Modelling word retrieval issues in aphasic patients in a neural network

by
Amélie J. Croshere

BA thesis Linguistics
University of Amsterdam
Supervised by Paul Boersma
July 2024

1. Introduction

Neural modelling has been around for several decades, and has many different applications in many different fields. One of those fields is linguistics. Within linguistics, neural modelling can be used for several different things. In this thesis, neural modelling will be used to simulate an impairment in word-retrieval in people with aphasia and the effects the therapy those people received has on their impairment.

Aphasia is a language disorder that can impact several areas of speech, both in comprehending and in producing speech. It is usually acquired as a consequence of brain damage, after, for example, a stroke, but it can also be part of a neurodegenerative disease like Alzheimer's Disease.

Modelling a word-retrieval impairment in people with aphasia has been attempted before by Grasemann et al. (2021), who created the BiLex model to simulate the effect of treatment on lexical access in bilingual people suffering from aphasia.

To do this, they first created a model that could simulate pre-stroke lexical access in each of the participants, taking into account their age and language exposure to each of their two languages. Their model is based on Self-Organising Maps (SOMs), with one corresponding to the semantic system, which is shared between the two languages, and two corresponding to the two phonetic systems of the participants' languages. The maps were connected via bidirectional associative connections, and the weights between those connections were based on language dominance. Not all neurons on one layer were connected to all neurons on the next layer. Grasemann et al. (2021) does not explain which neurons are connected and which are not, but it may be assumed that the neurons that are connected are located near each other. A visualisation of the model can be seen below.

