ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA CESENA CAMPUS

DIPARTIMENTO DI DEPARTMENT OF ELECTRICAL, ELECTRONIC, AND INFORMATION ENGINEERING -"Guglielmo Marconi" – DEI"

SECOND CYCLE DEGREE IN BIOMEDICAL ENGINEERING
Class: LM-21
Curriculum Biomedical Engineering for Neuroscience

THESIS TITLE

Semantic decoding of natural speech production from sEEG via deep learning

Graduation thesis in
Neurorobotics and neurorehabilitation

Supervisor
Prof. Silvia Orlandi

Candidate
Anna Zanuccoli

Co-supervisors
Prof. Lorenzo Chiari
Dr. Ibrahim Kiremitci
Dr. Julia Berezutskaya
Prof. Nick Ramsey

Academic Year 2021/2022
Session III

Table of Contents

Abstract

One of the most devastating neurological diseases that result in the loss of communication is amyotrophic lateral sclerosis (ALS) which is characterized by loss of voluntary muscle control and affects around 5 persons per 100,000 population. For people suffering from this disease, brain-computer-interface (BCI) technology aims to provide a means of communication to improve their quality of life and, to improve the communication rate of a BCI system, semantic decoding can be exploited.

In this context this master thesis, developed in collaboration with Nick Ramsey's BCI lab at UMC Utrecht, focuses on the feasibility of decoding semantic representations of speech, in the form of word embeddings, measuring neural activity through stereotactic electroencephalography (sEEG). The study involves two participants who read aloud a chapter from the Harry Potter book while sEEG electrodes recorded their brain activity. Word embeddings are generated using Bidirectional Encoder Representations from Transformers (BERT) model and Recurrent neural networks (RNN) are employed as decoding models to decode word embeddings directly from high frequency band [60-150 Hz] neural activity.

This project is the first study that tried to decode word embeddings directly from brain activity and, due to the complexity of the task and the sparse coverage of sEEG electrodes, the decoding model achieves an accuracy only of 52%.

Despite achieving poor accuracy, the findings of this exploratory study may support future projects in further developing speech BCI applications based on semantic decoding.

Acknowledgments

I would like to express my sincere gratitude to professor Silvia Orlandi who has been an incredible mentor throughout my this journey. Her guidance, support, and feedback have been fundamental in improving my professional skills and personal values.

I would also like to express my gratitude to Dr Ibrahim Kiremtci, my other supervisor, who supported me during my time abroad, and all the people working in Professor Ramsey's lab. It was a privilege to work in such a stimulating and collaborative environment. Thank you all for your encouragement and assistance.

I would also like to thank my family for their love and support, and for always believing in me. Their encouragement and motivation have been a driving force behind my academic achievements, and I am grateful for their constant presence in my life.

Last but not least, I would like to extend my appreciation to my friends for their encouragement, understanding, and support throughout this journey.

Thank you all for your contributions to my success, and for being a part of this exciting chapter of my life.

List of Tables

List of figures

List of acronyms

WHO	World health organization
ALS	Amyotrophic lateral sclerosis
BCI	Brain-computer-interface
FMRI	Functional magnetic resonance imaging
ECoG	Electrocorticography
sEEG	Stereotactic electroencephalography
NLP	Natural language processing
BERT	Bidirectional encoder representations from transformers
RNN	Recurrent neural networks
FNIRS	Functional near-infrared spectroscopy
EEG	Electroencephalography
MEG	Magnetoencephalography
MFCCs	Mel-frequency cepstral coefficients
ELMO	Embeddings from language models
LSTM	Long-short-term- memory
GPT	Generative pretraining transformer
HFB	High frequency band
IEMU	Intensive epilepsy monitoring unit
CAR	Common average reference method
STFT	Short-time Fourier transform
DPSS	Digital prolate spheroidal sequence
WER	Word error rate
PCA	Principal component analysis
GRU	Gated recurrent unit
MSE	Mean square error

Chapter 1: Introduction

1.1 Motivations

Speech is the most convenient and natural form of communication, however, for people suffering from severe forms of paralysis, the ability to communicate thoughts and feelings can be limited.
According to the World Health Organization (WHO), 3.6% of the worldwide population has severe communication impairment that affects their community participation and social relationships [1,2].

One of the most devastating neurological diseases that result in the loss of communication is amyotrophic lateral sclerosis (ALS) which is characterized by loss of voluntary muscle control and affects around 5 persons per 100,000 population [3]. For people suffering from this disease developing technology that can support or recover their communication ability could really improve their quality of life [4–6].

In this context brain-computer-interface (BCI) technology aims to provide a means of communication to these individuals.
In general, BCI is a computer-based system that translates brain signals into commands that are passed onto an external application or appliance, so as to facilitate the user's intention [7]. In the field of communication, speech BCIs are devices that produce a form of speech output (e.g., words, sentences, synthesized speech) from a measure of the user's brain activity. [8]

In recent years, the potential for such systems in assisting individuals with language impairments has been investigated and showed that BCIs could be reliably used in real-world applications. [9–14]
Despite the recent advances in speech BCIs the available technologies allow for transmitting only 10 words per minute, which is a rate far slower than the average of 150 words per minute of natural speech[15].
This is because usually decoding algorithms used in speech BCIs rely on spelling-based approaches characterized by serial communication paradigm, thus resulting in a slow

communication rate [16]. To overcome this limitation a possible solution is to develop a speech BCI that exploits the semantic representation of language.

Speech decoding based on semantic representation aims to identify the meaning of words rather than the specific words or sounds that are being spoken.
Words can be represented based on their semantic content either identifying semantic categories or generating their "word embeddings" representation. Words can be represented as semantic categories by simply clustering them in groups (like objects, animals, people etc) and, using this representation, semantic decoding reduces to a classification problem. Alternatively, a semantic representation of words can be constructed by applying a language model that maps words into vectors, called word embeddings, such that vectors of words with similar meanings are closed in the vector space. Exploiting this semantic representation, the decoding method aims to predict each word separately (as a vector of real numbers), hence decoding consists of a regression problem.
Semantic decoding has the potential to achieve a form of parallel communication with BCIs, improving their communication speed considerably. For example, identifying the multi-bit semantic concept of 'hunger' directly from neural data could be much faster than spelling out 'I–A-M–H-U-N-G-R-Y' in series.
Additionally, semantic representation can allow for more complex and nuanced communication, as it can take into account the context and intent of the speech.

However, it is important to note that semantic decoding from the neural activity is a new field of research and is still in its early stages.
The majority of studies in this field tried to decode semantic representation using functional magnetic resonance imaging (fMRI) [16]. However, fMRI has enormous limitations for semantic decoding, because semantic representation is involved in the early stages of speech production (usually starts 200 ms before speaking) and, due to its low temporal resolution, fMRI is not able to detect the dynamics of semantics.

Better results can be achieved using invasive methods for measuring brain activity, such as electrocorticography (ECoG) or stereotactic electroencephalography (sEEG), because they are characterized by both high spatial and temporal resolution.
Specifically, in the field of semantic decoding, exploiting sEEG to record neural activity might be the best option. While ECoG provides higher density coverage over a limited cortical region

(typically unilateral), sEEG provides sparser coverage spanning more, bilateral brain regions including deeper structures [17]. Considering that semantic representation of speech is encoded in deep and widely spread regions of the brain, sEEG coverage has the potential to better measure neural activity related to semantics as compared to ECoG.

In this context, the overall goal of this project was to decode speech, exploiting its semantic representation, from neural activity measured through sEEG electrodes.
The aim of the decoding model was to predict each word separately according to its semantic representation, thus we generated word embeddings using Bidirectional Encoder Representations from Transformers (BERT)[18] model, which is one of the most accurate language models recently developed in the natural language processing (NLP) field.
Considering the increasing applications of deep learning models in the BCI field (due to their potential of learning complex relationships between sources and targets [12,15,19–22]) recurrent neural networks (RNN) are used in this project as decoding models.
Another key aspect of this project was referring to the applied natural speech production task, during which, patients read aloud a chapter from Harry Potter's book. This paradigm allows us to investigate and decode brain activity during a natural form of speech production, providing insights into neural activity during speech production in everyday life.

To the best of our knowledge, this is the first time that semantic decoding of word embeddings is performed using sEEG during a natural speech production task. Since no one else before has ever tried to decode word embeddings directly from brain activity measured through sEEG, this study was conducted as a feasibility study on two participants. This project addresses a completely new field of research and has the potential to advance speech BCIs based on semantic decoding.

1.2 Project overview and thesis objective

This project was an exploratory study that aimed to determine the feasibility of semantic decoding based on sEEG data. The following research questions were posed:

1. Is it possible to decode semantic representation of speech, in the form of word embeddings, using sEEG during a natural speech production task?
2. If not, which is the complexity of the problem?

1.3 Thesis roadmap

This thesis is divided into five chapters. Chapter 1 introduces the motivation, research question, objectives, and thesis outline. Chapter 2 provides background on data acquisition methods for speech decoding and language models applied to build word embeddings (semantic representation of words). Chapter 3 details the methodology in the development of the pipeline to preprocess data, generate word embeddings and decode them from brain activity. The results and discussion are presented in chapters 4 and 5 respectively. Lastly chapter 6 reports the conclusions of the project.

Chapter 2: Background

2.1 Neural speech decoding: speech BCIs

2.1.1 Brain signals for speech decoding

Methods to record brain activity are categorized into invasive and non-invasive, but only some are suitable for speech decoding.

Among non-invasive techniques, there are methods based on the detection of metabolic signals for brain activity, such as functional Magnetic Resonance Imaging (fMRI) and functional Near-Infrared Spectroscopy (fNIRS). Both methods are characterized by a high spatial resolution but also by a low temporal resolution because the metabolic processes that they measure are slow and take several seconds to complete. Considering that speech production occurs in the range of tens milliseconds, fMRI and fNIRS are not suitable measurement systems for speech decoding [16].

The other non-invasive methods that can be used for speech decoding are Electroencephalography (EEG) and Magnetoencephalography (MEG) which offer a temporal resolution of 0.01 to 0.1 seconds, making them better suited for capturing the rapid dynamics of speech production processes. However, both methods are characterized by a low signal to noise ratio and high sensitivity to artifacts which are relevant limitations in the field of speech decoding.

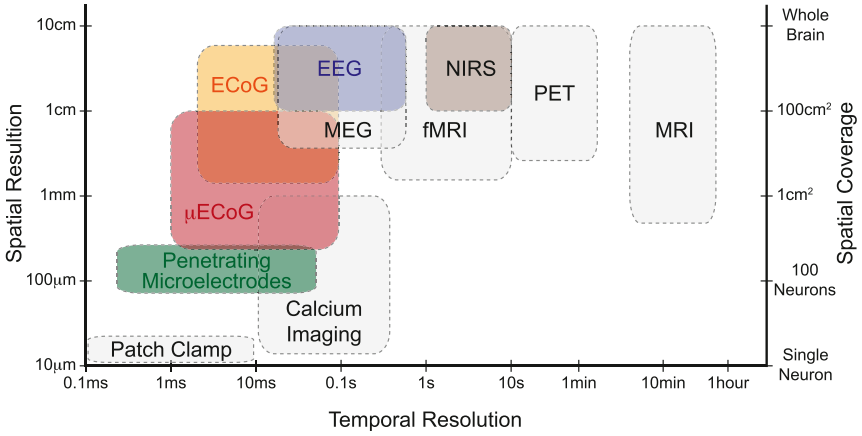


Figure 1 Spatial and temporal resolution and spatial coverage of various acquisition methods of brain activity

Specifically, recent previous studies focused on decoding proved that invasive recording methods, such as ECoG and sEEG, are the once that achieve the best accuracy[8] because these methods have excellent temporal and spatial resolutions. In addition, they are characterized by a high SNR and are less affected by artifacts. However, their implantation involves a risk as they are invasive. Most of what has been learned about ECoG and sEEG signals for speech decoding comes from studies on patients with medically intractable epilepsy who undergo implantation for seizure localization. During the time that patients spend at the hospital for such medical treatment (about a week), they may volunteer for studies that allow researchers to correlate strictly defined tasks with resulting ECoG and sEEG signals [23,24].

For ECoG the implantation involves a craniotomy to place strips or grids of electrodes directly on the cortex. Electrodes are typically placed over specific localized regions of the cortex (based on the clinical needs of the patients), and they only provide access to the cortical surface and not key deeper brain structures.[17]

Regarding sEEG the penetrating depth electrodes are implanted through small burr holes in the skull and they are positioned using stereotactic guidance, thus the modality is referred to as stereotactic EEG (sEEG). [17]

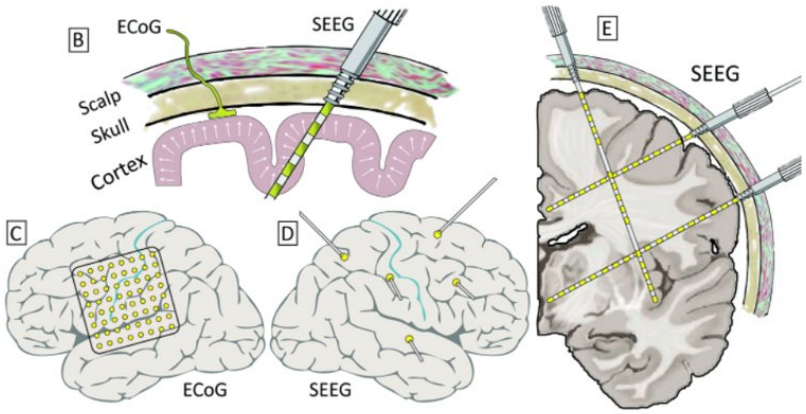


Figure 2 Implantation of invasive electrodes. The figure shows the difference between ECoG and sEEG electrodes. ECoG are implanted under the skull while sEEG electrodes penetrate inside the cortex.

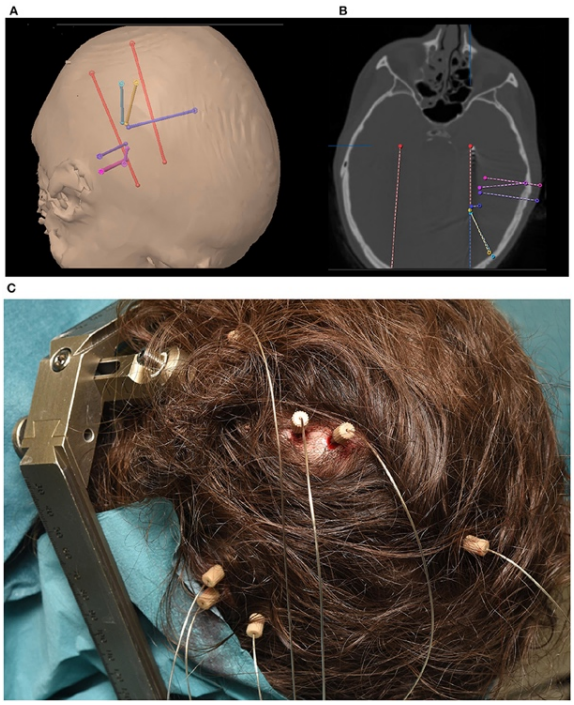


Figure 3 sEEG electrodes: (A)Trajectory planning for 8 sEEG electrode shafts. (B) Computer Tomography showing implanted electrode shaft locations.(C)Implanted electrode shafts. sEEG requires only small, localized burr holes compared to the comparatively large craniotomies required. Figure from reference[17]

While both signals have been used for speech decoding, most studies have been conducted using ECoG. However, specifically for semantic decoding, sEEG holds great potential because it is characterized by a broader coverage as compared to ECoG, thus allowing for better measuring of brain activity related to semantic processes, which are widely spread over the brain.

Another advantage of sEEG is that it offers access to brain structures not reachable with ECoG, thus allowing for better investigation of semantic representation of speech that is also encoded in-depth brain structures. In addition, since sEEG electrodes are implanted through small holes in the skull, the implantation leads to fewer surgical complications rather than for ECoG which requires a craniotomy [25].

Considering all these advantages we believe that sEEG holds great potential for semantic decoding, that's why we have used this measurement technique in this project.

2.1.2 Speech representations for decoding

There are several types of speech representations that can be exploited for speech decoding in BCIs. These representations can be categorized into three main groups - semantic, acoustic, and articulatory – and they are involved at different points along the speech production pathway.

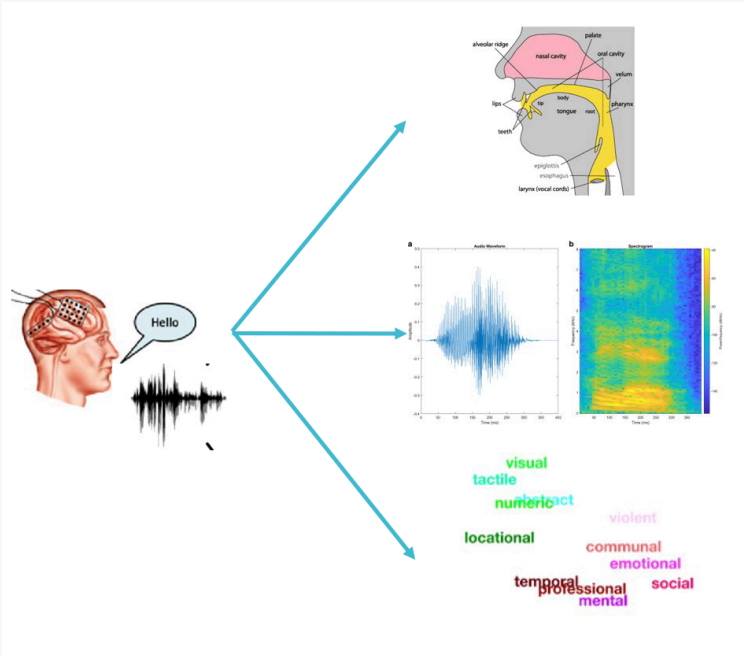


Figure 4 Speech representations applied for speech decoding.

Regarding semantics, speech can be represented as the underlying concepts or discrete words conveyed by that speech, which are likely intercepted at one of the earliest stages of speech production in various regions of the brain. [16,26]

Semantic features that can be exploited for decoding can be either categories (objects – animals - emotions) or words. Decoding models based on semantic representation apply different language models to construct a vector representation of words – word embeddings – that is the target of the decoding model (see section 2.2 for more details).

