Deep Semantic Decoding: Reconstructing Semantic information from sEEG Data in Epilepsy Patients

Student: Emiel Eliens

Student ID: 7683634

Project Supervisor: Prof. Nick Ramsey

Daily Supervisor 1: Dr. Julia Berezutskaya

Daily Supervisor 2: Dr. Zachary Freudenburg

Second Supervisor: Dr. Samson Chota

Department of Information and Computing Sciences
Utrecht University
Artificial Intelligence
February 2024

Abstract

This thesis, developed in collaboration with Nick Ramsey's BCI lab at UMC Utrecht, investigated the feasibility of decoding semantic representations of speech from Stereotactic EEG data, recorded from epilepsy patients during a natural speech production task. These representations were either encoded as static or dynamic word embeddings, generated through the use of the Word2Vec and BERT models respectively, or as automatically generated semantic categories, derived from either these embedding, or from WordNet, a large lexical database. Furthermore, auto encoders were utilized in order to reduce the dimensionality of the word embeddings, whilst keeping important semantic information intact, which in turn reduced the complexity of decoding such embeddings. Additionally, auto encoders were applied to the sEEG data, for the purposes of de-noising, automatic feature selection, and dimensionality reduction. A preliminary motor decoding task, in the form of syllable classification was also included, to determine whether the sEEG data held any decoding potential in the first pace, as this task demonstrated a more salient relationship with the sEEG data. Lastly, data from 2 individual subjects was combined, after having been compressed by the auto encoders, in order to investigate whether limitations regarding the sparse distribution of sEEG electrode placements could be mitigated, through the incorporation of more data. The results of this study have demonstrated that auto encoders could successfully compress both the brain data and the semantic vectors, whilst keeping important information intact. Furthermore, the results indicated that while motor information could be decoded to a certain extent, semantic information could not be decoded from the available sEEG data. This was likely caused by a combination of the sEEG data's inability to sufficiently encapsulate semantic processing, given its sparse and distributed nature, as well as a lack of clear separability between the different semantic representations. Despite these findings with respect to semantic decoding, the efficacy of the auto encoders could hold great potential for future semantic speech decoding studies. Future work should focus on generating more separable semantic representations, incorporating data from more subjects, and on designing tools that can properly quantify the relationship between different electrode activations and semantic processing.

Acknowledgements

I would like to express my sincere gratitude towards Dr. Julia Berezutskaya, Dr. Samson Chota, Dr. Zachary Freudenburg, and Professor Nick Ramsey for allowing me to work on this project and for supporting me during my time at UMC Utrecht.

I would like to especially thank Dr. Julia Berezutskaya for helping me during the early stages of this project and for the knowledge she imparted on me early on, as my experience in neuroscience was limited at best, when i first started working on this project. Furthermore, I would also like to give special thanks to Dr. Zachary Freudenburg for his extended daily support during Dr. Julia's maternity leave. Without his guidance, I would not have been able to finish this project by myself, so thank you for allowing me to move forward with this project during uncertain times.

Contents

List of Figures

List of Tables

1 Introduction

Communication, one of humanity's fundamental characteristics, plays a crucial role in social interactions, facilitating expressions of thoughts, emotions, and ideas. It fosters relationships, knowledge transfer, empathy, and community. Losing this ability to express oneself can therefore have a devastating impact not only on one's professional life, but also on one's mental state. This is especially true in cases where a healthy mind is locked inside a body that can no longer convey the notions, ideas, and feelings that are still present within, as is the case for people with locked-in syndrome. This condition, which can result from a brain stem stroke, trauma or a neurodegenerative disease, is characterized by severe whole-body paralysis, with varying degrees of motor control loss, depending on the root cause and the extent of the neural damage [15]. The loss of motor control can either be progressive, when caused by a neuro-degenerative disease such as amyotrophic lateral sclerosis (ALS), or remain relatively stable, as is the case with brain stem lesions or strokes [87] [40]. The severity of motor control decline can range from loss of fine motor control during the early stages of ALS, to losing all forms of motor control except for eye movement and even complete loss of motor function, resembling patients in a vegetative state, in certain ALS and brain stem lesion cases [87] [40]. People presenting with locked-in syndrome can still think, reason, and feel, however, and may make use of computer-based communication technologies, if they are still able to at least control eye movement [87]. Such forms of communication are limited, however, as they are relatively slow, do not form a natural way of communication, and can only be used by patients who are still able to control eye movement [87] [75].

To enable communication in cases of complete motor control loss and to create a more natural form of communication, patients with locked-in syndrome can make use of so-called Brain-computer Interfaces (BCI). A BCI's general aim is to utilize its user's neural activity in order to restore or augment some of their capabilities, which in this case would refer to restoring the capability to communicate [96]. A BCI for the purpose of restoring natural communication would thus produce either spoken or written forms of speech, words, or other linguistic information relevant for the purpose of communication and would derive such information from the user's brain activity.

Recent investigations into the potential for such systems have demonstrated that BCI systems could reliably be used to assist individuals with speech impairments [86] [92] [28] [59] [8]. However, most of these methods have focused on decoding speech directly, and are characterized by slow communication speeds stemming from their serialized manner of decoding [75]. Therefore, recent work has also gravitated towards focusing on specific representations of speech. These representations of speech can generally be categorized as belonging to the semantic, auditory or articulatory pathways of speech, each focusing on different aspects of speech production [72].

Most research pertaining to the potential use of BCI technology for speech decoding has been focused on the auditory and articulatory pathways, with many approaches placing a major emphasis on the motor cortex and other more localized brain areas [29] [86] [98]. However, insights gathered from neurocomputational models of semantic cognition have shown that the semantics of speech are represented in a much more distributed manner in the brain, meaning that locally recording brain activity might not adequately represent the manner in which semantic concepts are processed [23] [68]. Such semantic information could hold potential for BCI technology as it could allow for the decoding of isolated words in a faster and less ambiguous manner than current BCI technologies [75]. This, in turn, would warrant the need to explore BCI technologies capable of spanning multiple brain areas, in order to fully encapsulate the neural processing behind semantic representations of words. Doing so may not only increase the efficacy of future BCI systems but may also increase our understanding of how the concepts behind speech arise from brain functioning and give us more insight into the neural mechanisms underlying the comprehension of semantic concepts.

In terms of modalities for recording semantic processing, most studies have focused on non-invasive neural recording methods, such as functional magnetic resonance imaging (fMRI), magnetoencephalography (MEG), functional near-infrared spectroscopy (fNIRS), and electroencephalography (EEG), in order to gather relevant brain data [23] [76] [47] [95] [68] [90]. Methods like fMRI, however, are characterized by slow temporal recording capabilities, which limits their practical applications for semantic decoding[90] [75] [26]. More invasive methods, such as electrocorticography (ECoG) and stereotactic EEG (sEEG) recordings, are available in cases where human subjects either undergo clinical treatment for epilepsy or participate in a clinical BCI trial [96]. Epilepsy patients temporarily implanted with ECoG or sEEG present a unique opportunity in which more invasive methods can ethically be applied in order to gain much more detailed and accurate brain data, when compared to non-invasive methods, especially in terms of temporal recording capabilities. In particular, the distributed nature of sEEG methods happens to lend itself particularly well to the task of decoding semantic features from brain data, due to the similarly distributed manner in which semantic concepts are represented in the brain [23] [68] [32]. sEEG's sparsely distributed nature, on the other hand, could inhibit adequate coverage of brain areas related to semantic processing [32].Therefore, despite uncertainty regarding the extent to which semantic processes are shared across individuals, combining sEEG data from multiple subjects could potentially mitigate its sparse sampling limitations [23].

