneous activation to emerge.

Figure 6 shows an example of a simulation of this sort. As usual, the chart shows you the number of activated words in the network after each update, and the red dots show you the points at which events are implemented. Here every update is accompanied by an upgrade event. The chart shows that there is a tiny amount of activity in this network when it is first initialised. However, the network almost immediately settles into an attractor state where none of its words are active. After about 500 upgrade events, we get a small but steady uplift in the number of ON words in this network. At update 850 there is a sudden massive uplift, and when the upgrade events stop at update 900, the network finds itself in a new attractor state where about 880 words have become permanently ON.

The question for us to ask now is whether this is a typical response pattern for a network. You can investigate this by choosing a range of values for the NTWK parameter, and for each of these values run a set of simulations that vary the number of upgrade events. The answer to our question seems to be that while the response of most networks looks like Figure 6, not all networks respond in this way. Some networks do not experience an uplift, while others do show an uplift, but one that is smaller than the one shown in Figure 6.

Now find a network that does not show the typical uplift and change the value of the rEv parameter. This parameter determines which words change their status from AND to OR when an event takes place. Surprisingly, perhaps, changing the rEv parameter will sometimes result in a marked change in the behaviour of a network, and you may find that your unresponsive network becomes responsive when you choose a different value for the rEv parameter.

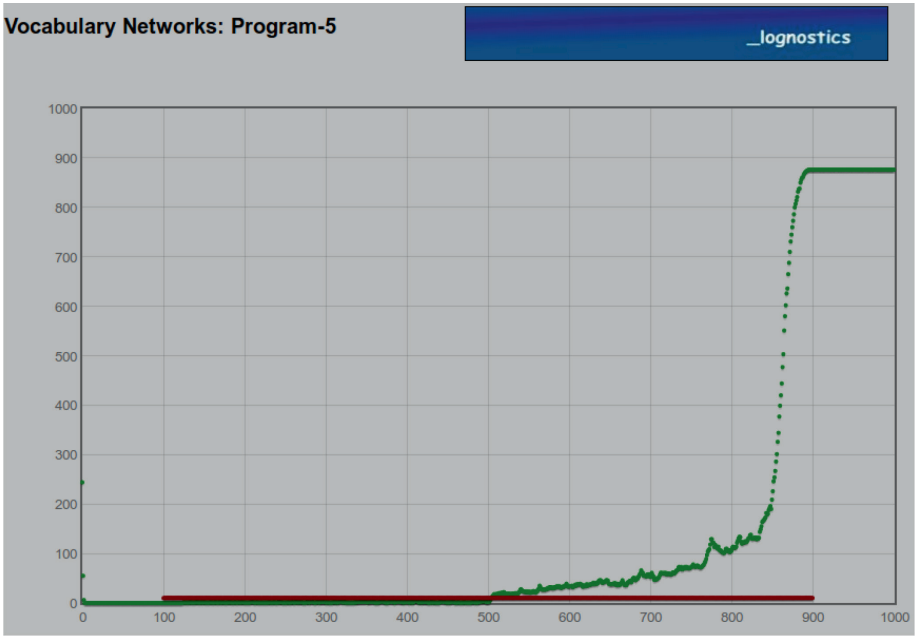

Figure 6 *The report page for Program-5 when NTWK:2000; INIT:1234; ORWds:0; nEv:800; rEv:1234.*

As usual, this data should be prompting you to ask questions about what is going on here. Some of the more obvious questions are listed below, and you should be able to see how you can investigate them by varying the value of the parameters when you run the simulations.

- **Does a series of events always result in a permanent uplift?**
  We have seen that uplifts do not always appear in these simulations, but maybe that is because we have limited the number of upgrade events to 800. What would happen if a greater number of upgrade events was allowed?
- **How many upgrade events are necessary for the uplift to occur?**
  You can examine this question by changing the value of the nEv parameter. Try slowly reducing the number of events until you reach a point where the network does not experience an uplift in the number of ON words.
- **Do all the networks behave in this way? Do some networks experience a series of smaller uplifts, rather than a single massive one?**
  You can examine this question by varying the value of the NTWK parameter. Note how many different ways of responding that you find.
- **Does a network always respond in the same way, irrespective of which words are being upgraded? You can examine this question by varying the value of rEv.**
  This will cause the program to select different words for upgrading.