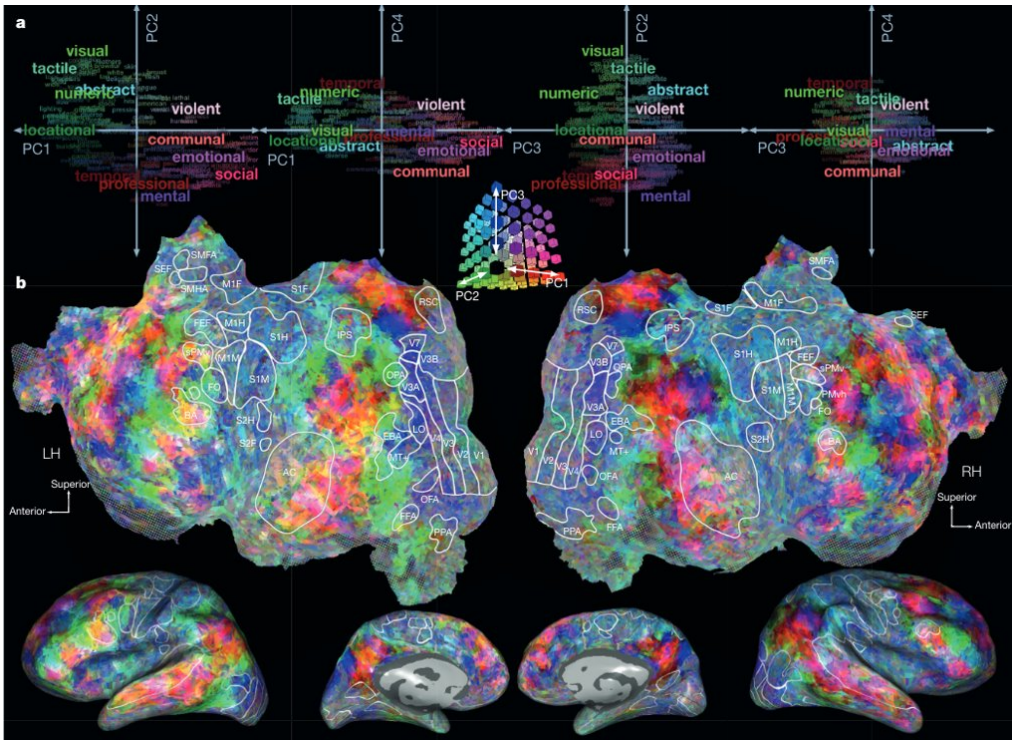


Figure 5 Principal components of voxel-wise semantic models. Principal components analysis of voxel-wise model weights reveals four important semantic dimensions in the brain. An RGB colourmap was used to color both words and voxels based on the first three dimensions of the semantic space. Figure reused with permission from ref[27].

In terms of acoustic representation, speech can be segmented into its corresponding spectrotemporal features, which can also be further segmented into phonemes. Neural representations of phonemes or related acoustic features are likely intercepted for decoding in the middle stages of speech production. Acoustic features that are usually exploited for decoding are Mel-frequency cepstral coefficients (MFCCs) or logarithmic mel-scaled coefficients.[8]

For both semantic and acoustic-based decoding approaches it is necessary simultaneously acquire microphone recordings of produced speech and related brain activity.

Finally, in terms of articulatory features, speech can be represented as a sequence of phonation and articulatory gestures generated by the muscles and organs within the vocal tract and mouth. This representation would likely be intercepted in the later stages of speech production, especially in the speech-related motor integration areas of the brain [8].

However, for the development of such a speech decoding approach, articulatory movements need to be recorded during speech simultaneously with the brain activity, which is rarely feasible in ECoG research due to the additional burden it would inflict on the participants.

It is important to note that there are many different representation types and models that can be used for speech decoding and the choice of representation will depend on the specific application and the goals of the BCI.

In this project, we have exploited the semantic representation of speech because it's a new field of research and, to the best of our knowledge, no one before has ever tried to decode semantic features from sEEG during a natural speech production task. The advantage of semantic-based decoding is that it allows faster communication because semantic representation is involved in the early stages of speech production, and it allows a more natural form of communication.

2.2 Semantic representation: word embeddings & language models

As previously mentioned, the goal of the project is to decode words from neural activity based on their semantic representation, but how can we derive the semantic representation of words? The ability of humans to represent the meaning of words is based on mental representations, however, we have no direct access to those representation[28]. Hence, to represent the meaning of words, we apply language models that rely on the distributional semantic hypothesis, according to which "words with similar meaning occur in comparable linguistic context"[29]. Based on this hypothesis, a word is represented as a vector in a high dimensional space, and vectors of words with similar meanings are close in the vector space [30] (as we can see from figure 6 in which "cat" and "kitten" are very close in the space, and they are also quite close to "dog", while both are far from "house").
This language modelling method that maps words from a vocabulary into vectors of real numbers is called word embeddings.

When constructing a word embedding space the goal is to capture some sort of relationship in that space, be it meaning, morphology, or context. Word embeddings can be constructed applying different language models, but the embeddings always have some key characteristics:

- Every word in the vocabulary has a unique word embedding (or "vector"), which is just a list of numbers for each word.
- The word embeddings are multidimensional: typically for a good model, dimensions are between 50 and 500 in length.

- Similar words end up with similar embedding values.

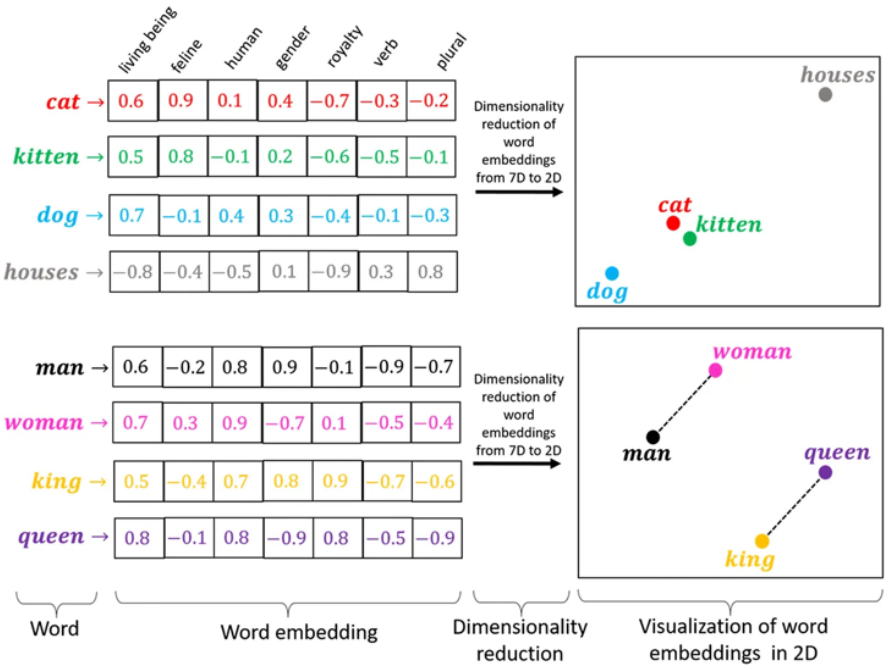


Figure 6 Representation of word embeddings. On the left side of the figure, we can see the vectors of numbers use to represent words as word embeddings, while in the right part of the figure the words are represented in the space as vectors.

Word embeddings can be calculated through many different language models and the recent advances in NLP have shown that the most efficient models are the ones whose architecture is based on neural networks.

Two well-known approaches for calculating word embeddings are Word2vec[31] and GloVE[32]. The Word2vec model predicts a target word from a list of context words using a simple feed-forward neural network. The GloVe model, in addition to using local context windows such as Word2vec, tries to predict a word also considering the global word co-occurrence counts.

This architecture is very simple and can perform well enough when the sentences are short, however, the main limitation of these models occurs when sentences are long because the models are trained at the word level and do not use the information about all words in the sentence.

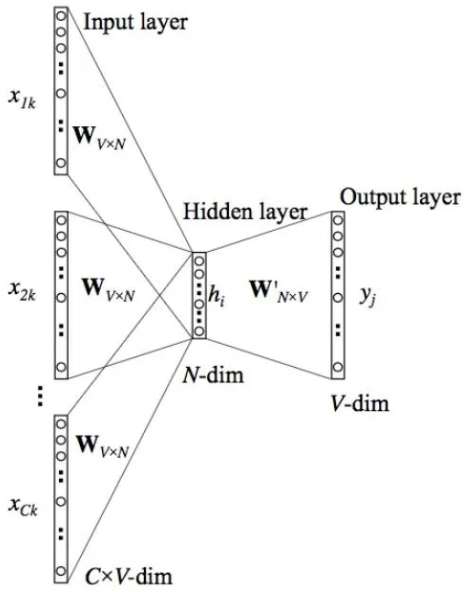


Figure 7 The word2vec model architecture, figure from reference[31].

To overcome these limitations, recent models adopted a more complex architecture based on RNN. The advantage of language models based on RNN is that they take all previous input into account when computing the output, hence they compute the word embedding of each word considering all previous words in the sentence.

One of the most well-known language models based on RNN is ELMO (Embeddings from language models) [33]. The key characteristic of ELMO is that it utilizes a bidirectional Long-Short-Term- Memory (LSTM), which considers the context before and after the word counts.

Despite the good performance that RNN based models can achieve they require long training time when dealing with long sequences, due to their sequential way of computing. Another great limitation of RNNs when working with long sequences is that their ability to retain information from the first elements is lost when many new elements are incorporated into the sequence.

Recent advances in NLP solved this problem by adopting transformers-based architecture[34]. These models exploit a mechanism called "Attention" by which the model can learn which inputs deserve more attention than others in certain cases [35]. Attention allows the model to make predictions by looking at the entire input and selectively attending to some parts of it. This selection is determined by a set of weights that are learned during training.

The key characteristic of transformers is that their encoder-decoder structure makes use only of attention mechanisms and does not apply recurrent or convolutional layers. Nowadays the two

most applied transformer models are Generative Pretraining Transformer (GPT)[36] and Bidirectional Encoder Representations from Transformers (BERT)[18].

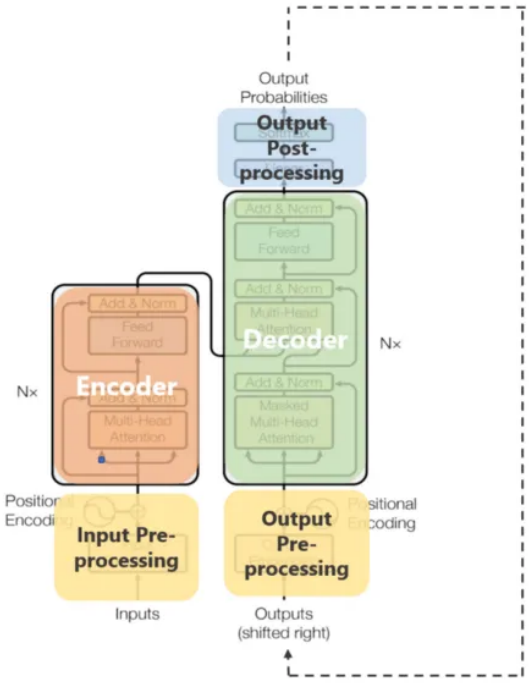


Figure 8 Architecture of an encoder-decoder based language model

Both models are based on encoder-decoder architecture and apply attention mechanisms but, while GPT is unidirectional because it is trained only to predict the next word from the current word, BERT is bidirectional and computes embeddings considering the context before and after the analyzed word.

The main advantage that has been introduced by those models is that they have been pre-trained on a large amount of data (BERT: 800 million words) and they can be applied for different NLP tasks just fine-tuning the models to the specific task.

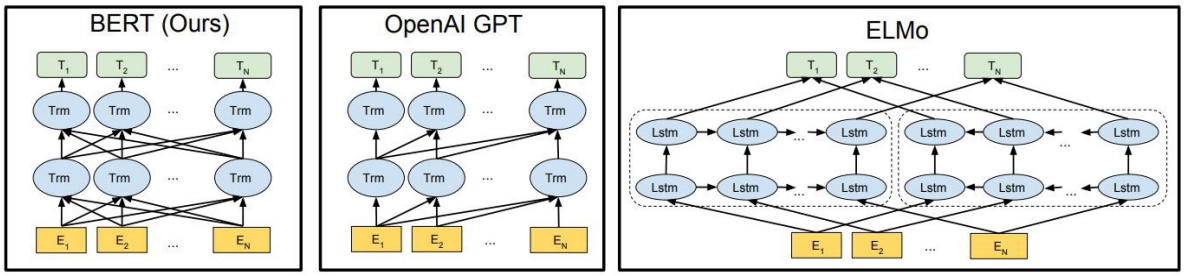


Figure 9 Difference in model architecture. BERT uses a bidirectional transformer. OpenAI GPT uses left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-let LSTMs. Figure from reference[18]

To compute the word embeddings of words in our task we decide to apply the BERT model because several previous works have shown that it achieves high performances. More details about the BERT model and how we have applied it to generate embeddings are described in Chapter 3.

2.3 Previous works

Semantic decoding from neural activity is a new field of research that has not been explored a lot yet. A recent review by Rybar and colleagues[16] provides a nice overview of the advances of semantic decoding.
This review includes studies that apply different acquisition methods to measure brain activity for decoding purposes, and authors report that the majority of the studies use fMRI as an acquisition method (45), while only a few studies make use of invasive methods such as ECoG (5) and sEEG (1). This result highlights once again the exploratory nature of our project that uses sEEG signals to decode the semantic representation of words.

In the literature, studies that investigate semantic decoding from neural activity usually aim to decode two different types of outputs:

- The semantic categories
- The probability of the word that comes next given the knowledge of previous words (represented as word embeddings)

Studies that aim to decode semantic categories first group words into categories, and then classify brain activity related to the pronunciation of each word into one of the categories. One recent study that classifies words according to semantic categories has been conducted by Na et al.[37] In this paper authors proposed a semantic-hierarchical structure for classifying brain activity (corresponding to listened words) by means of a support vector machine classifier. Na et al. obtained good accuracy results (10% improvement of classification accuracy when compared with a non-hierarchical structure), however, the potentials of decoding categories of words for the development of speech BCIs are limited because ideally, we would like a speech BCI that directly decodes the word that the user wants to communicate and not a semantic category.

A different strategy is to derive a semantic representation of words by applying language models and thus obtaining word embeddings. Studies that represented words using word embeddings, such as Goldstein et al. and Tang et al., aim to decode the probability of the word that comes next given the knowledge of previous words.

Goldstein and colleagues[38] trained a convolutional neural network to predict the word embeddings of the next word given previous words from ECoG data (10 subjects) acquired during a listening task. The authors of this last paper compared the performance of two language models, GloVe and GPT model, and proved that better results can be obtained using contextual transformer language models such as GPT (65% accuracy using GloVe, 75% accuracy using GPT). Similarly, Tang et al.[39] introduced a non-invasive fMRI-based decoder (data recorded from 3 participants during a listening task) that predicts the next word coming in a sequence.

Even if those papers decode words from brain activity, they can decode only words within a sequence of words and given the knowledge of the previous words.

In order to develop a speech BCI based on semantic decoding we would like to decode words independently, and not predict the next word coming in a sentence. However, to the best of our knowledge, no one before has ever tried to independently decode words, based on word embedding representation, from brain activity.

In this context, the novelty of our project becomes evident because, for the first time, we tried to decode word embeddings of words from neural activity exploiting the potentials of sEEG.

Given the unexplored nature of this decoding task, the aim of the project was to conduct a feasibility study.

Chapter 3: Methodology

This chapter describes the architecture of the decoding process implemented to achieve the main goal of the thesis such as predicting word embeddings from brain activity through sEEG signals.

The entire pipeline is reported in the following paragraphs and summarized in the following flowchart (figure 10).

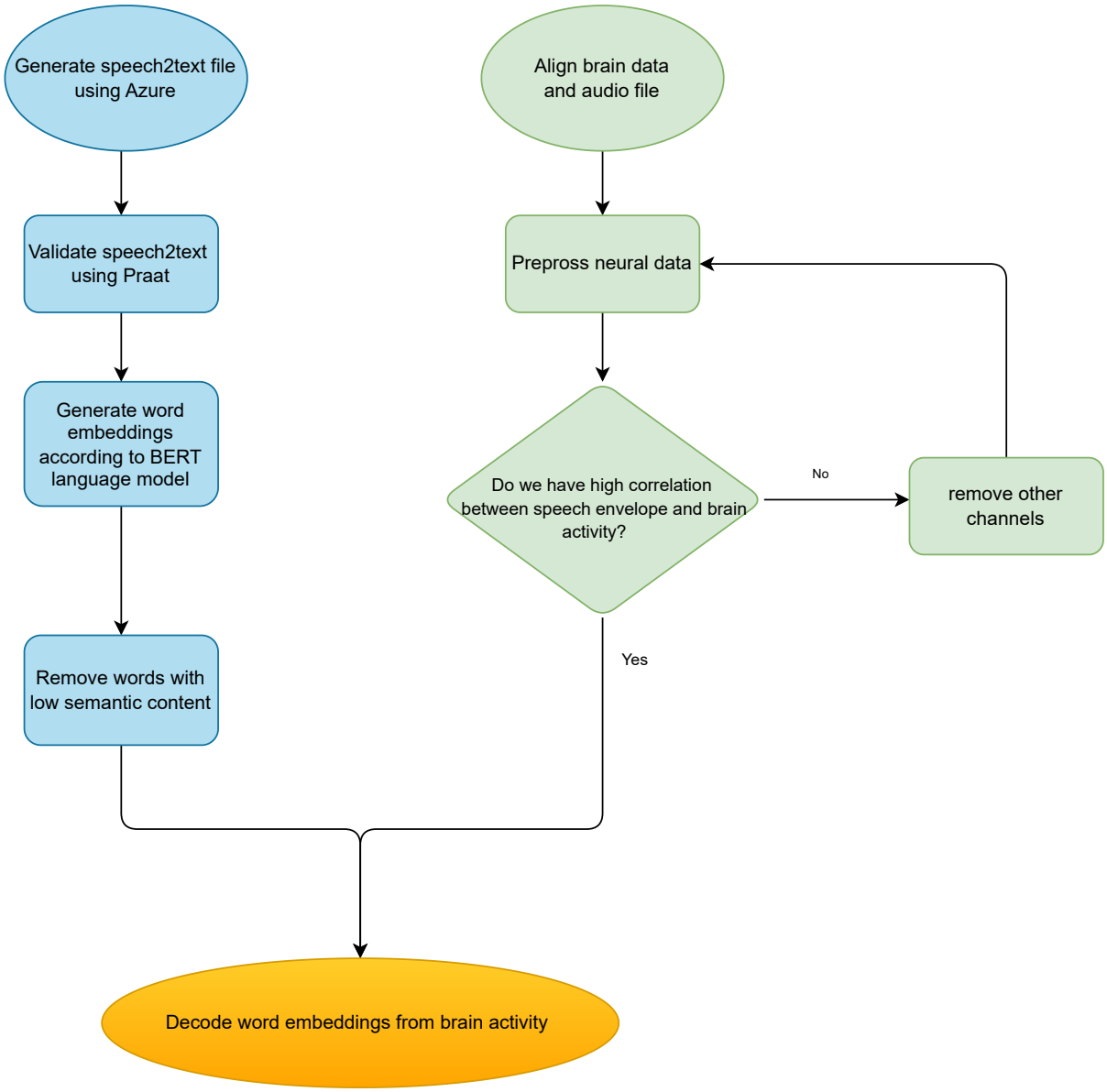


Figure 10 Flowchart that represents the main steps of project implementation

In order to decode word embeddings from brain activity we needed to perform two parallel analyses: on the one hand, we needed to preprocess brain data (represented by green flowchart

symbols in figure 10), and on the other hand, we had to generate the word embeddings (represented by blue flowchart symbols in figure 10).

