Language Production Dynamics with Recurrent Neural Networks

Jesús Calvillo[1,2] and **Matthew W. Crocker**[1]
Saarland University[1]
Saarbrücken, Germany
Penn State Applied Cognitive Science Lab[2]
University Park, PA USA
{jesusc,crocker}@coli.uni-saarland.de

Abstract

We present an analysis of the internal mechanism of the recurrent neural model of sentence production presented by Calvillo et al. (2016). The results show clear patterns of computation related to each layer in the network allowing to infer an algorithmic account, where the semantics activates the semantically related words, then each word generated at each time step activates syntactic and semantic constraints on possible continuations, while the recurrence preserves information through time. We propose that such insights could generalize to other models with similar architecture, including some used in computational linguistics for language modeling, machine translation and image caption generation.

1 Introduction

A Recurrent Neural Network (RNN) is an artificial neural network that contains at least one layer whose activation at a time step t serves as input to itself at a time step $t + 1$. Theoretically, RNNs have been shown to be at least as powerful as a Turing Machine (Siegelmann and Sontag, 1995; Siegelmann, 2012). Empirically, in computational linguistics they achieve remarkable results in several tasks, most notably in language modeling and machine translation (e.g. Sutskever et al., 2014; Mikolov et al., 2010). In the human language processing literature, they have been used to model language comprehension (e.g. Frank et al., 2009; Brouwer, 2014; Rabovsky et al., 2016) and production (e.g. Calvillo et al., 2016; Chang et al., 2006).

In spite of their success, RNNs are often used as a black box with little understanding of their internal dynamics, and rather evaluating them in terms of prediction accuracy. This is due to the typically high dimensionality of the internal states of the network, coupled with highly complex interactions between layers.

Here we try to open the black box presenting an analysis of the internal behavior of the sentence production model presented by Calvillo et al. (2016). This model can be seen as a semantically conditioned language model that maps a semantic representation onto a sequence of words forming a sentence, by implementing an extension of a Simple Recurrent Network (SRN, Elman, 1990). Because of its simple architecture and its relatively low dimensionality, this model can be analyzed as a whole, showing clear patterns of computation, which could give insights into the dynamics of larger language models with similar architecture.

The method that we applied is based on Layer-wise Relevance Propagation (Bach et al., 2015). This algorithm starts at the output layer and moves in the graph towards the input units, tracking the amount of relevance that each unit in layer l_{i-1} has on the activation of units in layer l_i, back to the input units, which are usually human-interpretable. For a review of this and some other techniques for interpreting neural networks, see Montavon et al. (2017); for related work to this paper see Karpathy et al. (2015); Li et al. (2015); Kádár et al. (2017); Arras et al. (2016); Ding et al. (2017).

Our analysis reveals that the overall behavior of the model is approximately as follows: the input semantic representation activates the hidden units related to all the semantically relevant words, where words that are normally produced early in the sentence receive relatively more activation; after producing a word, the word produced activates syntactic and semantic constraints for the production of the next word, for example, after a deter-

Proceedings of the Eighth Workshop on Cognitive Aspects of Computational Language Learning and Processing, pages 17–26
Melbourne, Australia, July 19, 2018.

miner, all the nouns are activated, similarly, after a given verb, only semantically fit objects are activated; meanwhile, the recurrent units present a tendency for self-activation, suggesting a mechanism where activation is preserved over time, allowing the model to implement dynamics over multiple time steps. While some of the results presented here have been suggested previously, we present a holistic integrative view of the internal mechanics of the model, in contrast to previous analyses that focus on specific examples.

The next subsection describes the semantic representations used by the model. Section 2 describes the language production model. Section 3 presents the analysis. Discussion and Conclusion are presented in sections 4 and 5 respectively.

1.1 Semantic Representations

The semantic representations were derived from the Distributed Situation Space model (DSS, Frank et al., 2003, 2009), which defines a *microworld* in terms of a finite set of *basic events* (e.g., *play(charlie,chess)*). Basic events can be conjoined to form *complex events* (e.g., *play(charlie,chess)∧win(charlie)*). However, the microworld poses both hard and probabilistic constraints on event co-occurrence, where some complex events are very common, and some others impossible to happen.

Frank et al. (2009) defined a microworld consisting of 44 basic events centered around three people. Then they built a *situation space* by sampling 25,000 observations, where each observation is encoded by setting basic events that are the case to 1 and 0 otherwise (see Table 1). The resulting matrix encodes then all knowledge about the microworld, where each column, also called *situation vector*, represents the meaning of each basic event in terms of the observations in which the event is true. Finally, they reduced the dimensionality of the situation vectors to 150 dimensions using a competitive layer algorithm.