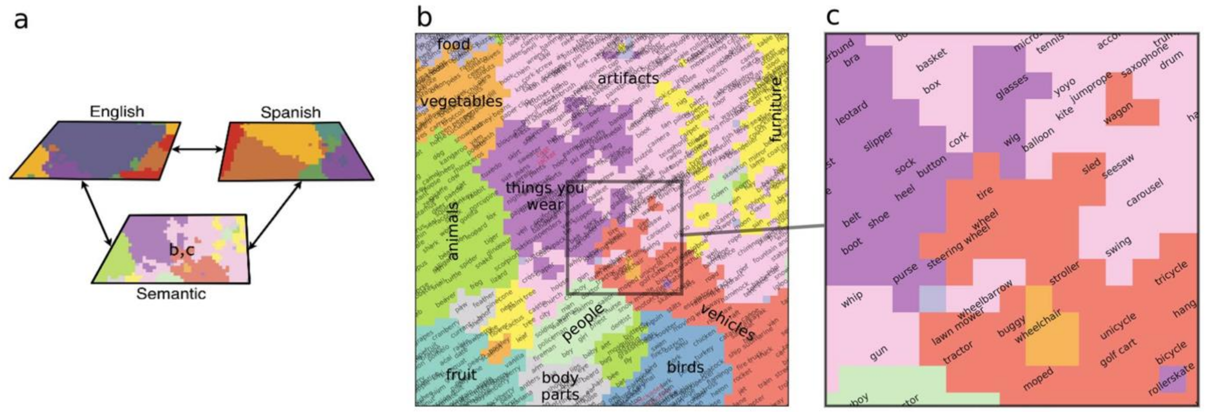

Figure 1: a visual representation of Grasemann et al. (2021)'s model

A SOM algorithm was used to train both languages, but only one language at a time, by training one of the two phonetic maps, namely the one belonging to the language being trained, and the shared semantic map together. The weights were adapted through Hebbian learning. This pre-stroke simulating model was trained until its scores matched those of the participant pre-stroke. The model takes a semantic representation of a word as input, and then transfers it from the semantic map to one of the phonetic maps. The output was interpreted as correct if the most highly activated phonetic unit matched the input word. Next, to simulate the effect of the stroke on word retrieval, circular areas containing neurons and their connections were removed from all maps, and after that further damage was done to the semantic map until it matched the patient's PAPT score. The PAPT (pyramids and palm trees test) is a metric used to evaluate semantic processing regardless of language, and is administered by giving the participant one word, and then presenting them with two options to choose between, with both options representing a similar concept to the presented word. The participants then have to pick the option that is more similar to the presented word. However, when modelling the patient's PAPT score in the model, the model did not develop a lexical impairment large enough to match the patient's, so further damage was done to the two phonetic maps until the lexical impairment in the model was sufficient for both languages. Finally, to simulate the effect of the treatment, a retraining cycle was developed, which attempted to mirror the actual treatment protocol the patients underwent. Firstly, thirty words were selected at random from a corpus containing concrete nouns. Those nouns were then used to retrain the semantic map. Afterwards, to simulate the exposure to the target words in the treated language, the phonetic map of the treated language and the semantic map were retrained, along with the connections between them. Lastly, the model was given the opportunity to recover the correct associations in the untreated language. This was done by transferring the activation from the phonetic map of the treated language to the phonetic map of the untreated language through the connections between the two phonetic maps, which could then be used to retrain the connections between the phonetic map of the untreated language and the semantic map. Furthermore, the activations from the semantic map were also transferred to the phonetic map of the untreated language through the connections between the semantic map and the phonetic map of the untreated language, which could then be used to retrain the connections between the phonetic map of the untreated language and the phonetic map of the treated language. Every time the patient attended a therapy session, this cycle was used to retrain the model. After every therapy session, and thus cycle, the model was tested in both languages, and the results were compared to those of the patients. The model was also trained to account for exposure to language in between therapy sessions. Six parameters were found using an Evolutionary Algorithm and used to determine learning rates and conditions for how the untreated language was retrained.

The results of this study showed that this model can model aphasia and therapy outcomes in English-Spanish bilinguals fairly well, especially in the treated language. The modelling of the untreated language also showed some success, but to a lesser extent.

Boersma et al. (2022) uses a different model, in which they model a small, invented language. Their model uses three levels: an auditory-phonetic level, a morphology/meaning level and an emergent phonological level in between those two. The auditory-phonetic level is represented by basilar membrane frequencies on which representations of the F1 and F2 of certain sounds can be mapped, which can be seen on the bottom left side of Figure 2. So, one utterance, which is, in their case, a vowel, causes two Gaussian bumps on the membrane. In the model, each of the used morphemes has its own node on the bottom layer, as can be seen on the bottom right side of Figure 2. If the input is a sound, represented on the basilar membrane, the expected output is the representation of the associated morpheme on the morpheme level. If the input is a meaning, represented by a morpheme, the expected output is the associated sound, represented on the basilar membrane. The middle layer of nodes, which represent the emergent phonological level, consists of an arbitrary number of nodes, which are linked to the input with weights determined during the training of the model. The top layer of nodes also consists of an arbitrary number of nodes, which are connected to the middle layer. Thus, the model works from the input layer, to the middle layer, to the top layer, back to the middle layer and then back to the bottom layer to provide an output. A visualisation of the model can be seen below.

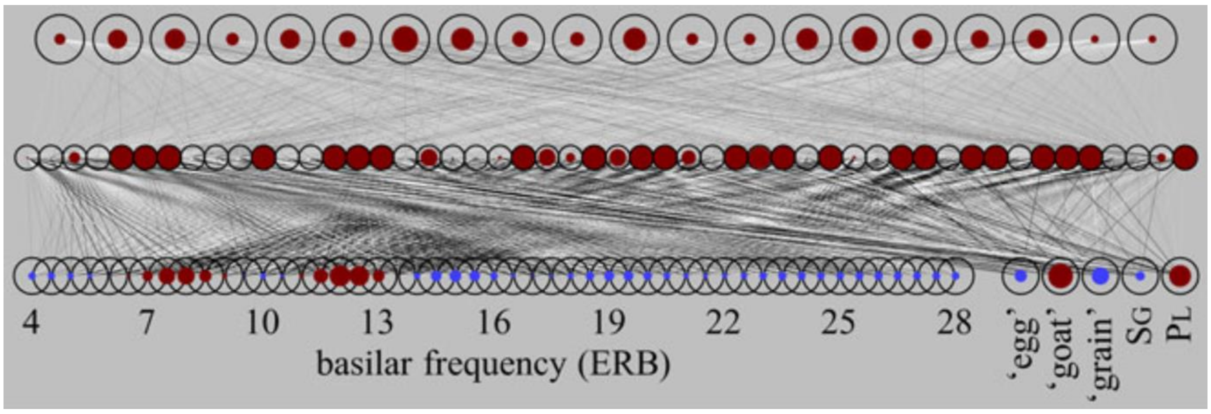

Figure 2: a representation of Boersma et al. (2022)'s model

To start training this model, all of the biases and weights are set to zero. Throughout the model, all connections need to be symmetric. This means that the influence node 1 has on node 2 is the same as the influence node 2 has on node 1. The training of the model consists of four phases per sound/word. First, the initial settling phase, in which the activities that were just applied spread to the next layers of the model, but the input layer remains unchanged. Next, there is the Hebbian learning phase, in which the weights and biases are adapted according to the 'neurons that fire together, wire together' principle. Thirdly, there is the dreaming phase, in which the input layer can now change, and the activities are not constant anymore. Lastly, the anti-Hebbian learning phase takes place, in which some of the knowledge, that already existed, is deleted. These four phases are repeated for new pairs of sounds and meanings until the model contains as much existing knowledge as new knowledge, meaning the model is properly trained.

