How Brain-Like is an LSTM's Representation of Nonsensical Language Stimuli?

by

Maryam Hashemzadeh

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

The representations generated by many models of language (word embeddings, recurrent neural networks and transformers) correlate to brain activity recorded while people listen. However, these decoding results are usually based on the brain's reaction to syntactically and semantically sound language stimuli. In this study, we asked: how does an LSTM (long short term memory) language model, trained (by and large) on semantically and syntactically intact language, represent a language sample with degraded semantic or syntactic information? Does the LSTM representation still resemble the brain's reaction? We found that, even for some kinds of nonsensical language, there is a statistically significant relationship between the brain's activity and the representations of an LSTM. More exceptional, a character-based LSTM's representation of pseudoword sentences is significantly correlated to EEG collected while people listened to those sentences - even though the pseudowords were not in the LSTM training data. This indicates that, at least in some instances, LSTMs and the human brain handle nonsensical data similarly.

Preface

Parts of this thesis appeared as a publication at the Proceedings of the 2020 Conference on Empirical Methods in Natural Language, Findings (EMNLP-2020 [13]).

Acknowledgements

First and foremost, I would like to thank my supervisors, Alona and Martha, for their strong and consistent support in all facets of life as a graduate student. They were looking out for my best interests and, for that, I cannot thank them enough. I feel very fortunate to have had both of them as my supervisors throughout these years and, without a doubt, it has been one of the most important factors contributing to my enjoyment of my Master's studies. I would also like to thank Carrie Demmans Epp for being on my thesis committee. Finally, thank you to all the members of the Amii and RLAI groups who are my wonderful friends with their countless thought-provoking discussions.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The human brain has a unique capacity to integrate information from one word to the next word and from one context to the next context. With the help of contextual information, humans are able to integrate the next words or sentences or even infer other people's thoughts. This ability is due to the capability of the semantic meaning representation [3], [6], [36]. To find how representations of semantic meaning are neurally encoded, we can use a mapping from computational models of language (e.g. word embeddings, neural network hidden representations) to patterns of human brain activation. Language encoding models are a strong tool to interpret language processing in the human brain as they can capture high-level semantic meaning that incorporates context.

When people listen to or read a language, brain imaging studies have shown us that the brain's activity correlates to LSTM (Long Short Term Memory) state representations for the same text [16], [33]. In those studies (and others like them) the stimuli used to test for this correlation was based on language with no errors. This implies that during language processing, for within-sample data (i.e. well-formed sentences), LSTMs and the human brain show similar representational patterns [1].

But what happens when language is out-of-distribution (e.g. nonsensical [2] sentences or pseudo-words)? Can we expect that an LSTM will still compute contextual states in a way that resembles how the human brain reacts? I.e. is

[1]Here we speak of decoding studies only.

[2]Nonsensical language is often used when measuring Event Related Potentials.

there a correlation between LSTM representations and neural activity when the stimuli is not a predictable language sample? Answering these questions could provide evidence that an LSTM is able to generalize to new data in a human-like way, even when the new data is unlike anything it encountered during training. Our answers could also help psycholinguists reason about the efficacy of nonsensical sentences and pseudo-words as syntax-only stimuli controls.

Here we use brain imaging data (Electroencephalography, EEG) collected in three conditions, *Sentence*: well-formed grammatical sentences, *Jabberwocky*: pseudo-word sentences that preserved word order, morphosyntax, and sentential prosody without lexical or compositional semantics, and *Word-list*: the words of the *Sentence* condition in a pseudo-random order without sentence prosody, syntax, or compositional meaning. We ran a character-level LSTM model on the stimuli, and trained a *decoding model* to predict the LSTM's internal representations from EEG signals. Using data from the *Sentence* condition, we corroborated previous results and showed that LSTM representations are correlated with brain activity for within-distribution language. But, when it came to nonsensical language stimuli, it was unclear if LSTM representations would still correlate to brain activity. Our original hypothesis was that LSTM representations for out-of-distribution language would no longer correlate to brain activity. However, we found that our decoding model worked quite well even when all content words of the stimuli were pseudo-words (*Jabberwocky*).

In the following, we define some terms which we use in this text repeatedly.

- Decoding: here is a mapping function between stimuli and word representation vectors.
- Nonsensical language: a language that have little or no meaning; or make little or no sense.
- Out-of-distribution language: an irregular language or a language with errors or not well-formed sentences.

1.1 Contribution

Our results show that there are similarities between the way the brain and an LSTM represent stimuli from both the *Sentence* (within-distribution) and *Jabberwocky* (out-of-distribution) conditions. More specifically, the contributions of this work include:

- Our decoding models work well in both the *Sentence* and *Jabberwocky* conditions, but not in the *Word-list* condition. This shows there is a relationship between the semantic and/or syntactic information as represented by the brain and by LSTM representations for a regular and nonsensical language.

- The syntactic signatures available in *Sentence* and *Jabberwocky* LSTM representations are similar and can be predicted from either the *Sentence* or *Jabberwocky* EEG. The results also present evidence for the lack of semantic information in *Jabberwocky* LSTM representations.

- For some LSTM representations, the decoding model's *weight maps* generalize between *Jabberwocky* and *Sentence* EEG data. The results imply that the brain's representation for the syntax in both the *Sentence* and *Jabberwocky* conditions takes a similar form, at least with respect to the syntactic information represented in earlier and middle layers.

- From our results, we can infer which LSTM layers encode semantic and/or syntactic information. We confirm using syntactic and semantic probing tasks. The probing tasks show different LSTM layers encode the different amounts of semantic and syntactic information.

1.2 Related Work

The first example of mapping brain responses onto corpus-derived representations appeared in [26]. This study encoded word meaning into vectors of word co-occurrence features. The authors showed that a trained linear regression model could predict fMRI activation in response to single concrete noun stimuli. From there, decoding models were shown to work with dependency-parse-based representations [27] and with concept-relation-features extracted from topic models [30]. Anderson et al. [2] demonstrated that decoding models can learn the pattern of the brain's response to abstract concepts/nouns.

Some of the first examples of decoding language *in context* were from [34] and [14]. The first models used a combination of (non-contextual) corpus-derived, acoustically-derived and/or hand-coded representations. Several groups then began to experiment with encoding models based on *contextual* language representations, like those in recurrent neural network (RNN) language models [16], [33], [35]. These models showed that vectors incorporating contextual information could be decoded from brain imaging data, and contextual models actually outperformed non-contextual word vectors. We confirmed those findings here.

Though there are fewer decoding models trained on EEG, there are a few recent examples. Hale et al. [12] showed that the operations performed by an RNN-grammar trained to parse sentences correlated to EEG collected while people listened to a story. Schwartz et al. [32] found connections between bi-LSTM representations and the ERPs (event related potentials) more classically used to study language in the brain. Our work adds to the new body of work showing that EEG can be a powerful data source in this space.

Chapter 2

Background Material

In this chapter, we review the background of this thesis. Firstly, we introduce Language Modeling (LM) and Long Short-Term Memory (LSTM) neural networks which are used to find representations of stimuli in this research. Then, we describe Dutch language and Electroencephalography (EEG), an electrophysiological monitoring method to record the electrical activity of the brain which respond to auditory stimuli in our study.

2.1 Long Short-Term Memory Neural Network

A recurrent neural network is a type of artificial neural network in which previous outputs are used as inputs beside the current feed of data. It is well-suited for sequential or time-series data such as natural language processing, language translation, speech recognition, and image processing. So, its outputs are influenced by prior sequences of data as they take information from both prior and current inputs. If h_{t-1} is the previous hidden state, and x_t is the current input (e.g. the word at time t), then the current hidden state h_t is:

$$h_t = f(x_t, h_{t-1}) \tag{2.1}$$

where f is usually a nonlinear function.

Long Short-Term Memory (LSTM) neural network is a type of recurrent neural network that avoids vanishing gradient problem. It can keep track of the long-term appendices in the input sequences beside allowing gradients to flow unchanged by a forget gate. A usual LSTM unit is comprised of a memory cell, an input gate, an output gate and a forget gate. The memory cell rem denoted by c_t remembers information over time and the three gates control the flow of the information into and out of the cell. The forget gate denoted by f_t decides how much information throw away from the cell state. It uses a sigmoid function to produce a number between 0 and 1 where 0 means completely throw away and 1 represents completely keep this.

$$f_t = \sigma(W_f \times [x_t, h_{t-1}] + b_f) \tag{2.2}$$

where W_f and b_t are the matrix weights and bias vectors which need to be learned during training. x_t is the input vector to the LSTM unit and h_{t_1} is the hidden state vector at time-step $t-1$.

Then by the input gate i_t, it regulates how much new information is going to store in the cell state. It consists of two parts. The first part is a sigmoid function called "input gate" on x_t and h_{t_1}:

$$i_t = \sigma(W_i \times [x_t, h_{t-1}] + b_i) \tag{2.3}$$

where W_i and b_i are the matrix weights and bias vector. The second part is a tanh layer creating a new candidate value, $\tilde{c_t}$ that is added to the state.

$$\tilde{c_t} = \tanh(W_c \times [x_t, h_{t-1}] + b_c) \tag{2.4}$$

After that, i_t and $\tilde{c_t}$ are combined to create an update to the state.