The language production model of Calvillo et al. (2016) uses the same microworld as Frank et al. (2009), however, the situation vectors were converted to *belief vectors*. Each dimension of the latter is equal to the conditional probability of each basic event given the original 25k-dimensional situation vector associated to each sentence[1]. The result is a 44-dimensional vector where each dimension gives an intuition of the state-of-affairs that is being represented. For example, for the sentence "Charlie plays chess.", the dimension corresponding to the basic event *play(charlie,chess)* would have a value of 1.0, the basic event *play(charlie,bedroom)* would also have a value of 1.0 because that is the only place where chess can be played, nonetheless, the dimension of *play(heidi,chess)* would be less than 1.0 because Heidi does not always play chess whenever Charlie does.

	play(ch,chess)	play(ch,hide&seek)	play(ch,soccer)	...	manner(win,diff)
observation$_1$	1	0	0	...	1
observation$_2$	0	1	0	...	1
...	.	.	.	...	.
observation$_{25000}$	0	1	0	...	0

Table 1: Situation Space.

2 Language Production Model

The model architecture can be seen in Figure 1. It consists of a 45-dimensional input layer, containing the semantic representation dss of the sentence to be produced, plus one bit indicating the model to produce an active (1) or passive (0) sentence.

At each time step t, activation of the input layer propagates to a 120-dimensional hidden recurrent (sigmoid) layer[2]. This layer also receives a copy of its own activation h_{t-1} at time-step $t-1$ (zeros at $t = 0$) through *context units*; and the identity of the word mon_{t-1} produced at time-step $t-1$ (zeros at $t = 0$) through *monitoring units*, where only the unit corresponding to the word produced at time-step $t-1$ is activated. More formally, activation of the hidden layer is given by:

$$h_t = \sigma(W_{ih} \cdot dss + W_{hh} \cdot h_{t-1} + W_{mh} \cdot mon_{t-1} + b_h) \quad (1)$$

where W_{ih} is the weight matrix connecting the input layer to the hidden layer, W_{hh} is the weight

[1]This vector is computed by calculating the dot product between the situation space matrix and the original $25k$-dimensional situation vector, and then normalizing each dimension of the resulting vector by the sum over the dimensions of the original $25k$-dimensional situation vector.

[2]While the model by Calvillo et al. (2016) uses an htan activation function, here we use a sigmoid activation because it simplifies the analysis, however there was no difference in performance between the two configurations.

matrix connecting the hidden layer to itself, W_{mh} is the matrix connecting the monitoring units to the hidden layer, and b_h corresponds to the bias unit of the hidden layer.

Then, the activation of the hidden layer h_t is propagated to a 43-dimensional softmax output layer, yielding a probability distribution over the vocabulary:

$$output_t = softmax(W_{ho} \cdot h_t + b_o) \quad (2)$$

where W_{ho} is the weight matrix connecting the hidden layer to the output layer and b_o is the vector corresponding to the output bias unit.

The word produced at time-step t is defined as the one with highest probability. The model stops when a period has been produced.

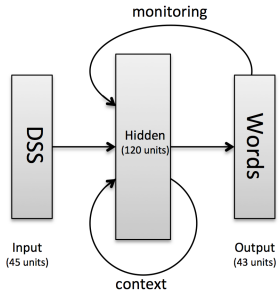

Figure 1: Model architecture.

2.1 Examples Set

The dataset that was used consists of a set of pairs $\{(dss_1, \varphi_1), \ldots, (dss_n, \varphi_n))\}$ where each dss_i corresponds to a belief vector plus one bit indicating the model to produce an active sentence (1) or a passive one (0); and $\varphi_i = \{sent_1, \ldots, sent_k\}$ where $sent_j$ is a sentence, a sequence of words $word_1, \ldots, word_n$, expressing the information contained in dss_i. Each set φ_i represents all the possible sentences that express the information contained in dss_i and in the expected voice.

The sentences are those generated by the microlanguage defined by Frank et al. (2009). This microlanguage consists of 40 words that can be combined into 13556 sentences according to its grammar. The grammar was minimally modified by introducing the determiners "a" and "the", and adding a period to the sentences, leaving a total of 43 vocabulary items.

Sentences that expressed unlawful situations according to the microworld rules, and therefore whose situation vectors were empty, were discarded; leaving a total of 8201 lawful sentences and 782 unique DSS representations. Each dss_i is related on average to 6.91 ($\sigma = 7.13$) sentences, with a maximum of 130.

2.2 Training and Evaluation

The model was trained using cross-entropy backpropagation (Rumelhart et al., 1986) with weight updates after each word. All weights on the projections between layers were initialized with random values drawn from a normal distribution $\mathcal{N}(0, 0.1)$. The weights on the bias projections were initially set to zero.