This thesis will simulate the effects of aphasia, as presented in Graseman et al. (2021), but with the model presented in Boersma et al. (2022).

In the following section, the model structure, the data and the training and testing procedure will be described. After that, the results section will describe the outcomes of the presented model. Lastly, the discussion will discuss this model and the outcomes, and conclude this paper.

2. Method

2.1 The model[1]

The model is made up of three layers of nodes. The number of nodes on the bottom layer is determined by the number of phonetic and semantic features the input words have as per Grasemann et al. (2021), which in this case is 544. 144 of those are for the phonetic features, and the other 400 are for the semantic features. The bottom layer also functions as the output layer; if the given input is phonetic, and thus only covers the phonetic nodes, the semantic nodes will function as the output layer, and if only semantic input is given, which only covers the semantic nodes, the phonetic nodes will function as the output layer. The middle and top layer have a semi-arbitrary number of nodes, in this case 500 nodes on the middle layer and 465 on the top layer. All nodes are connected to all nodes on adjacent layers through connections with weights, which are bidirectional; they perform the same actions no matter which direction the activity goes. A visual representation of the model can be seen below.

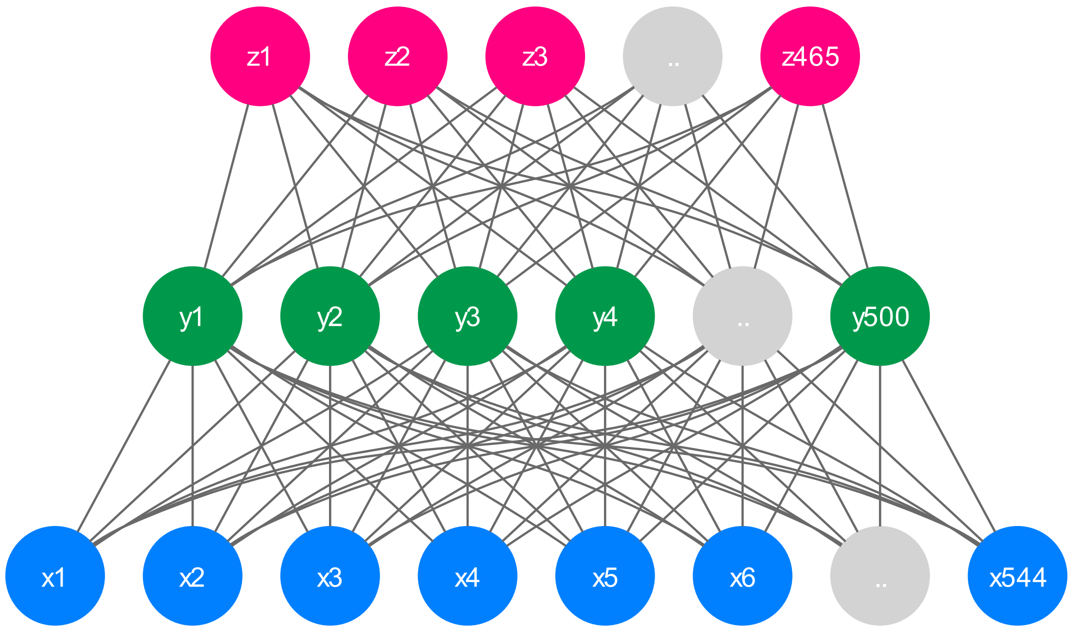

Figure 3: a representation of the model used here. The nodes on the bottom layer are signified by their blue colour, those on the middle layer by their green colour, and those on the top layer by their pink colour. Not all nodes are visualised here due to there being too many nodes to visualise properly and in a clear manner.

This model's algorithm is the same as in Boersma et al. (2022), and has already been described briefly in the introduction. Here, a more thorough explanation is provided.