The first step of the generation of word embeddings was to obtain the transcript of the audio file of each participant and to derive the onset and offset time at which each word was pronounced. The generation of the transcript was automatically done via Azure software, explained in paragraph 3.5.3. Then the transcript, generated by the automatic tool, was validated through Praat (see 3.5.3) to verify if the alignment was correct and manually modify transcription errors accordingly.
After these steps, we generated the word embeddings using the BERT model (section 3.6.2) and in the end, after removing words with low semantic content (paragraph 3.6.3), we obtained the word embeddings of words pronounced by the participants, that will later be used as targets of the decoding model.

Regarding the processing of brain data, the first step was to check the alignment between the brain data file and audio file (see 3.5.1.1), because later in the processing we assumed that the time instant at which each word is pronounced is aligned with the neural activity. After this, we preprocessed neural data, and computed its power spectrum (paragraphs 3.5.1.2 and 3.5.1.3).
Later, before moving to the semantic decoding, we did a simpler analysis to check if the neural activity was correlated with speech and verify that the preprocessing pipeline was performed correctly. We analyzed the speech envelope entrainment by computing the correlation between neural activity in high frequency band (HFB) and speech envelope. If the previous preprocessing were performed correctly, we expected to find good speech envelope tracking, in terms of high correlation, at least for the neural activity of some electrodes (details are reported in section 3.7).
After the speech envelope entrainment analysis, we were ready to use neural activity in HFB as input for the decoding model.

3.1 Participants

Two participants with medication-resistant epilepsy, were admitted to intensive epilepsy monitoring unit (IEMU) for diagnostic procedures after they underwent temporary

implantation sEEG electrodes to determine the source of seizures and test the possibility of surgical removal of the corresponding brain tissue. All participants were native Dutch speakers. In addition to clinical procedures, participants gave written informed consent to participate in scientific research that accompanied sEEG recordings and could be conducted between clinical procedures.

The study was approved by the Medical Ethical Committee of the University Medical Center Utrecht in accordance with the Declaration of Helsinki (2013).

The patients underwent a pre-operative MRI scan and post-operative CT scan for placement as well as localization of the electrodes. Depth electrodes were 0.8 mm in diameter with 5 to 18 contact points. The electrodes were placed into the left (4 patients) or bilateral (2 patients) hemispheres, mainly including temporal, parietal, and occipital cortices.

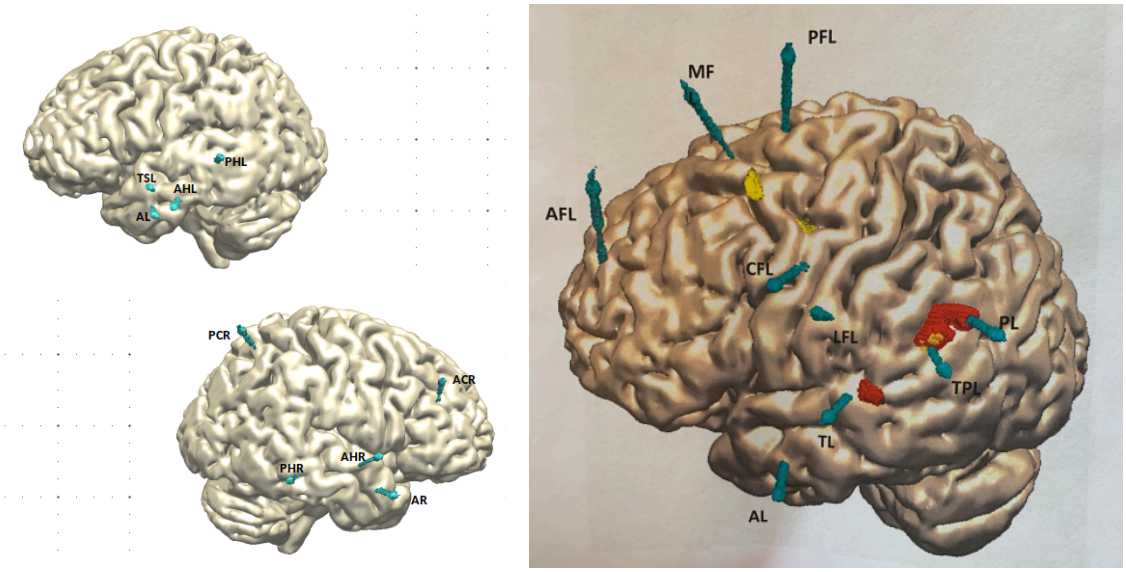


Figure 11: sEEG Electrodes implantation for participant 1 (left) and participant 2 (right)

3.2 Experimental procedure

Data collection was conducted at University Medical Centre in Utrecht in the IEMU unit. During the experiment, words were presented to the participants on separate pages on a tablet (Microsoft Prime) using a software called Presentation software (Neurobehavioral Systems)[40]. Participants hold the tablet according to their own comfort to read, and they read the pages at their own pace. sEEG data were acquired using Micromed at a sampling rate of 2048 Hz (filtered at 0.3-500 Hz).

In addition to the neural data, audio recordings were acquired using an external microphone (Samson Q2U dynamic USB microphone), connected directly to the tablet, and controlled within the Presentation software. Microphone data were recorded at 16000 Hz. Synchronization of the neural recordings with the task, as well as with the recorded audio, was achieved based on event codes sent from stimulus presentation software to the Micromed system.

3.3 Task

The experiment consisted in a naturalistic task during which subjects read aloud chapter one from Harry Potter and the Sorcerer's Stone (in Dutch). The same chapter was selected with the study of Wehbe et al[41]. The task was split into two runs, dividing the chapter into 2 roughly equal parts in terms of the number of words. Each run lasted approximately 15 minutes.

3.4 Python libraries

The project was implemented using Python (version 3.6)[42], a popular programming language used in data science and machine learning applications. In addition, Anaconda[43], a distribution platform that provides a comprehensive set of libraries, was used to facilitate the implementation process. Anaconda's package manager allows the installation and management of the necessary libraries, such as Numpy, Pandas, and Matplotlib. A complete list of all the libraries needed to implement this project can be found in table 1.

Table 1: List of python libraries applied to implement the project.

Library	Version	Characteristics
bidso	0.8	Transparent Object-Oriented Approach to BIDS in Python
torch	1.10.2	PyTorch is a Python package that provides two high-level features: Tensor computation (like NumPy) with strong GPU acceleration and Deep neural networks built on a tape-based autograd system
wonambi	7.11	Package for the analysis of EEG, ECOG, and other electrophysiology modalities. It allows for visualization of

		the data and sleep stage scoring in a GUI. Provides automatic detectors for spindles and slow waves
soundfile	0.11	The soundfile module is an audio library based on libsndfile, CFFI and NumPy
nltk	3.6.7	NLTK (the Natural Language Toolkit) is a suite of open source Python modules, data sets, and tutorials supporting research and development in Natural Language Processing
transformers	4.18	Transformers provides thousands of pretrained models to perform tasks on different modalities such as text, vision, and audio
librosa	0.9.2	Librosa is a python package for music and audio analysis
azure	4.0.0	The open-source Azure libraries for Python simplify provisioning, managing, and using Azure resources from Python application code
textgrids	1.5	Python classes for Praat TextGrid and TextTier files
scipy	1.5.4	Open-source software for mathematics, science, and engineering. It includes modules for statistics, optimization, integration, linear algebra, Fourier transforms, signal and image processing, ODE solvers, and more
pyNSL	0.0.1	Python version of the NSL MATLAB toolbox, used to compute audio spectrogram following the biological model of sound processingby the cochlea
sklearn	0.24.2	scikit-learn is a Python module for machine learning built on top of SciPy and is distributed under the 3-Clause BSD license.

3.5 Data preprocessing

The data preprocessing included: the preprocessing of neural data, the computation of the power spectrum, and the generation and validation of the transcript of the audio file, that was later used to obtain word embeddings.

3.5.1 Neural data preprocessing

3.5.1.1 Aligment

During the task, the Presentation software sent event triggers to the Micromed system to mark the start and end of the task, as well as the start and end of the audio recording. These time markers were used to align the brain and audio data.

3.5.1.2 Preprocessing

Data were loaded and preprocessed exploiting the Wonambi Python library [44], which can read brain data in different formats (BCI2000, Blackrock, Brainvision, EEGlab, and more) and represents data using a ChanTime class, with axis channels and time.

After loading, data were preprocessed using a custom pipeline that started from bad channel identification and removal. As a first step data were visualized and all channels with noisy or flat data (based on visual inspection) were excluded from further analysis. Then also channels with NaN values or extremely weak signals (1e-15) were removed.
The final step of bad channels identification consisted in removing channels considered “bad by deviation” following recommendations provided by Bigdely-Shamlo et al.[45] This method computed a robust z score by replacing the mean with the median and standard deviation with the robust standard deviation (0.7413 times the interquartile range) for each channel. Then a channel was considered “bad by deviation” if its robust z score was greater than 5.

The second step of preprocessing pipeline consisted in re-referencing the signals. To do that we applied common average reference method (CAR), which computes the mean of the signal over all channels and then subtracts it from every channel. Since each sEEG electrode was composed of multiple channels we applied CAR for each electrode separately.

The last step in the preprocessing pipeline was filtering the signal to remove power line noise. To do that we applied a notch filter to remove the frequency at 50 Hz and its harmonics.

3.5.2 Time-frequency analysis

Spectral analysis of the sEEG signal was performed using the Short-Time Fourier Transform (STFT), in which the signal was divided into small sequential or overlapping data frames and fast Fourier Transform was applied to each one[46].

A DPPS window[47] (window having maximal energy concentration in the main lobe given by the digital prolate spheroidal sequence DPSS) of 0.3 seconds, with an overlap of 0.01 seconds, was used in the STFT computation. Then HFB amplitude, in the range between 60 and 150 Hz, was obtained by averaging amplitudes for the entire range of extracted frequencies. Lastly, the HFB power spectrum was resampled at 100 Hz for further analysis.

3.5.3 Transcript generation & validation

The Microsoft Azure Cognitive Services were used to generate transcripts from audio recordings. In particular, we used the Azure speech-to-text service that provides the transcript in a TextGrid format, obtaining information about each time instant at which words were pronounced (onset and offset time).

Even if Microsoft Azure Cognitive Services have been proven to be accurate in the generation speech to text files [48] they always need to be validated, because it is important to check the alignment between onset and offset time with the audio, and because there is the need to check the accuracy of the transcript.

Figure 12 figure contains words pronounced by participants during the reading task (words from the first chapter of Harry Potter and the Sorcerer's Stone book). The figure was generated using the wordcloud python library https://pypi.org/project/wordcloud/.

The validation of the transcript was performed using Praat. Praat is a free and open-source software program designed for the phonetic analysis of speech signals [49]. Exploiting this software, we were able to listen to the entire audio file to check the alignment and, at the same time, to check the number of words transcribed wrongly.

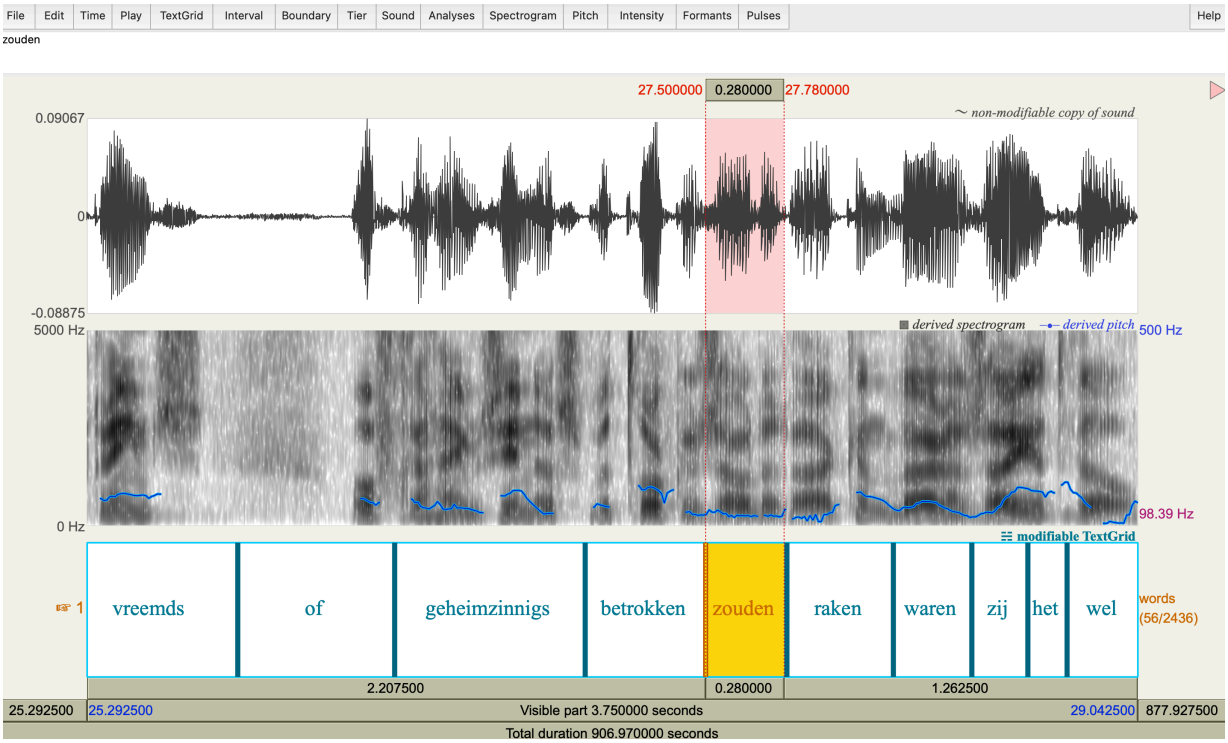


Figure 13 Praat interface

The standard way to measure Speech-to-Text accuracy is word error rate (WER). To compute WER the number of incorrect words identified during validation were counted and then divided by the total number of words.

Table 2.1: results of manual validation of subject's 1 transcript.

sub-1			
	word detection count	error detection count	error percentage [%]
run 1	2437	132	5,42
run 2	2375	122	5,14
total	4812	254	5,28

Table 2.2: results of manual validation of subject's 2 transcript

sub-2			
	word detection count	error detection count	error percentage [%]
run 1	2594	133	5,13
run 2	2051	157	7,65
total	4645	290	6,24

3.6 Word embeddings using BERT model

After the validation of speech-to-text transcript we were ready to generate the word embeddings of words spoken by the participant during the experiment. Those word embeddings constitute the semantic representation of words that will be later used as targets of the decoding model.

3.6.1 BERT language model

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a state-of-the-art pre-trained language model developed by Google [18]. It is a neural network-based approach that has significantly improved the performance of natural language processing tasks such as question answering, sentiment analysis, and text classification.

BERT is a transformer-based model, meaning that it uses a self-attention mechanism to identify important contextual information within the input text. The key characteristic of BERT is that it's a bidirectional model that takes into account both the left and right context of a word when generating its embeddings. The model architecture is constituted of 12 layers and 768 hidden states, hence the word embeddings generated by BERT have 768 dimensions.

BERT is a pre-trained model, meaning that it has been trained on a large corpus of text data, before being fine-tuned for specific NLP tasks. The pre-training process involves training the model on a large corpus of text data, such as the entire Wikipedia corpus or the BookCorpus dataset. Once pre-trained, BERT can be fine-tuned on a specific NLP task by training it on a smaller dataset of labeled examples.

However, for the purpose of our study, we did not need to apply any fine-tuning for a specific NLP task, we only used a pre-trained BERT model to generate word embeddings for our specific dataset.

BERT offers an advantage over simpler and static models like Word2Vec [18], because while each word has a fixed representation under Word2Vec, regardless of the context within which the word appears, BERT produces word representations that are dynamically informed by the words around them.
For example, given two sentences:

"The man was accused of robbing a bank."
"The man went fishing by the bank of the river."

Word2Vec would produce the same word embedding for the word "bank" in both sentences, while under BERT the word embedding for "bank" would be different for each sentence. Aside from capturing obvious differences like polysemy, the context-informed word embeddings capture other forms of information that result in more accurate feature representations, which in turn results in better model performance [18].

Since our participants were speaking Dutch during the reading task, we needed to apply a BERT model pre-trained on Dutch corpora. To do that, we decided to apply a monolingual Dutch BERT model called "BERTje" which was developed by the University of Groningen [50].

We preferred to apply this model rather than the multilingual model m-BERT (that also includes Dutch) developed by Google because applying BERTje the word count of vocabulary was only 355 out of 5458 while applying m-BERT we obtained 874 words out of vocabulary.

3.6.2 Generation of word embeddings

The generation of word embeddings using the BERT model required several steps:

- remove words out of vocabulary from our words list
- load pre-trained BERT model
- load the pre-trained model tokenizer
- input formatting: preprocess the input in a way that the BERT model can interpret it (figure 14)

- add a special token called [CLS] at the beginning of a sentence and another special token called [SEP] at the end of the sentence (where [CLS] is a special symbol added in front of every input example and [SEP] is a special separator token).
- tokenize words (token embeddings)
- associate an index to each token (positional embedding)
- associate the same id to tokens that belong to the same sentence (sentence embedding)

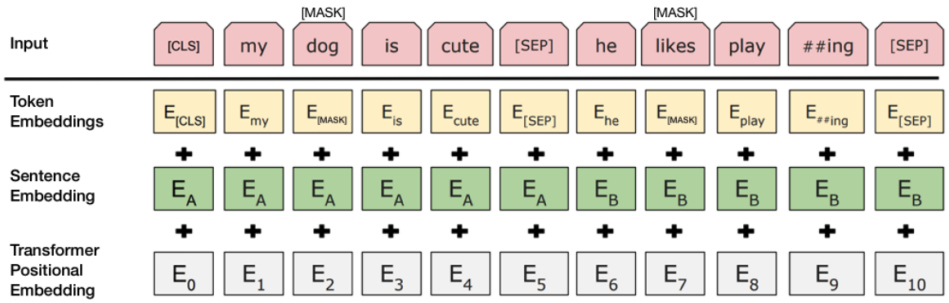


Figure 14 BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings. Figure from reference[18]

- apply the model to each sentence: BERT model is constituted by 12 layers of transformer encoders and each output from each layer can be used as word embeddings. According to Liu et al.[51] (as shown in figure 15) the best performance is provided by the sum of the embeddings obtained by the last 4 layers, hence we generate the word embeddings taking the sum of the outputs provided by the last 4 layers of the model.