During training, the monitoring units were set at time t to what the model was supposed to produce at time $t-1$ (zeros for $t = 0$). During testing, the monitoring units are set to 1.0 for the word that is actually produced and 0.0 everywhere else.

The model was trained for a maximum of 200 epochs, each epoch consisting of a full presentation of the training set, which was randomized before each epoch. Each item in this set is a pair $(dss_i, sent)$, where $sent$ is a sentence related to dss_i, such that there is one training item per sentence related to each dss_i. We employed an initial learning rate of 0.124 which was halved each time there was no improvement of performance on the training set during 15 epochs. No momentum was used. Training halted if the maximum number of epochs was reached or if there was no performance improvement on the training set over 40 epochs.

The model was evaluated using a 10-fold cross-validation schema, with 5 testing conditions assessing different levels of generalization. A full report can be seen in Calvillo et al. (2016). For a given semantic representation dss_i, a Levenshtein similarity value was obtained comparing the sentence produced with the most similar sentence in φ_i. The performance was very high, obtaining an average across conditions of 97.1% in similarity scores, with 88.57% of perfect matches.

3 Production Dynamics

We based our analysis on Layer-wise Relevance Propagation (Bach et al., 2015). The algorithm consists of identifying for each unit in layer l_i, the units in the layer immediately before l_{i-1} that are most important for the activation of that unit. The process starts with the units in the output layer and moves toward and up to the input units, similar to the backpropagation algorithm.

An aspect that facilitates the analysis of this ar-

chitecture is that the activation of all layers is positive, ranging from 0 to 1. Then, the difference between activation or inhibition of any unit onto another is given by the sign of the connection weight between them. Thus, units inhibiting a particular unit u_i will be those with a negative connection weight to u_i, and activating units will be those with a positive connection weight to u_i.

Having this in mind, we performed the analysis. In this architecture the output layer depends solely on the activation of the recurrent hidden layer. Thus, we will first analyze the influence of the hidden layer onto the output layer, and later we will see how monitoring, input and context units affect production via the hidden layer.

3.1 Word-Producing Hidden Units

As the first step, we would like to know which hidden units are most relevant for the production of each word. We begin by identifying the hidden layer activation patterns that co-occur with the production of each word. In order to do so, we fed the model with the training set. For each training item, the model was given as input the corresponding semantic representation, and at each time step the monitoring units were set according to the corresponding sentence of the training item. This is very similar to one epoch of training, except that no weight updates were made. During this process, for each time a word had an activation greater than 0.2, the activation of the hidden layer was saved. This value was chosen in order to record activation patterns where the target word was clearly activated. At the end, for each word o_k we obtained a set of vectors, each vector corresponding to a pattern of activation of the hidden layer that led to the activation of o_k. Then we averaged these vectors, obtaining a vector that shows which hidden units are active/inactive during the production of o_k, in general and not just for a single instance, providing us with a more general perspective of the dynamics of the model for each word.

Having these patterns, we can further infer the direction and magnitude of their effect by looking at the connection weights that connect the hidden layer to the output layer.

A hidden unit h_j having a high average activation a_j when producing a word o_k means in general that h_j is relevant for o_k. However, if the weight connecting h_j to o_k is close to 0, then the production of o_k will not be so affected by h_j. In this case, it could be that h_j is only indirectly affecting the production of o_k by activating/inhibiting other words.

Intuitively, hidden units can lead to the production of o_k directly by activating o_k or indirectly by inhibiting other words. Similarly, they can lead to the inhibition of o_k directly by inhibiting o_k, or indirectly by activating other words that compete against o_k. Because of the large number of configurations that can possibly influence production, we will only focus on direct activation/inhibition.

For the case of activation, we obtain a score $A_{h_j o_k}$ conveying the relevance of hidden unit h_j on the activation of word o_k, equal to the average activation that o_k receives from h_j when o_k is produced, normalized by the sum of all activation that o_k receives:

$$A_{h_j o_k} = \frac{a_j^k w_{jk}^+}{\sum_{j'} a_{j'}^k w_{j'k}^+} \quad (3)$$

where a_j^k is the average activation of unit h_j when the word o_k is produced, and w_{jk}^+ is the positive weight connecting h_j to o_k. This score is only defined for hidden units with a positive connection weight to o_k, which we call activating units.