In its initial state, all the model's activations, weights and biases are zero. When the first input, which can be a phonetic sequence, a semantic sequence, or a combination of both, is applied, this changes. The activities spread from the bottom layer to the middle layer, adapting the excitation accordingly. This is done by multiplying the excitation of each node with the weight placed on the connection that connects that node to the adjacent layer, and adding up the bias of that node and the sum of those multiplications, after which a logistic function is applied. Because the middle layer is connected to two layers, namely the bottom and the top

[1] The code for this model can be found on https://github.com/ameliejcr/thesis

layer, both the sum of the multiplications for the nodes on the bottom layer and the sum of the multiplications for the nodes on the top layer will be added up with the bias to become the activation of the nodes on the middle layer, after applying the logistic function. Then, the activities are spread from the middle layer to the top layer in the same way. Even though this step spreads to the middle and top layer, no spreading to the bottom layer takes place here, this will happen in the dreaming phase. After having repeated the spreading to the middle and top layer ten times, the first phase, the initial settling phase, is complete. This phase is also expressed in the formulas below, which are cited from Boersma et al. (2022).

$$y_l \leftarrow \sigma\left(b_l + \sum_{k=1}^{K} x_k u_{kl} + \sum_{m=1}^{M} v_{lm} z_m\right)$$

Formula 1: the logistic function σ is applied to the sum of the bias of middle node l (b_l), the sum of, for all nodes on the bottom layer, the activities of bottom node k (x_k) multiplied by the weight u_{kl}, which goes from bottom node k to middle node l, and the sum of, for all nodes on the top layer, the activities of top node m (z_m) multiplied by the weight v_{lm}, which goes from the middle node l to the top node m. This becomes the activation of node y_l on the middle layer.

$$\sigma(x) \coloneqq 1/(1 + \exp -x)$$

Formula 2: the logistic function used in Formulas 1, 3, 10 and 11

$$z_m \leftarrow \sigma\left(c_m + \sum_{l=1}^{L} y_l v_{lm}\right)$$

Formula 3: the logistic function σ is applied to sum of the bias of node m (c_m) and the sum of, for all nodes on the middle layer, the activities of middle node l (y_l) multiplied by the weight v_{lm} between middle node l and top node m. This becomes the activation of node z_m on the top layer.

In the second phase, a Hebbian learning rule is applied. Hebbian learning states that neurons that fire together, wire together. This is implemented by changing all of the biases of all the nodes and all of the weights between all three layers. The new biases are computed by adding the current bias to a multiplication of the learning rate, which is 0.001 in this case, and the excitation of the node in question. The new weights are computed by adding the current weight to the multiplication of the learning rate times the activity of the connecting node on one side of the connection of the weight times the activity of the node on the other side of the connection of the weight. This phase is visualized in the following formulas, cited from Boersma et al. (2022):

$$a_k \leftarrow a_k + \eta x_k$$

Formula 4: the bias of node k on the bottom layer (a_k) becomes the old bias, plus the learning rate η, which in this case is 0.001, multiplied by the activation of bottom node k

$$b_l \leftarrow b_l + \eta y_l$$

Formula 5: the bias of node l on the middle layer (b_l) becomes the old bias, plus the learning rate multiplied by the activation of middle node l

$$c_m \leftarrow c_m + \eta z_m$$

Formula 6: the bias of node m on the top layer (c_m) becomes the old bias, plus the learning rate multiplied by the activation of top node m

$$u_{kl} \leftarrow u_{kl} + \eta x_k y_l$$

Formula 7: the weight between bottom node k and middle node l becomes the previous weight plus the learning rate multiplied by the activation of bottom node k and the activation of middle node l

$$v_{lm} \leftarrow v_{lm} + \eta y_l z_m$$

Formula 8: the weight between middle node l and top node m becomes the previous weight plus the learning rate multiplied by the activation of middle node l and the activation of top node m

In the dreaming phase, the bottom layer receives activities that are spread from the middle layer, though a formula similar to the one used in the initial settling phase, minus the logistic function. To the middle and top layers, a Bernoulli distribution is applied to Formula 1 and Formula 3, both with the logistic function. Those distributions will return a probability in the form of a number between zero and one for each node on both layers. Per node, the probability will be compared to a randomly generated number between zero and one, and if the randomly generated number is equal to or higher than the probability, the excitation of the node becomes zero. Otherwise, it becomes one. This phase is also repeated ten times. This phase can also be found in the formulas below, cited from Boersma et al. (2022).