Furthermore, because most research into speech decoding has been focused on these auditory and articulatory representations of speech, most speech decoding tasks revolve around continuous next word prediction, or prediction aided by statistical language models, in which neural activity plays a more cooperative role in word prediction as it is coupled with a sequence of context words which are used to select probable words based on a language model[88] [90] [31]. The task of predicting words or any sort of semantic information without context is sparsely documented in the speech decoding community with only a few documented cases focusing on semantic decoding specifically for BCI purposes [76] [24] [60] [93]. This lack of documentation, therefore, warrants a closer look into the feasibility of such a task. Furthermore, because words can have many interpretations and are often context dependent and thus highly ambiguous when isolated, speech decoding systems should aim to decode brain data in a more holistic manner, when compared to approaches solely focused on the articulatory and acoustic aspects of speech production [68]

A simple way of representing semantic information involves using semantic categories, which separate words based on their semantic meaning. Such categories have been successfully used in previous works and are often manually generated [76] [60] [93].They can also be generated automatically, based on large semantic databases, such as the WordNet database, which has been used for tasks such as information retrieval, text categorization, semantic feature selection, and word sense disambiguation [57] [85] [13]. However, using semantic categories limits the granularity of the semantic information that can be decoded from brain data. One way of capturing word semantics, whilst keeping word differentiability intact, is by representing words in semantic embedding spaces, generated from word co-occurrences in natural speech, which capture word similarity through the positions of the words in that space [38]. These models, along with the use of deep learning in general, have been met with increasing interest in the field of speech decoding, because of their ability to represent semantic information mathematically [72] [79] [80] [27]. Such embeddings may be static or contextual in how they represent words semantically. Static word embedding models, such as Word2Vec learn one fixed embedding for each word, whereas dynamic models, such as the BERT model developed by google, can be different for the same word in different contexts and are often more complex as a result [38] [18] [56]. The increased complexity of dynamic models makes them far more accurate in complex NLP tasks such as word sense disambiguation and could therefore allow for more intricate semantic encoding [18]. Conversely, the simplicity of Word2Vec may increase the feasibility of successful semantic decoding, and such models have been shown to work better in isolation due to their static nature [68]. Therefore, both models ought to be tested in order to see which model is better suited for semantic decoding.

One major drawback of directly decoding word embeddings from brain data,is the high dimensionality of such embeddings, which makes reconstructing such vectors a complex task. Simplifying such vectors should therefore constitute a high priority task, which can be achieved through the use of auto encoders, a specific type of neural network that is often used for dimensionality reduction purposes. These models have been shown to outperform linear techniques like PCA in tasks such as image compression, and have been successfully applied on word embeddings [48] [2] [51] [94]. It could therefore be interesting to see whether applying auto encoders could potentially enhance the feasibility of semantic vector decoding. These models have also successfully been used in order to denoise and compress the brain data used for decoding as well, which may improve decoding capabilities [12]. Decoding these complex semantic representations, in turn, could be achieved through the use of temporal neural network architectures, such as recurrent neural networks (RNN), gated recurrent units (GRU), temporal convolutional neural networks, and Long short-term memory networks, as such models have been successfully applied to sEEG data in previous works [30] [45].

Therefore, the purpose of this study was to investigate the potential of utilizing deep learning approaches in order to decode semantic representations from sEEG data and aimed to answer the following research question: *Is it possible to decode semantic representations of speech, encoded as either word embeddings or semantic categories, using sEEG during a natural speech production task?*

Towards that goal, semantically relevant words, pronounced by two subjects during a Dutch natural language reading task, were represented either as semantic vectors, generated through both static and dynamic embedding models, like Word2Vec and BERT, or as semantic categories, which were automatically generated from these models as well as from WordNet. The decoding models consisted of the temporal neural network architectures, mentioned before. Furthermore, in order to make the decoding task more computationally feasible, both the sEEG data and semantic vectors underwent dimensionality reduction through the use of auto encoders. Additionally, in order to assess the usefulness of the sEEG data, a preliminary motor decoding task, in the form of syllable quantity classification, was performed. Lastly, the sEEG data from both subjects was combined, in order to investigate whether an increase in available electrodes could mitigate the sparsity of the sEEG electrode coverage and whether this could positively modulate decoding performance.

As a concluding remark, it is important to note that utilizing sEEG data to decode isolated semantic representations in the form of word embeddings obtained from natural speech represents a novel task, which has not been publicly studied before. Therefore, this study could be considered a feasibility study, which could hold major implications for semantic decoding in general.

2 Related Work

This section provides an overview of the relevant background with regards to speech decoding. Towards that end, an overview of the different brain recording methods, used in general speech decoding, is provided first, after which the different representations of speech are introduced. The following section, on the other hand, discusses work related to how specific semantic representations of speech can both be encoded and decoded, and discusses how such work may be applied, when aiming to decode semantic representations from sEEG data.

2.1 Brain signals for speech decoding

Generally, methods to record brain data for the purpose of speech decoding can be categorized into invasive and non-invasive methods, with invasive methods only available in epilepsy patients or participants of BCI clinical trials.

2.1.1 Non-invasive methods

Non-invasive methods, based on the detection of metabolic signals for brain activity, include functional magnetic resonance imaging (fMRI) and functional near-infrared spectroscopy (fNIRS), which can reach the majority of the brain surface with good spatial coverage and resolution [75] [26]. An added benefit of using non-invasive methods, is the fact that patients require no surgical implants in order for these methods to work, which in turn lowers the risk of complications and eliminates some of the more physical entry barriers towards introducing BCIs to the larger population. However, fMRI and fNIRS represent an indirect measure of neural activity by recording BOLD changes, leading to a delay in recordings and overall low temporal resolution of the signal, which combined with the high number of words per seconds used in natural language, means that each brain image can be affected by a multitude of words [90] [75] [26]. This, in turn causes non-invasive models to lack the temporal resolution necessary for capturing brain data pertaining to real time speech synthesis [83].

Despite those limitations, recent work on non-invasive fMRI decoders has still demonstrated the feasibility of generating decoders capable of generating continuous intelligible word sequences that recover the meaning of perceived and imagined speech [90]. Although promising, the use of fMRI is still limited in the sense that decoding speech using fMRI was unable to capture individual semantic representations as word sequence generation relied on auto-regressive priors in order to generate candidate sequences, which highlights fMRI's inability to predict isolated semantic representations or individual words [90]. furthermore, cases in which fMRI could predict isolated words exist, but relied on the careful selection of words, which were temporally separated to a sufficient extent, which highlights fMRI is unsuitable for decoding natural speech, in which such temporal separations are virtually non-existent [68]. FNIRS has also successfully been utilized to differentiate between imagined semantic concepts belonging to two separate semantic categories [76]. The need to utilize long hemodynamic response measurement intervals (several seconds), however, highlights the lack of temporal resolution that is inherent to both FNIRS and fMRI [76]. Therefore, these slower but more global imaging methods are more often used in order to gather insights into the inner workings and relevant structures involved in speech production and comprehension. [47] [95] [49] [49] [17].

In contrast, electro-physiological neuroimaging methods, such as electroencephalography (EEG) and magnetoencephalography (MEG), offer far better temporal resolution in terms of their recording capabilities, when compared to fMRI and FNIRS [75] [16]. EEG, for instance, has successfully been used to model auditory speech processing [58]. Furthermore, EEG recordings have been used in order to enhance classification performance on common NLP tasks, such as sentiment classification and relation detection, although these methods were contingent on combining EEG recordings with independent semantic features which had no neural basis [34]. Unfortunately, EEG is not without its drawbacks. Whereas it has excellent temporal resolution, in contrast to fMRI, it severely lacks in the area of spatial resolution and has poor signal quality, caused by the fact that the electrodes are not in direct contact with the cortex of the brain [75] [23]. MEG, on the other hand suffers less from signal distortion and has a higher spatial resolution than EEG, and there are multiple studies describing and testing its viability for the task of both imagined and attempted speech decoding [15] [14] [16]. However, MEG still has worse spatial resolution, a lower signal-to-noise ratio, and motion artifacts, when compared to the more invasive methods that are available in cases where human subjects undergo clinical treatment for epilepsy or participate in a clinical BCI trial [96].