Inhibiting hidden units are units with negative weights to a word o_k. For inhibition the average activation of an inhibiting hidden unit during the production of o_k is expected to be close to 0. Then, the connection weight is irrelevant, as the product would be close to 0 as well. Thus, for inhibition we do not take into account the average activation, but rather its complement. That is, for each hidden unit h_j, with average activation a_j, we obtain $1 - a_j$ and multiply it by the corresponding connection weight. The result gives us the relevance regarding inhibition of each hidden unit on a particular word:

$$I_{h_j o_k} = -\frac{(1 - a_j^k) w_{jk}^-}{\sum_{j'} (1 - a_{j'}^k) w_{j'k}^-} \quad (4)$$

Based on these definitions, for each hidden unit we obtained activation/inhibition relevance scores for each word in the vocabulary. This gives us an idea of the function of each hidden unit. Examples for some hidden units are shown in Figure 2, where columns represent hidden units and rows are words in the output layer[3]. The first 5

[3]For better readability, all heatmaps presented here are also available at https://plot.ly/~jesusctCogACLL

columns show a sample of the relevance patterns in general, while the rest were chosen because they show some kind of specialization. With the exception of Figure 4, the words in these heatmaps are ordered intuitively according to syntactic and semantic similarity, having in order: determiners, nouns related to persons, nouns related to toys and games, nouns related to locations, verbs, adverbs, prepositions and the period.

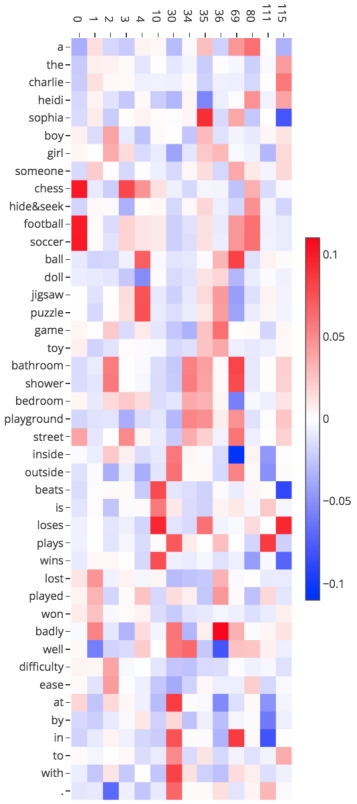

Figure 2: Relevance scores of some hidden units on output units. Red represents activation, blue inhibition.

One can see that the model takes advantage of redundancy and context sensitivity, where hidden units activate many different words depending on the context. As a result, production of a specific word depends on the combined behavior of the hidden units, where a word is produced if it receives support from several units.

Nonetheless, some units suggest a specialization (see also Karpathy et al., 2015), activating/inhibiting related words: there are units related to games (e.g., 0, 80), toys (e.g., 4, 36), places (e.g., 30, 34, 35, 69), people (e.g., 35, 115), winning/losing (e.g., 10, 115), prepositions (e.g., 30, 36, 111) and adverbs (e.g., 36).

We can also see that similar words have similar relations with the hidden neurons, suggesting syntactic/semantic categories. A clear example are synonyms, with almost identical relevance patterns, as shown by the rows corresponding to football/soccer, jigsaw/puzzle and bathroom/shower.

3.2 Monitoring Units

Having the relevance values of the hidden layer, we can infer the influence that monitoring units have on the production of each word by looking at their influence on the hidden layer.

The monitoring units feed the hidden layer with the identity of the word produced at the previous time step, where only the unit related to that word is activated (set to 1). Consequently, their effect on the hidden layer depends only on their connection weights. Then, total relevance R_{ik} of the monitoring unit i on the output unit k, is given by:

$$R_{ik} = \sum_j w_{ij} R_{jk} \tag{5}$$

where w_{ij} is the weight connecting monitoring unit i to the hidden unit j, and R_{jk} is the relevance score of hidden unit j onto output unit k, which can be activation ($A_{h_j o_k}$) or inhibition ($I_{h_j o_k}$).

Having this, we can further separate and normalize, giving activation A_{ik} and inhibition I_{ik}:

$$A_{ik} = \frac{R^+_{ik}}{\sum_k R^+_{ik}} \tag{6}$$

$$I_{ik} = -\frac{R^-_{ik}}{\sum_k R^-_{ik}} \tag{7}$$

Figure 3 presents these scores. In general, each monitoring unit promotes the activation of words that are allowed after it. Determiners activate the possible nouns that can follow them: “a” activates all toys, “game” and “girl”; and “the” activates “boy” and all locations. Nouns referring to people (e.g., “charlie”) activate all present tense verbs and the adverbs “inside” and “outside”. Games and toys activate “is”, in order to form passive constructions. Given that locations appear always at the end of the sentence, they activate the period “.”. Verbs activate words that can serve as their complements, for example “beats” activates all person-related nouns. Similarly, prepositions activate all their possible complements.