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c_t} \tag{2.5}$$

In the last step, again a sigmoid layer on x_t and h_{t-1} creates the output gate o_t. Then $tanh(c_t)$ multiplies by the output gate to produce the hidden state h_t.

$$o_t = \sigma(W_o.[x_t, h_{t-1}] + b_o)h_t = o_t \odot \tanh(c_t) \tag{2.6}$$

2.2 Neural Language Model

A statistical language model is providing a probability distribution over a sequence of words used in a text. A trained language model can predict the probability of occurrence word w_t given preceding words, $w_1, w_2, w_3, ..., w_{t-1}$. Language models can be applied on character level, n-gram level, sentence level, or even paragraph level. In an n-gram model, the probability $p(w_1, w_2, ..., w_t)$ is approximated by the chain rule:

$$p(w_1, w_2, ..., w_t) = \Pi_{i=1}^{t} p(w_i|w_1, ..., w_{i-1}) \approx \Pi_{i=1}^{t} p(w_i|w_{i-(n-1)}, ..., w_{i-1}) \quad (2.7)$$

where

$$p(w_i|w_{i-(n-1)}, ..., w_{i-1}) = \frac{\text{count}(w_{i-(n-1)}, ..., w_{i-1}, w_i)}{\text{count}(w_{i-(n-1)}, ..., w_{i-1})} \quad (2.8)$$

Then, to train the language model, it wants to maximize equation 2.7 which is equivalent to maximize:

$$\log p(w_1, w_2, ..., w_t) = \sum_{i=1}^{t} \log p(w_i|w_1, ..., w_{i-1}). \quad (2.9)$$

Also, to evaluate a language model, we normalize it and take natural exponential to find the word perplexity:

$$\exp\left(\frac{1}{t}\sum_{i=1}^{t} \log p(w_i|w_1, ..., w_{i-1})\right). \quad (2.10)$$

A neural language model is a language model based on neural networks, utilizing their ability of continuous representation to alleviate the impact of the curse of dimensionality when the number of unique words is increasing in a corpus. The neural networks can be feed-forward or recurrent. Usually, they are trained to learn a probabilistic classifier, probability distribution over a vocabulary, to predict the probability of the next words given some context, $p(w_t|\text{context})$.

To train the neural network, it typically does back-propagation of the gradient of the loss function with respect to the network parameters through the network. The context can be a fixed-size window of previous or future words with an arbitrary length T.

When a recurrent neural network is used, then the gradient is accumulated over timestep and back-propagate through time (BPTT). Then it encounters problems such as exploding gradient and vanishing gradient. To avoid this problem, usually, they truncate the gradients to T timestep and then called it T-step PBTT.

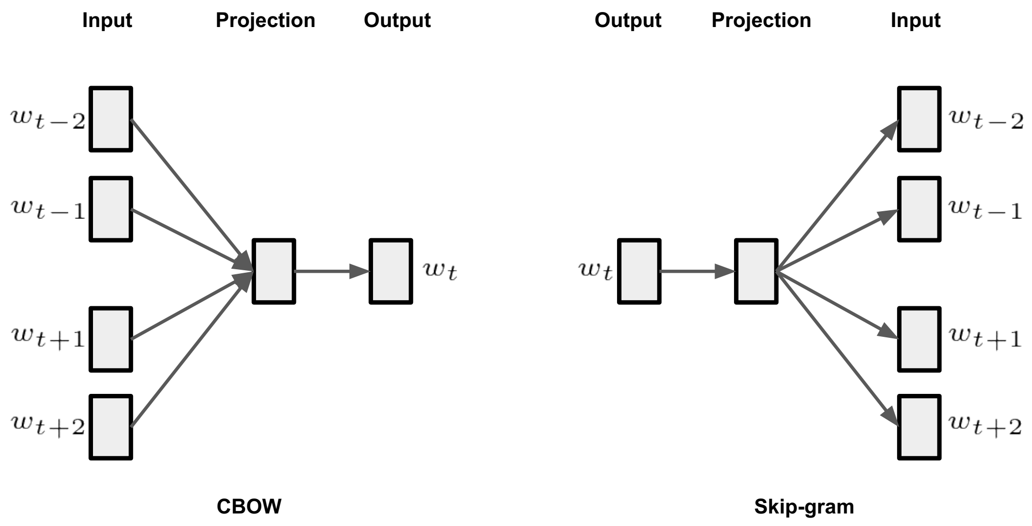


Figure 2.1: Scheme of CBOW and Skip-gram [25]. As it shows, CBOW is trained to predict the centred word using surrounding words. In contrast, Skip-gram uses the centred word to predict the surrounding words.

2.2.1 Bag-Of-Words Model

Bag-Of-Words (BoW) Model is a simple word representation that disregards grammar or word order while keeping multiplicity. It counts the occurrence of words within a document and then can be used as a feature of the given text. Continuous Bag of Words Model (CBOW) and Skip-gram are two types of BoW models in which the word representations is learned by using neural networks.

- Continuous Bag of Words Model (CBOW): In this model, the centre word is predicted by its surrounding words. So, it has a context passing to a neural network to predict the target word.

- Skip-gram Model: This model takes a target word and tries to predict its neighbouring words. For defining a neighbouring word, a window size as a hyper-parameter is considered.

2.3 Dutch language

In this research, the stimuli are in Dutch. Dutch is a West Germanic language spoken by the population of the Netherlands and about 60% of the population of Belgium. It is the third most widely spoken Germanic language and is close to English and German. Dutch orthography uses the Latin alphabet. It consists of 26 letters of the basic Latin alphabet where five letters are vowels. It also has a shallow orthography meaning that phonemes are highly predictable by seeing letters. This property helps the language model to have a good presentation when it is training by text instead of phonemes in this research.

For the vocabulary, it is German origin and because of that, it is very similar to other German languages. For grammar, it is simpler than German. Stress in Dutch words usually happens on the first syllable. This language has two genders, common and neuter, and two numbers, singular and plural.

2.4 Electroencephalography

Electroencephalography (EEG) is an electrophysiological recording method to capture the macroscopic activity of the brain in the study of human language processing. It has a scalp holding electrodes that comprise small metal discs and thin wires. The electrodes can detect very small oscillations coming from the brain electric potentials. However, a generated electric potential of an individual neuron is very tiny to recorded by EEG. Then, EEG activity reflects the summation of the synchronous activity of millions of neurons which have similar spatial orientation.

Recording signals are transmitted to an EEG system which has amplifiers, filters, and a computer monitor. EEG has a broad application in clinical circumstances to detect abnormal changes in brain activity that is helpful for diagnosing or treating brain disorders such as epilepsy, brain tumour, brain dysfunction, stroke, sleep disorders and more disorders.

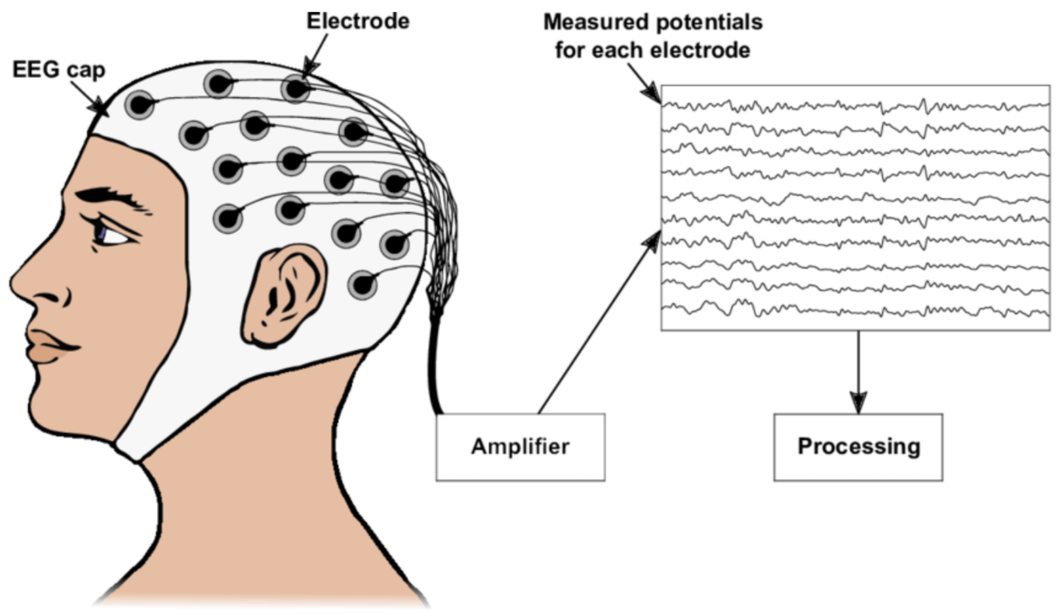


Figure 2.2: Electroencephalography from [28]. EEG is measuring by the electrical activity using electrodes that are fixed on the scalp. For each electrode, signals are amplified and can be shown in a monitor for other applications.

Chapter 3

Materials and Methods

In this chapter, we describe the dataset used in this study. Also, we present our decoding model which learns the correlation between LSTM representation and EEG signals. Finally, an evaluation method is proposed to evaluate the performance of the decoding model.

3.1 Data description

Our data was originally collected to contrast the brain's response to language samples that vary the amount of semantic and syntactic information [17]. The dataset consists of EEG recordings (64 channels, 500 Hz sampling rate) of 27 native Dutch speakers (9 males; mean age= 23). The participants listened to a native Dutch speaker in three conditions: *Sentence*, *Jabberwocky*, and *Word-list*. Each condition has 80 sentences, and all *Sentence* and *Jabberwocky* stimuli sentences share the same grammatical structure.

The *Sentence* stimuli contain two coordinate clauses and a conjunction with the structure *[Adj N V N Conj Det Adj N V N]* where *N* denotes $Noun$, *V* is $Verb$, *Adj* is $Adjective$, *Conj* is $Conjunction$, and *Det* is $DefiniteArticle$. This condition contains lexical semantics, compositional semantics, and syntax. *Word-list* consists of the same ten words as *Sentence* but in a pseudo-random order with infeasible syntactic structures (either *[V V Adj Adj Det Conj N N N N]*, or *[N N N N Det Conj V V Adj Adj]*). The *Word-list* condition leaves orthography/phonology intact and contains lexical semantics, but not compositional semantics or syntax. For *Jabberwocky*, words from the

Sentence condition are replaced with pseudo-words created with the Wuggy generator [18]. Crucially, the ***Jabberwocky* pseudo-words appear in the same order as the corresponding words in the *Sentence* condition**. The Wuggy generator alters words in a way that obeys the phonotactic and morphosyntactic constraints of a language, but eliminates semantic meaning. The *Jabberwocky* condition contains syntax and morphosyntax, which is preserved by Wuggy.

Amongst psycholinguists and cognitive neuroscientists, it is widely accepted that Jabberwocky does not contain lexical or compositional semantics, and a Jabberwocky condition is often used to control for semantics [8], [9], [15]. Anecdotally, native Dutch speakers typically cannot guess the true word when presented with the pseudo-word.

