Language emergence with embeddings from a language model

Dewi E. Timman
12419273

Bachelor thesis
Credits: 12 EC

Bachelor *Cognition, Language and Communication*

University of Amsterdam
Faculty of Humanities
Spuistraat 134
1012 VB Amsterdam

Supervisor
Dr. R. G. Alhama

Institute for Logic, Language and Computation
Faculty of Science
University of Amsterdam
Science Park 107
1098 XG Amsterdam

October 15, 2024

Abstract

Contents

Chapter 1

Introduction

We are able to talk and communicate with each other. Our Languages are very complex. But how did language originate in the first place? In this paper...

1.1 Literature review

1.1.1 Agent-based linguistic communication

In this research, there will be looked at agent-based linguistic communication. This means that there is communication between two or more agents with the use of language. Usually, that means the agents need to solve a task together.

1.1.1.1 Compositionality

1.1.2 Referential communication games

To research this agent-based linguistic communication, agents can participate in a referential communication game. In such a game, agents talk about objects or other entities in a specified world. To do this, they need to come up with a language to communicate about their world. Usually, the agents do not have any prior linguistic knowledge.

A more formal definition of the referential communication game can be found in section 2.2.

1.1.3 Neural networks

In this research, three types of neural networks will be used.

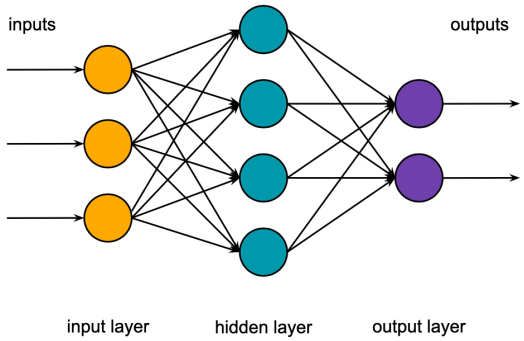


Figure 1.1: Schematic overview of a MLP. The orange nodes represent the input layer, the blue nodes the hidden layer and the purple nodes the output layer.

Multilayer perceptron. A multilayer perceptron (MLP; Rosenblatt, 1958) is one of the most simple types of neural networks. An overview can be seen in figure 1.1. It is fully connected, which means that, from two connected layers, from every node (represented by circles) of the first layer, to every node of the second layer, a connection is made (represented by arrows). The nodes can be seen as neurons and all connections have weights, reflecting the strength of that connection. There are three types of layers: an input layer, a hidden later, and an output layer. The input and output layers consist of nodes with the input and output, respectively. The hidden layer processes the inputs and converts them to the outputs. A MLP can have multiple or no hidden layers.

Thus, a MLP converts an input into an output. The amount of inputs does not need to match the amount of outputs. Therefore, a MLP can be used for converting data to a smaller size, i.e. making a dense representation. To this network type is referred as linear layer in chapter 2. In this case, the MLP does not have any hidden layers. Moreover, an activation function is used. An activation function transforms the inputs of a neuron into an output determining if the neuron should be activated or not. In this research, the hyperbolic tangent function (tanh; Szandala, 2020) is used.

Long-short term memory. A Long-Short Term Memory (LSTM; Hochreiter & Schmidhuber, 1997) is a type of recurrent neural network (RNN). This means that the input goes through the network one by one, instead of all at once (like with a MLP), i.e. word by word. This makes it possible for the model to process information with different kinds of lengths, i.e. sentences with different amounts of words where all words are transformed into vectors with the same length. Moreover, a LSTM takes previous input into account. Therefore, it takes some context into account making it a great option for generating language.

Convolutional neural networks (ConvNets). A Convolutional neural network (ConvNet; LeCun, Bengio, & Hinton, 2015a) has one or more convolutional layers. A convolutional layer has a filter (or kernel) that slides over the input data. This kernel is learned during the training of the network. Therefore, this network is especially good in extracting certain features in images. The network can also have multiple kernels to extract more information. The amount of kernels is called channels. The kernel can slide over the input one-by-one or it can skip some inputs. This is called the stride. A stride of 1 means that no inputs are skipped. After a convolutional layer, often an activation function is used. It is common to use the rectified linear unit activation function (ReLU; Fukushima, 1969).

