Understanding Interpersonal Relationships with an LSTM

Mara Fennema
7021461

July 6, 2020

Abstract

This research looks at the number of individuals where an LSTM can still understand interpersonal relationships. This was done by using both standard and bidirectional LSTMs, varying the number of individuals, while keeping the number and types of relationships the same. The results show a steady decline in accuracy when the number of individuals increases. Additionally, the accuracies of multiple models trained on a dataset of the same number of individuals can vary greatly, sometimes up to 0.762. It was concluded that the maximum number of individuals for a unidirectional LSTM is at 12 individuals, and that of a bidirectional LSTM is at 8, which shows that both models are not capable of fully understanding such relationships.

1 Introduction

Interpersonal relationships can be quite difficult to maintain, but the relation between to individuals can be quite simple to understand, such as that of a parent and their child. As humans, we have the ability to remember such relationships, not just those of a parent and their child, but others too. Relationships like friends, enemies, lovers and colleagues, to name just a few. Dunbar (1992) discovered that humans are, on average, able to remember around 150 different individuals and their relationships. The goal of this research is to find out where the maximum lies for a computer, if it is comparable to the number defined by Dunbar.

Keysers et al. (2019) looked at how to measure compositional generalisation, for they believed the results of machine learning algorithms were often very bad due to the absence of a realistic benchmark. In their research, they proposed a Distribution-Based Compositionality Assessment (DBCA), which measured the adequacy of a dataset split for measuring compositional generalisation. In other words, it checked if the distribution over the train and test set is adequate. DBCA uses a setting where all sentences in the dataset are comprised of atoms and compounds, with atoms being the primitive elements used in the sentences, and multiple atoms together create compounds. To ensure optimal learning while still being able to conclude whether or not the algorithm is able of compositional generalisation, they propose that the trainingset should contain all existing atoms, but not all compounds. This structuring of the dataset was then tested using three different machine learning algorithms, none of which were able to generalise compositionally.

Hupkes (2018) looked at two different types of neural networks (TreeRNNs and RNNs) to see which one would yield higher results with a dataset of arithmetic language. These arithmetic sequences consisted of a number of elements, numbers ranging from -10 to 10, the mathematical operators + and -, and brackets (and). There were many sequences of differing length, and both the recursive TreeRNN and the recurrent RNN were trained and tested using this dataset. They concluded that RNNs could be trained to compute the meaning of arithmetic expressions and

generalise these meanings to expressions of greater length, and that the TreeRNN nearly perfectly approximated the recursive solution of the arithmetic sequences.

Paperno (2018) did exploratory research as to how an LSTM generalises compositional interpretations, specifically regarding to interpersonal relationships. This was tested with both a standard RNN and a Long Short-Term Memory RNN (LSTM), with both left-branching phrases (e.g. A's friend B) and right-branching phrases (e.g. B friend of A). The RNN failed at the task of understanding recursive relationships, whereas the LSTM, for a small dataset of four individuals and only left-branching phrases, could generalise recursively, if it was trained on a very extensive dataset of such recursive data.

As was stated before, the goal of this research is to find the number of people where an machine learning algorithm can no longer understand the interpersonal relationships in a dataset. A good algorithm to attempt this with is a LSTM, as was shown by Paperno (2018). It is predicted that this breaking point of the LSTM will lie below the 150 of Dunbar's number (Dunbar, 1992). Dunbar linked the number of individuals and their relationships that could be remembered to the size of one's brain, for he also researched it for other animals, not just humans. While an LSTM is a specific type of architectural structure of a neural network, which are modelled after human brains, the current state of the art is not at the same level as that of the human brain. Therefore, it is predicted that the breaking point of the LSTM will be below to the breaking point of the average human brain. Seeing as this is exploratory research, it is impossible to create a more specific hypothesis than this, for there is no literature to base it on.

2 Method

In order to find the breaking point where the LSTM can no longer understand the interpersonal relationships, an LSTM that understands these relationships had to be implemented. This was done using Python 3.7 (Van Rossum, 2009), specifically the Pytorch package (Paszke et al., 2017). Before implementing the LSTM, the dataset of individuals and their relationships needed to be generated. How this was done is explained in Section 2.1, and how the implemented LSTM works is explained in Section 2.2. Both the code to generate the data as the code for the LSTM were based on the code used by Paperno (2018).