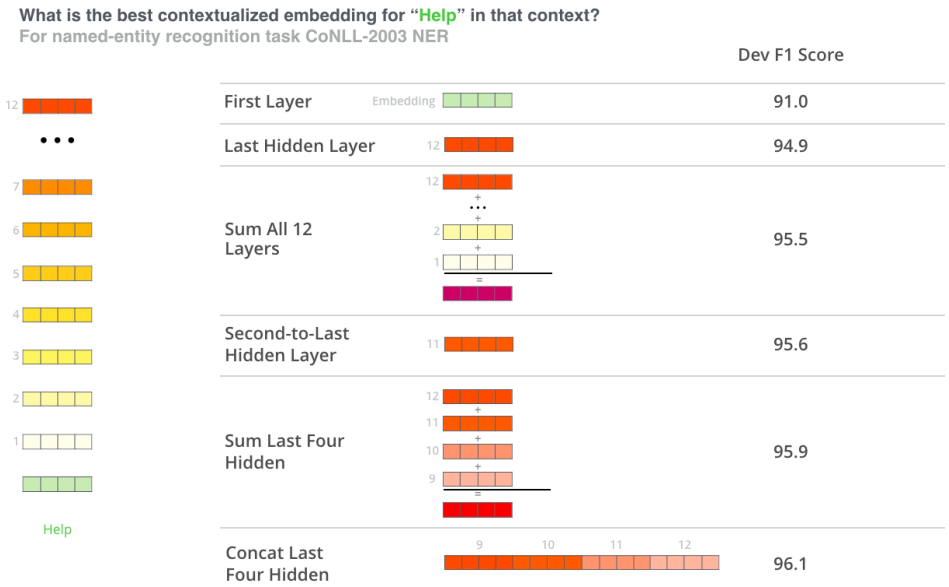


Figure 15 Accuracy scores obtained combining layers of BERT model in different arrangement[52].

At the end of this pipeline, we obtained word embeddings for all words pronounced by the participant, and each embedding was represented by 768 dimensions (as the number of hidden units in each layer of the model).
This analysis was done exploiting Transformers library [53], to load the pre-trained model and the model tokenizer, and *nltk* library [54], to split the transcript into sentences.

To represent the word embeddings, we applied t-SNE which is a tool (from sklearn[55]) to visualize high-dimensional data in a 2D space. As we can see from figure 16, words with similar meanings are close in the embeddings:

- meneer (sir), mevrouw (madame), zoontje (son), baby and zus (sister) → words related to people
- straat (street) and hoek (corner)
- nek (neck) and snor (mustaches)
- mantel (coat) and laarzen (boots)

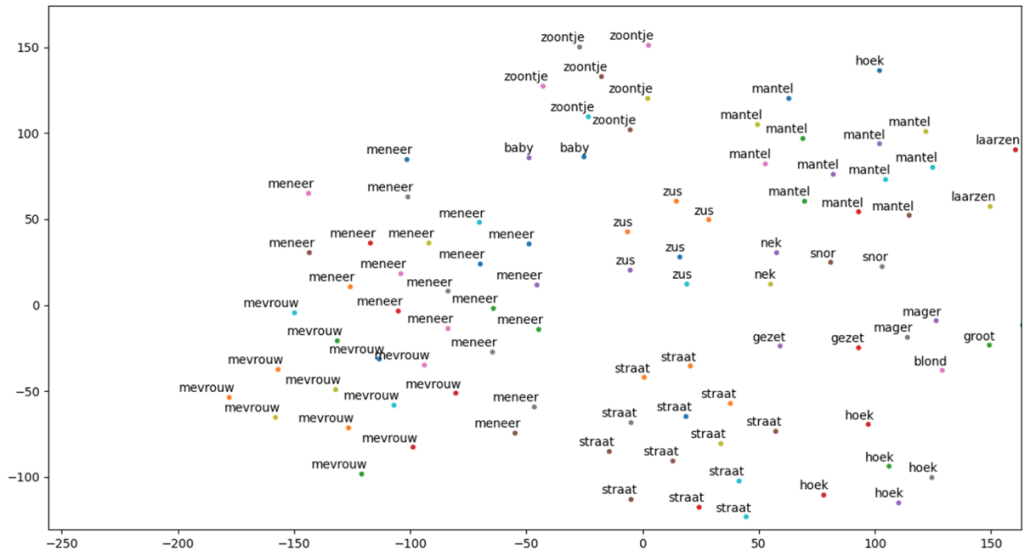


Figure 16 Figure generated using t-SNE python library. It represents words in a 2D space.

In addition, to visualize the variance of similarity between words in our dataset we plot the heatmap of cosine similarity between words. In figure 17, we can see the heatmap, with labels, of words present in the dataset and repeated at least five times (this choice was done to better visualize data, otherwise considering all words the labels would not be readable). As we can see there is a nice variance between the words in our dataset.

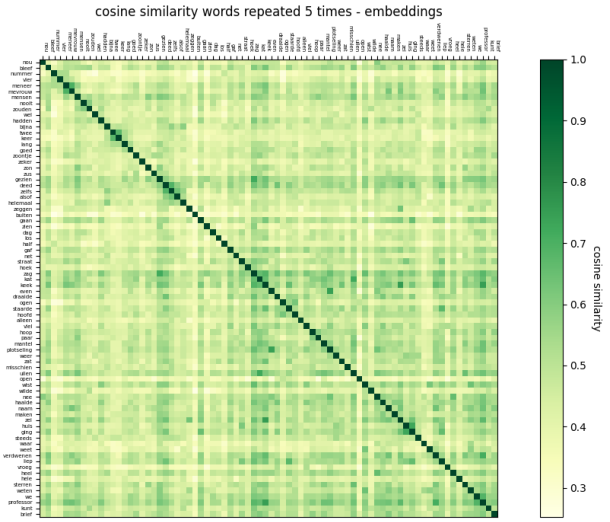


Figure 17 Heatmap of cosine similarity between words repeated at least five times during the task.

3.6.3 Post processing of word embeddings

Once word embeddings were generated, we removed words with low semantic content (usually called stop words) because we were not interested in decoding the meaning of those words. Stop words can be pronouns, linkers, and adverbs. To remove them a list of Dutch stop words provided by *nltk* python package was used.

By removing those words, we ended up with 1889 words for sub 1 and 1809 words for sub 2 (see table 3.1 and 3.2 below for details).

Table 3.1: total number of words, for subject 1, used to train the decoder.

sub-1		
	word count	word count after removing stopwords
run 1	2437	961
run 2	2375	928
total	4812	1889

Table 3.2: total number of words, for subject 2, used to train the decoder.

sub-2		
	word count	word count after removing stopwords
run 1	2594	1019
run 2	2051	790
total	4645	1809

3.6.4 Dimensionality reduction of word embeddings

Since the number of dimensions of embeddings space of the BERT model is really high (768), and since the ability of the decoding model to predict such a high number of features for each word was a concern, we decided to apply dimensionality reduction through principal component analysis (PCA).
PCA was performed using the *sklearn* python library[55]. We first computed the model for 50 principal components, then we fit and transformed it on the embedding space. Thus, we obtained only the first 50 principal components, out of the 768 total components, that contained the 75% percent of the variance of the entire embedding space.

3.7 Speech envelope entrainment

After preprocessing of neural activity and generation of word embeddings we were ready to start the decoding. However, before moving to the decoding analysis we decided to do a simpler analysis of brain data in order to check their quality, that brain activity correlated with speech, and that there were no errors in the preprocessing pipeline.

The simpler checkpoint analysis that we did was the tracking of speech envelope[56,57]. From previous studies[58,59] it has been proved that there is a correlation between the variability of speech envelope and the variability of electrical potentials and currents in the human cortex. This effect has been proved to be stronger in regions situated relatively early in the auditory pathway (belt areas) compared to other regions involved in speech processing, including the superior temporal gyrus and the posterior inferior frontal gyrus (Broca's region).

Since our task was reading, we were expecting to see a high correlation between brain activity and speech envelope. Of course, not all the electrodes were placed in the regions related to the auditory pathway but, if our preprocessing steps were performed correctly, we were expecting high correlation at least for some electrodes.

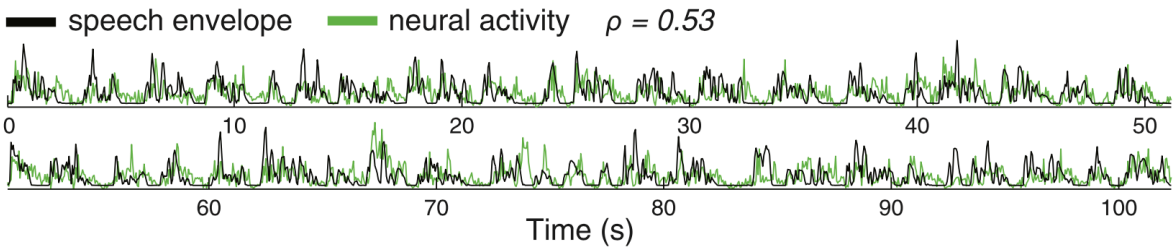


Figure 18 High gamma activity in human auditory cortex tracks the envelope of speech

To perform this analysis the first step was the computation of the speech envelope. The sound spectral envelope was extracted using the python library pyNSL which is the python equivalent of the Matlab NSL toolbox developed by Chi et al [60]. This toolbox computes the sound spectrogram following the biological model of sound processing by the cochlea (Chi et al., 2005). The spectrogram was extracted at 8 ms frames along 128 logarithmically spaced frequency bins in the range of 180–7,200 Hz. We then averaged the spectrogram data over the frequency bins to obtain a 1D spectral sound envelope. The resulting spectral envelope was downsampled to 100 Hz to match the sampling rate of the sEEG HFB time courses

Then, to assess the relationship between the speech envelope and HFB activity, we computed the nonparametric Spearman correlation coefficient per channel. To account for the time lag between these two quantities, we computed the correlation for each lag between the two signals in the range -200ms to 200ms with 10 ms steps. Then we took the maximum correlation over time lags, and we obtained one value of correlation for each electrode.

At the end of this analysis, we obtained a high correlation at least for some electrodes for each subject, and this proved that our preprocessing of neural activity was performed correctly and that we had a correlation between brain activity and speech.

3.7.1 Statistical analysis

To verify that our results were significant we applied bootstrapping statistical analysis. Bootstrapping is a statistical technique used to estimate the properties of an estimator by repeatedly sampling from the observed data. The method involves generating a large number of resamples, each of which is obtained by random sampling with replacement from the original data set. Then, the resamples are used to obtain estimates of the estimator or test statistic of

interest, and the distribution of these estimates is used to make inferences about the properties of the estimator[61].

The advantage of bootstrapping is that can be applied to any distribution of data, there is no need to check that the data have a normal distribution.

The main steps of the bootstrapping method are to:

1. take a sample from a population with sample size n.
2. draw a sample from the original sample data with replacement with size n, and replicate B times, each re-sampled sample is called a Bootstrap Sample, and there will be a total of B Bootstrap Samples.
3. evaluate the statistic of θ for each Bootstrap Sample. In the end, B estimates of θ will be obtained.
4. construct a sampling distribution with these B Bootstrap statistics and use it to make further statistical inferences, such as:
   - estimating the standard error of statistic for θ.
   - obtaining a Confidence Interval for θ.

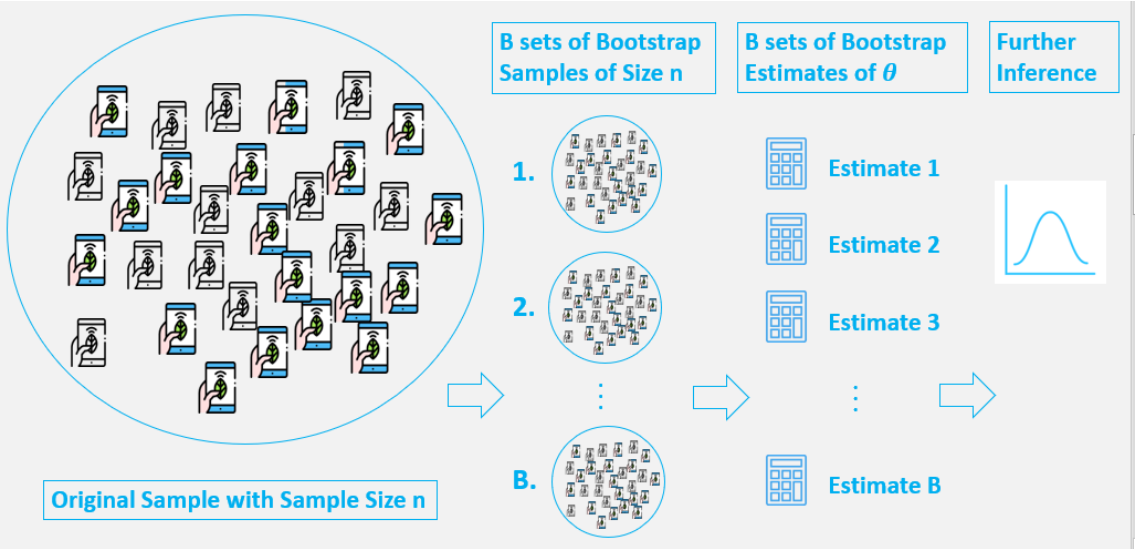


Figure 19 Working principle of bootstrapping method. Figure from https://towardsdatascience.com/an-introduction-to-the-bootstrap-method-58bcb51b4d60

For our statistical analysis, we aimed to test whether the correlation coefficients obtained for each channel were significant.

To do that we first split the brain data into 10 seconds blocks. Then the blocks were shuffled, to obtain a random distribution of brain activity. Later we computed the correlation between the random distribution of brain activity and speech envelope for each electrode (only for the time lag that provided the highest correlation in the previous analysis).

This process was repeated 1000 times, saving the results of correlation for each electrode for each iteration, and we ended up with a matrix with rows equal to the number of iterations and columns equal to the number of electrodes. We sorted the values along each row and then, for each electrode, we computed how many correlations obtained from random distribution were greater than the actual correlation. In the end, we computed a p-value for each electrode as the ratio between the number of correlations greater than the actual correlation and the number of iterations (1000) [62].

In this way, we obtained a p-value for each electrode and if the value was lower than 0.01, we considered a significant correlation for that value, otherwise, the correlation for that electrode was not significant.

3.8 Semantic decoding through Recurrent Neural Networks

After the preprocessing pipeline, we started the decoding of the word embeddings from the neural activity. The diagram below summarizes the decoding framework:

- input = brain activity in HFB [60 – 150] during the reading task;
- output = word embeddings for all words pronounced by the participant;
- network = RNN, GRU, LSTM

In the following paragraphs, we will describe in detail the characteristics of the network, the data aggregation, training and testing procedures.

Figure 20 Block diagram representation of the decoding model

3.8.1 Recurrent Neural Networks (RNN)

Deep learning is a subfield of machine learning that is inspired by the structure and function of the brain [63]. In the context of BCI, deep learning algorithms can be used to classify brain signals, extract relevant features, and perform various tasks related to decoding and control.
One of the main advantages of deep learning is that it can learn to recognize patterns in the data that are not easily detectable by traditional machine learning algorithms. This can lead to improving the accuracy of the BCI systems. Another advantage is that deep learning algorithms can learn to recognize patterns in the data in an end-to-end manner, which eliminates the need for manual feature selection.
Taking into account the increasing and promising applications of deep learning models in the field of BCIs [12,15,19–22], we decided to apply deep learning methods to decode word embeddings. At this point, we needed to understand which was the most suitable neural network (NN) for the decoding task of our project.

Considering that words have different lengths, the brain activity corresponding to the pronunciation of each word also has a different sequence length. Therefore, the input data to our model, from which we want to predict word embeddings, are characterized by variable sequence length. Due to this consideration, we decided to apply RNN as a decoding method, because RNNs are the most suitable networks to predict sequence data and because RNNs can handle arbitrary sequence lengths.

The main feature of an RNN is its ability to keep in memory all the previous elements of the sequence through a state vector. The most simple RNN is shown in figure 21. As can be seen, an RNN has a feedback connection that connects the hidden neurons across time.

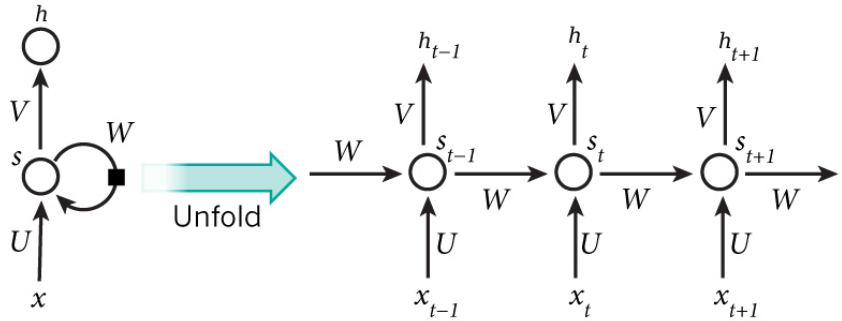


Figure 21 A standard RNN. The left hand side of the figure is a standard RNN. The state vector in the hidden units is denoted by s*. On the right hand side is the same network unfolded in time to depict how the state is built over time. Image adapted from reference*[63]

At time *t*, the RNN receives as input the current sequence element x_t and the hidden state from the previous time step s_{t-1}. Next, the hidden state is updated to s_t and finally the output of the network h_t is calculated. In this way, the current output h_t depends on all the previous inputs x'_t. U is the weight matrix between the input and hidden layers, W is the weight matrix for the recurrent transition between one hidden state to the next, V is the weight matrix for hidden to output transition and f represents the activation function for the output layer.
Since the same weights are used for each time step, RNNs can process variable-length sequences. At each time step new input is received and, due to the way the hidden state s_t is updated, the information can flow in the RNN for an arbitrary number of time steps, allowing the RNN to keep all the past information.

RNNs can be divided into three categories:

1. Traditional Recurrent Neural Network (RNN)
2. Long-Short-term-Memory Recurrent Neural Network (LSTM)
3. Gated Recurrent Unit Recurrent Neural Network (GRU)

3.4.4.1. Traditional RNN

As previously explained, due to the recurrent nature of RNN, each cell has two inputs: the output of the previous state and the current input. This way of using information from the past results in a short-term memory mechanism.