- **How do the networks respond if they already have a small number of OR words when the simulation starts?**
  You can examine this by changing the value of the ORWds parameter. For example, you could take the network in Figure 6 and raise the value of the ORWds parameter to 200. This will give you a network with 200 OR words when it is initialised. Is this sufficient to prevent the network falling into an inert attractor state? Does it change the shape of the curve in the report page? Does it result in an even higher final attractor state?
- **How do the parameters interact with each other?**
  If you raise the number of OR words in the network when it is initialised, can you reduce the number of events that are needed to generate the characteristic uplift in activation? You can examine this question by systematically varying the values of the ORWds parameter and the nEv parameter.
- **What is the minimum number of upgrade events needed for a network to find a new attractor state?**
  You can investigate this question by setting the value of ORWds to 500, and finding a network whose natural attractor state has about half of its words ON. Set the value of nEv to 1, and run the simulation. Gradually increase the number of upgrade events and watch what effect this has on the overall activity level in the network? How many events are needed before an uplift occurs? Does this surprise you?
- **Can you predict when a network is about to experience an uplift?** You could ask, for instance, whether a small increase in activation predicts that a larger increase will appear within a further 20 updates.

Once again, these questions do not have any "correct" answers. Your conclusions will depend to some extent on the specific choices you make for the different parameter values. Program-5 definitely shows that some relatively small changes to a network can indeed dramatically affect the overall level of activation it displays. However, if you are thinking critically about this simulation set, then you might be feeling that Program-5 makes some rather big assumptions about the environment where these events take place in. In particular, you might be thinking that it is a bit odd for us to look at the way that a totally inert vocabulary, where every word is OFF, responds to changes in the Type of its words. What is driving these changes? Why would a word that is OFF change its Type in the absence of some sort of activity in the network? Program 6 addresses this problem.

Program-6: Working with unstable networks

Program-6 allows you to explore how upgrade events that change words from AND status to OR status react differently when the network is in an unstable state. We will not go through this in any detail in this part of the Workshop. Instead, we will just provide you with a few pointers to guide your own investigations.

Program-6 works in exactly the same way as Program-5, but it lets you explore what happens in a network which is NOT in a stable attractor state when words are

upgraded. In Program-5, the vocabulary networks were allowed to settle into their stable attractor states before any events took place, and with low numbers of OR words, this usually resulted in an attractor state where no words were activated. Clearly, this is a rather unusual state of affairs, and it leaves us with the problem of why upgrade events should spontaneously occur in a network which is inert. The obvious solution is to allow some random activation to take place in the network that we are studying. In real world terms, this would be the equivalent of providing a context for a word that is being upgraded.

The main feature of Program-6 is that it contains a new parameter: **WdsON**. When you run this simulation, the program sets up the network and lets it settle into a stable attractor state. From updates 100 to 900, the program systematically nudges the network into an unstable configuration by turning a small number of words ON at each update. The effect of this is to artificially raise the level of activation in the network. When this activation stops, then the network returns to its stable attractor state (see Figure 7.)

Set the value of ORWds to 0 and experiment with a range of different values for WdsON. This should show you that some non-permanent activity can emerge, even when all the words in a network are difficult to turn ON. Now gradually raise the value of the nEv parameter to add upgrade events to your network. This will increase the number of OR words in your network.