2.1.2 Invasive methods

Invasive brain recording methods represent the most commonly used brain recording techniques for developing BCI systems and notably include electrocorticography (ECoG) and stereotactic EEG (sEEG), which require patients to receive surgical implants in order for their brain data to be recorded. While these methods inherently carry more risk in terms of surgical complications, they offer far superior spatial and temporal resolution in return and do not suffer from poor signal quality as is the case with most non-invasive methods [96]. ECoG entails placing sheets of electrodes directly onto the brain, just below the dura, which allows it to directly measure the synaptic field potentials of the underlying neurons in a manner inaccessible to fMRI, and without the signal quality issues of surface EEG [72] [9]. Its invasive nature, on

the other hand, has caused ECoG to most often be applied in cases of drug resistant epilepsy, in which patients receive these implants strictly for medical purposes [72] [52]. Despite this practical drawback, numerous studies have shown the potential of using ECoG in speech decoding, with many approaches yielding favorable results [72] [3] [27] [93] [62]. A more fundamental issue pertaining to the use of ECoG relates to its inability to penetrate into the deeper structures of the brain, such as the hippocampus, insula, Herschl's gyrus and basal ganglia [32]. Furthermore, ECoG often fails to have broad coverage over different brain areas, since the electrode sheets are most often placed over specific and localized regions of the brain based on the affected area of the patient's brain [32].

The other notable invasive method, stereotactic EEG, entails the employment of penetrating depth electrodes that are directly implanted through burr holes in the skull and are placed using stereotactic guidance [32]. sEEG is most often used in conjunction with ECoG but the general consensus is that sEEG by itself carries fewer risks of complications than ECoG, which makes it a more desirable target when looking for more permanent implants [32] [36]. Furthermore, while sEEG is characterized by a much sparser cortical sampling rate than ECoG, its potential for employing a multitude of distinct recording sites in different areas of the brain potentially allows it to better investigate semantic representations as these are believed to be widely distributed throughout the brain [32] [72] [68]. Lastly, sEEG's stereotactic guidance approach allows it to reach deeper structures of the brain, which allows for analyzing previously unreachable brain areas such as the hippocampus, insula, Herschl's gyrus and basal ganglia [32]. There are limited studies available which have focused on speech synthesis and speech activity recognition, as well as semantic processing, utilizing sEEG data. [45] [1] [61] [84]. These existing studies demonstrated promising results however, directly attributed to sEEG's ability to more accurately decode memory related processes, which are linked to semantic representations, its ability to reach multiple brain areas when placed in larger numbers, and its potential to access deeper brain structures [32] [45][61] [84] [1]. These recording advantages, combined with sEEG's relative low risk profile, as well as the lack of attention it has received when compared to the more established ECoG method, make it a suitable target for exploratory research with regards to semantic decoding.

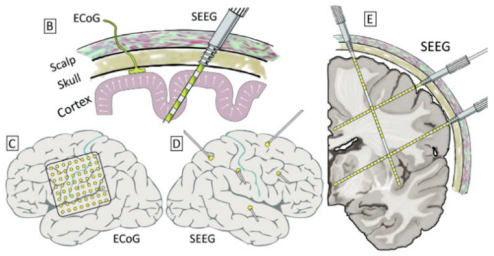


Figure 1: Implantation difference between ECoG and sEEG electrodes. ECoG grids are implanted under the skull whereas sEEG electrodes penetrate inside the cortex.

Figure 2: Implanted electrode shafts. sEEG requires only small, localized burr holes compared to the comparatively large craniotomies required for ECoG implants. [32].

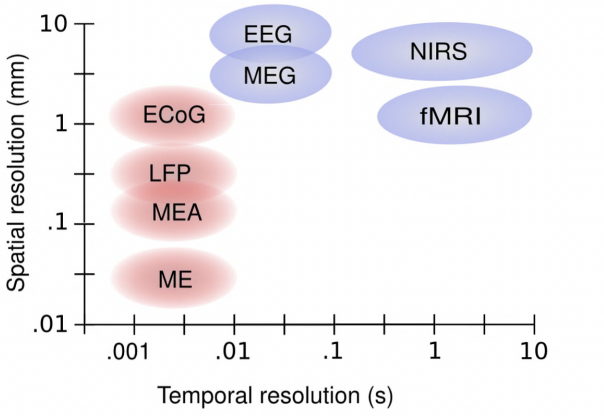


Figure 3: Schematic overview of the scale of spatial and temporal resolution of measurement methods used for BCI. Measurement methods are electroencephalography (EEG), magnetoencephalography (MEG), near-infrared spectroscopy (NIRS), functional magnetic resonance imaging (fMRI), electrocorticography (ECOG), local field potential (LFP) recordings, micro-electrode array (MEA) recordings, and microelectrode (ME) recordings. Non-invasive methods are shown in blue and invasive methods are shown in red. [91].

2.2 Representations of speech

Before the models for the purpose of semantic encoding and decoding can be discussed, a clear understanding of the desired output representations for such models is required. Generally, speech can be analyzed through three main representations, which are each involved in a different part of the speech production process and are termed the semantic, auditory and articulatory pathways [72].

Semantic representations are likely one of the earliest encountered representations in the process of speech production and are involved with the meaning of words and concepts [72] [75]. Therefore, these representations rely on both speech formation as well as memory, vision, and higher cognitive functioning related areas of the brain such as the parietal, frontal, temporal, and occipital lobes [72] [75]. This entails that semantic representations are highly distributed in the brain and that they likely cannot be fully represented by examining just one localized brain area. In computational terms, semantic information can be represented through semantic categories, which group semantically similar words together into clusters. More advanced methods focus on mapping the semantic information to a higher dimensional space, in which each dimension describes some feature. These higher dimensional spaces are termed embedding spaces in the realm of natural language processing and utilize the mathematical properties of vector spaces in order to represent semantic information [38]. More specifically, utilizing vector spaces allows computers to mathematically compare semantic information, because words are embedded in a contextualized vector space, in which words with similar meaning are physically closer to each other in that space [38]. This similarity is contextually derived in the sense that it originates from the notion that similar words occur in similar contexts, which is also called the distributional hypothesis [38] [55]. Furthermore, mathematical rules valid in vector spaces, such as addition and subtraction, make it possible to establish mathematical relationships between different semantic concepts [38]. More details on how these word embedding are generated can be found in the next section concerning encoding and decoding approaches. One drawback of relying on word embeddings to represent semantic information is the fact that precise word decoding becomes much more difficult as many words can have similar meanings. This can also be viewed as a strength for the case of semantic decoding, however, as semantic information is more concerned with conceptual information, rather than precise wording, where semantic vectors can serve to mitigate the ambiguous nature of words, when viewed in isolation [20]. This means that contextual embeddings could potentially be more informative than words themselves in certain cases, as they give computers more context pertaining to the actual meaning of embeddings than words usually can. Embedding spaces have successfully been used in several studies pertaining to neural speech decoding, in particular with regards to fMRI brain activation prediction for semantic categories as well as for contextualizing brain data such as EEG and iEEG recordings [65] [34]. They have also been used in conjunction with fMRI data in order to reconstruct semantic information directly, as is discussed later on [68]. Most BCI research, however, is geared towards decoding discrete words and studies that do use semantic vectors, generated through large language models, use them in continuous language decoding tasks in order to generate likely next word candidates [27] [90]. Semantic categories are more commonly used in BCI studies, as they can simplify the decodable semantic space, which allows for easier classification [62] [93]. Ways of generating semantic categories are discussed alongside semantic vector generation in the next section.

In terms of audition, speech can be represented in terms of its acoustic wave-forms and corresponding spectrotemporal features, which can further be separated into phonemes whose neural representations could be used for decoding in the middle stages of speech production [72]. such spectrotemporal features have been used in both ECoG and sEEG studies for the purpose of auditory speech feature reconstruction and have found that auditory representations of speech can in fact be reconstructed from such features [54] [3] [45]. Lastly, the articulatory pathway of speech can be represented as a combination of vocation and articulatory actions generated by the mouth, vocal tract and tongue and could be assessed through a combination of visual and attention correlates as well as neural features gathered through either invasive or non-invasive methods focusing mostly on the motor cortex [72] [15] [97]. As informative as the auditory and articulatory representations of speech may be, semantic representations more directly relate to the purpose of this current study, although it should be noted that a combination of multiple representation pathways could positively modulate general decoding capabilities. Such a task is currently outside the scope of this project however, which is why solely semantic representations are considered for decoding in this study.