1.1.4 Word embedding vectors

A word embedding vector is a word that is representated as a vector using numbers. For example, the word 'car' can be represented by the vector $\begin{bmatrix}1 & 7 & 4\end{bmatrix}$. This is also how Large Language Models (LLMs) represent text. LLMs learn the word embedding of a word by looking what other words appear in its context. Therefore, the more similar two words are, the closer they are in the vector space. Thus, there is a certain structure present in these word embedding vectors.

Bidirectional encoder representations from transformers (BERT). BERT is a LLM made by Google (Devlin, Chang, Lee, & Toutanova, 2018). BERT is based on the transformer network architecture (Vaswani et al., 2017). Transformers are based on attention (Bahdanau, Cho, & Bengio, 2016). Attention determines the importance of a word in a sentence relative to the other words in that sentence. Transformers therefore do not have any recurrent units, which is the case by, for example, LSTMs. Because of this, transformers have a shorter training time than LSTMs.

BERT is used as a baseline in many natural language processing tasks (Rogers, Kovaleva, & Rumshisky, 2020). The original versions have 110 and 340 million parameters, which is quite large. Therefore, a lot of smaller, different versions are available nowadays, for example, ALBERT (Lan et al., 2019). In addition, BERT is also available for other languages than English, for example, the Dutch BERTje (de Vries et al., 2019).

Rogers et al. (2020) researched what BERT actually seems to know. They found that BERT embeddings contain information about syntax, e.g. part of speech and syntactic roles. Moreover, BERT encodes semantic knowledge such as semantic roles and entity types (Tenney et al., 2019).

In this research, BERT is used to extract embeddings from the labels of labeled images.

1.2 Current research

Gap: Research with prior linguistic knowledge -> Do the agents do something with the structure/semantics already present in the word embedding vectors and can this say something about language emergence?
RQ: What is the effect of word embedding vectors from a language model on the emergence of agent-based linguistic communication from referential communication games? (simplify?) SubRQ: Do the agents communicate succesfully? SubRQ: Do they communicate using compositionality?

Chapter 2

Method

To research what the effect is of word embeddings on language emergence, two models are made: one model that has images as input, and one that has word embeddings as input. The method is based on Lazaridou, Hermann, Tuyls, and Clark (2018).

2.1 Data

A downsampled ImageNet dataset with an image size of 64×64 pixels with 3 color channels is used (Chrabaszcz, Loshchilov, & Hutter, 2017). The training dataset, which is used for this research, consists of 1,281,167 images with 997 different labels. From this dataset, 20,000 games are created by randomly sampling 5 items (1 target item and 4 distractor items) from the dataset. To ensure that there are no duplicates in the word embedding model, games that have 2 or more equal labels are resampled. The dataset is then split into a train, validation and test set with 12,000, 4,000 and 4,000 games, respectively. For each model, the dataset is further adjusted to the needs of that model. These adjustments and more detail about the datasets are discussed in sections 2.3.1 and 2.4.1.

2.2 Experimental design

In this research, agents participate in a referential communication game. The game is adapted from Lazaridou et al. (2018) and is a variant of the Lewis signaling game (Lewis, 1969). More formally, the game works as follows. There is a set of items. From this set, K items are drawn at random. One of those items is the target t. The other items are distractor items, $D = \{d_1, \ldots, d_{K-1}\}$. Together, these are the candidate items, $C = \{t \cup D\}$.

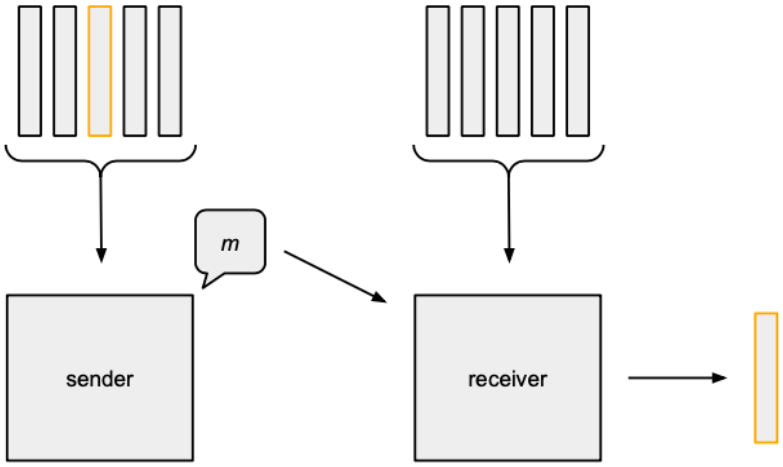


Figure 2.1: Schematic overview of the referential communication game. The item with the orange border represents the target item.