2.1 Data generation

The code used for data generation code creates a world of a given number of individuals their relationships. The relationships within this world are parent, child, friend and enemy. The code works in such a way that when individual A's parent B is true, B's child A is true, too. The relationships of enemy is randomly generated in such a way that if A's enemy B is true, B's enemy A is also true. The same concept is implemented for the relationship of friend. Because both relationships are initiated separately from each other, it is possible that A's friend B is true, and A's enemy B is also true, but the chance of that occurring is slim. Because of the implementation of the relationships, only even numbers of individuals can be used. As Paperno (2018) showed that left-branching phrases are easier learnt by an LSTM, this research only focuses on those, and keeps right-branching phrases out of the scope of this research.

The phrases such as `A's child B` are reduced to short strings, each word being represented by one letter. So, the aforementioned sentence `A's child B` becomes `ascb`, where the `a` and `b` stand for `A` and `B` respectively, and `sc` stands for `'s child`. Besides these phrases with complexity 2, i.e. pertaining to two different individuals, the dataset used also contained sentences with both complexity 1 and complexity 3. Sentences of complexity 1 are simply the individuals' names, so

the sentence `a` represents individual `A`. Sentences of complexity 3 are a bit longer and reference 3 individuals, but only two by name. An example of such a sentence would be `A's parent's friend C`, represented by the string `aspsfc`. The name of the second individual is omitted, in order to ensure that focus of the sentence is on the relationships specified, and not just on the end of the sentence. The LSTM was implemented in such a way that all three types of sentences should be understandable.

For the LSTM to function properly, an input and an output must be defined. The goal is that the LSTM outputs the character denoting the individual, so the last character of the previously explained strings. The input of the LSTM will thus be all the previous characters of the string. So, in the case of `aspb`, the input will be *asp* and the output will be `b`, and a similar case for a phrase of complexity 3, where the input would be in the shape of `aspsf`, and the output would be `c`. For the phrases with a complexity of 1 (e.g. `a`), the last character is also the only character, so there the input is the same as the output.

The data-generator is structured in such a way that a list of tuples can be created of all the relationships in the dataset with a given complexity. The first element of the tuple is the input string, and the second element is the output string for the LSTM. From this list of all the relationships, the train, validation and test sets were constructed. To ensure an accuracy that was as high as possible, the trainingset contains all the strings of complexity 1 and 2, and 80% of all strings of complexity 3. Other percentages have been tried as well, but 80% yielded the highest accuracy. This division is in line with the research by Keysers et al. (2019), with the atoms being the phrases of complexity 1 and 2, and the compounds being the phrases of complexity 3. The trainingset contains all the strings of complexity 1 and 2, in order to ensure that it knows all of the individuals and the base relationships. Both the validation and test set only contain strings of complexity 3, which are not used in the training set. The ratio between the validation and test set is 55:45, this both being of the remaining 20% of the list of strings of complexity 3. Thus, the training set contains 11% of of strings from the list of complexity 3, and the test set contains 9% of strings from this same list.

As an LSTM uses sequences of numbers, not strings, each string was converted to such a sequence of integers. Each integer in the input vector represents a single character. For example,, the strings `asp` and `bsp` could become the vectors `[0, 1, 2]` and `[3, 1, 2]`, respectively. Thus, each character gets its own unique identifying index. A similar thing was done for the ground truths, where a smaller amount of indices was used, for there are less individuals than total number of characters, as some characters denote relationships. Thus, in the input vector the individual `a` could be represented by the number `0`, but in the output vectors it could be `1`. However, given that it is a classification problem, the LSTM needs to know how to classify each input, but it is not required that when the input in string form is the same as the output, this is also the case for the integer form. It is only required that each character in the string is consistently transformed into integers for the input, and consistently transformed for the output, but these values do not need to be exactly the same.

2.2 LSTM

As was described in the beginning of Section 2, the LSTM used was created utilising Pytorch. The input of the LSTM is a vector of integers representing a phrase, and the output of the LSTM is a list with predictions of the output being each label possible. The index of the maximum value is used to get the label the LSTM deems most likely.