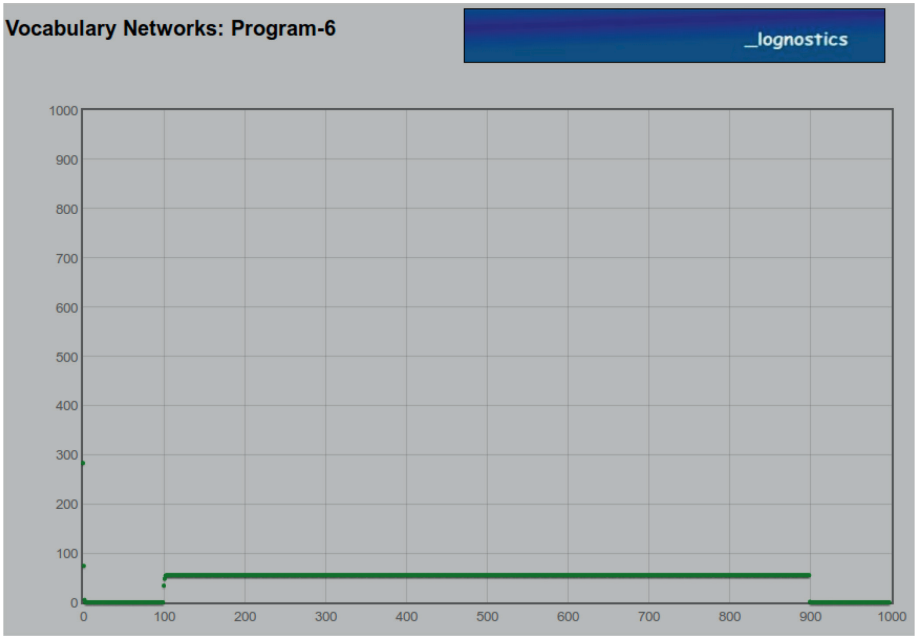

Figure 7 *The report page for Program-6 when NTWK: 5000; ORWds: 0; INIT: 1234; nEv:0; rEv: 5678 and WdsON: 200.*

Questions to ask with Program-6

- How many different ways of responding can you identify in a network?
- How important is the threshold that begins to appear when the number of OR words in the network reaches 500 words?
- Will these models scale up to larger vocabularies?
- How would smaller vocabulary networks behave? Would you expect to get the phase shift effect with a small vocabulary network of, say, 200 words? How many upgrade events are necessary for the phase shift to emerge in a vocabulary network of this size?
- How would you expect upgrade events to interact with growth events in a Boolean vocabulary network?
- In real life, what kinds of activity could contribute towards a word being upgraded?
- In this simulation set, words come in only two Boolean types: OR words that are easy to activate, and AND words that are less easy to activate. How would you expect the models to work if there were three or four levels of activation potential? Is the binary AND/OR distinction a reasonable simplification?
- In this simulation set, upgrades occur randomly - for each event, the program selects a random word and upgrades it. In reality, it is more likely that upgrades are the result of some other activity in the network. For example, perhaps words are more likely to upgrade if they are activated frequently. What other features can you think of that could as triggers for an upgrade event?
- How would the networks in this simulation set react if the change from AND status to OR status was temporary, rather than permanent?

As usual, there are no right answers for these questions. Their purpose is to help you think about the simulations in a critical way.

Discussion

There are three main points to take away from this Part of the Workshop.

The first idea to note is that "making words easier to activate" looks like a very effective way of increasing the number of active items in a Boolean vocabulary network. In Program 4, we saw that activating words directly, by simply turning them ON, did not usually have a lasting effect on the overall levels of activation in a vocabulary network. In contrast, the simulation sets in this Part of the Workshop showed that making words easier to activate generally does have a lasting effect on the overall activation level of a network. Of course, it is important to remember that the simulations in Part 1 of the Workshop are not meant to be realistic models of the way vocabularies work, but they should make you think about how real vocabularies work. Making words easier to turn ON massively affects the way these model vocabulary networks perform, so you should be asking whether real vocabularies are affected in a similar way. Are there perhaps a handful of simple processes which determine how a real vocabulary performs?

The simulation sets in this Part of the Workshop have emphasised that "making words easier to activate" looks like a fundamental process that underpins the way vocabularies work, but on its own it does not tell us how words in a vocabulary actually come to be easier to activate. What the simulations DO suggest is that this feature is one that we need to investigate further.