2.3 Semantic encoding and decoding models

The last section focuses on specific ways to both encode and decode semantic information. Based on the previous sections, the types of semantic representation relevant for the purposes of this study are represented by semantic categories and word embeddings. Therefore, the first subsection discusses how such representations could be generated from natural speech. Afterwards, ways in which more complex semantic representations, as well as the brain data itself, can be simplified, whilst retaining important semantic information, are discussed. Subsequently, suitable decoding models for sEEG data are presented, after which the possible enhancement of sEEG data through the combination of data from multiple subjects is discussed.

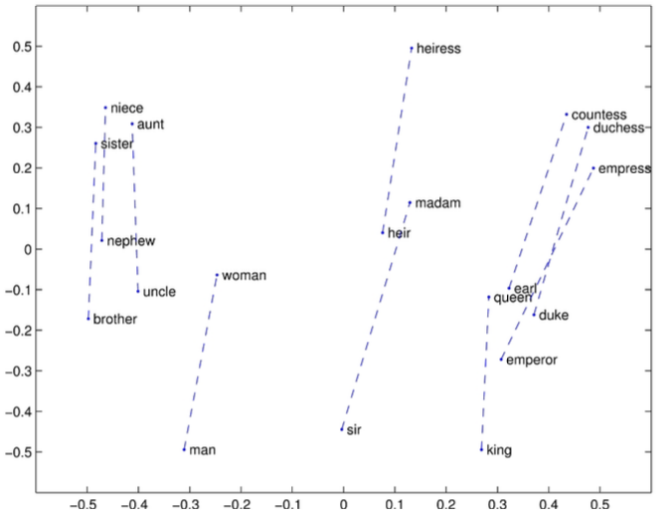


Figure 4: relational properties of embedding spaces [38]

2.3.1 Encoding semantic representations of speech

Encoding approaches, for the purposes of this study, consist of methods towards generating semantic semantic categories, which cluster target concepts based on their semantic characteristics, or word embeddings, the dense and short vectors used for representing semantic information [38]. Semantic clusters can be generated either manually, by focusing on salient semantic differences between different words, or automatically by using large language models or semantic databases. Manually generating semantic categories is particularly useful when the words one aims to decode can be separated into clearly defined clusters, which is why manual semantic category selection is often used when decoding a relatively small set of words, which have a relatively clear relationship with the class they belong to, with classes such as tools and animals [76] [60] [93]. However, such an approach becomes less advantageous when using words recorded during natural language tasks, which often do not display such clear distinctions, as such words are not selected to be categorically distinct from one another, meaning that derived categories need to be much more general in order to encapsulate all the words or the number of categories must be substantially increased, and manually annotating words with semantic categories is labour intensive work [35]. Therefore, automatically categorizing words based on semantic databases or embedding models can both decrease the labour intensiveness of such tasks as well as infer semantic categories without the need to define such categories beforehand. Clustering the words into semantic categories can be done either trough semantic databases, such as WordNet, or through the use of large language models, which are introduced later on. WordNet is a large lexical database which also contains semantic relationships between different words, and has mainly been used for tasks such as information retrieval, text categorization, semantic feature selection, and word sense disambiguation [57] [85] [13]. WordNet connects nouns, verbs, and adjectives hierarchically based on a class system, where parent classes are more general than their sub-classes [57]. This hierarchical structure builds knowledge from specific concepts to broader classes, such as people and dogs both falling under the entity superclass. This hierarchical structure could potentially be used in order to cluster different words based on how similar their positions are in this hierarchical structure and possible ways of achieving this goal are discussed in the methodology of this study. Lastly, while employing semantic categories proves highly effective in simplifying the complexity of speech decoding tasks, it is important to acknowledge that the broad categorization of words into semantic clusters imposes limitations on the specificity of information that a semantic decoder can decode. This can be mitigated by increasing the number of semantic categories, but this in turn increases decoding complexity, which is what semantic categories are supposed to mitigate in the first place. This is where word embeddings can offer additional value over semantic categories for the purpose of semantic decoding, as word embeddings fully encapsulate the meaning of each concept, without the need for categorization.

Whereas semantic categories can be generated manually, embeddings are exclusively generated through the use of large language models (LLMS), which are trained on large quantities of unlabeled text, also called a training corpus [38]. Generally, such models can either generate static or dynamic embeddings, where static methods learn one fixed embedding for each word in the vocabulary, whereas dynamic embeddings can be different for the same word in different contexts [38]. The two most well known methods for generating static embeddings are GloVe and Word2Vec, which work in slightly different ways but produce similar outputs [56] [67]. GloVe, short for global vectors, is based on capturing global corpus statistics and captures ratios of probabilities from word to word co-occurrence matrices, and serves as a hybrid between count based methods and methods like word2Vec, due to its linear substructure used for assessing vector similarity [38] [67]. Word2Vec, on the other hand, is a predictive based method, which uses logistic regression to distinguish between the context and non-context words of a given target word, and uses the learned weights of that classifier as the word embeddings [38] [56]. Both models have been applied to semantic representation encoding and as a way to generate semantic categories, although they are generally outperformed by dynamic embedding methods in cases where they are used for next word prediction [10] [90] [80] [68].

Dynamic embedding models, in contrast to their static counterparts, generate multiple embeddings for each word, based on the different contexts in which that word is used. One of the most well known dynamic embedding models is ELMO (embeddings from language models), which utilizes bidirectional

LSTMs, allowing it to consider context words both before and after the target word for encoding [69]. Elmo, however, is limited in the sense that it considers the context from both directions independently from one another, which limits its ability to capture certain contextual dependencies [18]. Furthermore, its training process is computationally expensive, its performance is very dependent on the quality of the pretraining tasks on which it was trained and it can still struggle with longer sequences because of the vanishing gradients problem inherent to almost all recurrent neural network architectures [38]. Therefore, more advanced models like BERT and GPT have taken over ELMO's prevalence in the Natural language processing scene.

Such models are based on the transformer architecture, a deep sequential learning approach which employs self attention layers [38] [18]. This allows transformers to directly extract and use information from arbitrarily large contexts, without needing to pass that information through intermediate recurrent connections, unlike Recurrent Neural Network based approaches, which can suffer from forgetting context that is further removed from the target input [38]. Transformers thus allow for creating a more sophisticated manner of representing how words contribute to the representation of longer inputs. Transformers can serve as the basis for powerful language models, which can be applied to embedding generation tasks, but the unidirectional processing of standard transformer encoders cannot take context information from the right of the target word into account, which inhibits their potential to fully account for the context they attribute to their generated embeddings [38]. Despite this limitation, Unidirectional transformer models, like GPT have still become wildly popular in the field of NLP and have been successfully used in semantic decoding studies as well [27] [90]. More advanced models, such as BERT, short for Bidirectional Encoder Representations from Transformers, overcome this limitation by adopting a bidirectional transformer model which allows self attention to span the entire input sequence rather than just the preceding context of a target word [38]. Models such as BERT have been utilized successfully in speech decoding, semantic labeling, and semantic processing studies, and generally outperform simpler methods in next word prediction tasks [10] [80] [90] [47].

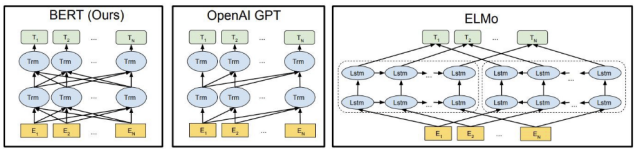


Figure 5: Different model architectures for contextual embeddings. BERT uses a bidirectional transformer. GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs [18].

It is important to note, however, that none of the studies focusing on word embedding encoding have actually used such semantic vectors for decoding isolated semantic representations directly. The semantic embeddings generated though these methods, rather always played a more supportive role, in the form of next word prediction candidate generation or by using them in order to show how neural processes can follow similar semantic trends as observed in such embeddings [27] [90] [47]. The closest attempt towards direct embedding decoding was conducted by Pereira and colleagues utilizing fMRI data, who selected isolated concepts from clusters derived from Glove and wWrd2Vec embeddings, and were able to show that the reconstructed semantic vectors were more similar to the true semantic vectors than they were to other words [68]. Such work, however, involved the use of fMRI, which limited the number of words that could be decoded, as word had to be sufficiently separated in terms of their temporal extent, due to fMRI's lacking temporal recording capabilities [68]. This also implies such an approach would hold no practical value for decoding purposes, as it would be incapable of keeping up with the fast pace of natural speech [26] [83]. Furthermore, the words used for decoding were sampled from clusters which were generated to be semantically different from one another, which also limits practicality with regards to handling natural speech [68]. In terms of the methods with practical applications towards semantic decoding, such as ECoG and sEEG, attempts towards direct semantic vector decoding have not been made to date.