$$x_k \leftarrow a_k + \sum_{l=1}^{L} u_{kl} y_l$$

Formula 9: the bias of bottom node k is added to the sum of, for all nodes on the middle layer, the activities of middle node l multiplied by the weight u_{kl} between bottom node k and middle node l. This becomes the activation of node k on the bottom layer.

$$z_m \sim \mathcal{B}\left(\sigma\left(c_m + \sum_{l=1}^{L} y_l v_{lm}\right)\right)$$

Formula 10: the Bernoulli distribution is applied to Formula 3 to generate the new activity for top node m

$$y_l \sim \mathcal{B}\left(\sigma\left(bl + \sum_{k=1}^{K} x_k u_{kl} + \sum_{m=1}^{M} v_{lm} z_m\right)\right)$$

Formula 11: the Bernoulli distribution is applied to Formula 2 to generate the new activity for middle node l

Lastly, the anti-Hebbian learning phase reverses the Hebbian learning phase, by subtracting the learning rate times the excitation from the bias to form the new bias, and subtracting the learning rate times the activity of the node on one end of the connection times the activity of the node on the other end of the connection from the weight to form the new weight. This phase is also expressed in formulas cited from Boersma et al. (2022), which can be seen below.

$$a_k \leftarrow a_k - \eta x_k$$

Formula 12: the bias of node k on the bottom layer (a_k) becomes the old bias, minus the learning rate η multiplied by the activation of bottom node k

$$b_l \leftarrow b_l - \eta y_l$$

Formula 13: the bias of node l on the middle layer (b_l) becomes the old bias, minus the learning rate multiplied by the activation of middle node l

$$c_m \leftarrow c_m - \eta z_m$$

Formula 14: the bias of node m on the top layer (c_m) becomes the old bias, minus the learning rate multiplied by the activation of top node m

$$u_{kl} \leftarrow u_{kl} - \eta x_k y_l$$

Formula 15: the weight between bottom node k and middle node l becomes the previous weight minus the learning rate multiplied by the activation of bottom node k and the activation of middle node l

$$v_{lm} \leftarrow v_{lm} - \eta y_l z_m$$

Formula 16: the weight between middle node l and top node m becomes the previous weight minus the learning rate multiplied by the activation of middle node l and the activation of top node m

For mimicking the effect of aphasia, a method was designed in which a number between zero and one, which can be multiplied by 100 to create a percentage, could be entered. The percentage pertains to how many percent of all nodes are to be removed, along with all their connections. All of the nodes that are to be deleted are selected randomly, by having each node generate a random number between zero and one, and comparing that to the decimal. If the randomly generated number is lower or equal to the percentage, that node and its connections will be removed.

A much smaller instance of this model was later created for the purpose of being able to go through all the phases faster than the model described above, so all phases could be run through 10,000 times, since that is how Boersma et al. (2022) train their model, for it to be in a moderately advanced state. This model has twenty nodes on the bottom layer, ten on the middle layer, and five on the top layer. The bottom nodes are divided into ten nodes for phonetic input, and ten for semantic input.

2.2 Data

The data used to train and test this model consists of two parts. Firstly, there is the data containing the encoded words. In total, 638 words were used, and all of them are concrete nouns. Each one of those words has a phonetic encoding and a semantic encoding. For the phonetic encoding, the words are divided up into syllables. Those syllables are then divided up into onsets, nuclei and codas. Every syllable could have up to two onsets, two nuclei and two codas maximum. For every onset and coda, the onset/coda receives a value matching its place, manner, whether it is voiced or not and whether it is lateral or not, according to the IPA. Every nucleus receives values for how open and front the nucleus is, and whether it is long and/or rounded. Those values range from zero to one, with every value matching a quality the sound could have. Every word can have up to six syllables. An example of the structure of the phonetic data is provided below.

	nucleus				**coda**			
	open	**front**	**long**	**rounded**	**place**	**manner**	**voiced**	**lateral**
word								
ant	0	0	0.5	0	0.3	1	1	0

Table 1: an example of how the phonetic data is structured. Here, only one nucleus and one coda are shown, but one syllable can have up to two onsets, two nuclei and two codas. A zero for 'open' means the vowel is open, a zero for 'front' means that the vowel is a front vowel, a 0.5 for 'long' means the vowel is short, and a zero for 'rounded' means the vowel is unrounded. Regarding the coda, a 0.3 for 'place' means signifies the consonant is alveolar, a one for 'manner' indicates the consonant is nasal, a one for 'voice' means the consonant is voiced, and a zero for 'lateral' means the consonant is not lateral.