In addition, there are two agents: a sender and a receiver. The sender can see t, while the receiver sees C. However, the receiver does not know which of the items is the target t.

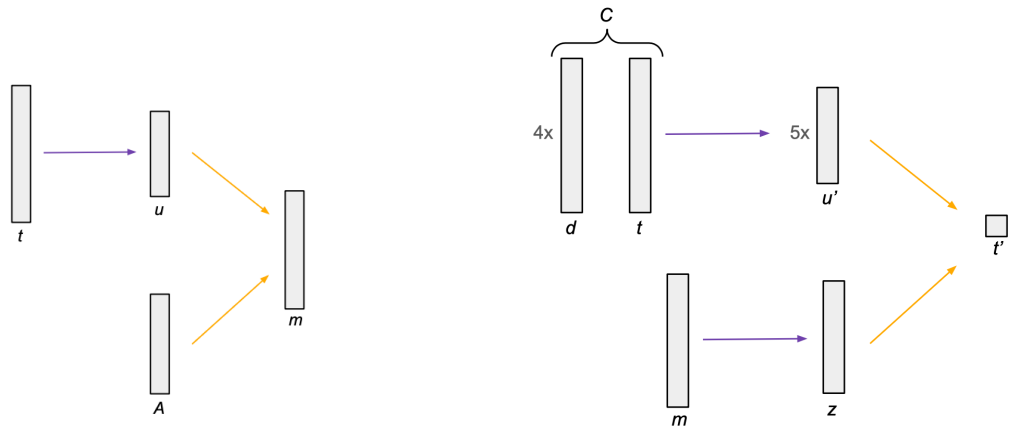
Moreover, there is a alphabet A of size N. The sender chooses one or more symbols to describe t. This is called the message m. The receiver does not know the target. However, it does see the message from the sender. The receiver then tries to guess the target image based on the message. This is the predicted target, t'.

If the receiver guesses the right target, so if $t = t'$, the agents achieve communicative succes. In return, the agents get a reward. If no communicative succes is achieved, the agents do not receive a reward. Based on this reward, the agents learn how to communicate with each other. A schematic overview of this game can be seen in figure 2.1.

2.2.1 Model architecture

The model consists of a sender and a receiver. The only connection between the two agents is the message m that is generated by the sender. Both agents consist of an encoder-decoder model. An overview can be seen in figure 2.2. The model is programmed in Python 3.12.2, using the EGG library (Kharitonov, Dessì, Chaabouni, Bouchacourt, & Baroni, 2021).

Sender. The sender receives the target item t and alphabet A and outputs the message m. To do this, it encodes the target item into a dense representation u.



(a) sender (b) receiver

Figure 2.2: Schematic overview of the data representations in the agents. Purple arrows indicate processes that are part of an encoder, while orange arrows indicate processes that are part of a decoder.

Then, u, together with the alphabet A, is decoded into m. See figure 2.2a for a schematic overview and sections 2.3.2 and 2.4.2 for more detail about the encoder and decoder for each model.

Message. The message consists of one or more words from A. The message is generated by sampling symbols from A until a stop symbol is generated or the maximum length L is reached. The words in A do not have any a priori meaning. Therefore, A can be seen as a list of possible words that the agent can use, but need to define themselves.

Receiver. The receiver receives the message m, and the set of candidate items C and outputs a predicted target t'. In order to do so, the agent first encodes each candidate item into a dense representation $u' \in U'$. In addition, m is encoded into another representation z. Finally, U' and z are decoded, and the receiver determines the predicted target item t'. See figure 2.2b for a schematic overview and sections 2.3.2 and 2.4.2 for more detail about the encoders and decoder for each model.