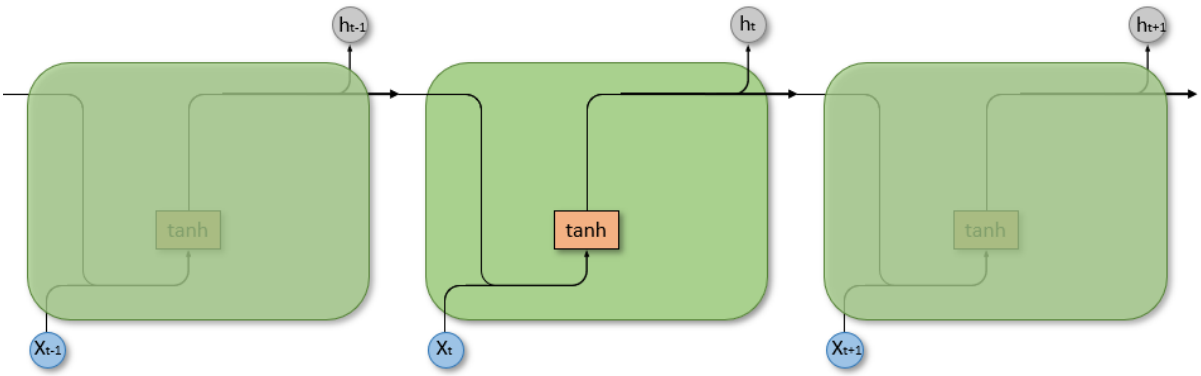


Figure 22 RNN architecture. Figure from reference[64]

In each cell, the input of the current time step x (present value), the hidden state h of the previous time step (past value), and a bias are combined and then limited by an activation function (tanh in this example) to determine the hidden state of the current time step.

$$\boldsymbol{h}_t = \tanh\left(\boldsymbol{W}_x\ \boldsymbol{x}_t + \boldsymbol{W}_h\ \boldsymbol{h}_{t-1} + \boldsymbol{b}\right)$$

The main disadvantage of simple RNN is that it suffers from the vanishing gradient descent problem, which means that the gradients that are used to update the weights during backpropagation become very small. Multiplying weights with a gradient that is close to zero prevents the network from learning new weights. This stopping of learning results in the RNN forgetting what is seen in longer sequences. As the RNN only keeps recent information, the model has problems considering observations that are far from the past. Hence, the RNN has only short-term but not long-term memory.

3.4.4.2. Long-Short-term-Memory (LSTM) RNN

LSTMs are a special type of RNN that intend solve the vanishing gradient problem.

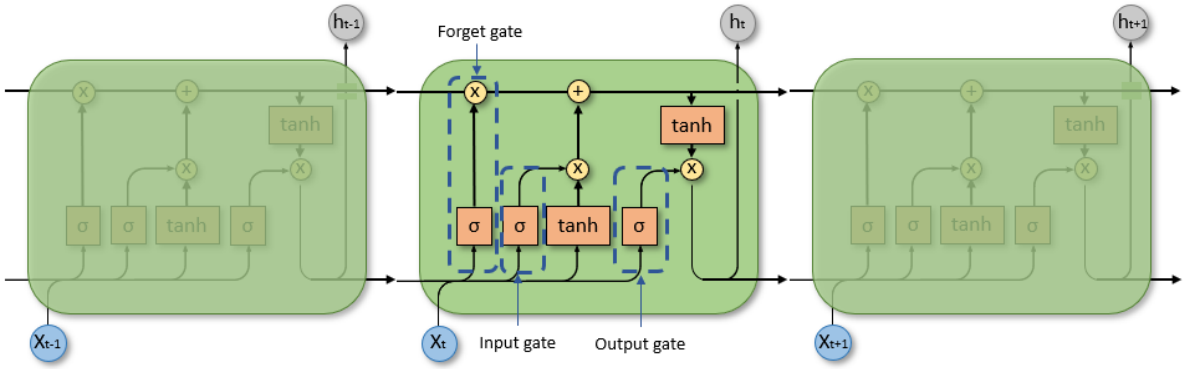


Figure 23 LSTM architecture, figure reference[64]

The key characteristic of LSTMs is the presence of a cell state, computed through three gates, which is passed from the input to the output of a cell. The cell state allows information to flow along the entire chain with only minor linear actions through three gates. Hence, the cell state represents the long-term memory of the LSTM. The three gates are called the forget gate, input gate, and output gate. These gates work as filters and control the flow of information and determine which information is kept or disregarded.

The forget gate decides how much of the long-term memory should be kept. For this, a sigmoid function is used which states the importance of the cell state. The output varies between 0 and 1 and states how much information is kept, The output is determined by combining the current input x, the hidden state h of the previous time step, and a bias b:

$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}_{f,x}\,\boldsymbol{x}_t + \boldsymbol{W}_{f,h}\,\boldsymbol{h}_{t-1} + \boldsymbol{b}_f\right)$$

The input gate decides which information should be added to the cell state and thus the long-term memory. Here, a sigmoid layer decides which values are updated.

$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{W}_{i,x}\,\boldsymbol{x}_t + \boldsymbol{W}_{i,h}\,\boldsymbol{h}_{t-1} + \boldsymbol{b}_i\right)$$

The output gate decides which parts of the cell state build the output. Hence, the output gate is responsible for short-term memory.

$$\boldsymbol{o}_t = \sigma\left(\boldsymbol{W}_{o,x}\,\boldsymbol{x}_t + \boldsymbol{W}_{o,h}\,\boldsymbol{h}_{t-1} + \boldsymbol{b}_o\right)$$

As can be seen, all three gates are represented by the same function. Only the weights and biases differ. The cell state is updated through the forget gate and the input gate.

$$\boldsymbol{c}_t = \left(\boldsymbol{f}_t \circ \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \circ \tanh\left(\boldsymbol{h}_{t-1}\right)\right)$$

The first term in the above equation determines how much of the long-term memory is kept while the second term adds new information to the cell state. The hidden state of the current time step is then determined by the output gate and a tanh function which limits the cell state between -1 and 1.

$$\boldsymbol{h}_t = \boldsymbol{o}_t \circ \tanh\left(\boldsymbol{c}_t\right)$$

3.4.4.3. Gated Recurrent Unit (GRU) RNN

Similar to LSTMs, the GRU solves the vanishing gradient problem of simple RNNs. The difference to LSTMs, however, is that GRUs use fewer gates and do not have a separate internal memory, i.e., cell state. Hence, the GRU only relies on the hidden state as a memory, leading to a simpler architecture

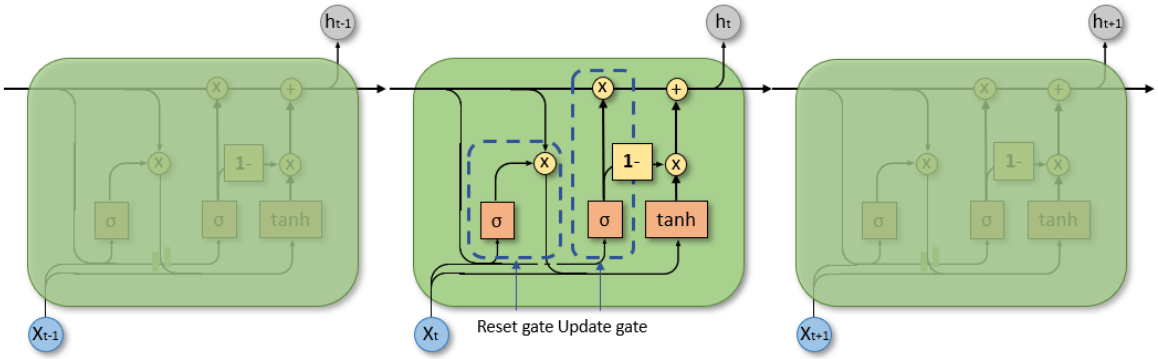


Figure 24 GRU architecture, figure from reference[64]

The reset gate is responsible for short-term memory as it decides how much past information is kept and disregarded.

$$\boldsymbol{r}_t = \sigma\left(\boldsymbol{W}_{r,x}\,\boldsymbol{x}_t + \boldsymbol{W}_{r,h}\,\boldsymbol{h}_{t-1} + \boldsymbol{b}_r\right)$$

The values in the vector r are bounded between 0 and 1 by a sigmoid function and depend on the hidden state h of the previous time step and the current input x. Both are weighted using the weight matrices W. Furthermore, a bias b is added.

The update gate, in contrast, is responsible for long-term memory and is comparable to the LSTM's forget gate.

$$\boldsymbol{u}_t = \sigma\left(\boldsymbol{W}_{u,x}\,\boldsymbol{x}_t + \boldsymbol{W}_{u,h}\,\boldsymbol{h}_{t-1} + \boldsymbol{b}_u\right)$$

As we can see the only difference between the reset and update gate are the weights W.

The hidden state of the current time step is determined based on a two-step process. First, a candidate hidden state is determined. The candidate state is a combination of the current input and the hidden state of the previous time step and an activation function. In this example, a tanh function is used. The influence of the previous hidden state on the candidate hidden state is controlled by the reset gate

$$\hat{\boldsymbol{h}}_t = \tanh\left(\boldsymbol{W}_{g,x}\,\boldsymbol{x}_t + \boldsymbol{W}_{g,h}\left(\boldsymbol{r}_t \circ \boldsymbol{h}_{t-1}\right) + \boldsymbol{b}_g\right)$$

In the second step, the candidate hidden state is combined with the hidden state of the previous time step to generate the current hidden state. How the previous hidden state and the candidate hidden state are combined is determined by the update gate.

$$\boldsymbol{h}_t = \boldsymbol{u}_t \circ \boldsymbol{h}_{t-1} + (1 - \boldsymbol{u}_t) \circ \hat{\boldsymbol{h}}_t$$

If the update gate gives a value of 0 then the previous hidden state is completely disregarded and the current hidden state is equal to the candidate hidden state. If the update gate gives a value of one, it is vice versa.

For our project, we built three RNNs considering either 1 or 2 hidden layers and adding a fully connected linear layer.

3.8.2 Data aggregation

The first step of data aggregation was the concatenation of brain activity of the two runs of the task. Then, to decode word embeddings from brain activity associated with each word, we needed to select the HFB activity corresponding to each word and then give it as the input to the RNN.

The 2D input matrix of HFB activity was arranged as a number of samples (or sequence length) x number of electrodes (input features).
To select brain activity correspondent to each word we used the onset and offset time, at which each word was spoken, derived from the TextGrid file. In addition, since brain activity related to semantics is anticipatory with respect to word pronunciation, we selected brain activity -200 ms before the onset time. By selecting the HFB activity of each word, the input had dimensions equal to the number of words multiplied by the number of samples for each word multiplied by the number of electrodes.
Considering that each word had a different sequence length, the input data was aggregated as a list of tensors and not a 3D matrix (because the second dimension was different across words). The methods that we applied to deal with input data of variable sequence length are detailed in section 3.8.3.

The output of the model was arranged as a 2D matrix with the number of words x dimensions (features) of embedding space. Since the BERT model generates word embeddings using 768 dimensions the network had to predict 768 values for each word.

The decoding pipeline was implemented using *Pytorch*[65]. The *torch.nn.RNN* requires the input and output in a specific format:

- input must be a tensor of shape: N x L x H_{in} (when batch- first = True)
- output is provided in the format: N x L x H_{out} (when batch- first = True)

where:

- N = batch size
- L = sequence length
- H_{in} = input size, that is the number of features in the input
- H_{out} = hidden size, that is the number of features in the hidden state

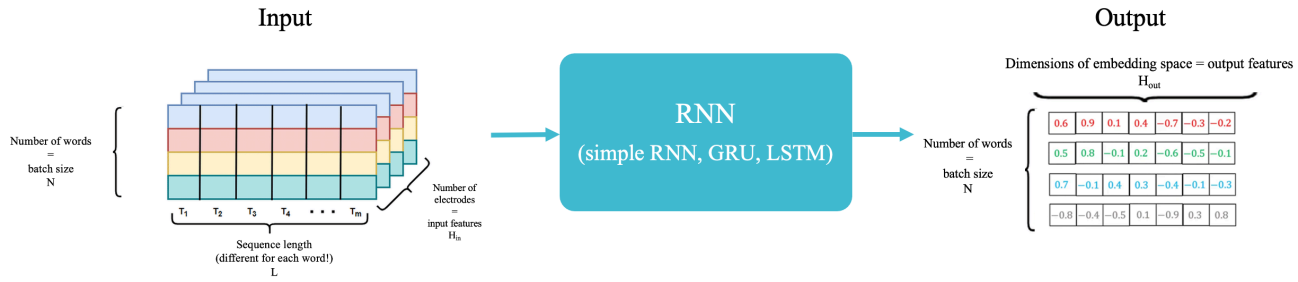


Figure 25 Block diagram of input and output data applied to the decoding model

3.8.3 Training

The main advantage of an RNN is that it can deal with varying length inputs. However, when training the model in batches, the items within a batch are still required to be of the same size. This problem can be dealt with in three ways:

1. padding
2. using a batch size of 1
3. forcing an equal batch size

We tried all three approaches. When we padded the data, we implemented a dynamic RNN that ignored the padded values, while when we applied the other strategies, the RNN was a classical one. For all three batching methods we trained simple RNN, GRU, and LSTM networks.

The dataset was randomly split in:

- training set 80% (x words for sub-1, x words for sub-2)

- test set 20% (x words for sub-1, x words for sub-2)

Each model was trained to minimize the mean square error (MSE) and all models were trained for 400 epochs. For training with batch size = 1 we applied SGD optimizer, in the other two cases we applied Adam optimizer.

The table below contains a list of the hyperparameters of each model.

Table 3.3: Hyperparameters applied during training.

Hyperparameters	
Parameter	Values
number of layers	1,2
number of hidden units in each layer	ranging from 50 to total number of features in the output
fully connected layer	yes or no
learining rate	0.0001, 0.0005
weight decay	0, 0.00001
batch size	12, 24, 32 *
number of epochs	200, 300, 400

The steps for the training and testing procedure are summarized in the following flow chart. After setting the hyperparameters configuration the environment was set up (all models were trained using a single graphics processing unit NVIDIA GeForce RTX 2080 Ti). Then the model was set up, as previously said we tried 3 different types of RNN and in each case, the hidden states were initialized to zero.

After this, the training in batches starts. First input data were normalized according to the mean and standard deviation computed on the entire training set. Then the output of the network was computed, during forward propagation, and the training loss (according to MSE) was also computed. Then the backpropagation of the error was performed, and weights were updated.

After the training loop, the validation loop starts, and all previous steps were repeated except for back propagation.

Hence, we computed the training and validation loss for each epoch and we checked the state of the model after every 50 epochs. Since the number of features (number of dimensions in embedding space) was extremely high (768) we also trained the model to predict the word embeddings obtained from PCA considering only the first 50 principal components.

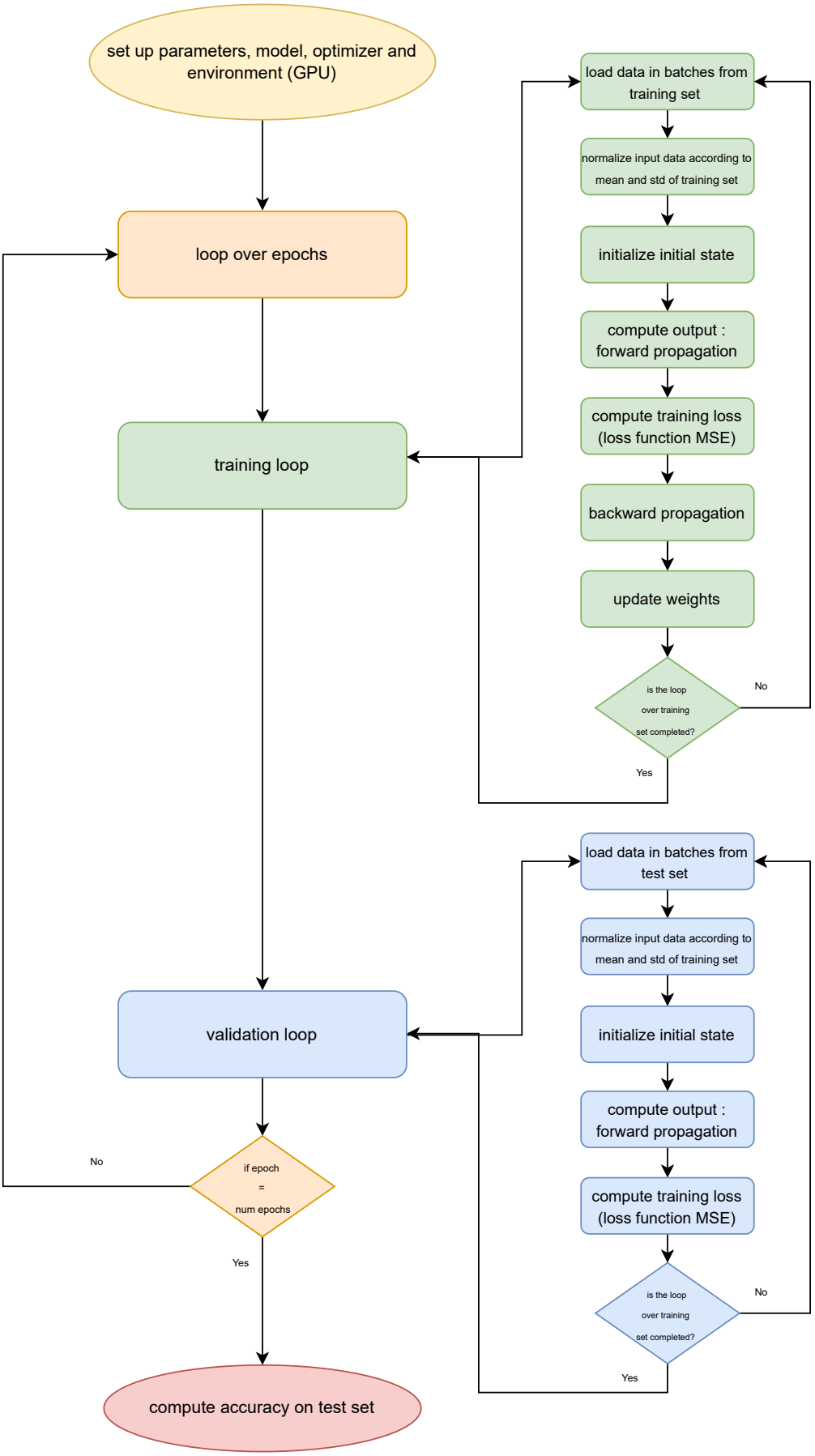


Figure 26 Flowchart representing steps to train an test the decoding model

3.8.4 Testing

As previously mentioned, to the best of our knowledge, there are no studies in the literature that try to decode word embeddings from brain activity. Therefore, to measure the accuracy of the decoding we set up a new method.