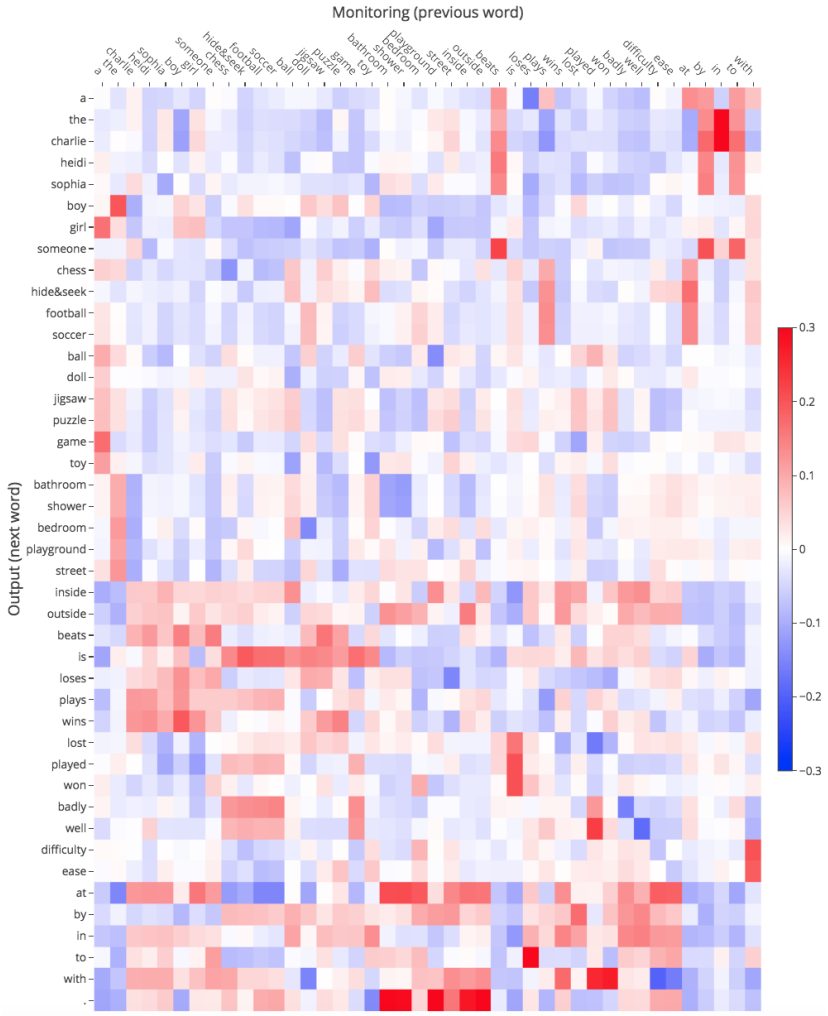

Figure 3: Relevance scores of monitoring units on output units. Red represents activation, blue inhibition.

Inhibition works very similarly, where monitoring units inhibit words that should not follow them. For example, determiners inhibit all prepositions, nouns inhibit other nouns as two nouns never occur together, prepositions inhibit also other prepositions, etc. Finally, some words inhibit themselves avoiding repetitions, for example "well" and "badly".

In general we can see that the monitoring units enforce patterns related at least to bigrams in the training set, with possibly more long distance dependencies introduced via context units.

3.3 Input Units

Using equation 5, we also computed activation and inhibition scores for the input units, where i would be in this case the index of each input unit. In contrast with monitoring units, many input units can be active simultaneously. Because of that we would like to infer not only the direction of their effect, but also its magnitude in relation to other input units. Hence, we skipped the normalization introduced by equations 6 and 7. In this case activation and inhibition correspond respectively to positive and negative values of R_{ik} in equation 5. The resulting scores are shown in Figure 4.

In general, the input units activate words that are related to their semantics. For example, the input unit *play(sophia,soccer)* activates words related to sophia, soccer and places where soccer is played (in the street). Similarly, the input unit *manner(win,difficulty)* activates "beats", "difficulty" and "with", which are used to convey this aspect. At the same time, each input unit inhibits words that are in conflict with its semantics. For example, the unit *play(charlie,hide&seek)* inhibits words concerning other games and the place where that game is not allowed (in the street). This behavior of activation and inhibition can be seen to