Surprisingly, there is relatively little research on this topic: the whole area of vocabulary fluency or automaticity is something of a black hole in L2 vocabulary research. A number of people have suggested that something like accessibility might be a fundamental property of a vocabulary (e.g., Henriksen, 1999; Meara, 1996). These authors use a dimensional model of lexical development, trying to account for important features of lexical development in terms of a small number of "dimensions", such as SIZE, ORGANIZATION and AUTOMATICITY. It is easy to see that SIZE might be a fundamental characteristic of a vocabulary, and it is relatively easy to see how we might be able to measure how big someone's vocabulary is. It is also easy to see how we might describe a vocabulary in terms of the way it is structured: we can think about a vocabulary as a set of words that are linked together in some way, and we can describe the patterns these links make. In fact, there is a whole branch of mathematics devoted to describing the way networks are linked together—graph theory—and there are plenty of tools available to describe and measure the complexity of networks (cf. Meara, 2007). Unfortunately, the dimensions idea works much less well when we come to measure automaticity. The main difficulty is that it is not obvious how you could come up with a single measure that would describe a whole vocabulary in terms of how easily its words can be accessed. In a simulation like the ones in this Part of the Workshop, the solution to this problem is obvious: we simply count the number of words that are easily activated. In a real vocabulary, however, we cannot do this. So we are stuck with what looks like an important idea, but one which we cannot investigate in real life because we do not have the necessary tools. All we can say at the moment is that accessibility looks like an important feature of a real vocabulary, and that this idea needs to be explored. Some interesting ideas about automaticity in an L2 are developed in Segalowitz and Gatbonton (1995), Segalowitz et al. (1998), Segalowitz (2003), Oberg (2012), Miralpeix and Meara (2014), and Pellicer-Sánchez (2015).

One type of vocabulary which might be worth exploring in more detail is cognate vocabulary. Cognates are words which are etymologically related in two languages – for instance *bleu, blau* and *blue* in French, German and English respectively, or *Mann* and *man* in German and English. Not all cognate pairs are as obvious as these examples – for instance, English *chain* and Spanish *cadena* are technically cognate words (they both go back to the Latin *catena)*, but most people would not readily recognise this. And some apparent cognates are actually **false friends**—words that look similar but have divergent meanings: in Spanish, for example, *embarazada* looks as though it should have the same meaning as English *embarrassed*. It actually means *pregnant*. Nevertheless, some language pairs have very large numbers of cognates (cf. Otwinowska, 2015). These words ought to be very easily activated in an L2 vocabulary because they are very similar to words in the learner's L1. It is interesting

to speculate whether a large number of cognate words in an L2 might form the core of easy-to-activate words that seems to be necessary for a large active vocabulary. Cognate vocabulary has been extensively researched, but largely from a linguistic point of view (cf. for example Hammer and Giauque, 1989, and Granger, 1993). The psycholinguistics of cognates is much less well researched (but see Costa et al. 2000, for some relevant experimental work).

Cognacy is not the only feature that can make words easy to process. Another feature that probably plays a role in making words easy to turn ON is frequency. In real life, L2 words which are encountered frequently seem to have a special status, especially if they are encountered frequently over a short time span. Recency effects could also facilitate turning words ON. We could model effects of this sort, but for the moment, such complex effects lie beyond the scope of the simple models we are working with so far. However, you should be asking yourself what additional parameters would make our model vocabularies more realistic in this respect.

The second idea that emerges from the simulations in this Part of the Workshop is that it may not be a good idea to assume that vocabulary acquisition as a process always works in the same way. Most vocabulary research assumes that the processes involved are basically linear: we teach the words using whatever method seems appropriate, and we watch them being learned – or not. From this approach, it follows that the interesting things to research are the conditions which allow words to be learned (e.g., are some words easier to learn than others?). The simulation sets in Program-5 and Program-6 suggest that things might be more complicated than this. The basic problem seems to be that we have a threshold effect, rather than a simple linear effect. This means, at the very least, that the same input will not necessarily produce the same output all the time. A single learning event that takes place before the rapid activation threshold has been reached may not appear to have any great effect, while a very similar event on the cusp of a threshold may appear to be massively important.