Since the original stimuli are in Dutch, we also brought them in both Dutch as the original ones and English. Here are stimuli examples on the three conditions:

- *Sentence*: Lange mannen bouwen huisjes en de lieve honden brengen planken. (Tall men build houses and the sweet dogs bring boards.) This example has the structure of *[Adj N V N Conj Det Adj N V N]*.

- *Jabberwocky*: Lalve wanzen botren raasjes en de reeve rorden brargen sponken. This one has the same structure as the structure in the *Sentence* condition, *[Adj N V N Conj Det Adj N V N]*, but real words are replaced with pseudo-words instead.

- *Word-list*: planken mannen huisjes honden de en bouwen brengen lange lieve (boards men houses dogs the and build bring tall sweet). This example has the structure of *[N N N N Det Conj V V Adj Adj]* where it serves the real words while breaks the grammatical structure.

In the *Jabberwocky* condition the determiners and conjunctions are not pseudo-words. To fairly compare the conditions, we removed these words from all three conditions during our analyses. Due to the nature of spoken language, the time-duration each of word/pseudo-word differs. To account for

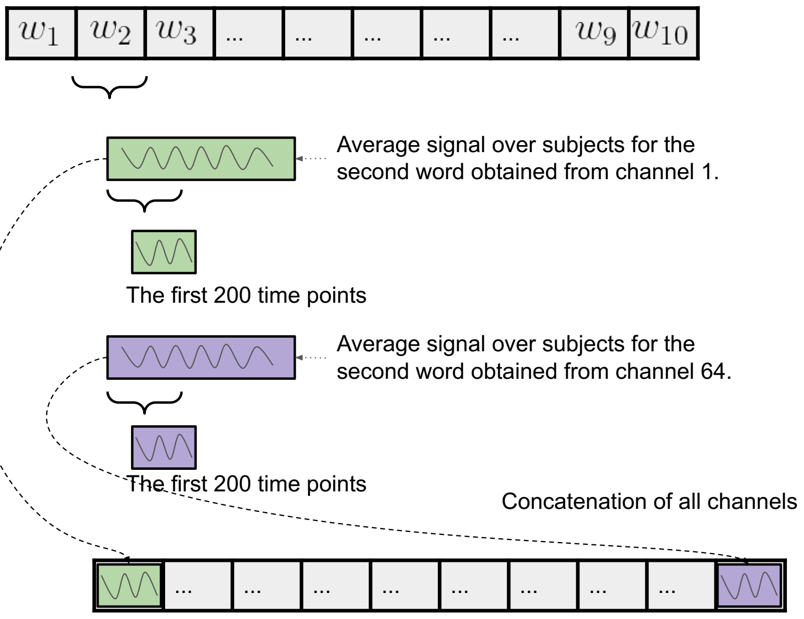

w_1
w_2
w_3
...
w_9
w_{10}
Average signal over subjects for the second word obtained from channel 1.
The first 200 time points
Average signal over subjects for the second word obtained from channel 64.
The first 200 time points
Concatenation of all channels

Figure 3.1: Concatenation of EEG signals in the model. For a given word, the average signals over all of the subjects are calculated separately at each electrode (channel). Then, the first 200 time steps of each channel are detached from the rest time steps to concatenate to the same time step of other channels.

this, we considered the first 400 ms of EEG after word/pseudo-word onset. Before we tested it for 200 ms and 600 ms and we found 400 ms time duration works better for LSTMS.

To improve the EEG's signal to noise ratio, we average the EEG recording for a given sentence across all subjects. Though this reduces participant-specific signal, we have found it to be the best way to decode from EEG data. For this data, models trained on only one subject did not perform above chance. For each word of each stimulus sentence S, we concatenated the recording from every electrode into one vector $R_t \in \mathbb{R}^{1\times D}$ where D is the total number of readings across all sensors (here $D = 12800$: 64 sensors $\times$ 200 time points). Fig 3.1 shows more details.

3.2 Decoding model

The aim of a decoding model is to find a mapping function $f(R_t) \rightarrow g(S_{1:t})$ between an EEG recording R_t of the brain's response to word w_t and a language

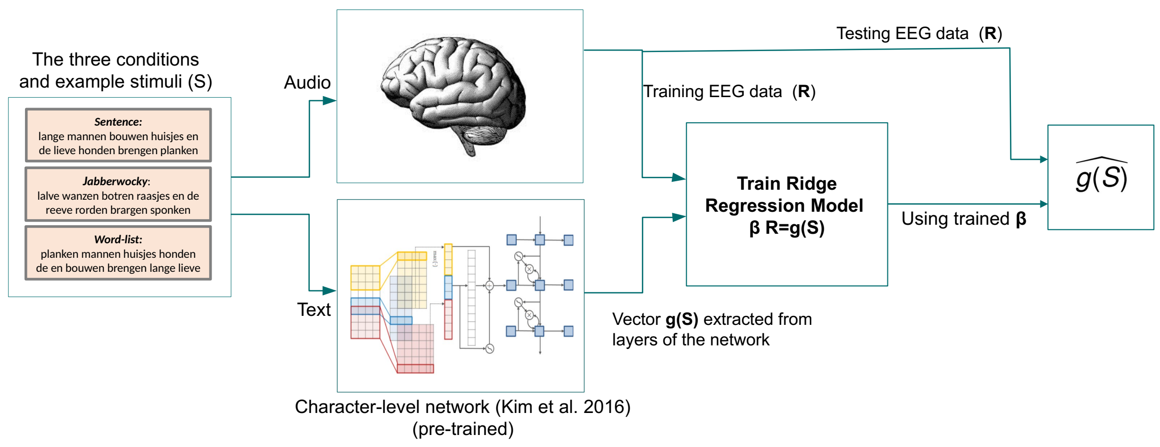

The three conditions
and example stimuli (S)
Sentence:
lange mannen bouwen huisjes en de lieve honden brengen planken
Jabberwocky:
lalve wanzen botren raasjes en de reeve rorden brargen sponken
Word-list:
planken mannen huisjes honden de en bouwen brengen lange lieve
Audio
Text
Testing EEG data (R)
Training EEG data (R)
Train Ridge Regression Model
β R=g(S)
Using trained β
$\widehat{g(S)}$
Vector g(S) extracted from layers of the network
Character-level network (Kim et al. 2016) (pre-trained)

Figure 3.2: Decoding model. Each stimulus sentence is fed to a pre-trained language model to create a non-linear context-based representation. The hidden representations for a sentence (S) are extracted from each layer $g(S)$. Our ridge regression model is trained to use the EEG signal R to predict $g(S)$.

model's representation of stimulus $S_{1:t}$ (the words of a sentence up to and including word w_t). Our methodology closely followed [16]. We instantiate our mapping function in two steps:

- $g(S_{1:t}) \in \mathbb{R}^{1\times P}$: a contextual representation of the current word, w_t. In our model, it is an LSTM's P-dimensional representation for word w_t, conditioned on context $w_1, \dots w_{t-1}$. This representation is a hidden layer from an LSTM language model.

- $f(R_t)$: a regularized linear regression to map the EEG signal R_t to $g(S_{1:t})$, $f(R_t) = R_t\beta$ where we use data to train weights β.

Figure 3.2 shows a schematic of the decoding model.

1- Specifying LSTM representations ($g(S)$): The *Jabberwocky* stimuli are made of pseudo-words that are out-of-vocabulary referring to vocabulary contained words that are not part of the normal/usual vocabulary lexicon in a natural language processing environment, and therefore will always be misrecognized. So we needed a language model that can handle out-of-vocabulary input. Character-level language models can be a good candidate since the input is character instead of word and then it can accept the inputs which are out-of-vocabulary. Character-level language model leverages sub-word information which outperforms morphologically rich languages like Germanic languages including Dutch.

We used a state-of-the-art character-level LSTM language model proposed by [19], but we added one more LSTM layer to have three LSTM layers based on the previous decoding work [16]. This model is able to encode rich semantic and orthographic features from characters only. The LSTM operates on the characters of incoming words (so it can handle pseudo-words), but it produces predictions at the word level. Each input character has its own embedding, which is concatenated and fed to convolutional layers. The convolved values are passed to a highway network, whose output is fed through three stacked LSTM layers before predicting the next word.

In the decoding analyses that follow, the $g(S)$ vectors we analyze are (1) the concatenation of the character embeddings called *Embedding* layer, (2) the concatenation of the Convolutional layers called *Conv*, and ({3-5}) the three LSTM layers called *LSTM1-3*. We will use the term *LSTM* to refer to the full character-based model, *LSTM representation* to refer to any of the $g(S)$ vectors types, and *LSTM layer* or *LSTM1-3* to refer specifically to the LSTM layers within the larger LSTM model.

We trained the LSTM on one million sentences from Dutch Wikipedia. We set the number of epochs to 40, batch size is 50, and sequence length is 20. We used a stochastic gradient descent optimizer with sparse categorical cross-entropy loss. The initial learning rate is 0.8 with inverse time decay rate 0.5.

For Dutch Wikipedia, the average test perplexity of our model is 108.12. When the inputs are the *Sentence* stimuli, the average perplexity is higher: 317.91. This is likely because the coordinate clauses within each stimulus are only 4 words long, which reduces the effective context. When the inputs are the *Jabberwocky* pseudo-words and the outputs are the corresponding *Sentence* next word, the perplexity is 325.12. These *Sentence* and *Jabberwocky* perplexities are not significantly different ($p = 0.967$). We calculated the average perplexity on the *Word-list* stimuli to be 1008.23, which indicates that (as expected) the network cannot predict the next words in the *Word-list* stimuli. This also shows that while the *Jabberwocky* and *Sentence* perplexities are higher than on Wikipedia, they are much lower than for stimuli with no contextual information.

For comparison, we also experimented with non-contextual word embeddings from [11]. This 300-dimensional model is pre-trained on Dutch Wikipedia using Continuous Bag of Words (CBOW) with position-weights.