Communicative success. When the predicted target t' is equal to the actual target t, communicative success is achieved. The agents are then rewarded. If t'

is not equal to t, the agents do not get a reward. More formally,

$$R = \begin{cases} 1, & t = t' \\ 0, & \text{otherwise,} \end{cases}$$

where R is the reward function. The parameters of the model are estimated using the `REINFORCE` update rule (Williams, 1992) with an Adam optimizer.

Hyperparameters. In both models, the agents use a dense representation with size 50. The learning rates of both the sender and the receiver are 0.001 and the Adam optimizer has a learning rate of 0.0001. Moreover, a batchsize of 32 is used. For the image model 25 epochs are used and for the word embedding model 50. In the referential game, 4 distractor items are used. In addition, two versions of the game are run for both models, one with an alphabet size of 100 and a maximum message length of 10 and one with an alphabet size of 17 and a maximum message length of 5. This in in line with earlier research from Kottur, Moura, Lee, and Batra (2017) suggesting that non-compositional language emerged in the case of overcomplete alphabets.

2.3 Model 1: images

In the first model, the data used consists of images.

2.3.1 Image data

The dataset described in section 2.1 is further adjusted. For this model, only the images are needed. Therefore, the labels are removed from the dataset. All images are represented by a tensor (a multidimensional matrix) with shape $64 \times 64 \times 3$.

2.3.2 Image model

As described in section 2.2.1, the model consists of two agents which are both encoder-decoder models. In this section, a more detailed description of those encoders and decoders will be given.

Sender. The sender consists of an encoder that converts the target t into a dense representation u and a decoder converting u and alphabet A into a message m (see figure 2.2a). The encoder consists of an 6-layer convolutional neural network (ConvNet; LeCun, Bengio, & Hinton, 2015b) followed by a linear layer with the hyperbolic tangent activation function (Szandala, 2020). The ConvNet is not

pre-trained so that the agent gets its own representations from the image. The ConvNet has 32 filters, a kernel size of 3, batch normalization, and strides [1, 2, 1, 2, 1, 2] for each layer. The decoder consists of a single-layer long short-term memory (LSTM; Hochreiter & Schmidhuber, 1997).

Receiver. The receiver consists of an encoder that encodes the items in C into dense representation U', an encoder that encodes the message m into representation z and a decoder that takes both U' and z to determine a target t'. The first encoder, to make dense representations U' from C, has the same architecture as the encoder of the sender. However, no weights are shared between the sender and the receiver. Therefore, the encoders do have the same architecture, but are not exactly the same.

The second encoder makes a representation z from the message m using a single-layer LSTM (Hochreiter & Schmidhuber, 1997).

Finally, the decoder takes the dot product between all encoded items in U and z. It then samples the target object t' from a Gibbs distribution calculated with this dot product.

2.4 Model 2: word embeddings

In the second model, the data used consists of word embeddings.

2.4.1 Word embedding data

The dataset described in section 2.1 is further adjusted. For this model, only the labels are needed. Therefore, the image part of the dataset is removed. Furthermore, the labels are given to the BERT model (Devlin et al., 2018) to obtain word embeddings of the labels. This is done as a preprocessing step rather than a step inside the model, since we are not training the BERT model. Therefore, this is a deterministic process for all labels and this significantly speeds up the training of our model. The word embeddings of the BERT model are represented as a 768-dimensional vector.

2.4.2 Word embedding model

The word embedding model is almost the same as the image model described in 2.3.2. The only difference is the architecture of the encoders. In the word embedding model, the encoders consist of only the linear layer followed by a tanh activation function instead of a ConvNet followed by the linear layer and the activation function.

Chapter 3

Results

To evaluate the models, multiple metrics are used. First, lexicon size is used to see if the messages send differed from each other. Second, accuracy is used to see how well the communication between the agents is: did the agents create a language? Last, compositionality is used to measure how human-like the language is. The results on the validation set can be seen in table 3.1 and on the test set in table 3.2. The test set has 3992 different images and 562 different word embeddings. Both models were evaluated two times with different values for the alphabet size and maximum message length. The results will be further explained in the following sections.