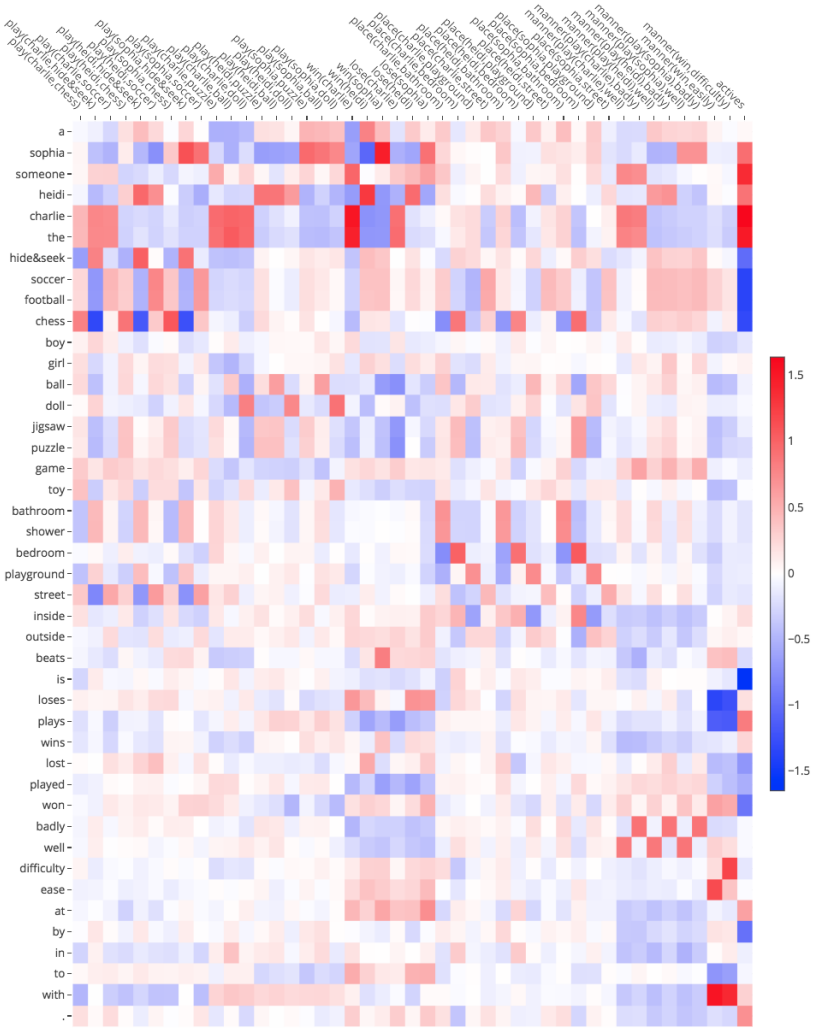

Figure 4: Relevance scores of input units on output units. Red represents activation, blue inhibition.

some degree in all input units.

Of special interest is the last input unit (*actives* in Figure 4), which marks whether the model should produce an active or a passive sentence. When this unit is active, words concerning people are activated (e.g., "charlie", "someone", "heidi"); at the same time, this unit inhibits words concerning games and passive constructions (e.g., "chess", "hide&seek", "is", "by"). Thus, production of active or passive constructions seems to be determined by giving relatively more activation to words related to people for actives, or games for passives. This seems to reflect experimental evidence that shows that more conceptually available elements are placed in more prominent grammatical roles (Bock and Warren, 1985; Ferreira, 1994). In this case, the *actives* unit learns to promote the activation of hidden units related to specific concepts depending on the voice of the sentence to be produced.

At time step 0, the activation of monitoring and context units is equal to 0. Consequently, the activation of the hidden layer at this point only depends on the input semantic representation. Then, we would expect that the input units should activate more the words that can appear at the beginning of a sentence, relative to other words. This would ensure that the words starting a sentence are correct, afterwards, monitoring and context units would be able to enforce syntactic and semantic sequential constraints, such that the resulting sentence is coherent. The words shown in Figure 4 are ordered similarly to the other figures, except that the first 10 words are those that can appear at the beginning of a sentence. As one can see, those words receive relatively more activation/inhibition than the rest. Furthermore, the *actives* unit has a very strong relevance, such that when an active

sentence is queried, the words that can start an active sentence are more activated.

In sum, the input units influence production by activating hidden units that are related to the semantics that is to be encoded, while additionally giving an idea of the word order that they should follow, specially at time step 0.

3.4 Context Units

At each time step, context units feed the hidden layer with its own activation at the previous time step, providing the model with some kind of memory over possibly unlimited time steps.

We will use the notation h_i to refer to a hidden unit i in the hidden layer, and c_i to refer to the corresponding context unit which contains the activation of h_i at the previous time step.

A way to preserve information over time is by reverberating activation over different time steps. For example, if the hidden unit h_a gets active, then the corresponding context unit c_a will be active at the next time step; if the weight connecting c_a to h_a is such that the activation of c_a causes the activation of h_a, then this would form a cycle in which h_a will be active indefinitely or until other units introduce inhibition, breaking the cycle.

We analyzed the connection weights between the context and hidden layers in order to see if these cycles were present. In such cases, the effect of h_a in the current time step would be similar to the effect of c_a in the next time step. Thus, for each pair (h_i, c_i), if the effect of c_i is similar to the one of h_i, it would mean that c_i is mainly activating h_i or units similar to h_i, forming a cycle. Note that if c_i does not activate h_i directly but other units similar to h_i, it would mean that while the activation of the specific unit might not be preserved, the model would still remain in the same area within the hidden space.