2- Regularized linear regression ($f(R)$): We used ridge regression to test if the EEG data correlates with the word/pseudo-word representations. The regression function $f(R_t)$ is a linear transformation of R_t to predict the P-dimensional $g(S_{1:t})$: $f(R_t) = R_t\beta$ where $\beta \in \mathbb{R}^{D\times P}$.

3.3 Measuring model accuracy

We used Monte Carlo (MC) cross-validation to evaluate our decoding models. MC cross-validation affords a more stable estimate of model accuracy, and allows for statistically-sound comparisons of model performance. During each of our 500 MC samples, we swept the regression regularization parameter among the values in range $[0.1, 200]$ using 5-fold cross-validation on the training data only.

We use a 2 vs. 2 classification test to assess the performance of the learned model [10], [26]. During each cross-validation trial we randomly create groups of two from the held-out samples. Using a model fit to the training data, we produce predicted representations for the held-out samples. For simplicity, let $y_t^i = g(S_{1:t}^i)$ be the contextual representation for word w_t of sentence i. Then, for each group of 2 test samples $(S_{1:t1}^i, S_{1:t2}^j)$, we perform a *2 vs. 2 test* using the true representations (y_{t1}^i, y_{t2}^j) and predicted representations $(\widehat{y}_{t1}^i, \widehat{y}_{t2}^j)$. The 2 vs. 2 test compares the sum of cosine similarity for correctly matched the true and predicted vectors:

$$cos(y_{t1}^i, \widehat{y}_{t1}^i) + cos(y_{t2}^j, \widehat{y}_{t2}^j), \tag{3.1}$$

to the sum of cosine similarity of the mismatched vectors:

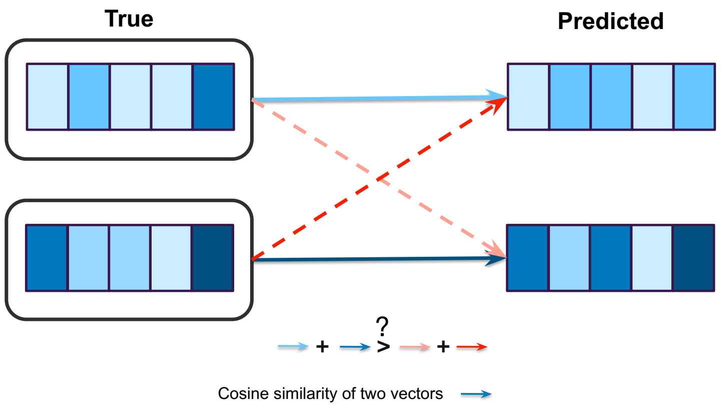


Figure 3.3: 2 vs. 2 classification test to assess the performance. It separates held-out samples into two groups. At each time, it takes one sample from each group and finds its prediction. Then, if the sum of cosine similarity of the true and corresponding predicted vectors of these two samples, light blue and dark blue arrow, is greater than the sum of cosine similarity of the true vectors and their mismatched predictions, pink and red arrow, we count one point for the 2 vs. 2 test. Its accuracy is equal to the percentage of the correct tests. Chance in this classifier is 50%.

$$cos(y_{t1}^{i}, \widehat{y}_{t2}^{j}) + cos(y_{t2}^{j}, \widehat{y}_{t1}^{i}). \tag{3.2}$$

If Eq 3.1 is greater than Eq 3.2, the 2 vs. 2 test passes. 2 vs. 2 accuracy is the percentage of correct 2 vs. 2 tests, and chance 2 vs. 2 accuracy is 0.5 (see Fig 3.3). In addition to 2 vs. 2 accuracy, we also report mean-squared-error of the learned model to see the other evaluation methods.

To test for statistical significance, we used permutation tests. The LSTM representations for the stimuli were randomly shuffled such that the true representations $g(S_t)$ were no longer correctly matched to the EEG data. We then trained and tested our decoding models as described above using more than 1000 (> 1000) random permutations. These results represent the expected distribution of 2 vs. 2 accuracy when there is no relationship between the EEG data and the LSTM representations. From that (null) distribution we can compute p-values for our observed accuracy on the un-permuted representations. We correct for multiple comparisons using the Benjamini-Hochberg-Yekutieli False Discovery Rate (FDR) procedure [4] using $\alpha = 0.05$.

For our models to perform above chance, there must be correlates of particular aspects of language (such as semantics or syntax) present in the brain activation data R, and in the corresponding contextual representation $(g(S))$. Furthermore, our decoding model assumes a linear relationship between R and $g(S)$. If our models do not perform above chance, any of the above conditions may be violated; our analyses are not designed to differentiate between the failure cases.

Chapter 4

Results

In this chapter, we are interested in comparing the representations generated by an LSTM to that of the human brain, in response to both within- and out-of-distribution language. To do so, we developed different analyses with respect to the conditions and semantic and/or syntactic information.

4.1 Experimental Questions

Our *Sentence* stimuli, which represent within-distribution language, contain semantic and syntactic information. We used two kinds of out-of-distribution stimuli: *Jabberwocky*, which was designed to have syntactic information only, and *Word-list*, which has only semantic information. We attempted to learn a mapping from EEG to LSTM representations (to test if the LSTM and brain handle the stimuli similarly). To begin, we examined the difference in the semantic and syntactic information encoded by each of the LSTM representations. Then, we developed analyses to test for a similarity in the representation of semantic and syntactic information across the experimental conditions. We investigated using the following questions:

1. Is there a difference in the semantic/syntactic information captured by the LSTM representations? (Probing tasks)
2. Can we learn a mapping from the EEG data to the LSTM representations in the *Sentences, Jabberwocky*, or *Word-list* conditions? Is there a difference in performance across the different LSTM representations? (Analysis 1: test for semantic and/or syntactic information)

Analysis	**Case**	**Train EEG**	**Train $g(S)$**	**Test EEG**	**Test $g(S)$**
1	**1**	*Sen*	*Sen*	*Sen*	*Sen*
	2	*Jab*	*Jab*	*Jab*	*Jab*
	3	*WL*	*WL*	*WL*	*WL*
2	**1**	*Sen*	*Jab*	*Sen*	*Jab*
	2	*Jab*	*Sen*	*Jab*	*Sen*
3	**1**	*Sen*	*Sen*	*Jab*	*Sen*
	2	*Jab*	*Jab*	*Sen*	*Jab*

Table 4.1: Data description for each analysis. Sen: Sentence, Jab: Jabberwocky, WL: Word-list. Analysis 1: EEG & $g(S)$ from the same condition. Analysis 2: $g(S)$ swapped between conditions. Analysis 3: EEG swapped between conditions *at test time only*.

3. If there is syntactic information present in the *Sentences* and *Jabberwocky* LSTM representations, is it exchangeable? (Analysis 2: swap the $g(S)$ conditions)
4. Do the actual *patterns* learned by the decoder generalize to EEG from the other condition? (Analysis 3: swap R at *test* time only)

The EEG analyses are summarized in Table 4.1.

4.2 Probing tasks

Probing tasks are used to ask what kind of linguistic information a neural language model is able to capture. Answering this question is not trivial as neural language models have usually multiple layers with non-linear transformations. Some benchmarks and downstream tasks are designed in a way to isolate some specific linguistic phenomena in order to measure that linguistic information encoded in a learned representation. If a probing classifier performs well on a probing task, it comes out that the learned representation encoded the linguistic information which the probing task has. Probing tasks also have been referred to as diagnostic classifiers, auxiliary classifiers or decoding.

In language processing, previous works have found that LSTM layers encode differing amounts of information about the semantic meaning and syntactic structure [24], [31]. To investigate the behaviour of our LSTM and find how

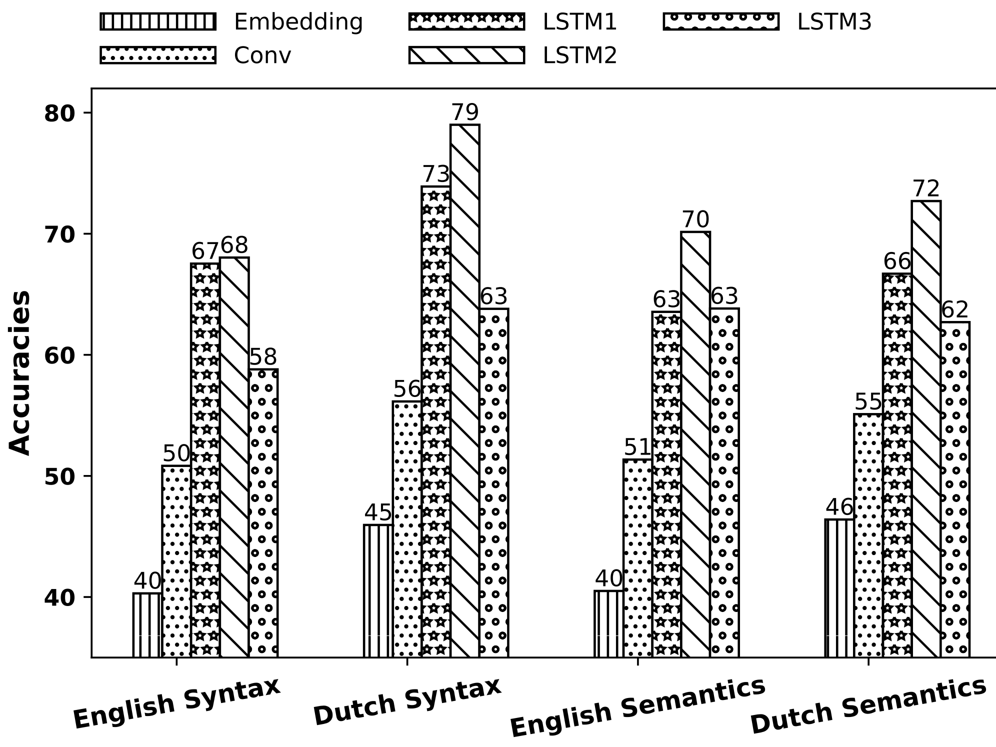


Figure 4.1: Average accuracies for the semantic/syntactic probing tasks using LSTM representations from Dutch or English LSTM language models.

much they capture different amounts of linguistic information, we used several probing task benchmarks. Because there are more available benchmarks for English, we used them besides a few available benchmarks for Dutch.