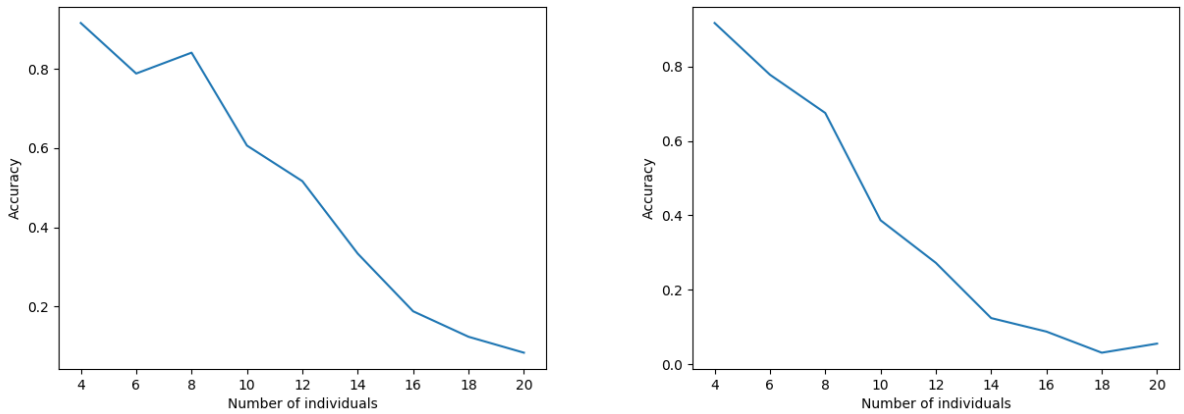
The learning rate of the LSTM was set to e^{-3} and a weight decay of $9 \cdot e^{-3}$, for these parameters were found to result in the highest accuracy, and the loss function used was a cross-entropy loss function. A cross-entropy log function, also known as the log-loss, calculates the loss on a logarithmic

scale, where the higher the predicted chance for the correct label is, the lower the loss. In this way, the predictions that are both confident and wrong get penalised heavily.

During the training phase of the model, curriculum learning is implemented. Curriculum learning ensures that the difficulty of the input for the first number of epochs is as easy as possible (i.e. only using inputs of complexity 1), and then slowly increasing the difficulty. By doing this, the model can first get a grasp of the different individuals, followed by each individual's direct relationships with other people, by introducing them to phrases of complexity 2. Lastly, after those have all been trained for a multitude of epochs, the phrases of complexity 3 were be introduced.

To find the breaking point of the understanding of the LSTM, the number of individuals was gradually increased. It was started at 4 individuals, as that was the value for which Paperno (2018) found positive results. All of the models were evaluated by calculating the accuracy of the predictions of the test set, and the number of individuals got increased until this accuracy dropped dramatically. For each number of individuals, ten different models were trained, each with a randomly generated seed, and the average accuracy of these models was taken as their final accuracy. Besides standard LSTMs, bidirectional LSTMs were also trained for all the number of individuals, to see if a difference in accuracy would occur.

3 Results



(a) Standard LSTMs (b) Bidirectional LSTMs

Figure 1: The accuracies for both standard and bidirectional LSTMs, plotted for 4 to 20 individuals.

In Figure 1, the average accuracies of all the models are shown for each number of individuals. The accuracy for four individuals is at 0.91 for both the standard and bidirectional LSTM. For 20 individuals, this value is is 0.083 for the standard LSTM, and 0.055 for the bidirectional LSTM. As these values were less than 0.1, increasing of number of individuals was stopped after 20 individuals. Both the average for the standard LSTM as the bidirectional LSTM decline in steady manner, starting at the same point and gradually declining to a value at less that 0.1 when 20 individuals are used. Notable is that for the standard model, shown in Figure 1a, the value of the average accuracy for 6 individuals, 0.788, is lower than the average accuracy of 8 individuals, which is 0.842, thus increasing with 0.054. A similar thing happens at the end of the bidirectional LSTM, where the accuracy at 20 individuals is slightly higher than when there are 18 individuals, but that difference

is only 0.015. Another thing to note in the graphs in Figure 1, is that for most data points, the accuracy for the standard LSTM is higher than that of the bidirectional LSTM.