This lack of research pertaining to decoding word embeddings directly is most likely caused by the complexity that is associated with such a task. For reference, BERT generates 768-dimensional embeddings, and even the smallest static embedding models, such as word2vec, generate vectors that are 100 dimensional [38] [18] [56]. Decoding such semantic representations directly, amounts to a task of vector reconstruction, which requires the generation of very high dimensional data, from an often limited number of available training samples in BCI studies. it could prove interesting to see whether clustering the embeddings discussed so far can result in semantic categories relevant for semantic decoding, as this constitutes a much simpler decoding task. However, as mentioned before, semantic categories limit the specificity of possible decoders. Hence, a key priority should be the reduction of the dimensional complexity in embedding models. This task is crucial for maintaining the distinguishability between various words, especially when the objective is to make decoding of semantic representations, in the form of word embeddings, feasible within the constraints of data availability prevalent in many speech decoding studies. The subsequent subsection addresses specific methods aimed at simplifying the complexity of semantic vectors.

2.3.2 Semantic vector and brain data dimensionality reduction methods

One of the main ways to reduce the embedding space's complexity is to make use of Principal Component analysis (PCA). PCA is a linear technique that works by transforming high-dimensional data into a lower-dimensional space through the identification and selection of principal components, which are eigenvectors representing directions of maximum variance in the original dataset. These principal components capture the essential information, allowing for dimensionality reduction while retaining the most significant features of the data. [63].

Recent research on auto encoders, an encoder-decoder architecture used for dimensionality reduction and reconstruction, has shown that such models are capable of demonstrating superior performance over PCA in tasks such as image reconstruction and that they can successfully be applied to reduce the dimensionality of word embeddings [48] [2] [51] [94]. This advantage is attributed to the non-linear nature of neural networks, the foundation of auto encoders, enabling them to learn complex patterns beyond the reach of linear techniques like PCA. Such architectures have also been utilized to effectively reduce the dimensionality of vocoder parameters in an ECoG and sEEG auditory speech reconstruction study [1].

Auto encoders are neural network architectures designed for unsupervised feature extraction and consist of an encoder and a decoder [22]. they learn to compress input data into a lower-dimensional latent representation using the encoder and then reconstruct the original data from this compressed representation, using the decoder. The training process encourages the network to capture meaningful features, which is encapsulated in the data after encoding. The reconstruction error can be calculated through commonly used loss functions such as Mean Squared Error.

It is important to emphasize the flexibility of both the encoder and decoder components, as they can be constructed with various architectures, such as simple fully connected layers, convolutional layers, or sequential layers. This adaptability allows them to accommodate inputs of diverse formats such as images, vectors or time series data [2] [48] [39] . Consequently, an auto encoder doesn't represent a rigid model but serves as a framework for organizing neural networks in different configurations, which can have different purposes. Furthermore, auto encoders may be hierarchical, in the sense that they can employ multiple layers in both their encoder and decoder, which allows them to learn important features of the input space at different levels of abstraction [50]. An auto encoder for the purpose of dimensionality reduction, with compression occurring through the encoder and reconstruction being taken care of by the decoder, can be observed in figure 6.

There also exists a probabilistically informed variant of the auto encoder architecture known as the variational auto

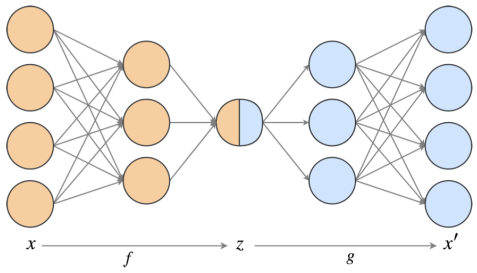


Figure 6: simple depiction of a 2-layer auto encoder. The encoder is in orange and the decoder in blue. The latent representation is the bi-colored node [22].

encoder (VAE). Unlike traditional auto encoders, VAEs are generative models with a non-deterministic latent space. They function by first projecting the input features into a Gaussian probability density distribution, from which samples are drawn in order to form the encoded latent space. Importantly, these samples are drawn by reparameterizing the sampling operation as a deterministic function,

$$z = \mu + \sigma * \varepsilon \tag{1}$$

where (μ) is the mean, (σ) is the standard deviation, and (ε) is a noise term drawn from the distribution [43]. This reparameterization trick ensures that the sampling operation is differentiable and allows for back-propagation with respect to the parameters of the latent encoding, concerning the mean and standard deviation [43]. The decoder then takes the sampled latent encoding and decodes it back into the input dimensionality. Furthermore, the loss function for Variational auto encoders include an additional regularization term, called the Kullback-Leibler (KL) divergence term, which forces the sampled latent space to be close to the predefined Gaussian distribution[22]. The formula for KL divergence can be observed in the equation below [43].

$$KL(P\|Q) = \sum_i P(i) \log\left(\frac{P(i)}{Q(i)}\right) \tag{2}$$

This equation measures the disparity between two probability distributions, P and Q. The KL divergence provides a quantification of the information loss incurred when Q is employed as an approximation for the true distribution P. By incorporating the KL divergence as a regularization term, the loss function forces the latent space to both capture important patterns from the input as well as to be close to the Gaussian distribution of the input. As a result, VAEs have the capability to generate diverse and meaningful samples while providing a continuous and robust latent representation [22]. Due to their stochastic nature, VAEs are most often employed for generative purposes, but they can also be used for dimensionality reduction, due to their robust representation of the latent space, which is especially beneficial in relatively small

datasets [53] [39]. A depiction of a simple variational auto encoder, for the purpose of dimensionality reduction, can be observed from figure 7.

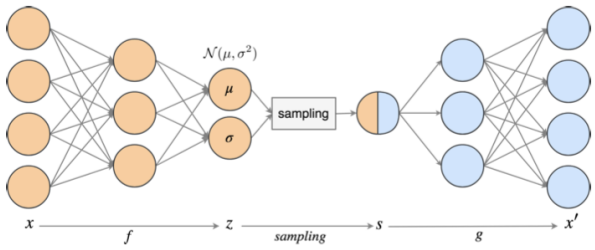


Figure 7: simple depiction of a 2-layer variational auto encoder. The encoder is in orange and the decoder in blue. The latent representation is the bi-colored node. note that the latent representation is sampled from a Gaussian density distribution, which is generated from the input data. [22].

In terms of use cases besides word embedding dimensionality reduction, as has currently been discussed, it is also important to note that auto encoders could hold great potential for reducing the complexity of the incoming brain signals used for decoding, especially when considering that this input data is equally sparse as the number of available semantic data, in speech decoding. Furthermore, auto encoders could potentially act as a way to denoise brain data which could make its semantically decodable features more salient. This potential is already demonstrated in work by Chikkankod and Longo who successfully utilized convolutional auto encoders for the tasks of dimensionaltiy reduction and artifact removal in EEG data [12].While sEEG data cannot benefit from spatial models like convolutional auto encoders due to its sparse and distributed nature, the flexibility of the auto encoder structure allows for the adoption of sequence-based or non-temporal fully connected architectures as well. Exploring whether such auto encoder architectures can similarly contribute to leveraging sEEG data represents an intriguing avenue for future research and is therefore discussed in more detail in the methodology section.