The English semantic and syntactic probing tasks are from [5], and the Dutch from [7]. A description of each task is given in Table 4.2. The tasks are from two categories, semantic and syntactic. For semantic, it includes *Tense* (present and past), *Subject number*, *Object number*, and *Coordinate inversion* (intact and modified). In *Coordinate inversion*, the dataset has sentences comprising of two coordinate clauses. In 50% of the sentences, the order of the clauses is inverted. The task is a binary task to indicate whether a sentence is inverted or not. For syntactic, the tasks are *Bigram shift*, *Tree depth*, *Top constituent sequence*, *Number* (singularity and popularity), *Part of speech*.

For the English benchmarks, we trained our LSTM architecture using English Penn Treebank (PTB) [21], to have the learned representation in English

Table 4.2: Description of the probing tasks. "En" shows the English datasets and "Du" shows the Dutch datasets. The gray and light green colours determine the tasks for the semantic and syntactic respectively.

Name	**Description**
Tense (En/Du)	*Tense of the main-clause verb (present/ past)*
Subject number (EN)	*Number of the subjects of the main clause*
Object number (EN)	*Number of the direct objects of the main clause*
Coordination inversion (EN)	*Indicate if a sentence is intact or modified*
Bigram shift (EN)	*Indicate having legal word orders*
Tree depth (EN)	*Depth of the hierarchical structure of sentences*
Top constituent (EN)	*Indicate top constituent sequence of sentences*
Number (Du)	*Indicate singularity and plurality of N/Adj/V*
Part of Speech (Du)	*Indicate the part of speech of a specific word*

in addition to Dutch. We also checked the probing task results for consistency against the Dutch results. For the Dutch benchmarks, the network is trained on Dutch Wikipedia.

For each probing task, we trained a multilayer perceptron (MLP) classifier with 2 hidden layers of 100 units. For the MLP input, we first find the learned representation vectors of each word for all of the layers in the LSTM network. Then, the average of these vectors for a sentence in a given probing task is calculated and it is passed to the MLP. The output is the predicted class of the sentence (e.g. past tense verb). Note that the sentences here are not from our stimuli, but rather from the probing tasks themselves.

The accuracy for each task appears in Table 4.3. To summarize, we show the average accuracies for the English and Dutch probing tasks in Fig. 4.1. We were reassured to see the performance of the English and Dutch LSTMs show similar patterns. Both the Embedding and Conv layers perform poorly on the semantic and syntactic tasks. We see the strongest evidence for syntax in LSTM1 and LSTM2, and the strongest evidence for semantics in LSTM2.

Table 4.3: Probing task accuracies. Each row shows the accuracies of a specific probing task described in Table 4.2. Columns correspond to the LSTM representation: "*Embedding*": Embedding layer, "*Conv*": concatenation of Convolutional layers, "*LSTM1-3*": an LSTM layers. "*Tense/En*" and "*Tense/Du*" denote the English and Dutch probing task for *Tense*, respectively.

Layers #	**Embedding**	**Conv**	**LSTM1**	**LSTM2**	**LSTM3**
Tense/En	43.2	53.2	63.2	70.7	63.9
Tense/Du	46.4	55.1	66.7	72.7	62.7
Subject number	38.8	53.5	65.5	72.1	64.3
Object number	39.5	52.1	66.8	71.7	65.8
Coord. Inv.	40.5	46.6	58.7	66.1	61.3
Bigram shift	43.1	53.1	70.8	69.4	58
Tree depth	39.3	45.6	56.3	58.6	54.3
Top constituent	38.5	53.8	75.5	76.1	64.1
Number	52.3	58.6	78.2	81.9	67.3
Part of Speech	39.6	53.7	69.8	76.1	60.3

4.3 Test for semantic and/or syntactic information (Analysis 1)

To test for the correlation of semantic and/or syntactic information between the EEG and LSTM representations, we measured the accuracy of a decoding model trained with data from the same condition. This is Analysis 1 from Table 4.1 where EEG and $g(S)$ are coming from the same conditions in both training and test. Lines in Fig. 4.2 show the results for this experiment.

Based on the probing results, for the *Sentence* stimuli we expected to see the highest performance for LSTM2 (contains semantic and syntactic information), and somewhat lower performance for LSTM1 (strong syntax performance, but lower semantics). For the *Jabberwocky* condition, we expected to see the strongest performance for the syntactically rich LSTM1 and LSTM2. For the *Word-list* condition, we were unsure if the contextual representations would work at all, given that the random ordering of words removes the sentence's context.

In the *Sentences* condition, the accuracy is statistically above chance for LSTM layers 1-3 (0.581, 0.600, and 0.569 respectively, $p < 0.05$, FDR cor-

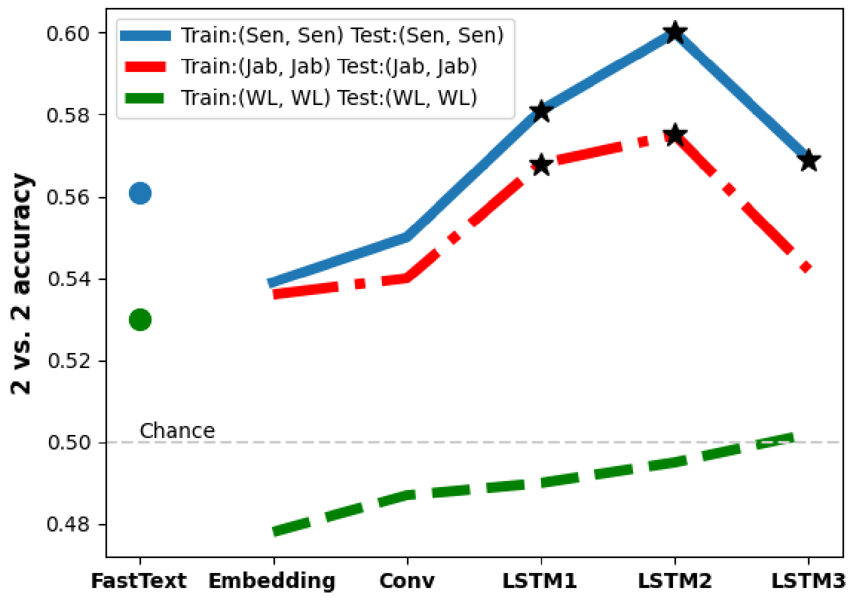


Figure 4.2: Analysis 1 (Test for semantic and syntactic information): 2 vs. 2 accuracy with $g(S)$/EEG from the same condition. The x-axis denotes LSTM representation ($g(S)$). Legend denotes EEG/LSTM representations used for train/test: (EEG condition, LSTM condition). "*Sen*": *Sentence*, "*Jab*": *Jabberwocky* , "*WL*": *Word-list*. $\star$: above chance (p < 0.05, FDR corrected). Dots show the performance for the word vectors from *FastText* [11].

rected). This matched our predictions based on the probing tasks and shows that LSTM3 has sufficient syntactic/semantic information for the decoding task. In the *Jabberwocky* condition, only the accuracies of the LSTM1 and LSTM2 are statistically above chance with (0.565 and 0.573 respectively, $p <$ 0.05, FDR corrected), which again matched our predictions based on the probing tasks. The *Sentence* condition conveys both semantic and syntactic information, and so the decoding model produces higher accuracy than the *Jabberwocky* condition, which lacks semantics. For both *Jabberwocky* and *Sentence* conditions, LSTM2 shows accuracy higher than LSTM1 and LSTM3, which is consistent with previous decoding work showing that middle LSTM layers outperformed early and late layers [16], [33].

Previous work has found that LSTM layers carry information about both the semantic meaning and syntactic structure [24], [31]. The EEG signals from the *Sentence* condition are in response to regular sentences, and so have both semantic and syntactic information. Because both information types are present, the *Sentence* condition produces higher accuracy than *Jabberwocky* condition. However *Jabberwocky* EEG does carry information, presumably about the syntactic structure. The *Jabberwocky* stimuli maintained prosodic rhythm, which supplies some hints as to e.g. part of speech. Our results show *Jabberwocky* signals can still be used to predict the LSTM layers, and so some of the information available in the EEG data correlates to what the LSTM is encoding in its state vectors.

This finding is consistent with the theory that the *Sentence* condition EEG/LSTM vectors contain both semantic and syntactic information, whereas the *Jabberwocky* condition EEG/LSTM vectors contain only syntactic information (thus there is less signal to be leveraged in the *Jabberwocky* stimuli, and lower 2 vs. 2 accuracy as a result). The *Jabberwocky* stimuli were carefully controlled to have a prosodic structure similar to the *Sentence* stimuli, and so it is reasonable to expect some syntactic signature to appear in the corresponding EEG. We also note that the perplexity of the trained LSTM run on the *Jabberwocky* stimuli to predict the next real word in the corresponding *Sentence* stimuli (i.e. pseudo-words as input, true words as output) produced perplexity very close to the same LSTM run on the *Sentence* stimuli (i.e. true words as input and output). This implies that the pseudo-words retain some correlates of word form, enabling the LSTM to produce somewhat reasonable next-word predictions.

When the decoding model is trained on data from the *Word-list* condition, no representation performs significantly different from chance ($p > 0.05$, FDR corrected). The *Word-list* condition is a random ordering of words, so the corresponding LSTM cannot leverage any contextual information. As participants listened to the stimuli in the *Word-list* condition they also could not predict upcoming words, as the stimuli lacked syntactic structure and carry only the lexical-semantic information from the words out of context. Thus, it is not surprising that the decoding accuracy for this condition is close to random. Because of this poor performance, Analyses 2 and 3 do not consider the

Word-list condition. The accuracies for the Embedding and Conv layers are not significantly above chance for any condition ($p > 0.05$, FDR corrected).