For the semantic encoding, every word gets assigned a specific value between zero and one based on how much a certain quality applies to that word, with zero meaning 'does not apply at all' and one being 'applies perfectly'. For example, one of the qualities is 'small', and one of the words is 'baby'. The word 'baby' gets assigned the value 0.99995000299985 for the quality 'small', which is high, because most people would view a baby as being small. An example of the structure of the semantic data is provided below.

word	**common**	**purchased**	**reusable**	**small**	**Is/are different sizes**	**heavy**	**Is a household item**
accordion	0.00001	0.999974	0.815768	0.00001	0.842083	0.00001	0.00001

Table 2: an example of how the semantic data is structured: how much each of these qualities apply to an accordion as an example word.

The second part of the data is how well the patients with aphasia performed on word retrieval tests. This data is synthetic, since I was unable to retrieve the patient data used in Grasemann et al. (2021). The synthetic data used here consisted of percentages, with the percentages signifying the nodes that are to be deleted, along with all of their connections. The percentages used were 20%, 40% and 60%.

For the smaller model, which was created for the purpose of being able to repeat all four phases 10,000 times, a small part of the testing data was used. Ten words were used for training, each with the first ten values from both the semantic features and the phonetic features for that word. The testing data consisted of another ten words, five of which were used to give the first ten values of the phonetic data as input, with the matching semantic data as expected output, and the other five were used to test the other way around.

2.3 Training and testing

To train the model, 608 of the words and their values from Grasemann et al. (2021) were used to create a list of tuples, with each tuple containing the values of one word. The other thirty words were saved for testing, but were imported from Microsoft Excel into PyCharm, the software that was used to create the model, at the same time in the same manner. In the training stage, the input was comprised of both the phonetic and the semantic values of the words, inputting one word at a time. During the testing stage, half of the words in the 'testing' list were used to test whether giving only the phonetic values would output the correct semantic values, and the other half was used to test whether giving only the semantic values would output the correct phonetic values, also inputting one word's values at a time. The first time the model was run, was to see how well the model would perform without the aphasia component, which would mean, it the outcomes were correct, it could mimic a human brain. This was done to test if the model itself worked correctly before drawing any conclusions based on the aphasia component. After that, the aphasia component was added in by deleting, at first, 20% of the nodes and all of its connections. Then, the trained model was run again on the testing data to see how the node deletion affected the output. The same was done for 40% and 60% node deletion.

3. Results

The outcomes of running through all 608 words used for training once and then testing the model with 30 words are not very promising. When presenting the model with phonetic input only, the model predicted, on average, 1.75% of the semantic nodes correctly. A node's value was seen as 'correct' when it differed less than four decimals from what the value was supposed to be. When presented with semantic input only, the model predicted 14.81% of the phonetic nodes correctly on average. Here, a node was seen as 'correct' if the node's value was the closest to the right phonetic encoding. For example, if a node signifies the manner of the sound, it could, according to the encoding of the features, potentially have the values 0.0, 0.1, 0.3, 0.5, 0.8 or 1.0. If the node's output is 0.4628, its closest match in the list of possible values is 0.5. If the node's output is supposed to be 0.5, the output is seen as 'correct'.

When running the model again, but looping through all of the training words twice, the phonetic output, generated by inputting semantic values, remained the exact same as when looping through the training words once. For the semantic output however, differences were found in 13/15 testing words. Since looping through all of the words another time means more training, and thus a model that should perform better, it was to be expected that the percentage of correct values would rise compared to the run in which the training data was only looped through once, but that was not the case. On 8/15 words, the percentage of correct values was lower than on the first run. The differences were small, with an average difference of 0.60625%. For 5/15 words, the percentage of correct values did rise, with an average of 2.87%. The average percentage of correct semantic values is 1.48%, which is lower than the average when looping through the training words only once.

The model was then run another time, this time looping through the training data 28 times. This yielded even more peculiar results. Like in the previous run, all of the phonetic output values remained the exact same as in the first run. The semantic output, however, did differ. For all of the 15 semantic testing words, the percentage of correct values was lower than in the initial run. This time, the differences were larger than those observed when comparing looping through the training data once and twice; the average difference was 1.18%. The average percentage of correct semantic values was 0.45%. An overview of these outcomes and the outcomes described above can be found in the table below.