As example, for the first 15 hidden units Figure 5 presents these values. For each pair of columns, the first column represents the direct effect of each hidden unit on the output, identical to the values in Figure 2, but normalized for each hidden unit; the second column represents the effect of the corresponding context unit at the next time step, calculated using the equations 5-7, where in this case i is the index of each context unit.

The column of the right side (DimCorr) presents for all hidden units, the correlations between the relevance values of the hidden units and the relevance values of the context units, related only to each specific word; intuitively showing the degree to which activation related to each word is preserved by all hidden units. The results suggest that the context units tend to preserve activation related to most words, but to different degrees, where activation of words related to toys, locations and adverbs is preserved more than activation of words related to people. Out of the 43 words, 11 presented moderate correlation ($0.4 \leq r < 0.6, n = 120, p < 0.00001$), and 15 weak correlation ($0.2 \leq r < 0.4, n = 120, p < 0.11$).

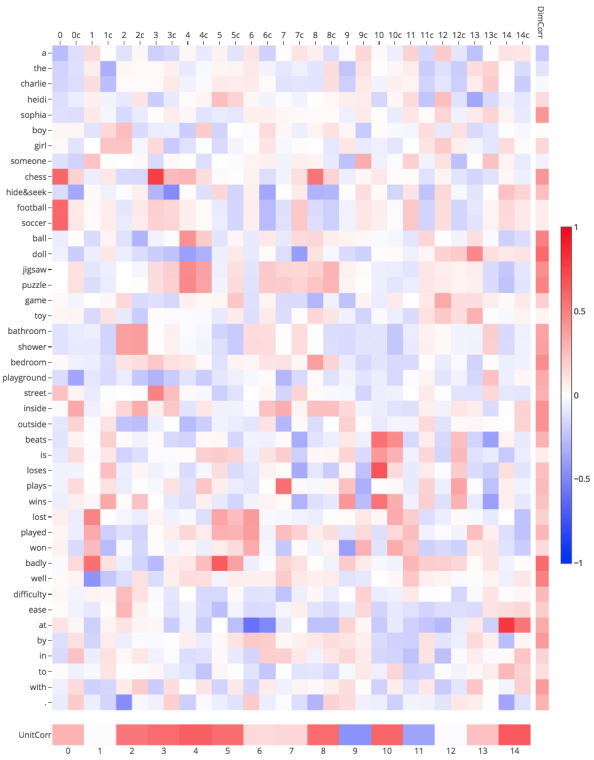

Figure 5: Relevance scores of hidden and context units. Right: correlations for each word between the relevance values of the hidden units and the context units. Bottom: correlations between all relevance values of each hidden unit and the corresponding context unit.

The row at the bottom (UnitCorr) presents correlations between all the relevance values of each hidden unit and the corresponding context unit, that is, between the values of the two columns above. As we can see, some units seem to behave like memory, while others seem to erase their content. For example, units 2, 3, 4, 5, 8, 10 and 14 have a high correlation between the hidden and context relevances, implying a cycle as described above, while units 9 and 11 present an an-

ticorrelation, which means that the context unit is actually inhibiting its corresponding hidden unit. Out of the 120 context units, 14 presented strong correlation ($r \geq 0.6, n = 43, p < 0.00001$), 26 moderate correlation ($0.4 \leq r < 0.6, n = 43, p < 0.006$) and 20 weak correlation ($0.2 \leq r < 0.4, n = 43, p < 0.2$). Regarding anticorrelation, there were 3 units with moderate anticorrelation ($-0.6 \leq r < -0.4, n = 43, p < 0.0036$) and 6 with weak anticorrelation ($-0.4 \leq r < -0.2, n = 43, p < 0.2$).

As we can see, about half of the context units have a tendency to preserve their activation, which varies according to each unit, and to the kind of information. This suggests a tangible mechanism that preserves information over time, which in the case of language is necessary in order to enforce long distance dependencies.

4 Discussion

In the above sections we separated the language production model into its different modules in order to see their function. Trying to integrate these parts into a global explanation of the internal mechanics of the model, we arrive to the following: production starts when the model is fed a semantic representation at time step 0. At this point, the semantic representation is the only source of information. Based on it, the model must produce a word that is in accordance to the semantics and that is syntactically plausible for the beginning of the sentence. As we saw, the input units seem to select the words necessary for production, and depending on the voice expected (active or passive) more activation is given to the words that can fulfill the first position. After the initial word has been produced, monitoring and context units gain influence. Monitoring units promote the production of words that can follow the previous word, and inhibit words that should not follow. At the same time, context units keep information regarding previous and current activation, suggesting a sort of memory, where information remains latent until the right time to be produced. This happens until a period is produced, which halts production.