To compare the results with non-contextual word embeddings, we also trained our decoding models with CBOW representations from *FastText* [11]. *Word-list* does not have context nor grammar structure, and so CBOW vectors—which do not use context—should perform better than LSTM vectors, which rely on context. We found the 2 vs. 2 accuracy to be 0.56 for the *Sentence* condition, and 0.535 for the *Word-list* condition, which are above chance ($p < 0.05$, FDR corrected). They are shown by dot on the left side of Fig 4.2. Since the *Jabberwocky* stimuli are pseudo-words, we cannot test the 2 vs. 2 accuracy using this word-level model.

In this experiment, *Sentence* and *Word-list* word vectors are the same. As we expected, in the *Sentence* condition, it performs worse than LSTM1-3 which is due to the non-contextual representation of CBOW. For *Word-list* , since the word vectors remain intact; then, it could work better versus LSTM representations. The results for *Word-list* are statistically lower than *Sentence* with a statistically significant p-value equal to $1.148e-05$ ($p < 0.05$, FDR corrected). That is because of *Word-list* EEG signals which are noisier. This result lets us conclude when we have stimuli with lexical-semantic meaning without grammar structure, word vectors that are non-contextual have higher performance.

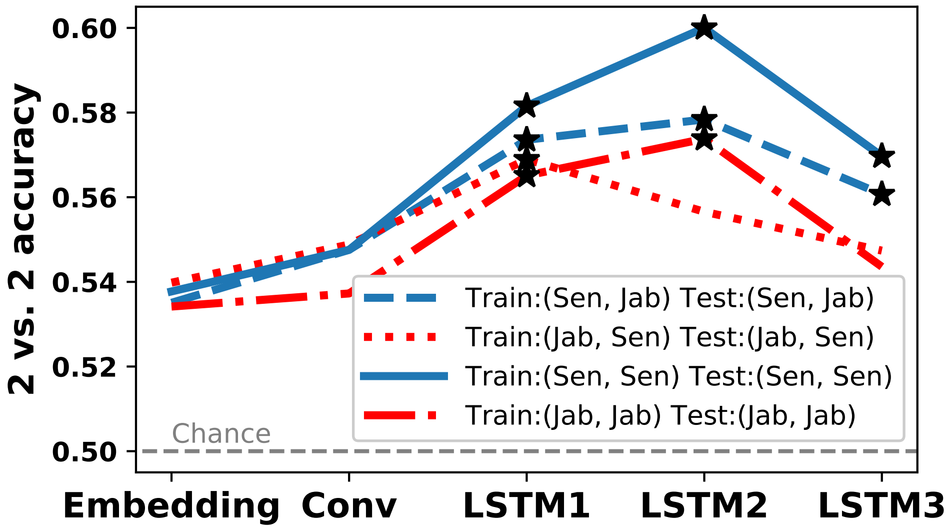
4.4 Swap the $g(S)$ conditions (Analysis 2)

Analysis 1 showed that some LSTM representations could be decoded in the *Sentence* and *Jabberwocky* conditions. This tells us there is a relationship between the information in some LSTM representations and the corresponding EEG data. But, the syntactic signatures that contribute to that relationship could be condition-specific. That is, the syntactic EEG signals driven by *Jabberwocky* could be fundamentally different from those in the *Sentence* condition.

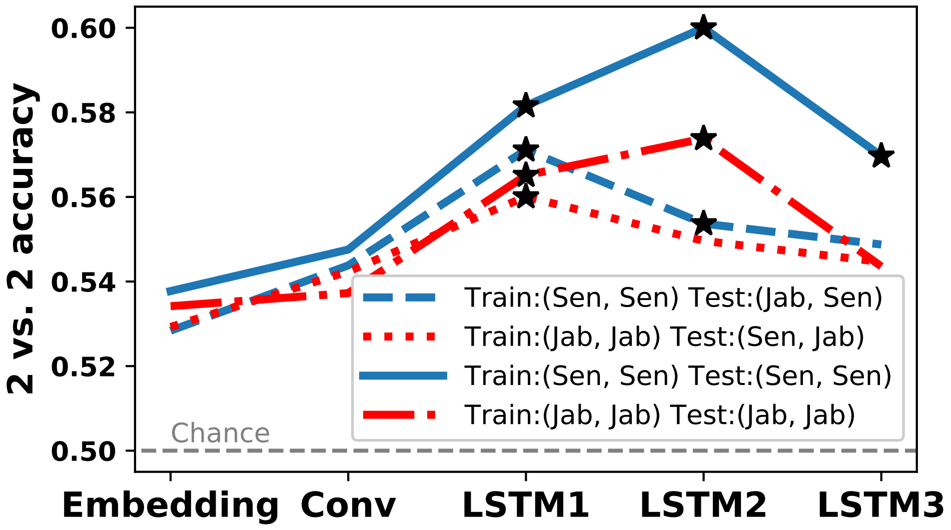
To test if the syntactic signatures in the *Sentence* and *Jabberwocky* conditions are exchangeable (i.e. similar in some way), we examined the accuracy of the decoding model in two cases: 1) using the EEG signals from the *Sentence* condition to predict the $g(S)$ vectors from the *Jabberwocky* stimuli, and 2) using the EEG signals from the *Jabberwocky* condition to predict the $g(S)$ vectors from the *Sentences* stimuli (see Table 4.1, Analysis 2). Because the *Jabberwocky* LSTM representations do not contain semantic information, this analysis will also tell us the degree to which the *Sentences* EEG/LSTM results in Analysis 1 leveraged semantic information. Because it is so central to this analysis, we again note that **the *Jabberwocky* stimuli are composed of pseudo-words derived from the *Sentence* stimuli, and the word order is maintained.** That is, the first word of sentence 1 in the *Jabberwocky* condition is a pseudo-word transformation of the first word from sentence 1 of the *Sentence* condition. Thus, we can interchange the corresponding representational vectors.

In Fig. 4.4 we see that the EEG signals from the *Sentence* condition can be used to predict the *Jabberwocky* LSTM representations (case 1). The accuracies for LSTM1-3 are 0.573, 0.578, and 0.560 which are all above chance ($p < 0.05$, FDR corrected). For the most part, the accuracies for case 1 are lower than the results from case 1 in Analysis 1 (EEG/LSTM representations from the *Sentence* condition), and we find there is a significant difference in the performance of LSTM2 ($p = 0.0006$). This is consistent with the hypothesis that *Jabberwocky* LSTM representations contain only syntactic information. Interestingly, the 2 vs. 2 accuracy when using *Sentence* EEG and *Jabberwocky* LSTM representations is higher than Analysis 1, where *Jabberwocky* EEG was paired with *Jabberwocky* LSTM representations. This is evidence that the syntactic information encoded in the *Sentence* EEG signals may be less noisy.

In case 2, when we use the *Jabberwocky* EEG to predict the *Sentence* LSTM representations, only the first LSTM layer shows above chance accuracy (0.568, $p < 0.05$, FDR corrected). This implies that the EEG signals from the *Jabberwocky* condition are not significantly correlated with the syntactic information in LSTM2 and LSTM3 vectors derived from *Sentence* stimuli. However, LSTM1 seems to encode syntactic information that is exchangeable.



Analysis 2 (Swap $g(S)$ vectors): Solid lines show the 2 vs. 2 accuracy of the decoding model that uses the *Sentences* EEG signals to predict the $g(S)$ vectors from the *Jabberwocky* stimuli, and vice versa.



(Swap R at test time): Solid lines show the 2 vs. 2 accuracy of the decoding model trained with EEG data and LSTM representations from the same condition, but tested with EEG data from the other condition.

Figure 4.3: Results from Analysis 2 and 3. Analysis 1 results appear as dashed lines. The x-axis denotes LSTM representation ($g(S)$). Legend denotes EEG/LSTM representations used for train/test: (EEG condition, LSTM condition). "*Sen*": *Sentence*, "*Jab*": *Jabberwocky* , "*WL*": *Word-list*. $\star$: above chance ($p < 0.05$, FDR corrected).

Though we did not explicitly test the correlation of the LSTM vectors for the *Sentence* and *Jabberwocky* conditions, Analysis 2 provides evidence that the two may encode correlated syntactic information. In addition, recall that the LSTM fed *Jabberwocky* can predict the next word of the corresponding *Sentence* stimuli with perplexity close to that of an LSTM fed *Sentence* stimuli. That predictability is another piece of evidence that the representations share information that could be leveraged in across the two decoding tasks.

4.5 Swap R at ***test*** time only (Analysis 3)

This analysis tests if a trained decoding model can generalize to EEG data from the other condition. For example, can a model trained with EEG signals and LSTM representations both from the *Sentence* condition still predict the *Sentence* vectors when tested on EEG from the *Jabberwocky* condition? This is Analysis 3 in Table 4.1. If the *pattern* leveraged to predict LSTM representations is similar across the two conditions, the 2 vs. 2 accuracy will remain above chance.

In Fig. 4.4, for case 1 (train on *Sentence* EEG, test on *Jabberwocky* EEG), the accuracies of the LSTM1 (0.571) and LSTM2 (0.553) are statistically above chance ($p < 0.05$, FDR corrected). Thus, the model trained using *Sentence* EEG can predict *Sentence* vectors from the corresponding *Jabberwocky* EEG. This implies that the brain's representation for the syntax in both the *Sentence* and *Jabberwocky* conditions takes a similar form, at least with respect to the syntactic information represented in LSTM1 and LSTM2. However, the performance of LSTM2 here is significantly lower than the performance of LSTM2 in case 1 of Analyses 1 and 2 ($p = 0.0001, p = 0.0005$ respectively). In fact, the performance for LSTM2 has dropped by a very large margin compared to Analysis 1, presumably because the semantic information leveraged in Analysis 1 is not available in the *Jabberwocky* EEG.

For case 2, (trained on *Jabberwocky* EEG/LSTM representations, but tested on *Sentence* EEG), only LSTM1 can be predicted with above chance 2 vs. 2 accuracy (0.560 with $p = 0.001$). So, as we saw in case 1, the LSTM1 model does generalize to EEG from the other condition. But, the same cannot be said for LSTM2, which is not significantly above chance in this case. That LSTM2 generalizes in one direction (case 1) but not the other (case 2) implies that the *Jabberwocky* EEG data is noisier, leading to a less robust model. So the pattern learned to predict the syntactic information in the *Jabberwocky* LSTM1 representations does generalize, but the same cannot be said about LSTM2.