	1x	**2x**	**28x**
% correct phonetic values	14.81	14.81	14.81
% correct semantic values	1.75	1.48	0.45

Table 3: an overview of the percentages of correct values for looping through the training data a certain amount of times

When including the testing data in the training set, the outcomes do not differ much from when the testing words are not included in the training set. The average difference between including the testing data in the training set or not is 0.6% for the semantic output. For the phonetic output, there was no difference.

When running the model without any training, the semantic results get noticeably better. On average, 95.63% of the semantic outcomes were correct. For the phonetic results, there was, once again, no difference.

The smaller model was used for three runs. In the first, the testing words were looped through once. When inputting semantic values only, and thus generating phonetic output values, the model got 0.69% of the values correct, for all of the words. When inputting phonetic values only, and thus generating semantic output values, it got 0.5% of the values correct for four out of five testing words, and 0.25% for one out of five testing words. This averages out to 0.45% for all five semantic testing words. The same criteria for correctness were applied here as in the normal model. The model was then run a second time, looping through all of the training words 1000 times. The results remained the exact same as when looping through the training words once. Then, the model was run once more, looping though the training words 10,000 times. This also did not yield any new or improved results. A table visualizing these results is provided below.

	1x	**1000x**	**10.000x**
% correct phonetic values	0.69	0.69	0.69
% correct semantic values	0.45	0.45	0.45

Table 4: an overview of the percentages of correct values for looping through the training data a certain amount of times

The model was also run with the aphasia component added in. It was run four times, with varying degrees of node deletion. In all of these, the training data was looped through once, and is therefore to be compared to the results that that yielded. Firstly, it was run with 20% of the nodes deleted. Of the semantic output, only 1.77% of the values were correct. The criteria for correctness remained the same as in the other runs. Of the phonetic output, 14.81% of the values were correct. These percentages are, remarkably, higher than and the same as those without any node deletion, which were 1.75% correct semantic values and 14.81% correct phonetic values.

When deleting 40% of all nodes, the percentage of correct phonetic values remains the same as when deleting 20% of all nodes: 14.81%. The percentage of correct semantic values does go down, to 1.22%. This is lower than the percentage of correct semantic values when no nodes are deleted, as opposed to when 20% of the nodes are deleted.

With 60% of all nodes deleted, the percentages of correct values are the exact same as when 40% of all nodes are deleted.

Lastly, when 80% of all nodes are deleted, the percentage of correct semantic values falls to 0.8%. The percentage of correct phonetic values still remains 14.81%, as is the case for the other runs with a certain percentage of nodes deleted. This also means that the percentage is the same as the percentage of correct phonetic values when all nodes remain intact. A table displaying all of these percentage is provided below.

	20%	**40%**	**60%**	**80%**
% correct phonetic values	14.81	14.81	14.81	14.81
% correct semantic values	1.77	1.22	1.22	0.8

Table 5: an overview of the percentage of correct values for different degrees of node deletion

4. Discussion

The results showed that this model does not work as it should. All outcomes, no matter how many times the training words have been looped through, are over 85% wrong, except when there is no training at all. In all cases, the percentage of phonetic values was higher than the percentage of semantic values, once again with the exception of the run where no training took place. This can for the most part be explained by the criteria for correctness. For the phonetic values, the closest match in the list of possible options is selected as the output of that node, after which the value is compared to the expected value for that node. For the semantic values, the number is rounded off to four decimals and then compared to the expected value. This means that the criteria for the phonetic values are a lot looser than those for the semantic values, and it therefore makes sense that the percentage of correct phonetic values is higher than the percentage of correct semantic values.

Another thing that is remarkable about the results is that the percentage of correct phonetic values remains the same throughout all runs. If the model were to work properly, those values would have changed depending on the percentage of node deletion or how many times the training data has been looped through, but that is not the case here. Even at 80% node deletion, the percentage remains at 14.81% on average. Currently, there is no explanation for why the model behaves in this manner.