3.1 Lexicon size

The lexicon size is the number of unique messages used when testing the model. It can be calculated by calculating the length of the subset of messages. A high lexicon size means that there are a lot of different messages used. A low lexicon size would mean a lot of messages are equal to each other.

model	max length	alphabet size	accuracy	topsim
image	10	100	76.9 %	0.201
	5	17	76.4 %	0.237
word embedding	10	100	86.0 %	0.055
	5	17	83.1 %	0.054

Table 3.1: Results of both models on the validation set. The topographic similarity metric is represented by 'topsim'.

model	max length	alphabet size	accuracy	topsim	lexicon size
image	10	100	77.1 %	0.195	73
	5	17	75.1 %	0.227	45
word embedding	10	100	86.8 %	0.055	56
	5	17	83.6 %	0.052	40

Table 3.2: Results of both models on the test set. The topographic similarity metric is represented by 'topsim'.

In table 3.2 it can be seen that the image model used more distinct messages than the word embedding model. Namely 73 and 45 for the image model and 56 and 40 for the word embedding model respectively. Moreover, the lexicon size is bigger when the alphabet is bigger and the maximum message length is shorter.

3.2 Accuracy

A low accuracy would mean that the agents fail to communicate well. From this, we could conclude that no language-like system emerges between the two agents. A high accuracy, however, would indicate that the agents are able to communicate and therefore created a language together. Accuracy is represented as a percentage and is calculated by:

$$\text{accuracy} = \frac{\#\text{ items where } t = t'}{\text{total }\#\text{ items}} \cdot 100.$$

When looking at table 3.1 and table 3.2, we see a higher accuracy for the word embedding model than the image model. In addition, the accuracies with a higher alphabet size and maximum message length are higher. Lastly, the accuracies are around 80%.

3.3 Topographic similarity

Topographic similarity is used to measure compositionality. It is represented by a number between -1 and 1. Here, a value of 0 means there is no compositionality at all. The more topographic similarity goes into the direction of -1 or 1, the more compositional the language is. Applying the definition of Brighton and Kirby (2006) to our situation, topographic similarity can be defined as the correlation of the distances between all the possible pairs of meaning (target items) and their corresponding message. The topographic similarity is therefore calculated by

taking the distances between all pairs of sender inputs and the distances between all pairs of messages. Then, the negative Spearmann correlation is taken.

In table 3.1 and table 3.2 all topographic similraity scores are positive. Moreover, a low topographic similarity score can be seen for the word embedding model. However, there is a higher score for the image model. In addition, we can see that when increasing the maximum message length and the alphabet size, the topographic similarity for the decreases for the image model but increases for the word embedding model.

Chapter 4

Conclusion

To see what the effect of word embedding vectors is on language emergence in a referential communication game multiple versions of the referential communication game were made. These versions are compared in the following ways: lexicon size, accuracy and topographic similarity. First, there is the question whether the agents do communicate succesfully. Second, there is the question whether the emerged language is compositional. These questions are answered in the following sections.

4.1 Lexicon size

If we compare the lexicon size with the amount of games in the test set, we can see that a lot of games got the same message. Moreover, if we take into account that the sender had 3992 different images and 562 different word embeddings as input, we can see that multiple target items generated the same message. Thus, these results indicate a high sense of ambiguity. This only becomes more evident when looking at the difference in lexicon size when comparing the same model with the different maximum message lengths and the alphabet sizes.

4.2 Accuracy

All accuracies are higher than that of a random baseline (20%). This indicates that the agents did achieve succesfull communication. In addition, a higher maximum length gives a higher accuracy, which is in line with earlier research of Skyrms (2010, p.131) and Lazaridou et al. (2018). This also makes sense from a logical point of view: the more symbols an agent can send the more distinct messages it can send. This is because the amount of possible messages is N^L, where N is the size of the alphabet and L the maximum message length. Therefore, the number

of distinct messages is dependent on the maximum length.

4.3 Topographic similarity

The results indicate that in the image model, some similar objects receive similar messages. However, in the case of the word embedding model, this is not the case. An interesting finding is that the topographic similarity increases for the word embedding model while it decreases for the image model when increasing L and N. However, this increase for the word embedding model is very small. If we look at the amount of distinct inputs of both models, we also see a big difference. This difference might explain the differences in topographic similarity for both models.