4.6 Measuring model accuracy by mean-squared-error

In addition to 2 vs. 2 accuracy, to see other validation methods, we also used mean-squared-error (MSE) to assess the performance of the decoding model. Here, the squared error is the difference between the predicted representations and the true representations. Figs. 4.4 and 4.5 show the results of MSE for analyses 1-3.

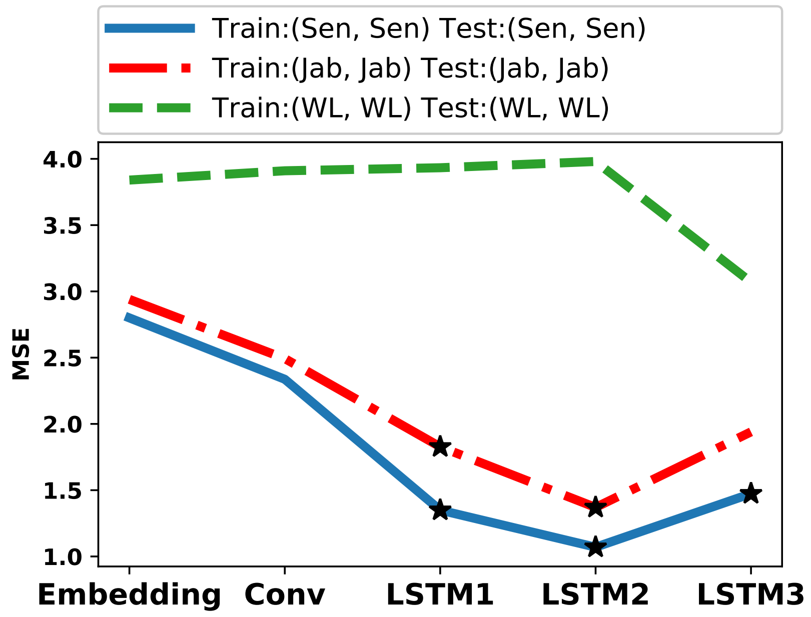
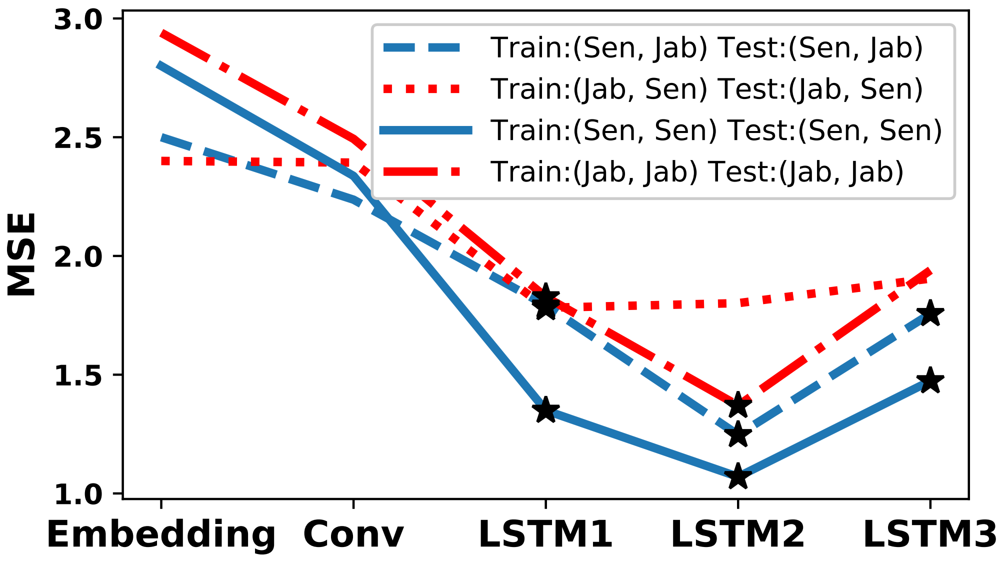
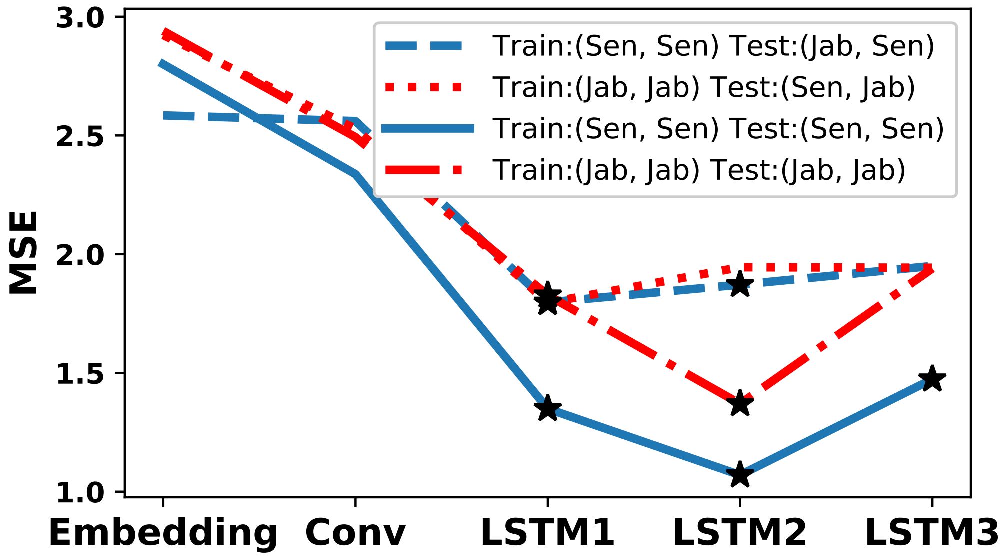


Figure 4.4: Analysis 1 (Test for semantic and syntactic information): MSE for $g(S)$/EEG from the same condition. The x-axis denotes LSTM representation ($g(S)$). Legend denotes EEG/LSTM representations used for train/test: (EEG condition, LSTM condition). "*Sen*": *Sentence*, "*Jab*": *Jabberwocky* , "*WL*": *Word-list*. $\star$: below chance ($p < 0.05$, FDR corrected).



MSE results from Analysis 2 (Swap $g(S)$ vectors): Solid lines show the MSE of the decoding model that uses the *Sentences* EEG signals to predict the $g(S)$ vectors from the *Jabberwocky* stimuli, and vice versa.



MSE results from Analysis 3 (Swap R at test time): Solid lines show the MSE of the decoding model trained with EEG data and LSTM representations from the same condition, but tested with EEG data from the other condition.

Figure 4.5: MSE results from Analysis 2 and 3. Analysis 1 results appear as dashed lines. The x-axis denotes LSTM representation ($g(S)$). Legend denotes EEG/LSTM representations used for train/test: (EEG condition, LSTM condition). "*Sen*": *Sentence*, "*Jab*": *Jabberwocky* , "*WL*": *Word-list*. $\star$: below chance ($p < 0.05$, FDR corrected).

Chapter 5

Discussion

Considering the results as a whole, several points become clear. There is a relationship between the semantic and/or syntactic information as represented by the brain and by LSTM representations, at least for the *Sentence* and *Jabberwocky* conditions. The probing results are quite consistent with the results of Analyses 1-3: LSTM1 has a strong signal for syntax, LSTM2 has syntax and semantics, and LSTM3 has some syntax and/or semantic signal, but the signal is weaker than for LSTM1-2.

LSTM1 shows only minor changes in performance in Analysis 2 and 3. So the syntactic information encoded in this layer is fairly consistent for stimuli from both the *Sentence* and *Jabberwocky* conditions, and it correlates well to either EEG data source. There is likely not much semantic information to leverage here, as the performance of models trained on *Sentence* EEG change by only a small amount in Analysis 2 and 3.

In Analysis 2 we saw similar drops in LSTM2 performance for both *Sentence* and *Jabberwocky* conditions. The drop in performance using the *Sentence* EEG could be attributed to the lack of semantic information in the *Jabberwocky* LSTM representations. However, we see a similar size drop in performance for the *Jabberwocky* condition, which implies that there is a mismatch even in the syntactic information available in LSTM2 for the two conditions. In Analysis 3, when we swap the test data, the pattern learned to predict LSTM2 in the *Sentence* condition (leveraging semantics and syntax) is not as effective when tested on *Jabberwocky* data.

The performance of LSTM3 is harder to explain, possibly because it has weaker semantic/syntactic signal (as evidenced by the probing tasks). There is a small performance hit when training on *Sentence* EEG data in Analysis 2, but a very large drop in Analysis 3. This pattern could result if LSTM3's representations of syntax are similar for *Sentence* and *Jabberwocky* stimuli, but the brain showed differing representations for the syntactic information in the two conditions. Then, it is possible that only the *Sentence* EEG would correlate to the syntactic information in LSTM3.

We wondered if there could be another explanation for our ability to decode in the *Jabberwocky* condition. One possibility is that the EEG and LSTM layers contain a correlate of the position in a sentence (1st word, 2nd word, etc.), and our models are using that information to decode (7/8 2 vs. 2 tests will use words at different positions). To test for this possibility, we trained a classifier to predict the ordering of two random words selected from a sentence, as suggested by [1]. The input to the classifier is the LSTM representation of the two words at their positions in a sentence, and the output is a binary decision for which of the two words appears sooner in the sentence. A model trained using our LSTM and the *Sentence* stimuli produced 80% accuracy on this task. Thus, we cannot say unequivocally that our results are not due in some part to positional information. However, our probing results are consistent with there being semantic/syntactic information in the representations, and those results are very consistent with the decoding analysis. This is strong evidence that our results are not entirely due to positional information.

We wondered also if the lexical semantics of the *Jabberwocky* stimuli could be leaking into the LSTM vectors, perhaps because the pseudo-words were repaired in the convolution step of the LSTM. Note, however, that lexical semantics are entirely intact in the *Word-list* condition, but the LSTM representations are of no use in that condition. Morphosyntax and syntax are maintained in the *Jabberwocky* condition, which appears to be enough to drive the correlation between LSTM representations and EEG recordings. The LSTM may be picking up on bi- and tri-gram signals related to morphosyntax cueing syntactic structure [22], [23], but more work is needed to rule out alternative explanations.