2.3.3 Semantic decoding models

As mentioned before, the sparsely distributed nature of sEEG limits the kind of deep learning architectures it can benefit from. This is true because spatial models, such as convolutional models are only really useful when the data is structured in a way that makes neighbouring channels carry related information. Because sEEG is so sparsely distributed, generating a spatial grid from its channels would generate a very sparse grid, which would need a lot of interpolated values to be densely populated. Because of this, deep learning models for decoding any sort of information from sEEG are mostly comprised of traditional sequence models, which do not take the spatial extent of the different channels into account. Such models present themselves in the forms of RNNs, GRUs, and LSTM models. The benefit of a simple RNN, when compared to traditional feedfoward neural networks, is their ability to capture temporal dependencies, since their hidden state can be influenced by earlier activation of the hidden state [38]. Although simple RNNs provide the right idea by focusing on more than just the current input, when determining their output, they have difficulties with capturing long range dependencies, because the weights in their hidden state have to be updated in order to both provide useful information regarding the current input as well as to carry forward information required for future time-steps [38]. They also need to back-propagate their prediction error signal through time, which involves repeated multiplications of the gradients, ultimately driving these gradients towards zero, thus losing the error signal as the signal propagates through the network [38]. This in turn inhibits learning long range context dependencies, and makes the information encoded in the hidden states of RNNs fairly local.

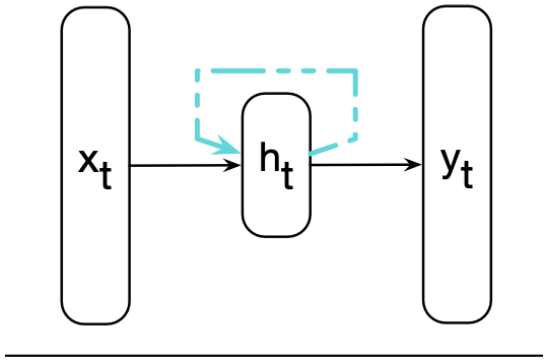


Figure 8: simple depiction of a recurrent unit, where the hidden unit contains a recurrent connection as part of its input, such that its activation is dependent on the previous activation of that same unit from a previous time-step. [38]

In order to overcome these issues, an extension of RNNs, called the LSTM, short for Long Short-Term Memory unit, can be applied. Such models tackle the long range dependency problem by removing information that is no longer needed from the context and by adding information that is likely needed later, through the use of explicit context layers and neural gates that control the flow of information in the network [38]. The three gates used are the forget, input, and output gates, and are used in each recurrent unit. In that sense, the hidden state of each unit represents the short term memory, whereas the recurrent unit, encompassing all the gates, represents the long term memory [38]. The forget gate deletes unnecessary context information from the context by computing a weighted sum of the previous hidden layer state as well as the current input and passes this sum through a sigmoid activation function, resulting in a number between 0 and 1 which is then multiplied with the context vector via

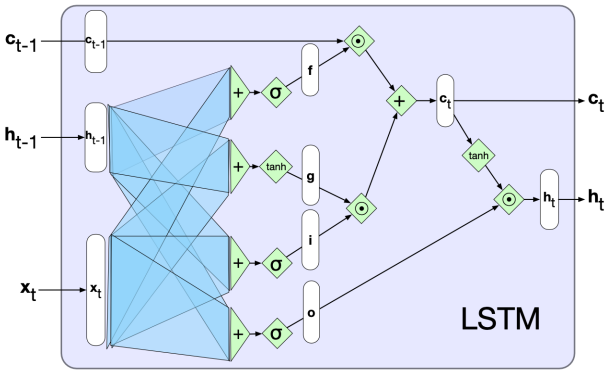


Figure 9: A single LSTM Unit, with current input xt, previous hidden state $ht-1$, previous context $ct-1$, hidden state ht and updated context ct [38]

dot product to remove unnecessary information. The forget gate is in essence a neural network of its own, which decides which pieces of the long-term memory (unit-state) should be forgotten, given the previous hidden state and the current data point in the sequence. Subsequently, the importance of the new information carried by the current input is quantified by adding the current input to the activation of the previous hidden state and passing the result through a Tanh activation function, resulting in a number between -1 and 1 [38]. Negative numbers allow for subtracting information from the cell state. The add gate is then used in order to determine which parts of the new input data should be remembered, acting as a filter, where outputs near zero indicate that the unit state should not be updated. The results of the Sigmoid and Tanh activations are then multiplied via dot product, and are added to the unit state, resulting in an update of the long-term memory of the network [38]. The input gate thus decides which information should be added to the unit state. Lastly, the output state decides the new hidden state, which in term decides the output of the entire unit for the current time-step. It does so by passing the current unit state through a Tanh function, after which the previous hidden state and current input are added and passed through a sigmoid activation neural network. The resulting vector acts as a filter, which is applied to the Thanh activated unit state in order to update the actual hidden state of the recurrent unit [38]. This complicated memory structure allows LSTMs to capture long range context dependencies by focusing on important context information and forgetting less relevant information, whereas the encapsulating nature of each unit allow for maintaining modularity, when combining LSTM architectures with other deep learning networks. Gated Recurrent Units, work in a similar manner as LSTMs but have fewer parameters and gates, which makes them faster to train but also less powerful [38].

It is important to note that more advanced versions of the aforementioned traditional sequence models do exist, such as bidirectional sequence models, sequence models with skip-connections, other wise called residual nets, and transformer models [38]. However, all of these models present drawbacks pertaining to increased complexity, which makes them unsuitable for most decoding tasks, given the constraints pertaining to data availability in most decoding studies. This increased complexity is caused by the increase in model parameters attributed to these more advanced models. Bidirectional models include a second hidden state for processing the sequence from right to left, which roughly doubles the model parameters for each layer [38]. Sequence models with skip-connections on the other hand, are only really useful when using deeper networks, and the networks used for decoding in most studies do not reach the depth at which having residual connections becomes advantageous, because there is simply not enough training data to utilize actually deep neural networks in the first place [38]. Lastly, transformer models, on which BERT is based, are some of the most complex sequence models to date and require far more data than is available for decoding [18]. This, combined with the fact that no pre-trained transformer models for the purposes of semantic decoding are available, makes their application infeasible with respect to semantic decoding.

The more straightforward traditional sequence models have been employed in certain sEEG studies. For instance, one study involving an sEEG subject employed an LSTM based model to automatically separate audio signals from mixed speakers, comparing each speaker with the user's neural data, and amplifying the speaker that best matched the neural data to assist the user [30]. Furthermore, Kohler and colleagues have used GRU based encoder-decoder networks, in combination with 1D-CNNs, which are introduced later on, in order to reconstruct audio from sEEG recordings [45]. As can be seen, the number of sEEG decoding studies is quite limited, which underlines the gap in research pertaining to the use of sEEG for any sort of decoding purposes. Therefore, besides focusing on methods that are purely relevant for sEEG, certain spatial methods could hold potential for sEEG as well.

For recording methods which incorporate a spatial structure, such as ECoG or EEG, deep learning approaches focusing on the spatial extent of the input data have also shown promising results. These approaches predominantly consist of deep convolutional neural networks, which in essence are basic neural networks with some additional prepossessing applied to them. This prepossessing is performed in the form of convolutional filters, which operate on a 1D, 2D, or 3D input signal and compress the signal based on element wise multiplication with the filter [66]. The paramaters such models aim to learn during training are the actual filter weights, which are shared between each convolutional layer. In doing so, convolutional neural networks can identify both spatial and hierarchical patterns in images and other forms of multi-dimensional input data, such as multi-channel EEG data [79]. The ability to analyze the spatial extent of the data has led CNN models to be particularly useful in tasks where the aim

is to decode multiple channels of brain signal data, where they perform admirably, when compared to more traditional decoding methods [78] [79].

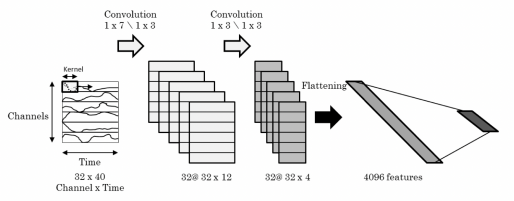


Figure 10: visualization of a sample CNN architecture for multi-channel EEG data. The dimensionality of the input data is reduced with every convolutional layer, in order to increase the CNN's receptive field with respect to the input data. After convolution, the resulting feature maps are flattened to a 1D vector, such that they may be fed into a fully connected layer [78].