4.4 The emerged language

To conclude, a language did emerge when using word embeddings as well as images. Nevertheless, the degree of compositionality in both models is different. The agents do not do something with the structure already present in the word embedding vectors of BERT. However, for the image model a more compositional language does emerge.

Chapter 5

Discussion

5.1 LLM

5.2 Dataset

5.3 Model

References

Bahdanau, D., Cho, K., & Bengio, Y. (2016). *Neural machine translation by jointly learning to align and translate.* Retrieved from https://arxiv.org/abs/1409.0473

Brighton, H., & Kirby, S. (2006). Understanding linguistic evolution by visualizing the emergence of topographic mappings. *Artificial Life*, *12*(2), 229–242. Retrieved from http://dx.doi.org/10.1162/artl.2006.12.2.229 doi: 10.1162/artl.2006.12.2.229

Chrabaszcz, P., Loshchilov, I., & Hutter, F. (2017). A downsampled variant of imagenet as an alternative to the cifar datasets. doi: 10.48550/arXiv.1707.08819

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, *abs/1810.04805*. Retrieved from http://arxiv.org/abs/1810.04805

de Vries, W., van Cranenburgh, A., Bisazza, A., Caselli, T., van Noord, G., & Nissim, M. (2019). Bertje: A dutch BERT model. *CoRR*, *abs/1912.09582*. Retrieved from http://arxiv.org/abs/1912.09582

Fukushima, K. (1969). Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, *5*(4), 322–333. Retrieved from http://dx.doi.org/10.1109/TSSC.1969.300225 doi: 10.1109/tssc.1969.300225

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

Kharitonov, E., Dessì, R., Chaabouni, R., Bouchacourt, D., & Baroni, M. (2021). *EGG: a toolkit for research on Emergence of lanGuage in Games.* https://github.com/facebookresearch/EGG.

Kottur, S., Moura, J., Lee, S., & Batra, D. (2017). Natural language does not emerge 'naturally' in multi-agent dialog. In M. Palmer, R. Hwa, & S. Riedel (Eds.), *Proceedings of the 2017 conference on empirical methods in natural language processing* (pp. 2962–2967). Copenhagen, Denmark: Association for Computational Linguistics. Retrieved from https://aclanthology.org/D17-1321 doi: 10.18653/v1/D17-1321

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, *abs/1909.11942*. Retrieved from http://arxiv.org/abs/1909.11942

Lazaridou, A., Hermann, K. M., Tuyls, K., & Clark, S. (2018). Emergence of linguistic communication from referential games with symbolic and pixel input..

LeCun, Y., Bengio, Y., & Hinton, G. (2015a). Deep learning. *Nature (London)*, *521*(7553), 436-444.

LeCun, Y., Bengio, Y., & Hinton, G. (2015b). Deep learning. *Nature*, *521*(7553), 436–444. Retrieved from http://dx.doi.org/10.1038/nature14539 doi: 10.1038/nature14539

Lewis, D. D. K. (1969). *Convention: a philosophical study*. Cambridge, Mass: Harvard University Press.

Rogers, A., Kovaleva, O., & Rumshisky, A. (2020). A primer in bertology: What we know about how BERT works. *CoRR*, *abs/2002.12327*. Retrieved from https://arxiv.org/abs/2002.12327

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. Retrieved from http://dx.doi.org/10.1037/h0042519 doi: 10.1037/h0042519

Skyrms, B. (2010). *Signals: Evolution, learning, and information*. Oxford University PressOxford. Retrieved from http://dx.doi.org/10.1093/acprof:oso/9780199580828.001.0001 doi: 10.1093/acprof:oso/9780199580828.001.0001

Szandala, T. (2020). Review and comparison of commonly used activation functions for deep neural networks. *CoRR*, *abs/2010.09458*. Retrieved from https://arxiv.org/abs/2010.09458

Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., ... Pavlick, E. (2019). What do you learn from context? probing for sentence structure in contextualized word representations. In *International conference on learning representations*. Retrieved from https://openreview.net/forum?id=SJzSgnRcKX

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *CoRR*, *abs/1706.03762*. Retrieved from http://arxiv.org/abs/1706.03762

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*(3–4), 229–256. Retrieved from http://dx.doi.org/10.1007/BF00992696 doi: 10.1007/bf00992696