Considering the high architectural similarity of the model of Calvillo et al. (2016) with other models of human language production (e.g., Dell et al., 1993; Chang et al., 1997), we expect that these results would also reflect their internal mechanics. Furthermore, some models used in computational linguistics also present an architecture where the main paths of computation are largely similar to the ones presented here: in language models, at each time step the word previously produced is fed to a recurrence that in turn feeds another layer yielding a probability distribution over the vocabulary (e.g., Mikolov et al., 2010); additionally, a semantics is fed into the recurrence in semantically conditioned models, such as some used in machine translation (e.g., Sutskever et al., 2014) or image caption generation (e.g., Chen and Lawrence Zitnick, 2015). One could argue that larger language models implement more complex interactions because of their higher dimensionality or the use of more complex hidden units such as LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014). Nonetheless, the individual results presented here are coherent with previous findings on larger architectures (for example, similar words are known to have similar word embeddings), suggesting that these results can be generalized to such models.

While some adaptation might be needed for larger models, the algorithm described above might serve as intuition of how those models work, and the methodology outlined here could serve to test such a hypothesis in future work.

5 Conclusion

We presented an analysis of the internal mechanism of a model of language production that uses a recurrent neural network at its core. The results show clear patterns of computation that permit to infer its internal mechanism. Because of architectural similarity, we expect that this mechanism could be generalized to other models of human language production (e.g., Dell et al., 1993; Chang et al., 1997), as well as models in computational linguistics, such as those used in language modeling (e.g., Mikolov et al., 2010) or machine translation (e.g., Sutskever et al., 2014). In future work, the methodology outlined here could also serve to test such a hypothesis.

Acknowledgments

This work was supported by DFG collaborative research center SFB 1102 'Information Density and Linguistic Encoding'. The first author was additionally supported by National Science Foundation grant BCS-1734304.

References

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2016. Explaining predictions of non-linear classifiers in nlp. *arXiv preprint arXiv:1606.07298*.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140.

J Kathryn Bock and Richard K Warren. 1985. Conceptual accessibility and syntactic structure in sentence formulation. *Cognition*, 21(1):47–67.

Harm Brouwer. 2014. *The Electrophysiology of Language Comprehension: A Neurocomputational Model*. Ph.D. thesis, University of Groningen.

Jesús Calvillo, Harm Brouwer, and Matthew W Crocker. 2016. Connectionist semantic systematicity in language production. In *Proceedings of the 38th Annual Conference of the Cognitive Science Society*. Austin, TX: Cognitive Science Society.

Franklin Chang, Gary S Dell, and Kathryn Bock. 2006. Becoming syntactic. *Psychological review*, 113(2):234.

Franklin Chang, Zenzi M Griffin, Gary S Dell, and Kathryn Bock. 1997. Modeling structural priming as implicit learning. *Computational Psycholinguistics, Berkeley, CA*, 29:392–417.

Xinlei Chen and C Lawrence Zitnick. 2015. Mind's eye: A recurrent visual representation for image caption generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2422–2431.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Gary S Dell, Cornell Juliano, and Anita Govindjee. 1993. Structure and content in language production: A theory of frame constraints in phonological speech errors. *Cognitive Science*, 17(2):149–195.

Yanzhuo Ding, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Visualizing and understanding neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1150–1159.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Fernanda Ferreira. 1994. Choice of passive voice is affected by verb type and animacy. *Journal of Memory and Language*, 33(6):715–736.

Stefan L Frank, Willem FG Haselager, and Iris van Rooij. 2009. Connectionist semantic systematicity. *Cognition*, 110(3):358–379.

Stefan L Frank, Mathieu Koppen, Leo GM Noordman, and Wietske Vonk. 2003. Modeling knowledge-based inferences in story comprehension. *Cognitive Science*, 27(6):875–910.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Akos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2017. Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics*, 43(4):761–780.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*.

Tomas Mikolov, Martin Karafit, Lukas Burget, Jan Cernock, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3.

Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2017. Methods for interpreting and understanding deep neural networks. *arXiv preprint arXiv:1706.07979*.

Milena Rabovsky, Steven S Hansen, and James L McClelland. 2016. N400 amplitudes reflect change in a probabilistic representation of meaning: Evidence from a connectionist model. In *Proceedings of the 38th Annual Conference of the Cognitive Science Society. Austin, TX: Cognitive Science Society*.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533.

Hava T Siegelmann. 2012. *Neural networks and analog computation: beyond the Turing limit*. Springer Science & Business Media.

Hava T Siegelmann and Eduardo D Sontag. 1995. On the computational power of neural nets. *Journal of computer and system sciences*, 50(1):132–150.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.