More recently, studies have shown that adopting methods which incorporate both the spatial and temporal of the input data can significantly improve decoding performance. Such spatiotemporal models present themselves in the form of convolutional LSTMs. Such models can be beneficial when a multitude of sensors are employed to record brain activity. Convolutional LSTMs have successfully been used in such instances in by integrating the spatial placement of sensors in order to improve decoding performance. One study transformed EEG data from multiple sensors into its time frequency distribution through wavelet and Fourier transforms, which allowed it to be fed into a CNN, whilst benefiting from the time dependencies introduced by the inclusion of an LSTM memory component, resulting in improved decoding performance for binary motor cortex activation [89]. In the realm of speech decoding, Convolutional LSTMs have been used in order to predict whether EEG data would match speech envelopes [58]. Although predictions like these were binary, in the sense that they would only predict whether a certain EEG and speech envelope pair were a match, their performance in these tasks was still impressive. There have also been efforts towards multi-class imaginary speech recognition systems through the use of Convolutional LSTMs, with one study achieving relatively high prediction accuracy for a small set of utterances, phonemes and words [74]. While most of the Convolutional LSTM approaches are hierarchical in nature, meaning that the results of convolutional layers are passed onto the LSTM layers, there have also been successful attempts focusing on disjointed architectures, which first separately learn the spatial and temporal features of the data [77]. These representations are concatenated afterwards and fed into a fully connected neural network, which can then exploit both types of features simultaneously, as depicted in figure 11. All in all, Convolutional LSTMs as well as traditional CNNs have been shown to hold a lot of potential in the realm of neural decoding, but their application to sEEg data remains limited, due to the sparsely distributed nature of sEEG electrode implantation.

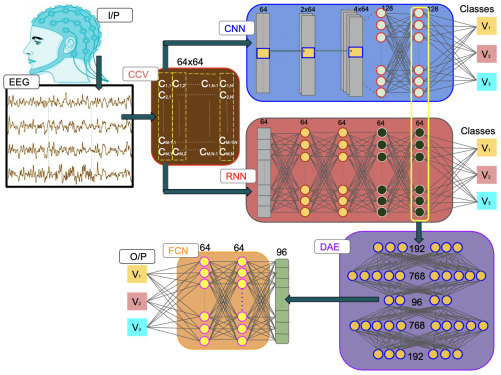


Figure 11: visualization of a sample disjointed CNN-LSTM architecture, in which temporal and spatial features are first learned separately, and concatenated afterwards [77].

However, not all spatial models require a dense and structured grid of data in order to be successfully utilized. Despite earlier having mentioned that spatial models are unsuitable in combination with sEEG, given the sparsely distributed nature of the sEEG data, convolutional models can still be successfully applied, if one shifts their attention from spatial properties in the data towards temporal properties. This can be achieved through the use of 1 dimensional convolutional neural networks, whose filters only span the temporal extent of the data and not their spatial extent, as can be observed in the left-hand image in figure 12 . In doing so, an alternative to traditional sequence models can be created, in which the model's convolutional filters exclusively learn local temporal features. While LSTMs and GRUs are theoretically stronger models and can in theory learn to model arbitrarily long dependencies between their inputs, the approach of applying convolutional filters at various positions in each channel sequence could potentially lead to learning interesting local patterns in the data [37] [73]. Such models have been successfully applied with sEEG data in work by Kohler and colleagues, as mentioned earlier, in which 1D-CNNs were used to down-sample the sEEG data into a lower temporal format whilst extracting relevant temporal features. [45]. It could, therefore, be interesting to see whether sEEG semantic decoding could benefit from such an alternative temporal model, inspired by spatial models.

Furthermore, although spatial temporal models hold no direct potential for decoding semantic information from sEEG signals, they could be incorporated if a multitude of sEEG electrodes were to be placed in a structured 3D grid. The number of electrodes required for such a spatial representation would make sEEG more invasive than ECoG, but only when all electrodes are placed within the same patient. This issue

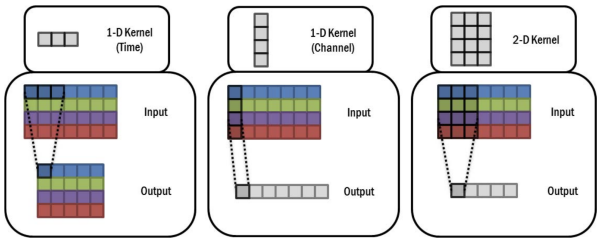


Figure 12: three types of convolution for the purpose of multichannel time series data (from left to right): convolution in time, convolution in channel, and convolution in both channel and time. colored squares represent independent time series, whereas gray values indicate mixed channel values [78]

could be mitigated if this structured grid was generated from electrodes placed among several subjects, which is why the following subsection delves into how combining sEEG data from multiple patients could take place.

2.3.4 Cross subject sEEG data combination

One of the difficulties in building neural decoders from brain data in general, pertains to the inter-patient differences in electrode placement and frequency, which makes direct comparisons between different users more difficult and inhibits the development of more generalized decoders, which incorporate data from multiple patients [70]. Work by Peterson and colleagues demonstrated that by mapping ECoG data from multiple patients onto common regions of interest, a more robust model for decoding motor activations could be created [70]. While their approach was quite complex in the sense that it did not simply map local electrode activations to a structured grid, but rather to common brain regions, they discuss that it would have been possible to do so. sEEG might benefit from a similar approach that aims to map these brain regions onto a structured grid. However, it remains uncertain whether integrating combined sEEG data would enhance semantic decoding performance. The study by Peterson and colleagues focused on motor decoding, and the debate on the shared representation of semantic processes is ongoing. Neuroimaging studies suggest some independent, contiguous, and similarly localized elements in neuro-semantic representation [23]. At the same time however, neurocomputational models of semantic cognition exhibit conjoint, anatomically dispersed, heterogeneous, and potentially differently localized representations across individuals [23]. Despite this uncertainty, it could be interesting to see whether the combination of sEEG data could yield semantic decoding performance improvements.

3 Methodology

This section elaborates on the methods used in this paper in order to answer its research question pertaining to the feasibility of decoding semantic information from sEEG data recorded from epilepsy patients. This chapter adopts a similar structure as was seen in the related work section, in that the brain data preparation steps are discussed first, after which the decoding tasks are elaborated on and instantiated. This order is maintained such that the inputs and outputs of the decoding models may be clear before they are introduced, negating any confusion regarding the tasks they aim to accomplish. Furthermore, since this study is based on previous internal work, conducted at the UMCU, many of the pre-processing steps had already been implemented, which is why a clear distinction is made between the parts of the project that were already in place, and the parts that were introduced in this study. a schematic overview of this study's pipeline can be viewed below, which serves to guide the individual subsections of the methodology.