Since the aim is not to exactly decode each value of the word embeddings, we decided to measure the accuracy using cosine similarity as a metric to assess the quality of the decoded embeddings, according to which similar words have cosine similarity around 1 and dissimilar words around 0.
For each predicted word we computed its cosine similarity with all the actual word embeddings. Then we sorted in descending order the cosine similarity matrix and we looked for the position of the word embedding of the word of interest. Ideally, the cosine similarity between the predicted and actual word embeddings of the same word should be equal to 1 and hence the position of the aforementioned word embedding on the ranked similarity matrix should be the first. However, since each embedding is represented with many dimensions (768 values) and since there are many similar words in this dataset, it is unlikely that this happens. Thus, we assessed the accuracy of decoding each word setting 4 ranges:

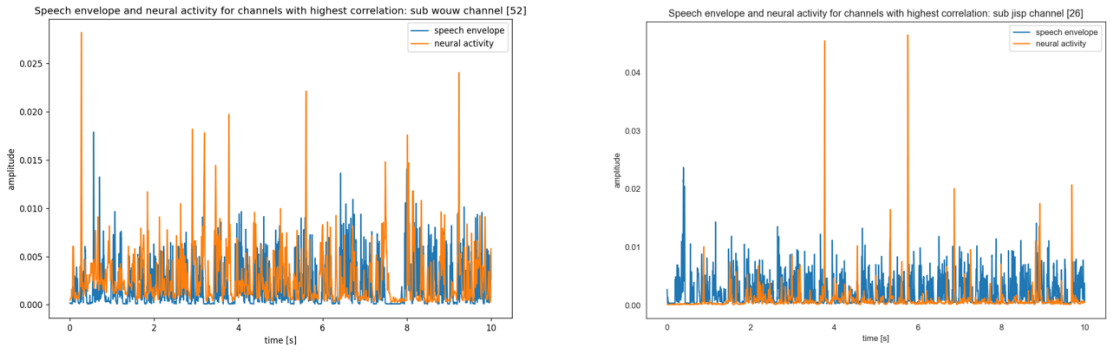
- if the position of the actual word was between the first 15% of words of the ranked cosine similarity matrix the accuracy was excellent
- if the position of the actual word was between the first 15% to 35% of words of the ranked cosine similarity matrix the accuracy was good
- if the position of the actual word was between the first 35% to 50% of words of the ranked cosine similarity matrix the accuracy was fairly good
- if the position of the actual word was between the first 35% to 50% of words of the ranked cosine similarity matrix the accuracy was bad

Chapter 4: Results

4.1 Speech envelope entrainment results

The aim of speech envelope tracking analysis was to check if there was correlation between brain activity and speech and to assess the quality of brain data.

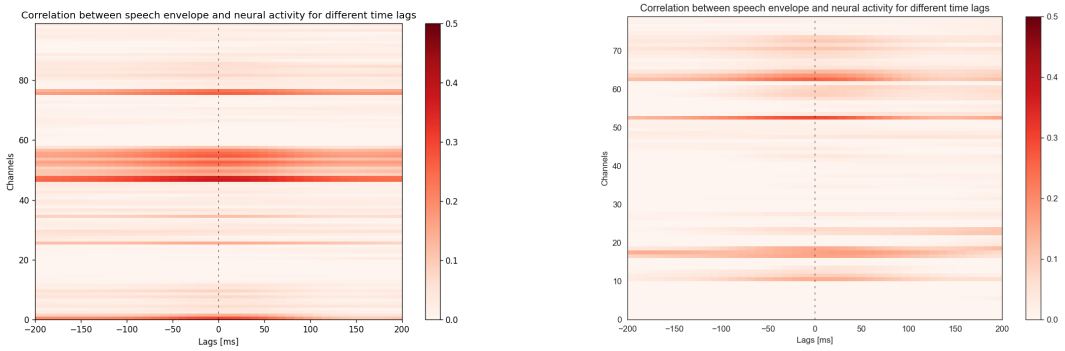
Figure 27 shows the time course of HFB activity of one of the channels superimposed on the time course of the envelope of the spoken story. The figures demonstrate that the neural signal faithfully tracks the speech envelope, in particular we can observe a better tracking is achieved for brain activity of participant 1.



(a) (b)

Figure 27 High gamma activity [60-150 Hz] tracks the speech envelope. In blue we observe the time course of the speech envelope while in orange it is reported the time course of HFB activity. Panel (a) shows results for sub-1, panel (b) shows results for sub-2.

Since we compute the correlation between brain activity and speech envelope for different time lags, between -200 and +200 ms, we also explored which was the time lag that provided the maximum correlation. Figure 28 a and b show the individual correlation profiles per channel over different time lags. As we can see for both subjects the maximum correlation is obtained for a time lag between -50 to 50 ms, which agrees with previous studies[57].



(a) (b)

Figure 28 shows individual correlation profiles per channel over different time lags. The x-axis represents cross-correlation lags (in milli seconds), where negative lags indicate that the neural activity precedes audio. As we can see the maximum correlation is achieved using a time lag between -50 to 50 ms and there are several electrodes (especially for sub-1, left panel) that have correlation coefficients higher than 0.3.

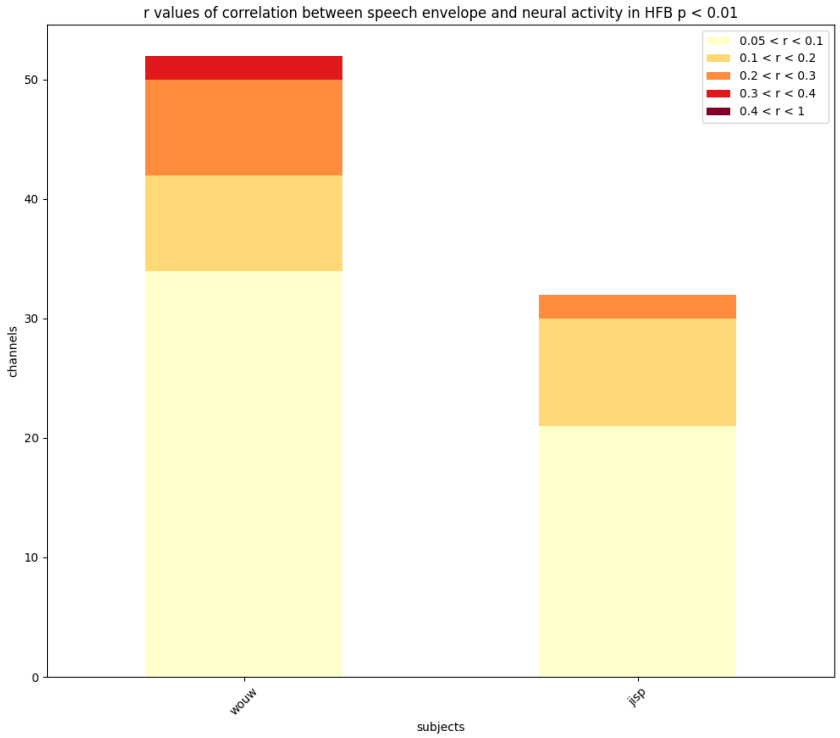


Figure 29 Channels with significant correlation ($p < 0.01$) between speech envelope and neural activity in HFB. The color bar represents different values of correlation. For each subject the figure shows the number of channels that have significant correlation coefficient.

A comparison between the correlation results of sub-1 and sub-2 is described in figure 29. The plot shows the number of channels that have a significant correlation ($p < 0.01$). As we can see most channels have correlation between 0.05 and 0.1, which is quite low, but this result was expected because not all electrodes were implanted in regions related to auditory processing.

The fundamental result of this analysis was that for both participants we obtained some electrodes with a high correlation between HFB activity and speech envelope (for sub-1 19 channels had correlation coefficient > 0.2, for sub-2 15 channels had correlation coefficient > 0.2). The presence of channels with high correlation between HFB activity and speech envelope proved that the brain activity of both individuals was correlated to speech and that brain data had high quality.
After this analysis, that was performed as a checkpoint of our preprocessing steps, we were ready to start the decoding pipeline.

4.2 Decoding results

The decoding results are described firstly considering findings obtained using the original word embeddings derived by the BERT model (section 4.2.1), (in which each word is represented using 768 dimensions) and then considering results obtained using the reduced embedding space (section 4.2.2), (in which each word is represented using 50 dimensions that are the first 50 principal components projected into the embedding space after PCA).

As previously mentioned, our input data to the RNN model had variable sequence lengths (due to the different duration of each word). We approached this problem considering padded input sequences, input sequences with batch size equal to 1, and input sequences forced to have the same length, thus allowing us to train the network with batch size greater than 1.
According to this different aggregation of input data, results are reported in each subsection of paragraph 4.2 separately, describing firstly results obtained training the network with batch size equal to 1 and secondly results obtained training the networks with batch size greater than 1.
Unfortunately, the results for padded input sequences were not even good during the training phase. The predicted values for each embedding dimension were almost close to zero and the network was not able to distinguish between different input sequences. This was due to the fact that there were too many padded values for many sequences and hence the network was not able to learn the pattern in the data. For these reasons results for input padded sequences were inconclusive are not reported in the following paragraphs of this thesis.

4.2.1 Decoding original word embeddings

4.2.1.1 Decoding with batch size equal to 1

As the first attempt, we tried to decode all dimensions of embedding space using RNNs with a number of hidden units, in the hidden layers, equal to the number of output features (768) and a fully connected linear layer as the output layer. As we can see from figure 30, that shows the training and validation losses over epochs, the results were not good because the network was overfitting a lot. We tried to change some hyperparameters (such as adding regularization and using 1 hidden layer instead of 2) and the type of RNN (GRU or LSTM) but even after several attempts, the high overfitting remained.

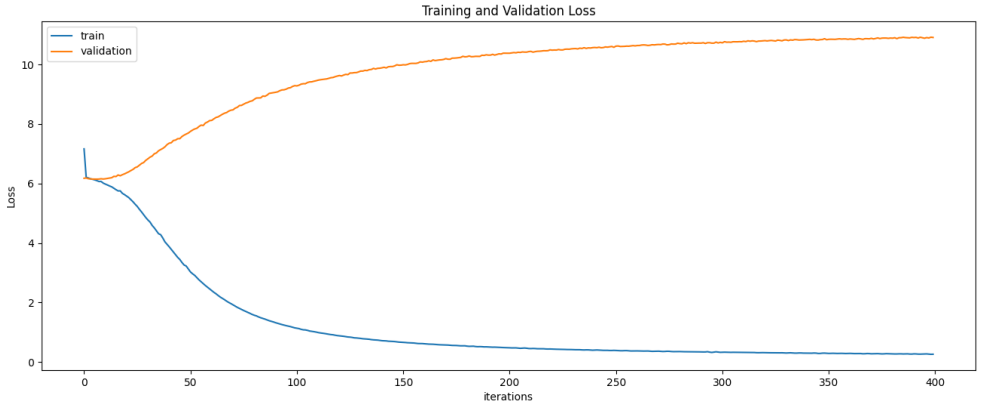
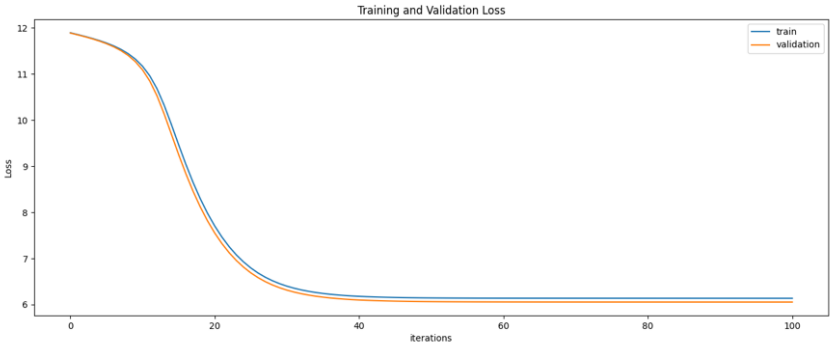


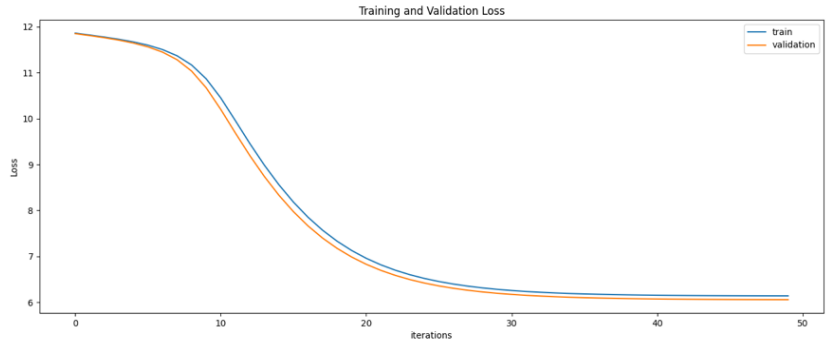
Figure 30 participant 2 Training and validation losses obtained using a GRU network with 768 hidden units on the hidden layer, 1 hidden layer and 1 fully connected layer.

Then we tried to solve the overfitting reducing the complexity of the network, by decreasing the number of hidden units in the hidden layers. Figure 31 shows results obtained using a GRU with 50 hidden units in the hidden layer and we can see that the results improved with respect to the previous configuration because now both training and validation losses decrease over epochs. However, using only 50 hidden units in the hidden layer, the MSE decreased by only 50% which was not enough to obtain a good accuracy performance (accuracy = 52%). So, we tried to optimize the network by changing the hyperparameters.

Figure 31 *a* shows results using a GRU with 50 hidden units and 1 hidden layer, while figure 31 *b* shows results for the same network using 2 hidden layers. From those figures we can see that there is no change in results using a network with 1 or 2 hidden layers (accuracy rates for both configurations of the network were always equal to 52%).



(a)



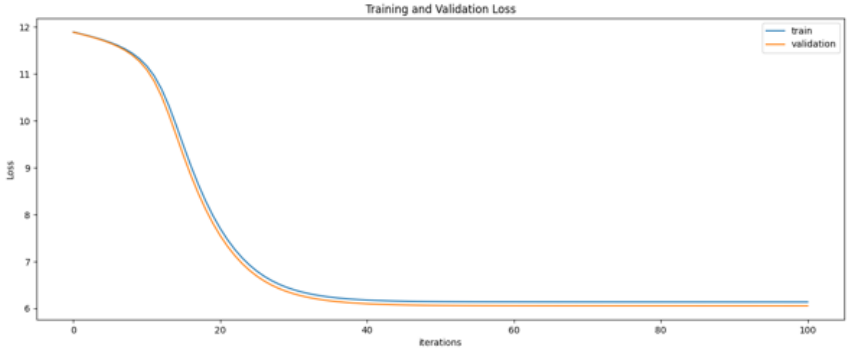
(b)

Figure 31

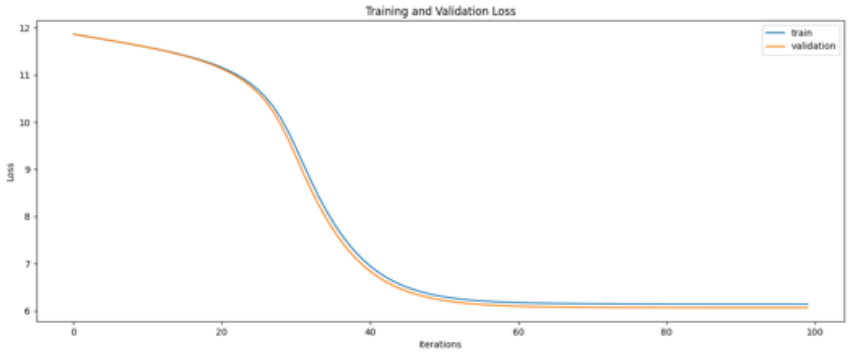
Panel (a) participant 1 GRU, 50 hidden units, 1 hidden layer and 1 fully connected layer, learning rate = 0.0005 weight decay = 0

Panel (b) participant 1 GRU, 50 hidden units, 2 hidden layer and 1 fully connected layer, learning rate = 0.0005 weight decay = 0

Then, considering the same number of hidden units, we changed the type of network to see whether this could lead to a higher minimization of MSE. Figures 32 *a* and *b* show the comparison between GRU and LSTM networks with 50 hidden units in the hidden layer and 1 hidden layer. As we can see there is no difference in the results given by the two networks, they are both not able to minimize the MSE by more than 50 %.



(a)



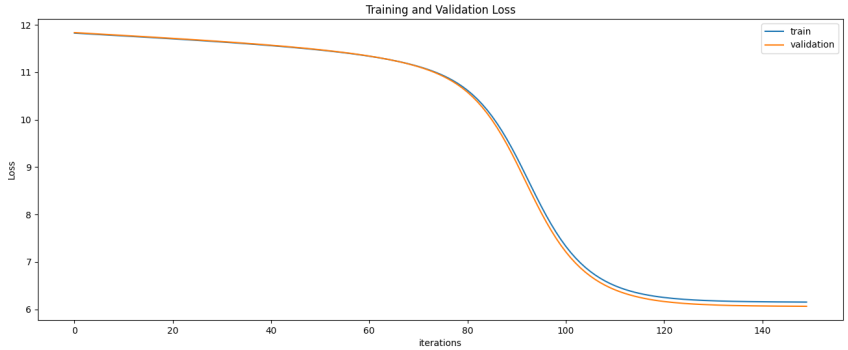
(b)

Figure 32

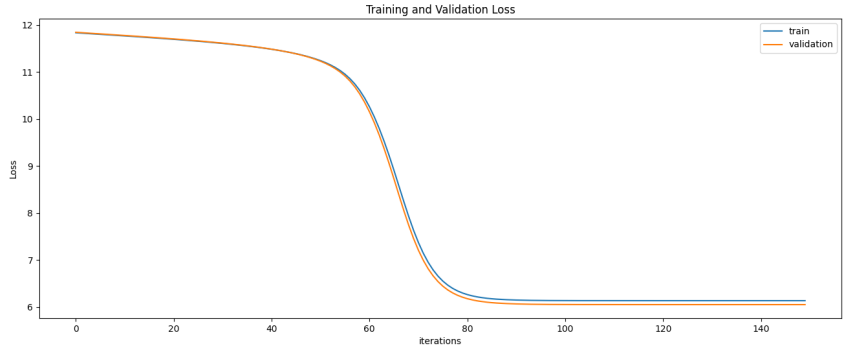
Panel (a): participant -2 GRU, 50 hidden units, 1 hidden layer and 1 fully connected layer, learning rate = 0.0005 weight decay = 0

Panel (b): participant -2 LSTM, 50 hidden units, 1 hidden layer and 1 fully connected layer, learning rate = 0.0005 weight decay = 0

Since the change in hyperparameters did not produce different results, we tried to increase a bit the complexity of the network by using more hidden units (200 and 400) on hidden layers. Figures 33 *a* and *b* show the comparison between results for LSTM with 200 hidden units (panel a) and LSTM with 400 hidden units. As can be seen, using 400 hidden units rather than 200 only modifies the speed of learning (using 400 hidden units the network minimizes the error of 50 % in 80 epochs, while using 200 hidden units the network minimizes the error of 50 % in 130 epochs) but does not change the learning capacity of the network. Thus, even increasing the complexity of the network the model is not able to minimize the MSE by more than 50 %.



(a)



(b)

Figure 33

Panel (a): participant 1 LSTM, 200 hidden units, 1 hidden layer and 1 fully connected layer, learning rate = 0.0001 weight decay = 0

Panel (b): participant 1 LSTM, 400 hidden units, 1 hidden layer and 1 fully connected layer, learning rate = 0.0001 weight decay = 0

After trying different configurations of the network and different values of hyperparameters we concluded that using batch size equal to 1, the network is, on one hand, able to minimize the MSE using 768 hidden units on hidden layers, but is not able to generalize because there is a lot of overfitting; and on the other hand, the network is not able to minimize the MSE more than 50% but reducing the complexity of the problem there was no more overfitting.

Accuracy results for all different decoding models are reported in Table 4.1. From the table, we can see that there is no difference among results provided by different models because training with batch size equal to 1 the network is not able to minimize MSE by more than 50% in no one of the considered configurations. In addition, we can see that accuracy results are almost the same between participant 1 and participant 2.