As was described in Section 2.2, for each number of individuals, ten different models were trained, and the average accuracy of each of these models was calculated. In Table 1, the accuracies of each of the models for the standard LSTM with 14 individuals is shown, together with the corresponding seeds used in the torch module. The same data of all other models, both standard as bidirectional, can be found in the Appendix. Notable in this table is that the accuracies of the models for the same number of individuals can differ greatly. One model resulted in an accuracy of 0.762, whereas another had an accuracy of 0.0, and, together with the accuracies of all other models, resulting in an average of 0.517. As can be seen in the tables in the Appendix, this difference was not always as large as in this particular case, but a similar difference was often present, both in the standard models as in the bidirectional ones.

Seeds used	**Accuracy**
53742635	0.571
533962705	0.0
2953382612	0.477
4177529557	0.571
2803547569	0.762
2218135072	0.190
3140730933	0.095
3363176955	0.095
567613884	0.048
2664190617	0.524
Average Accuracy	**0.517**

Table 1: Accuracies of the standard model with 14 individuals

4 Discussion

As is shown in Section 3 and in the Appendix, the accuracies of the models trained for each number of individuals vary a lot. With 14 individuals in the standard model shown in Table 1, the largest difference between two accuracies being 0.762. Some differences in the accuracies is expected, as the training, validation and test sets are randomly initiated, and each model works with a randomised seed. However, to get an accuracy of 0.0 is very unexpected, for that means that all of the classifications in the test set were incorrect, even after it was trained for a multitude of epochs using curriculum learning and a cross-entropy loss function. It could be possible that the ground truths of the test set just happened to be mostly the same specific individual, thus this one being underrepresented in the trainingset. If this were the case, that could point to overfitting of that specific model, more testing is required to be able to conclude that.

As is visible in Figure 1, the models' accuracy decreases when the number of individuals is increased. This is almost always the case, with the exceptions being the increase from 6 to 8 individuals for the standard LSTM resulting in an accuracy increase of 0.054, and at the increase from 18 to 20 individuals for the bidirectional LSTM, resulting in an accuracy increase of 0.015. This is possibly related to the difference in accuracies between the models of the same number of individuals, as is described above. Seeing as with the same number of individuals the models' accuracies can differ greatly, it is possible that it just so happened to be that the 10 models created for the case of 8 or 20 individuals had a relatively high accuracy, thus resulting in a higher average

accuracy than would be the case if more models were created. Or, in a similar manner, the models for the case of 6 or 18 individuals just happened to be lower than the average accuracy if more models had been created. However, when looking at the graph, the data point for the 6 or 8 individuals seems to be more in line with the other data points than the point for the 8 or 20 individuals, respectively. This could point to the accuracies for 8 and 20 individuals being outliers, but more testing should be done before that can be concluded.

Additionally, the accuracies for both the standard and bidirectional LSTMs for 20 individuals is quite low at 0.083 en 0.055. In the case of 20 individuals, the value of chance is 0.05, so the model is performing slightly above that. However, when looking at the significant decrease in accuracies calculated for the other models, it is clear that the interpersonal relationships for 20 individuals is clearly above the LSTM's maximum capacity of understanding. If the breaking point of the LSTM would be defined as the value of chance, this means that it would be at around 20 individuals. However, at this point, the accuracy is incredibly low, so a breaking point where the accuracy is still somewhat usable would be at around 12 individuals for the standard LSTM, and at 8 for the bidirectional LSTM, for this is where the accuracy is still above 0.5, thus being correct more than half of the time in its predictions.

Lastly, as is mentioned in Section 3, the standard LSTM seems to perform better than the bidirectional LSTM. One possible cause of this is due to the fact that the dataset created consists of left-branching phrases, as is described in 2.1. This means that the only individual present in the phrase is the very first character in the input string. This means that in the standard LSTM, the starting point of the calculation is also the most telling part, namely, the one individual present in the phrase that is identified. For the bidirectional LSTM, half of the calculation, i.e. the half that starts at the end of the phrase, does not know from what individual the relationship needs to stem until it gets the very last piece of information. This could possibly negatively influence the accuracy.

The hypothesis, as described in Section 1, predicted that the breaking point would lie below the breaking point defined by Dunbar (1992) for humans, which is at 150. As is described above, the breaking point for the LSTM, is between 8 and 12 individuals, depending on bidirectionality of the LSTM. Or, if the breaking point were to be defined as the last point above chance, it is around 20 individuals. This is indeed lower than the human breaking point, which lies as 150, thus the hypothesis can be confirmed.