Recall that the *Sentence* and *Jabberwocky* stimuli share some orthographic/ phonological information. Could our *Jabberwocky* results, and the results of Analysis 2 (swap $g(s)$), be due only to the EEG encoding phonological or orthographic information? If our models were able to leverage such information, we would expect to see comparable decoding results in Analysis 1 and the *Word-list* condition, where the stimuli are perfect orthographic matches to the EEG. However, that analysis did not produce significantly above-chance accuracy. Furthermore, if the information leveraged in Analysis 2 was at the character-level, we would expect to see significantly above-chance accuracy in the character embedding or convolutional layers. However, it is not until the first LSTM layer (where contextual information is first incorporated) that any decoding model performs significantly above chance in any condition. This is evidence that the information being leveraged is *not simply phonological or orthographic*.

Our stimuli are composed of two conjoined sentences. How much composition have Dutch listeners done by the time when they get to the conjunction word "en?" How does the processing differ between the first vs the second of the conjoined sentences? Previous work on the brain's processing of syntactic structures and coordinate clauses proposed an "active structure maintenance model", where neural activity increases as a function of syntactic complexity [20], [29]. They found that neural activity in certain left-hemispheric regions indeed increased when more constituents had to be integrated, for both sentences and jabberwocky stimuli. It may be that the second coordinate constituent in our stimuli sentences elicit stronger neural activity than the first, but more analysis would be required to verify this.

Chapter 6

Conclusion

In this study, we explored the correlation of a character-level LSTM with the brain's response for two kinds of out-of-distribution language. The *Jabberwocky* condition used pseudo-word translations of the *Sentence* stimuli (ablate semantics, preserve syntax). The *Word-list* stimuli was a pseudo-random reordering of the words in each of the *Sentence* stimuli (ablate syntax, preserve semantics). We ran a character-based LSTM to create contextual embeddings for the stimuli of each condition. Our linear-regression decoding models were trained to predict the various LSTM representations from the EEG signals.

Our results showed that the LSTM layers of this character-based LSTM do in fact correlate with EEG signals in both the *Sentence* and *Jabberwocky* conditions, but not in the *Word-list* condition. By training models with various alterations to the data, we were able to determine which LSTM representations carry semantic and syntactic information. We verified those results using a probing task on our Dutch LSTM, as well as an identical model trained on English.

There are multiple avenues for future work. One direction is to see if these results hold for different languages. Dutch has a fairly transparent phoneme-grapheme correspondence; would our results still hold for a language with deeper orthography? We were surprised to find that some LSTM representations resembled the *Jabberwocky* EEG signals. Are there other examples of out-of-distribution language where this relationship holds? And, perhaps more interestingly, where it does not hold?

Another avenue is to improve language modelling. Finding ways in which the brain's representations differ from an LSTM could help us to build models closer to the true nature of human language processing. By doing this, we might be able to improve natural language processing and language models. Finding how the human brain can understand different concepts in different languages besides translating would help natural language processing to develop a translator with more similar to the human ability. Finding a way that the human brain can generalize semantic meaning when it hears or reads new words, e.g. nonsensical language, is beneficial to improve computational models. On the other side, we can build computational models which interpret human language processing more truly to help psycholinguists, sociolinguistics, and neurolinguistics more effectively. In this way, they would be able to more effectively identify disorder patterns.

Note, in this research, our goal was to see the correlation and relationship between the human brain's activity recorded by EEG and a standard LSTM representation. We might find more correlation, however, with additions to the architecture, like auto-encoders, ELMO, or a multi-modal network in the decoding model. By doing this, we may be able to identify representations that are more similar to the brain's activity.

References

[1] Y. Adi, E. Kermany, Y. Belinkov, O. Lavi, and Y. Goldberg, “Fine-grained analysis of sentence embeddings using auxiliary prediction tasks,” *arXiv preprint arXiv:1608.04207*, 2016.

[2] A. J. Anderson, D. Kiela, S. Clark, and M. Poesio, “Visually grounded and textual semantic models differentially decode brain activity associated with concrete and abstract nouns,” *Transactions of the Association for Computational Linguistics*, 2017.

[3] M. Baroni, G. Dinu, and G. Kruszewski, “Don’t count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors,” in *Association for Computational Linguistics*, 2014.

[4] Y. Benjamini and D. Yekutieli, “The control of the false discovery rate in multiple testing under dependency,” *Annals of statistics*, 2001.

[5] A. Conneau, G. Kruszewski, G. Lample, L. Barrault, and M. Baroni, “What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties,” in *Association for Computational Linguistics*, 2018.

[6] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, “Indexing by latent semantic analysis,” *Journal of the American society for information science*, 1990.

[7] M. Eichler, G. G. Şahin, and I. Gurevych, “LINSPECTOR WEB: A multilingual probing suite for word representations,” in *Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, 2019.

[8] E. Fedorenko, A. Nieto-Castañon, and N. Kanwisher, “Lexical and syntactic representations in the brain: An fmri investigation with multi-voxel pattern analyses,” *Neuropsychologia*, 2012.

[9] A. D. Friederici, M. Meyer, and D. V. Cramon, “Auditory language comprehension: An event-related fmri study on the processing of syntactic and lexical information,” *Brain and Language*, 2000.

[10] A. Fyshe, G. Sudre, L. Wehbe, N. Rafidi, and T. M. Mitchell, “The lexical semantics of adjective–noun phrases in the human brain,” *Human Brain Mapping*, 2019.

[11] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning word vectors for 157 languages," in *International Conference on Language Resources and Evaluation*, 2018.

[12] J. Hale, C. Dyer, A. Kuncoro, and J. Brennan, "Finding syntax in human encephalography with beam search," in *Association for Computational Linguistics*, 2018.

[13] M. Hashemzadeh, G. Kaufeld, M. White, A. E. Martin, and A. Fyshe, "From language to language-ish: How brain-like is an lstm's representation of atypical language stimuli?" In *Empirical Methods in Natural Language Processing: Findings*, 2020.

[14] W. A. de Heer, A. G. Huth, T. L. Griffiths, J. L. Gallant, and F. E. Theunissen, "The hierarchical cortical organization of human speech processing," *Journal of Neuroscience*, 2017.

[15] C. Humphries, J. R. Binder, D. A. Medler, and E. Liebenthal, "Syntactic and semantic modulation of neural activity during auditory sentence comprehension," *Journal of Cognitive Neuroscience*, 2006.

[16] S. Jain and A. Huth, "Incorporating context into language encoding models for fmri," in *Advances in Neural Information Processing Systems*, 2018.

[17] G. Kaufeld, H. R. Bosker, S. Ten Oever, P. M. Alday, A. S. Meyer, and A. E. Martin, "Linguistic structure and meaning organize neural oscillations into a content-specific hierarchy," *Journal of Neuroscience*, 2020.

[18] E. Keuleers and M. Brysbaert, "Wuggy: A multilingual pseudoword generator," *Behavior research methods*, 2010.

[19] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," in *Association for the Advancement of Artificial Intelligence*, 2016.

[20] E. Lau and C.-H. Liao, "Linguistic structure across time: Erp responses to coordinated and uncoordinated noun phrases," *Language, Cognition and Neuroscience*, 2018.

[21] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of English: The Penn Treebank," *Computational Linguistics*, 1993.

[22] A. E. Martin, "Language processing as cue integration: Grounding the psychology of language in perception and neurophysiology," *Frontiers in Psychology*, 2016.

[23] A. E. Martin, "A compositional neural architecture for language," *Journal of Cognitive Neuroscience*, 2020.

[24] B. McCann, J. Bradbury, C. Xiong, and R. Socher, "Learned in translation: Contextualized word vectors," in *Advances in Neural Information Processing Systems*, 2017.

[25] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv:1301.3781*, 2013.

[26] T. M. Mitchell, S. V. Shinkareva, A. Carlson, K.-M. Chang, V. L. Malave, R. A. Mason, and M. A. Just, "Predicting human brain activity associated with the meanings of nouns," *science*, 2008.

[27] B. Murphy, P. Talukdar, and T. Mitchell, "Selecting corpus-semantic models for neurolinguistic decoding," in *Proceedings of the First Joint Conference on Lexical and Computational Semantics, Proceedings of the main conference and the shared task, Proceedings of the Sixth International Workshop on Semantic Evaluation*, Association for Computational Linguistics, 2012.

[28] S. Nagel, "Towards a home-use bci: Fast asynchronous control and robust non-control state detection," Ph.D. dissertation, Universität Tübingen, 2019.

[29] C. Pallier, A.-D. Devauchelle, and S. Dehaene, "Cortical representation of the constituent structure of sentences," *Proceedings of the National Academy of Sciences*, 2011.

[30] F. Pereira, M. Botvinick, and G. Detre, "Using wikipedia to learn semantic feature representations of concrete concepts in neuroimaging experiments," *Artificial intelligence*, 2013.

[31] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *North American Chapter of the Association for Computational Linguistics*, 2018.

[32] D. Schwartz and T. Mitchell, "Understanding language-elicited eeg data by predicting it from a fine-tuned language model," in *North American Chapter of the Association for Computational Linguistics*, 2019.

[33] M. Toneva and L. Wehbe, "Interpreting and improving natural-language processing (in machines) with natural language-processing (in the brain)," *Advances in Neural Information Processing Systems*, 2019.

[34] L. Wehbe, B. Murphy, P. Talukdar, A. Fyshe, A. Ramdas, and T. Mitchell, "Simultaneously uncovering the patterns of brain regions involved in different story reading subprocesses," *PloS one*, 2014.

[35] L. Wehbe, A. Vaswani, K. Knight, and T. Mitchell, "Aligning context-based statistical models of language with brain activity during reading," in *Empirical Methods in Natural Language Processing*, 2014.

[36] L. Wittgenstein, *Philosophical investigations.* John Wiley & Sons, 2009.