The main idea in this Part of the Workshop is that our simple simulations have identified a process—"making words easier to turn ON"—which looks as though it might play a critical role in the way vocabularies grow and develop. This idea does not play an important role in current thinking about vocabulary acquisition, partly because it is a dynamic feature of vocabulary development, whereas our current testing tools focus more on static descriptions of vocabulary knowledge. The important thing here is that the simulations have pushed us to look at vocabulary in a way that we probably would not have found without their help.

The simulations have also highlighted another feature of vocabulary networks that does not figure much in our current theorising about vocabulary. This is the idea that it might be important to look for threshold effects in L2 vocabulary growth. The simulations do not PROVE that these threshold effects exist, but they strongly suggest that threshold effects MIGHT be a standard feature in vocabulary acquisition if we knew where to look for them. Once again, the simulations seem to be nudging us in a direction that we probably would not have identified if we were using conventional research methods. In the next Workshop we will look at what happens when we allow the connections between words to change in response to an event.

References

Costa, A., Caramazza, A., & Sebastian-Galles, N. (2000). The cognate facilitation effect: Implications for models of lexical access. *Journal of Experimental Psychology: Learning Memory and Cognition, 26*, 1283–1296. https://doi.org/10.1037/0278-7393.26.5.1283

Granger, S. (1993). Cognates: An aid or a barrier to successful L2 vocabulary development? *ITL Review of Applied Linguistics, 99–100*, 87–105. https://doi.org/10.1075/ITL.99-100.03GRA

Hammer, P., & Giauque, G.S. (1989). *The role of cognates in the teaching of French*. Peter Lang. https://doi.org/10.2307/328851

Henriksen, B. (1999). Three dimensions of vocabulary development. *Studies in Second Language Acquisition, 21*, 303–317. https://doi.org/10.1017/s0272263199002089

Meara, P. M. (2007). Growing a vocabulary. *EuroSLA Yearbook*, 49–65. https://doi.org/10.1075/eurosla.7.05mea

Meara, P. M. (1996). The dimensions of lexical competence. In G. Brown, K. Malmkjaer, & J. Williams (Eds.), *Performance and competence in second language acquisition* (pp. 35–53). Cambridge University Press.

Miralpeix, I., & Meara, P. M. (2014). The written word. In J. L. Milton & T. Fitzpatrick (Eds.), *Dimensions of vocabulary knowledge* (pp. 30–44). Palgrave Macmillan. https://doi.org/10.1007/978-1-137-36831-7_3

Oberg, A. (2012). Receptive and productive vocabulary acquisition: examining processing time and memory threshold. *Indonesian Journal of Applied Linguistics, 2*(1), 23–40. https://doi.org/10.17509/IJAL.V2I1.71

Otwinowska, A. (2015). *Cognate vocabulary in language acquisition and use: Attitudes, awareness, activation*. Multilingual Matters. https://doi.org/10.21832/9781783094394

Pellicer-Sánchez, A. (2015). Developing automaticity and speed of lexical access: The effects of incidental and explicit teaching approaches. *Journal of Spanish Language Teaching, 2*(2), 1–15. https://doi.org/10.1080/23247797.2015.1104029

Segalowitz, N. (2003). Automaticity and second languages. In C. Doughty & M. Long (Eds.), *The handbook of second language acquisition* (pp. 382–408). Blackwell. https://doi.org/10.1002/9780470756492.ch13

Segalowitz, N., & Gatbonton, E. (1995). Automaticity and lexical skills in second language fluency: implications for CALL. *Computer Assisted Language Learning, 8*(2–3), 129–149. https://doi.org/10.1080/0958822940080203

Segalowitz, S., Segalowitz, N.S., & Wood, A. G. (1998). Assessing the development of automaticity in second language word recognition. *Applied Psycholinguistics, 19*(1), 53–68. https://doi.org/10.1017/S0142716400010572