Table 4.1 Accuracy results obtained training the network with batch size equal to 1. Participants are indicated as P1.

Network	Number of hidden units	Number of layers	Learning rate	Weight decay	Accuracy P1	Accuracy P2
GRU	50	1	0.0005	0	0.5238	0.5233
GRU	50	2	0.0005	0	0.5238	0.5233
LSTM	50	1	0.0005	0	0.5238	0.5233
LSTM	200	1	0.0001	0	0.5238	0.5233
LSTM	400	1	0.0001	0	0.5238	0.5233

4.2.1.2 Decoding with batch size greater than 1

We started training the network with small batch size, equal to 12. At the first attempt, we tried to decode all dimensions of embedding space using RNNs with 768 hidden units in the hidden layer. Figure 34 shows that the results obtained for this configuration are not good because the network is overfitting a lot. We tried to change some hyperparameters (such as adding regularization and using 1 hidden layer instead of 2) and the type of RNN (GRU or LSTM) but the high overfitting remained.

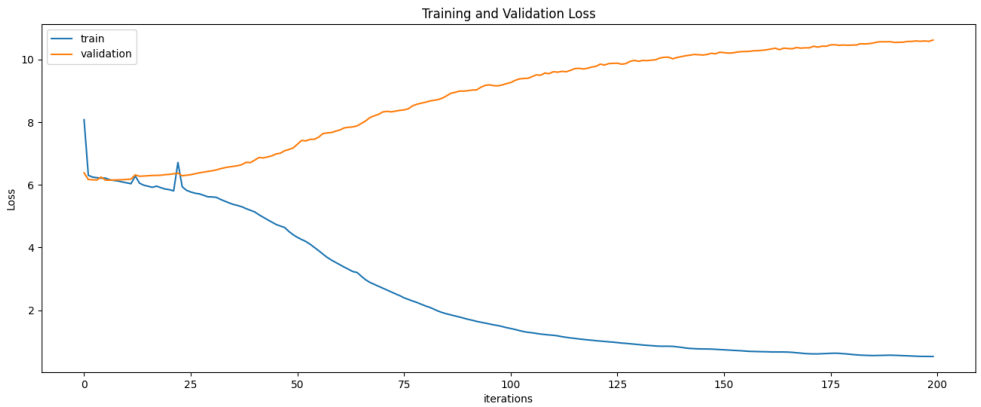
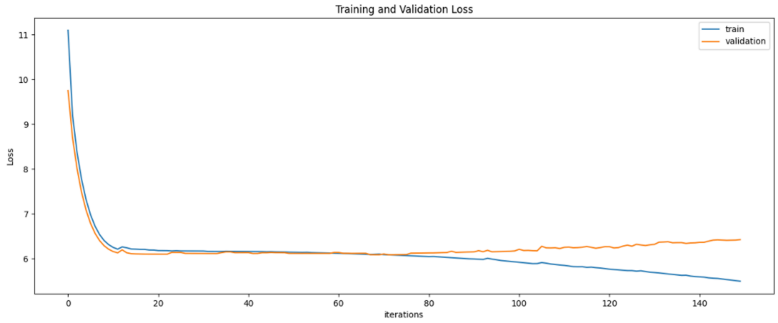


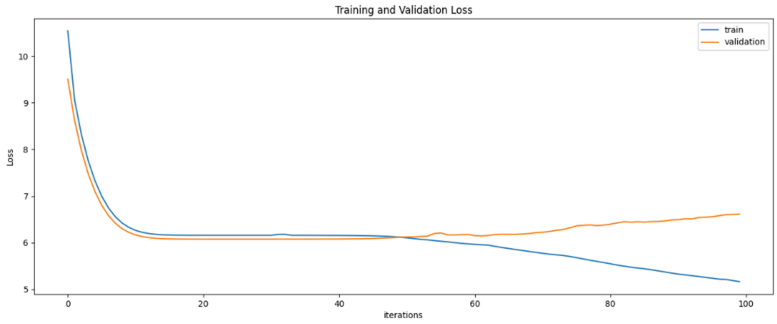
Figure 34 Participant 1 batch size = 12, GRU 768 hidden units, 1 hidden layer and 1 fully connected layer, learning rate = 0.0005 weight decay = 0

Then, as we did for training with a batch size equal to 1, we tried to solve the overfitting reducing the complexity of the network, by decreasing the number of hidden units in the hidden layers.

First, we tried to decrease the number of hidden units to 50. Figure 35 *a* shows results obtained using a GRU with 50 hidden units and 1 hidden layer and we can see that the results are better compared to figure 34 because both training and validation losses decrease over epochs.
However, in this configuration, the MSE decreased down to 50 % very quickly (in the first 10 or 15 epochs) but then it remains stable for 50 epochs both for training and validation loss and then training loss starts to decrease again while validation loss increases. This behavior shows that the network can learn very quickly but then it overfits.
To solve the overfitting problem, we tried to change some hyperparameters. Figure 35 *a* and *b* shows results obtained using a GRU with 50 hidden units and 1 hidden layer (panel *a*) and 2 hidden layers (panel b). As we can see the number of hidden layers does not change the results.



(a)



(b)

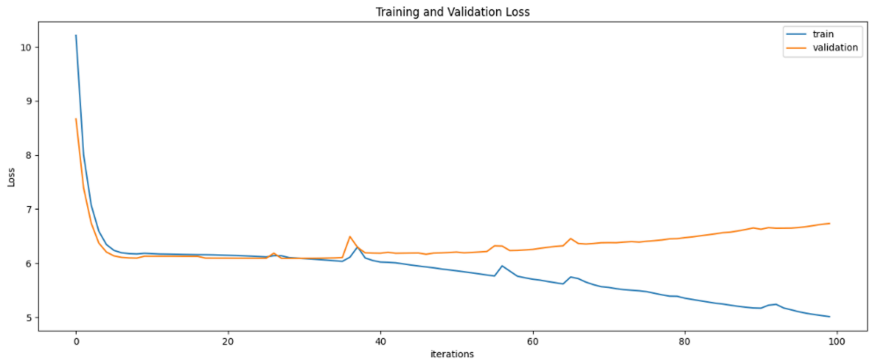
Figure 35

Panel (a): participant 1 Batch size = 12, GRU, 50 hidden units, 1 hidden layer and 1 fully connected layer, learning rate = 0.0005 weight decay = 0

Panel (b): participant -1 Batch size = 12, GRU, 50 hidden units, 2 hidden layer and 1 fully connected layer, learning rate = 0.0005 weight decay = 0

Then we tried to increase a bit the complexity of the network, to see if using more hidden units the ability of the network to learn could improve. Figure 36 shows the results obtained from training the GRU network with 200 hidden units (panel *a*) and 400 hidden units (panel *b*) on the hidden layer. As can be seen, using 200 hidden units does not change a lot the results

compared to 50 hidden units (fig 35 *a*) while increasing the number of hidden units up to 400 improves the learning capacity of the network, because minimizes the MSE down to 80%, but reduces its ability to generalize because the network is overfitting a lot.



(a)

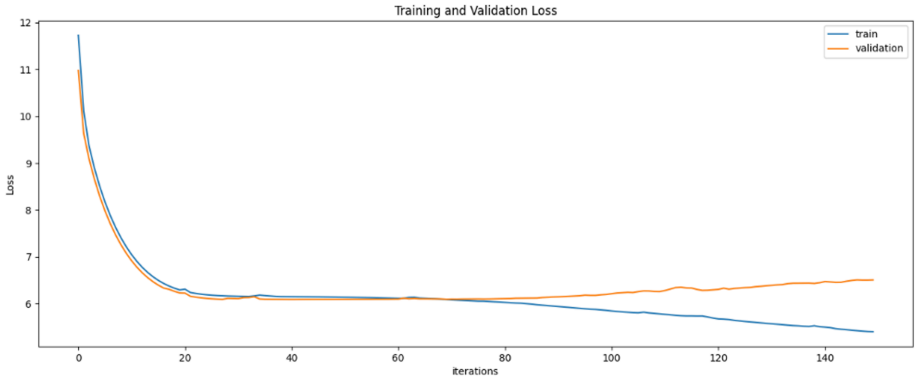
(b)

Figure 36

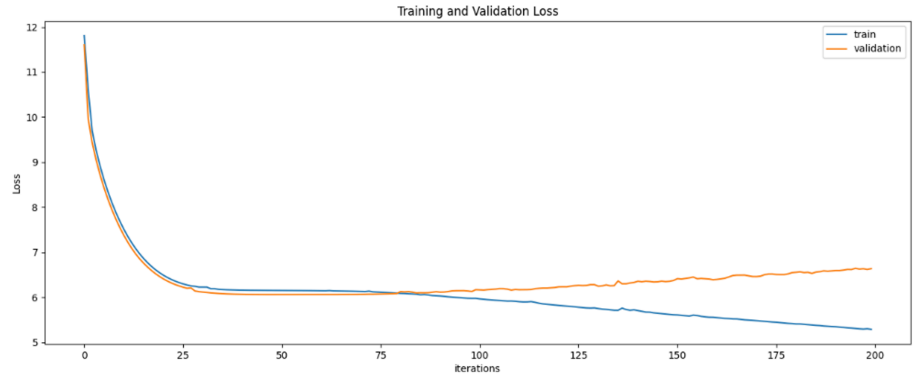
Panel (a): Batch size = 12, GRU, 200 hidden units, 1 hidden layer and 1 fully connected layer, learning rate = 0.0005 weight decay = 0

Panel (b): Batch size = 12, GRU, 400 hidden units, 1 hidden layer and 1 fully connected layer, learning rate = 0.0005 weight decay = 0

Then we tried to increase the batch size, but unfortunately training the network with a batch size equal to 24 (figure 37 *a*) or equal to 32 (figure 37 *b*), did not modify the behavior of the network.



(a)



(b)

Figure 37

Panel (a): Batch size = 24, GRU, 100 hidden units, 1 hidden layer and 1 fully connected layer, learning rate = 0.0005 weight decay = 0

Panel (b): Batch size = 32, GRU, 100 hidden units, 1 hidden layer and 1 fully connected layer, learning rate = 0.0005 weight decay = 0

After trying various network configurations and hyperparameter values, we reached the following conclusions. When we train the network with a batch size greater than 1 and a large number of hidden units in the hidden layers (200 or 400) the network is capable of reducing the mean squared error but overfitting becomes a significant issue, resulting in poor generalization performance. On the other hand, when the network is composed of a lower number of hidden units (50 or 100), it is not able to reduce the MSE by more than 50%, but overfitting is reduced. Furthermore, increasing the batch size beyond 12 up to 32 did not provide significant improvements in decoding performance.

Table 4.2 reports the accuracy results of various decoding models. The table show that there are no discernible differences in the results provided by the different models. This is because the network is not able to reduce MSE by more than 50% for any of the configurations tested. Additionally, the accuracy results show little variation between participant 1 and participant 2.

Table 4.2: accuracy results obtained training the network with batch size equal to 1. Participants are indicated as P1 and P2.

Batch size	Network	Number of hidden units	Number of layers	Learning rate	Weight decay	Accuracy P1	Accuracy P2
12	GRU	50	1	0.0005	0	0.1344	0.1353
12	GRU	50	2	0.0005	0	0.1344	0.1353
12	GRU	200	1	0.0005	0	0.114	0.1146
12	GRU	400	1	0.0001	0	0.118	0.1188
24	GRU	100	1	0.0001	0	0.134	0.1353
32	GRU	100	1	0.0001	0	0.1344	0.1353

4.2.2 Decoding word embeddings after dimensionality reduction

After failing to achieve good results while attempting to decode all 768 dimensions of the original embeddings we attempted to reduce the dimensionality of the embedding space by applying PCA and then decode the reconstructed embeddings using only a subset of the principal components.

Before to start the decoding of word embeddings reconstructed after PCA, we checked the quality of the new generated embeddings by comparing the cosine similarity between pairs of words on the original embedding space and on the reduced embedding space. Unfortunately, the dimensionality reduction process via PCA significantly corrupts the quality of the embedding space. Figure 38 *a* shows the heatmap of cosine similarity between words (repeated at least 5 times) computed using all the original embedding dimensions. At the same time figures 38 *b*, *c*, and *d* show the heatmap of cosine similarity of the same words computed using the first 50 (panel *b*),100 (panel *c*), and 200 principal components respectively (panel *d*).
As we can see from the figures, the quality of embedding space represented using PCA is inferior because the projection of principal components on the embedding space is able to represent quite accurately only the words that have high similarity (i.e., those corresponding to dark green squares in the heatmaps). In contrast, PCA decomposition fails to appropriately represent words with similarity below 0,6 reducing their similarity to 0.3 or below in the reduced embedding space.

These observations are further supported by Table 4.3 which represents cosine similarity between pairs of similar words related to the cluster “people”. The table shows that words with high cosine similarity in the original embedding space, such as ‘zoontje’ and ‘baby’ (highlighted in green) still have high cosine similarity in the embedding space represented with the first 50 principal components. However, words with lower, but still high, cosine similarity in the original embedding space, like “zus” and “kind” with similarity equal to 0.45 or “zus” and “jongetje” with similarity equal to 0.51 (highlighted in red), have very low cosine similarity in the embedding space represented with the first 50 principal components.

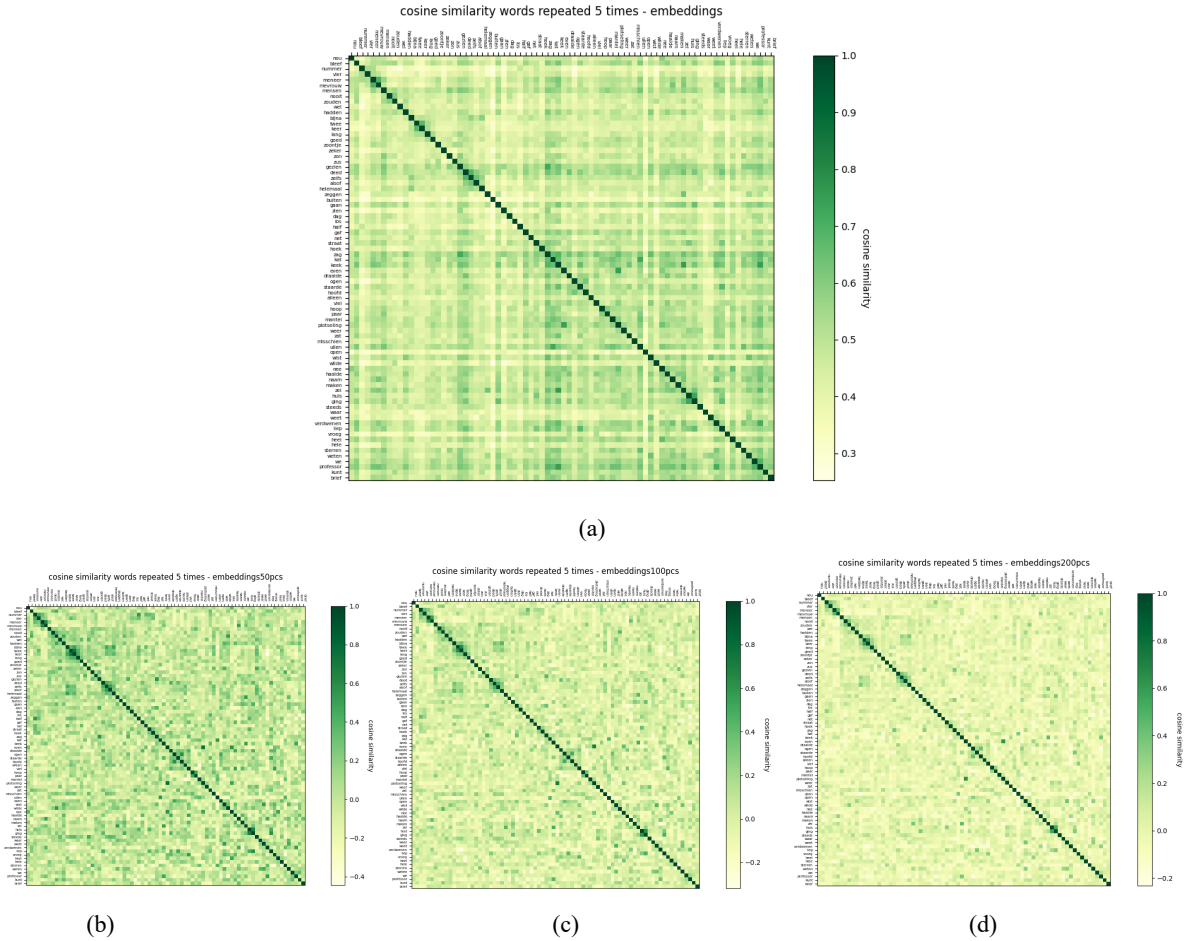


Figure 38 heatmap of cosine similarity between words repeated at least 5 times for participant 1. Panel (a) shows cosine similarity computed using all 768 dimensions of original embedding space. Panel (b), (c) and (d) respectively show results of cosine similarity computed using the first 50, 100 and 200 principal components of the embedding space.

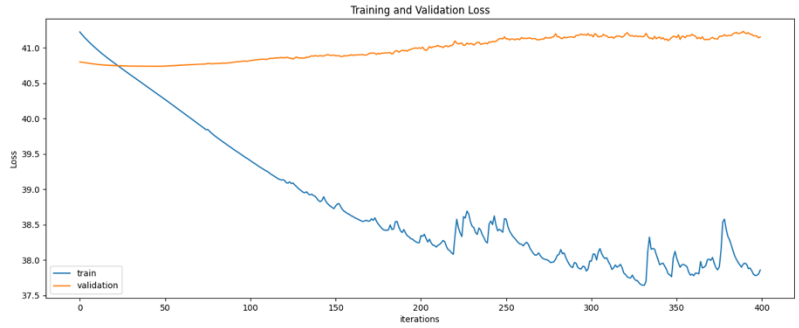
Table 4.3: Results of cosine similarity between words with similar meaning in the original embeddings space and in the embeddings, space represented with the first 50 principal components (all words are related to "people": zoontje = son, zus = sister, kind = child, jongetje = little boy, meneer = sir)

Word 1	**Word 2**	**Cosine similarity between words in the original embedding space**	**Cosine similarity between words in the reduced embedding space (50 principal components)**
zoontje	zoontje	1	1
zoontje	zus	0. 54	0.389
zoontje	kind	0.60	0.39
zoontje	jongetje	0. 67	0.54
zoontje	baby	0.75	0.74
zoontje	meneer	0. 53	0.13
zus	zoontje	0. 53	0. .39
zus	zus	1	1
zus	kind	0. 45	0.024
zus	jongetje	0. 51	0.075
zus	baby	0.53	0.35
zus	meneer	0. 48	0.08

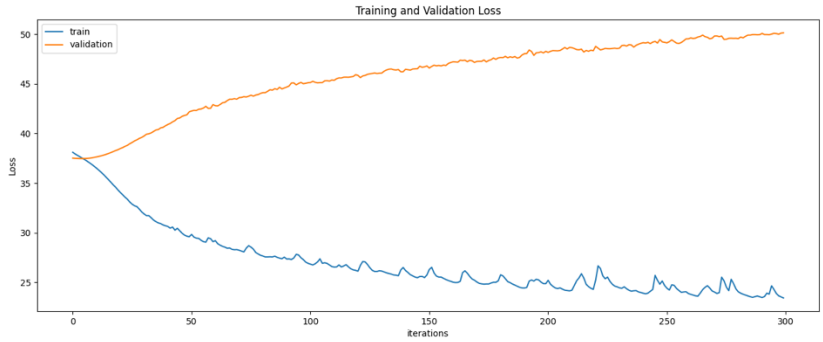
Due to this poor reconstruction of embedding space using only the first 50, 100, and 200 principal components we were not expecting to obtain high-quality results in terms of decoding computed on the reduced embedding space. Considering that by using the first 50 principal components we can preserve the largest amount of variance in the data (as we can see from figure 38) we decided to decode word embedding using only the first 50 principal components of embedding space.