However, the breaking point of the LSTM is significantly lower than that of a human, so it would be interesting to see if there is a possibility of increasing the accuracy for more individuals, possibly with another type of machine learning algorithm, such as the TreeRNN used by Hupkes (2018), or by structuring the dataset differently. Other possible future research would be to see if, when training more than ten models and calculating their average accuracy, if the lines in the graphs would be more resembling a straight line, with less presumed outliers. Also, training the model with a significantly increased number of individuals could yield interesting results to see if the accuracy stays at chance, or if when given a large number of individuals, the algorithm results in an accuracy lower than chance. Or, that is consistently stays at about 0.05, even when the value of chance is lower than that. Another expansion of this research would be to change the way the dataset is generated in such a way that also an odd number of individuals is possible, which is not the case for the current implementation.

In conclusion, the the breaking point of number of individuals for this LSTM lies at either 12 for the unidirectional LSTM, and at 8 for the bidirectional LSTM. At this point, the accuracy is greater than 0.5 on average, but some models have lower accuracies.

References

Dunbar, R.I.M. (1992). "Neocortex size as a constraint on group size in primates". In: *Journal of Human Evolution* 22.6, pp. 469 –493. ISSN: 0047-2484. DOI: https://doi.org/10.1016/0047-2484(92)90081-J. URL: http://www.sciencedirect.com/science/article/pii/004724849290081J.

Hupkes D., Veldhoen S. Zuidema W. (2018). "Visualisation and 'Diagnostic Classifiers' Reveal How Recurrent and Recursive Neural Networks Process Hierarchical Structure". In: *Journal of Artificial Intelligence Research*. DOI: https://doi.org/10.1613/jair.1.11196.

Keysers, Daniel et al. (Dec. 2019). "Measuring Compositional Generalization: A Comprehensive Method on Realistic Data". In: *Unpublished*.

Paperno, Denis (2018). *Limitations in learning an interpreted language with recurrent models*. arXiv: 1809.04128 [cs.CL].

Paszke, Adam et al. (2017). "Automatic differentiation in PyTorch". In:

Van Rossum G., Drake F. L. (2009). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace. ISBN: 1441412697.

Appendix

For an overview of all the code used, see `https://github.com/maradf/wordmeaning`.

Tables standard models

Seed used	**Accuracies**
1815894398	0.833
4011654815	1.0
2654792507	1.0
1244031616	0.667
2097838237	1.0
3525482108	1.0
3998013926	1.0
3571046819	1.0
1995146287	0.667
4067362136	1.0
Average Accuracy	**0.917**

Table 2: Accuracies of the standard model with 4 individuals

Seed used	**Accuracies**
3820382194	0.667
4216274121	0.667
1832279898	0.667
1724412701	0.778
3603256500	0.778
3357605669	0.889
3767776845	1.0
1400673806	0.889
4158063787	0.667
4074352732	0.889
Average Accuracy	**0.789**

Table 3: Accuracies of the standard model with 6 individuals

Seed used	Accuracies
4008194979	0.917
3061614617	0.583
3162152575	1.0
2856917375	0.833
1403075781	0.833
2048339479	0.833
3318730027	0.75
3296826900	0.917
1987244704	1.0
4065838595	0.75
Average Accuracy	**0.842**

Table 4: Accuracies of the standard model with 8 individuals

Seeds used	Accuracy
4142428618	0.333
4143584068	0.533
1749539812	0.733
3789183230	0.333
144356516	0.733
2663889340	0.667
2820900575	0.667
3198531999	0.667
2103559425	0.6
2571335671	0.8
Average Accuracy	**0.607**

Table 5: Accuracies of the standard model with 10 individuals

Seeds used	Accuracy
2670670264	0.111
3410780380	0.667
273286365	0.444
907877388	0.556
1973668210	0.556
4125574545	0.611
469977383	0.611
1804424321	0.389
1144952848	0.5
767632223	0.722
Average Accuracy	**0.517**

Table 6: Accuracies of the standard model with 12 individuals

Seeds used	**Accuracy**
53742635	0.571
533962705	0.0
2953382612	0.477
4177529557	0.571
2803547569	0.762
2218135072	0.190
3140730933	0.095
3363176955	0.095
567613884	0.048
2664190617	0.524
Average Accuracy	**0.333**