Furthermore, the percentage of correct semantic values under different circumstances is also confusing. It would make sense for the percentage to increase with more training, but as shown in the results, that is not the case here. When looping through the training data twice, the percentage is 0.27% lower than when looping through the training data once. And another decrease in the percentage is seen when looping through the data 28 times, but this time the percentage drops by 1.03%. An increase in the percentage of correct semantic answers can be seen when no training takes place at all, however. The smaller model does not clarify things unfortunately. The percentage of correct semantic values remains the exact same throughout the three runs, looping through the training data once, 1000 times and 10,000 times. Since the percentage of correct phonetic values is also static across the three runs, and both percentages are low, it could be due to the model being too small to offer any useful insights.

Lastly, it is also noticeable how the percentage of correct semantic values is higher when 20% of the nodes have been deleted than when no nodes are deleted. The difference is small, only 0.03%, but it was expected that the percentage would lower when nodes are deleted. This might be connected to the fact that without any training, the semantic outcomes are much better than with training. For 40%, 60% and 80% node deletion, the percentage does lower to under the percentage of correct semantic values without node deletion. The percentage of correct semantic values is the same for 40% and 60% node deletion, so the decline is not linear, as could be expected, but it does lower again to 0.8% when comparing the percentage for 60% node deletion and 80% node deletion, so it does not bottom out at 1.22%.

The fact that the model did not work as expected could be due to several different reasons, or a combination of them. Firstly, it could be due to a lack of data. In Boersma et al. (2021), the model was run with 10,000 new sounds, meanings or pairings. The data used in here contains only 630 words, which is significantly less than 10,000.

Another reason could be that this model has not been run with 10,000 training words, which would mean looping through the same words multiple times. This is because of time constraints, but it could have important consequences for the outcomes. This is, however, less likely than the explanation above, since the smaller model showed no difference between running the model once or running the model 10,000 times. This could, however, also be due to the scale of the smaller model.

Thirdly, there could be a mistake in the implementation of the model that was missed during the construction and several rounds of debugging of the model.

Because the model does not work as expected, it is very difficult to judge its ability to model aphasia. For the results that were gathered regarding aphasia, it can be said that, since the performance of the model did worsen, at least partly, it does have some capability of modelling word retrieval issues. However, only the percentage of correct semantic values was affected by the node deletion, and even that showed unexpected results, with the percentage for 20% node deletion being higher than no node deletion, and the percentage remaining the same for 60% node deletion and 80% node deletion.

In future research, after getting the model to work for this purpose and testing it with real patient data, it could be adapted to model recovery from aphasia over time, like Grasemann et al. (2021) attempts to do. Other applications for future research could include adapting it for bilingualism, which was also done in Grasemann et al. (2021), and comparing it to monolingual usage. In the future this model could also perhaps be used to model other aspects of aphasia, such as deficits in grammar or pronunciation. Firstly, however, it should be looked into why this model does not function as it should in its current state and how it can be adjusted so it can work as it should. There were several things that could have been done in this research to try to get better results, like changing the correctness criterium for the semantic output from having to match the correct answer up to four decimals to having to match the correct answer up to one decimal, but that was not possible because of time restraints.

In conclusion, the model this paper presents does not work in its current state. When testing the model, it did not get a lot of the expected phonetic values correct (< 15% on average per word), and even less of the semantic values (<2% on average per word, not counting the run where no training took place). Looping through the training data multiple times does not improve the performance of the model, it either worsens (as is the case for the semantic values) or it remains the same (as is the case for the phonetic values). The performance of the model only increases significantly when no training takes place, but only for the semantic outcomes. The smaller model did not yield any useful insights, which might be due to the scale of the model. The aphasia aspect also does not work as it should. It does lessen the percentage of correct responses for the semantic values, but the phonetic values remain the same throughout, no matter the percentage of nodes deleted. The percentage of correct semantic values when 20% of the nodes are deleted is, unexpectantly, higher than when no nodes are deleted. The fact that the model does not work could be explained by a lack of data, not running through the data enough, or a mistake in the model itself. In the future, it could be investigated how to get this model to work properly.

5. References

Bhogal, S., Teasell, R., & Speechley, M. (2003). *Intensity of Aphasia Therapy, Imact on Recovery*. *34*(4).

Boersma, P., Chládková, K., & Benders, T. (2022). Phonological features emerge substance-freely from the phonetics and the morphology. *Canadian Journal of Linguistics, 67*(4).

Grasemann, U., Peñaloza, C., Dekhtyar, M., Miikkulainen, R., & Krian, S. (2021). Predicting language treatment response in bilingual aphasia using neural network-based patient models. *Sci Rep, 11*(10497).