Figure 39 *a* shows the decoding results training the network to predict the word embeddings (using only 50 dimensions) with a batch size equal to 1, while figure 39 *b* shows the decoding results training the network with a batch size equal to 12. As we can see the performance of the network are even worse than the one that we obtained training the network to predict all dimension of embedding space, because there is a lot of overfitting and MSE is minimized by only 9 %, when training with batch size equal to 1 (panel a) and of 40% when training with batch equal to 12.

As previously said, those results were expected since the dimensionality reduction by means of PCA is not able to correctly preserve the similarity between words.



(a)



(b)

Figure 39

panel a: Batch size = 1, GRU, 50 hidden units, 1 hidden layer and 1 fully connected layer, learning rate = 0.0005 weight decay = 0

panel b: Batch size = 12, GRU, 50 hidden units, 1 hidden layer and 1 fully connected layer, learning rate = 0.0005 weight decay = 0

Chapter 5: Discussions

5.1 Discussions and limitations

This exploratory feasibility study aimed to decode the semantic representation of words (in the form of word embeddings) spoken during a natural reading task from brain activity measured through sEEG.

The decoding results, as reported in Tables 4.1 and 4.2 in Chapter 4, indicate poor performance, with the highest accuracy of 52% achieved training the network with a batch size equal to 1, which is nearly chance level.
To investigate the reason behind our poor results, in this chapter we attempt to identify potential mistakes that we may have committed and the main limitation of this exploratory study.

Considering that decoding performance is unsatisfactory both for participant 1 and participant 2 (see Table 4.1 and 4.2) we can exclude the possibility of poor accuracy due to the quality of participants' data (even if the study was conducted only with two individuals). The fact that participant's data is not the cause of poor decoding results is also supported by results of speech envelope entrainment analysis (paragraph 4.1) that proved that brain data for those individuals had good quality and were correlated to speech at least for some electrodes. Therefore, we can conclude that the cause of our poor results cannot be attributed to the subject's data but could be either due to the network or to the decoding task.

Results obtained training the network with 768 hidden units on the hidden layer, both considering bath size =1 and batch size greater than 1, show that using such a high number of hidden units allows minimizing the MSE almost close to zero, but on the other hand led to high overfitting (see Figure 30 and 34).
At the same time, results obtained training the network with a smaller number of hidden units in hidden layers (such as 100 or 200 hidden units) show that the network in this configuration can minimize MSE only down to 50 %, but on the other hand, is able to generalize because both training and validation loss decrease over epochs.

Considering those two aspects it becomes evident that for the task of our project is difficult to identify a tradeoff between the minimization of MSE and the ability of the network to generalize.

To try to solve the problem of overfitting we also trained the network using a batch size greater than 1 (because from the literature it has been proved that training with a batch size equal to 1 might lead the network to find a local minimum rather than to find a maximum minimum[66]). However, as we can see from figure 31 and figure 35, results obtained training the network with a batch size equal to 1 and a batch size greater than 1 are almost the same because in both cases the network cannot reduce the MSE by more than 50 %.
In addition, considering that decoding performance is almost the same even changing the type of network (RNN, GRU, and LSTM) and even changing hyperparameters values (see table 4.1 and 4.2) we can conclude that the poor quality of results does not depend on the network itself but is probably due to the decoding task that is too demanding.

Comparing our findings with those of other studies is challenging because no previous study has attempted to decode word embeddings from invasive EEG recordings. The only other study that performed a similar analysis to ours is the one conducted by Goldstein and colleagues[38] where they trained a convolutional neural network to predict word embeddings of the next word given previous words from ECoG data acquired during a listening task. Goldstein and colleagues compared the performance of different language models and achieved good accuracy results (65% accuracy using GloVe, 75% accuracy using GPT) training the network on data from 10 subjects. However, unlike our project, Goldstein et al. aimed to decode only the probability of the next word within a sequence, rather than decoding all words based on their word embedding representation. Therefore, it is evident that the complexity of our decoding task is much higher than the one conducted by Goldstein and colleagues.

On one hand, our decoding task might be too difficult because the coverage of the sEEG electrode for our participants might be too sparse. Unfortunately, we do not know exactly which are the brain areas penetrated by each electrode, but looking at Figure 11, which shows electrodes' implantation, we can see that electrodes are located quite sparsely over the brain and they might also penetrate brain areas with a lot of white matter that might not contain significant brain activity to decode semantic representation of speech.

On the other hand, decoding word embeddings directly from brain activity might be a too difficult task because of the language model that we are applying. The BERT language model that we applied in this project generates word embeddings using 768 dimensions to represent each word. This means that to reconstruct the output, the network must predict 768 values for each word, which might be a very demanding task.

Another possible cause of the poor decoding results that we obtained is related to the fact that the BERT model might not be the best choice to generate embeddings when we aim to decode word embeddings directly from brain activity. This might be due to the fact since the BERT model is contextual, the word embeddings generated by the model are too similar to each other, also depending on the context, and hence there is not enough variability between words to be decoded. For this reason, the BERT model might be more suitable for next word prediction tasks (as GPT, another transform model, which was used by Goldstein et al. to predict the next word in a sequence from brain activity). However, since no other study before has ever tried to decode word embedding from brain activity and since transformer models, such as BERT, are the state of the art of NLP, we decided to use the BERT model for our project. At the end of this exploratory research study, we can speculate that it might not be suitable to use a contextual language model to decode word embeddings without cooperating with the context they are used during the training of the decoder.

Despite the decoding results that we obtained are characterize by poor accuracy, this project is the first that attempts to decode word embeddings directly from brain activity measured through sEEG, and our findings could help future studies in further developing this application for speech BCIs based on semantic decoding.

5.2 Future work

To develop a decoding system based on semantic representation of speech, future works should attempt to reduce the complexity of decoding word embeddings directly from brain activity. One approach could be to explore the use of different language models. Even if BERT model is the state of the art of NLP, it is quite complex because is contextual and uses a large number of dimensions to construct the embeddings. Simpler models, such as word2vec or GloVe, can be applied to generate word embeddings using a lower number of dimensions. Although it is

well-established in the literature that those models may not generate embeddings as effectively as BERT, they can still be useful for a decoding task such as the one examined in our project. By using fewer dimensions, these models can produce word embeddings with less complexity, which may improve decoding performance.

To further advance the development of a semantic decoder, future studies should explore alternative electrode configurations and consider using ECoG electrodes instead of sEEG. While sEEG electrodes have the potential to record brain activity associated with semantics, due to their ability to penetrate deep regions of the brain, their sparse coverage may not provide sufficient information to decode semantic representation of speech. In contrast, ECoG grids, which are characterized by a higher density of electrodes, (although they are located only superficially and on the same grid) may enable a better recording of brain activity related to semantics. It is worth noting that these suggestions are speculative, as no prior attempts have been made to decode word embeddings directly from invasive EEG recordings.

Chapter 6: Conclusions

In this feasibility exploratory study we aimed to decode semantic representation of words, in the form of word embeddings, directly from brain activity measured though sEEG electrodes while two participants performed a natural reading task.

We applied the BERT language model to generate word embeddings and decoding model was implemented through recurrent neural networks (in different configuration such as simple RNN, GRU and LSTM). Even if we tried different network configurations, different hyperparameters values and different training strategies (including training the network with batch size equal to 1 and greater than 1) the highest accuracy achieved by our model was 52%, which is close to chance level.

After testing different strategies to improve the model's performance and after excluding that the source of error was due to the subject's data (as no difference was observed in performance of the two analyzed participants) we concluded that the poor results were due to the complexity of the decoding task we attempted to solve in our project. Based on our findings, future studies should try to reduce the complexity of the problem of decoding word embeddings directly from brain activity to obtain more accurate results.

Despite the poor decoding accuracy that we achieve, this project is the first study that attempts to decode word embeddings directly from brain activity measured through sEEG, and our findings could be a starting point for future studies to further developing this application for speech BCIs based on semantic decoding.

Bibliography

1. World Health Organization. & World Bank. *World report on disability*. (World Health Organization, 2011).
2. Chiò, A. *et al.* Global epidemiology of amyotrophic lateral sclerosis: A systematic review of the published literature. *Neuroepidemiology* vol. 41 118–130 Preprint at https://doi.org/10.1159/000351153 (2013).
3. Marin, B. *et al.* Variation in world wide incidence of amyotrophic lateral sclerosis: A meta-analysis. *Int J Epidemiol* **46**, 57–74 (2017).
4. Laureys, S. *et al.* The locked-in syndrome: What is it like to be conscious but paralyzed and voiceless? *Progress in Brain Research* vol. 150 495–511 Preprint at https://doi.org/10.1016/S0079-6123(05)50034-7 (2005).
5. Rousseau, M. C. *et al.* Quality of life in patients with locked-in syndrome: Evolution over a 6-year period. *Orphanet J Rare Dis* **10**, (2015).
6. Bruno, M.-A. L. *et al.* A survey on self-assessed well-being in a cohort of chronic locked-in syndrome patients: happy majority, miserable minority. doi:10.1136/bmjopen-2010.
7. Wolpaw, J. R., Birbaumer, N., Mcfarland, D. J., Pfurtscheller, G. & Vaughan, T. M. *Brain-computer interfaces for communication and control*. *Clinical Neurophysiology* vol. 113 www.elsevier.com/locate/clinphCLINPH2001764 (2002).
8. Rabbani, Q., Milsap, G. & Crone, N. E. The Potential for a Speech Brain–Computer Interface Using Chronic Electrocorticography. *Neurotherapeutics* vol. 16 144–165 Preprint at https://doi.org/10.1007/s13311-018-00692-2 (2019).
9. Stavisky, S. D. *et al.* Neural ensemble dynamics in dorsal motor cortex during speech in people with paralysis. *Elife* **8**, (2019).
10. Wandelt, S. K. *et al.* Decoding grasp and speech signals from the cortical grasp circuit in a tetraplegic human. *Neuron* **110**, 1777-1787.e3 (2022).
11. Guenther, F. H. *et al.* A wireless brain-machine interface for real-time speech synthesis. *PLoS One* **4**, (2009).
12. Moses, D. A. *et al.* Neuroprosthesis for Decoding Speech in a Paralyzed Person with Anarthria. *New England Journal of Medicine* **385**, 217–227 (2021).
13. Brumberg, J. S., Wright, E. J., Andreasen, D. S., Guenther, F. H. & Kennedy, P. R. Classification of intended phoneme production from chronic intracortical microelectrode recordings in speech-motor cortex. *Front Neurosci* 1–12 (2011) doi:10.3389/fnins.2011.00065.
14. Vansteensel, M. J. *et al.* Fully Implanted Brain–Computer Interface in a Locked-In Patient with ALS. *New England Journal of Medicine* **375**, 2060–2066 (2016).
15. Anumanchipalli, G. K., Chartier, J. & Chang, E. F. Speech synthesis from neural decoding of spoken sentences. *Nature* **568**, 493–498 (2019).
16. Rybar, M. & Daly, I. Neural decoding of semantic concepts: A systematic literature review. *Journal of Neural Engineering* vol. 19 Preprint at https://doi.org/10.1088/1741-2552/ac619a (2022).

17. Herff, C., Krusienski, D. J. & Kubben, P. The Potential of Stereotactic-EEG for Brain-Computer Interfaces: Current Progress and Future Directions. *Frontiers in Neuroscience* vol. 14 Preprint at https://doi.org/10.3389/fnins.2020.00123 (2020).
18. Devlin, J., Chang, M.-W., Lee, K., Google, K. T. & Language, A. I. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. https://github.com/tensorflow/tensor2tensor.
19. Angrick, M. *et al.* Speech synthesis from ECoG using densely connected 3D convolutional neural networks. *J Neural Eng* **16**, (2019).
20. Kohler, J. *et al.* Synthesizing Speech from Intracranial Depth Electrodes using an Encoder-Decoder Framework. (2021) doi:10.51628/001c.57524.
21. Sun, P., Anumanchipalli, G. K. & Chang, E. F. Brain2Char: A deep architecture for decoding text from brain recordings. *J Neural Eng* **17**, (2020).
22. Wilson, G. H. *et al.* Decoding spoken English from intracortical electrode arrays in dorsal precentral gyrus. *J Neural Eng* **17**, (2020).
23. Miller, K. J. A library of human electrocorticographic data and analyses. *Nat Hum Behav* **3**, 1225–1235 (2019).
24. Crone, N. E. *et al. Functional mapping of human sensorimotor cortex with electrocorticographic spectral analysis I. Alpha and beta event-related desynchronization*. *Brain* vol. 121 (1998).
25. Iida, K. & Otsubo, H. Stereoelectroencephalography: Indication and efficacy. *Neurologia Medico-Chirurgica* vol. 57 375–385 Preprint at https://doi.org/10.2176/nmc.ra.2017-0008 (2017).
26. Lebel, A. *et al.* A natural language fMRI dataset for voxelwise encoding models. doi:10.1101/2022.09.22.509104.
27. Huth, A. G., Heer, W. A. De, Griffiths, T. L., Theunissen, F. E. & Jack, L. Natural speech reveals the semantic maps that tile human cerebral cortex. *Nature* **532**, 453–458 (2016).
28. Wittgenstein, L. *Philosophical investigations.* (Basil Blackwell, 1968).
29. Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. & Harshman, R. *Indexing by Latent Semantic Analysis*.
30. Baroni, M., Dinu, G. & Kruszewski, G. *Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors*. http://ronan.collobert.com/senna/.
31. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. Distributed Representations of Words and Phrases and their Compositionality. (2013).
32. Pennington, J., Socher, R. & Manning, C. D. *GloVe: Global Vectors for Word Representation*. http://nlp.
33. Peters, M. E. *et al.* Deep contextualized word representations. (2018).
34. Vaswani, A. *et al.* Attention Is All You Need. (2017).
35. Luong, M.-T., Pham, H. & Manning, C. D. Effective Approaches to Attention-based Neural Machine Translation. (2015).
36. Openai, A. R., Openai, K. N., Openai, T. S. & Openai, I. S. *Improving Language Understanding by Generative Pre-Training*. https://gluebenchmark.com/leaderboard.

37. Na, Y., Choi, I., Jang, D. P., Kang, J. K. & Woo, J. Semantic-hierarchical model improves classification of spoken-word evoked electrocorticography. *J Neurosci Methods* **311**, 253–258 (2019).
38. Goldstein, A. *et al.* Shared computational principles for language processing in humans and deep language models. *Nat Neurosci* **25**, 369–380 (2022).
39. Tang, J., Lebel, A., Jain, S. & Huth, A. G. Semantic reconstruction of continuous language from non-invasive brain recordings. doi:10.1101/2022.09.29.509744.
40. https://www.neurobs.com/menu_presentation/menu_features/features_overview.
41. Wehbe, L. Simultaneously Uncovering the Patterns of Brain Regions Involved in Different Story Reading Subprocesses. *PLoS One* **11**, (2016).
42. https://www.python.org.
43. https://www.anaconda.com/products/distribution.
44. https://wonambi-python.github.io.
45. Bigdely-Shamlo, N., Mullen, T., Kothe, C., Su, K. M. & Robbins, K. A. The PREP pipeline: Standardized preprocessing for large-scale EEG analysis. *Front Neuroinform* **9**, 1–19 (2015).
46. Kiymik, M. K., Güler, I., Dizibüyük, A. & Akin, M. Comparison of STFT and wavelet transform methods in determining epileptic seizure activity in EEG signals for real-time application. *Comput Biol Med* **35**, 603–616 (2005).
47. Slepian, D. Prolate Spheroidal Wave Functions, Fourier Analysis, and Uncertainty-V: The Discrete Case. *Bell System Technical Journal* **57**, 1371–1430 (1978).
48. https://azure.microsoft.com/en-us/products/cognitive-services/.
49. Boersma, P., and Weenink, D. (2016). Praat: Doing phonetics by computer[Computer program], Version 6.0. 14.
50. de Vries, W. *et al.* BERTje: A Dutch BERT Model. (2019).
51. Liu, Q., Kusner, M. J. & Blunsom, P. A Survey on Contextual Embeddings. (2020).
52. http://jalammar.github.io/illustrated-bert/.
53. https://huggingface.co/docs/transformers/index.
54. https://www.nltk.org.
55. https://scikit-learn.org/stable/.
56. Berezutskaya, J., Baratin, C., Freudenburg, Z. v. & Ramsey, N. F. High-density intracranial recordings reveal a distinct site in anterior dorsal precentral cortex that tracks perceived speech. *Hum Brain Mapp* **41**, 4587–4609 (2020).
57. Kubanek, J., Brunner, P., Gunduz, A., Poeppel, D. & Schalk, G. The Tracking of Speech Envelope in the Human Cortex. *PLoS One* **8**, (2013).
58. Abrams, D. A., Nicol, T., Zecker, S. & Kraus, N. Abnormal cortical processing of the syllable rate of speech in poor readers. *Journal of Neuroscience* **29**, 7686–7693 (2009).
59. Abrams, D. A., Nicol, T., Zecker, S. & Kraus, N. Right-hemisphere auditory cortex is dominant for coding syllable patterns in speech. *Journal of Neuroscience* **28**, 3958–3965 (2008).
60. Chi T, Ru P, Shamma SA. Multiresolution spectrotemporal analysis of complex sounds. J Acoust Soc Am. 2005 Aug;118(2):887-906. doi: 10.1121/1.1945807. PMID: 16158645.

61. Efron, Bradley. & Tibshirani, Robert. *An introduction to the bootstrap*. (Chapman & Hall, 1994).
62. W, E. I. Correlation Analysis: The Bootstrap Approach. *Int J Sci Eng Res* **4**, (2013).
63. Lecun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* vol. 521 436–444 Preprint at https://doi.org/10.1038/nature14539 (2015).
64. Sit, M. *et al.* A comprehensive review of deep learning applications in hydrology and water resources. *Water Science and Technology* **82**, 2635–2670 (2020).
65. https://pytorch.org.
66. Glaser, J. I. *et al. Machine learning for neural decoding*.