Table 7: Accuracies of the standard model with 14 individuals

Seeds used	**Accuracy**
3440582600	0.083
752150208	0.083
2733207875	0.042
2843574753	0.042
4160443141	0.125
449672625	0.042
110044722	0.208
3674383012	0.167
2640172974	0.083
1174332481	0.0
Average Accuracy	**0.188**

Table 8: Accuracies of the standard model with 16 individuals

Seeds used	**Accuracy**
3316444014	0.154
1222784746	0.192
296306658	0.038
2587798984	0.154
2888600579	0.346
2621287482	0.078
3945714104	0.115
1152508991	0.078
99694822	0.078
1260354928	0.0
Average Accuracy	**0.123**

Table 9: Accuracies of the standard model with 18 individuals

Seeds used	Accuracy
2593018036	0.034
854617776	0.069
2596649823	0.103
1830916401	0.034
2789789136	0.103
2516684566	0.069
1580331591	0.103
3685530284	0.103
4190566548	0.207
49937797	0.0
Average Accuracy	**0.083**

Table 10: Accuracies of the standard model with 20 individuals

Tables bidirectional models

Seeds used	Accuracy
3730777409	0.333
4037173616	1.0
3244497724	1.0
643722334	1.0
2427012786	1.0
3922415840	1.0
2085010773	1.0
760548469	0.833
703633938	1.0
5778677	1.0
Average Accuracy	**0.917**

Table 11: Accuracies of the bidirectional model with 4 individuals

Seed used	Accuracies
694119582	0.667
632821785	0.778
1280222604	1.0
1970653190	0.667
2597132700	0.889
1944975383	0.222
3467185115	0.889
464280465	1.0
1943616838	0.778
2925459738	0.889
Average Accuracy	**0.778**

Table 12: Accuracies of the bidirectional model with 6 individuals

Seed used	**Accuracies**
969006221	0.5
76810757	0.583
1451627395	0.833
543420924	0.583
1689301785	0.667
50130808	0.833
2772165788	0.583
113666872	0.583
609078604	0.583
2381993770	0.75
Average Accuracy	**0.675**

Table 13: Accuracies of the bidirectional model with 8 individuals

Seeds used	**Accuracy**
2890029978	0.467
1110298312	0.067
3688463203	0.333
4037508376	0.067
1902869953	0.6
1939668247	0.333
2419275477	0.467
3520998996	0.8
45889126	0.067
3965113380	0.667
Average Accuracy	**0.387**

Table 14: Accuracies of the bidirectional model with 10 individuals

Seeds used	**Accuracy**
692928388	0.056
3090106004	0.167
3888067560	0.611
641903430	0.056
3051256347	0.167
4220426089	0.667
1223757389	0.0
1863789166	0.222
3711489798	0.167
195663617	0.611
Average Accuracy	**0.272**

Table 15: Accuracies of the bidirectional model with 12 individuals

Seeds used	**Accuracy**
1706973869	0.0
3149689724	0.190
3334162961	0.333
1235604171	0.238
2613743237	0.048
3067619913	0.095
1965435362	0.095
3006543106	0.142
1771130222	0.048
964679462	0.048
Average Accuracy	**0.124**

Table 16: Accuracies of the bidirectional model with 14 individuals

Seeds used	**Accuracy**
895272131	0.125
2375876811	0.042
922803535	0.292
3480723699	0.25
2965597583	0.042
4236302938	0.292
2716031909	0.208
2848047914	0.125
191069138	0.458
1142287440	0.042
Average Accuracy	**0.088**

Table 17: Accuracies of the bidirectional model with 16 individuals

Seeds used	**Accuracy**
2847613825	0.078
1771961486	0.038
541257753	0.038
3484718032	0.038
3710481168	0.0
430422815	0.0
1210848342	0.0
1340435345	0.038
1265693243	0.038
2279761242	0.038
Average Accuracy	**0.031**

Table 18: Accuracies of the bidirectional model with 18 individuals

Seeds used	**Accuracy**
965638482	0.034
1767917295	0.0
2854025374	0.0
811909444	0.069
2787429435	0.034
934955761	0.034
99623042	0.034
1339960308	0.069
4019359600	0.103
4089385780	0.172
Average Accuracy	**0.055**

Table 19: Accuracies of the bidirectional model with 20 individuals