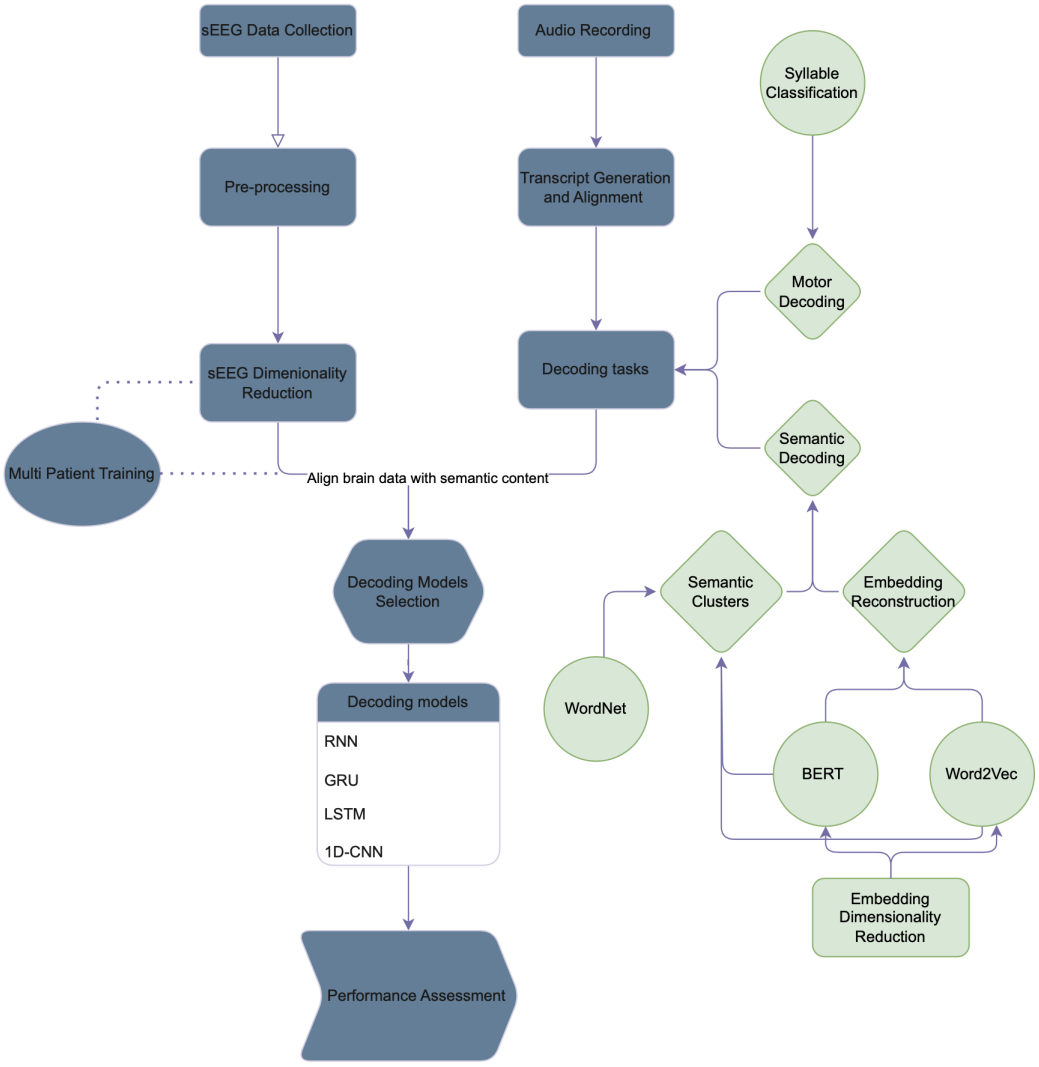


Figure 13: This study's main pipeline as described in the Methodology

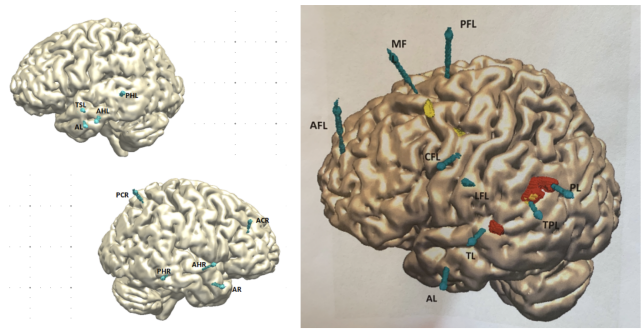


Figure 14: sEEG Electrodes implantation for participant 1 (left) and participant 2 (right)

3.1 Data Collection

This study's input sEEG data was obtained from two native Dutch participants with medication-resistant epilepsy, who underwent temporary implantation of sEEG electrodes, in order to localize the origin of their seizures and assess the possibility of removing the associated brain tissue. Participants gave written informed consent to participate in scientific research based on the utilization of their sEEG recordings, obtained between clinical procedures, and the use of these recordings was approved by the Medical Ethical Committee of the University Medical Center Utrecht in accordance with the Declaration of Helsinki. every patient underwent a pre-operative MRI scan and post-operative CT scan for the purposes of both localization and placement of the electrodes. Depth electrodes were 0.8 mm in diameter with 5 to 18 contact points. The electrodes were placed into the left (participant 2) or bilateral (participant 1) hemispheres, mainly including temporal, parietal, and occipital cortices, as can be seen in figure 14.

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 Microsoft Prime tablet using a software package called Presentation software (Neurobehavioral Systems) [64]. Participants could hold the tablet according to their own comfort and read the pages at their own pace. sEEG recordings were acquired using Micromed at a sampling rate of 2048 Hz, filtered at 0.3-500 Hz.

The experiment from which both the semantic data as well as the sEEG recordings were obtained consisted of a natural language task in which the subjects were instructed to read aloud chapter one of Harry Potter and the Sorcerer's stone in Dutch. The semantic data consisted of all the words pronounced by each subject as they read the chapter. The chapter was divided into two runs, dividing the chapter into roughly equal parts in terms of their word count, which was done in order to minimize the strain on the patients. Finishing each run took approximately 15 minutes, after which participants were given a short break. The 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.

3.2 Python Libraries

The tasks in this study were implemented using the Python programming language, which is one of the most popular languages for data analysis and machine learning [71]. The most commonly used packages in this study can be observed in table 1 below.

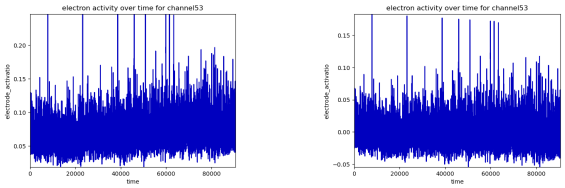
Table 1: Commonly used Python packages

Package	Purpose
Pandas	Data structure generation and manipulation
Torch	Deep learning and tensor computation
TensorFlow	Deep learning and tensor computation
Keras	Deep learning interface for TensorFlow
Transformers	Contains pre-trained transformer models
NumPy	Support for large, multi-dimensional arrays, matrices, and mathematical functions
Sklearn	Machine learning and statistical tools
NLTK	Natural language processing
Pyphen	Text hyphenation for Python
Matplotlib	Data visualization
Seaborn	Data visualization
PyNSL	python equivalent of NSL, used for spectogram audio computation
SciPy	Statistical analysis library
Azure	Management and use of Azure resources from Python application code

3.3 sEEG data pre-processing

The sEEG data utilized in this study had already been processed to an extent where it could be directly used in a decoding pipeline. These steps included filtering out noisy channels, re-referencing the signal through the Common Average referencing Method (CAR) and removing power line noise (50HZ) [6]. After these initial steps, the electrode data was converted into the time-frequency domain through the application of the Short-Time Fourier Transform (STFT), from which the activations in the High Frequency Band range (60-150HZ) were obtained by averaging amplitudes for the entire range of extracted frequencies [44]. Finally, the High Frequency Band power spectrum was re-sampled at 100HZ in order to facilitate analysis later on.

In addition to the already implemented pre-processing steps, some additional steps were taken in the current study, in order to further enhance the usability of the sEEG data for decoding purposes. These steps include de-trending the signal for each channel, as some long-term trends became apparent after visually inspecting each channel. Such long-term trends are likely irrelevant for decoding, as activations for each word rarely last longer than a second, meaning that these trends are likely not-informative for decoding individual words. the results of de-trending can be observed in figure 15.



(a) before de-trending (b) after de-trending

Figure 15: sEEG activations before and after de-trending

Furthermore, values that were more than two standard deviations away from the mean of each channel were clipped to be twice the standard deviation. This was done in order to reduce the effect that outliers could have on the data during data normalization, which in turn is necessary to facilitate convergence during decoding. Normalization is discussed at the end of this study's methodology, however, as it is more concerned with model instantiating than the more general data preparation discussed here.

Lastly, utilizing 1D convolutions as an alternative to the method of down-sampling the spectrum data to 100HZ was considered, due to the technique's ability to learn variable temporal patterns from the original data, rather than applying a static down-sampling technique. Its implementation was ultimately canceled, however, due to insufficient evidence for the current method's inefficacy to keep any existing temporal patterns in tact. 1D convolutions are applied in other areas of the methodology, however, as will become evident in later sections.

3.4 Speech Transcript generation and alignment

The audio data acquired during the task has been subjected to a similar amount of pre-processing as was seen in the sEEG data pre-processing section. The Microsoft Azure Cognitive Services were used to generate transcripts from audio recordings. In particular, Azure speech-to-text service was used to provide transcripts in a TextGrid format, obtaining information about each time instant at which words were pronounced (onset and offset time). Even though the Microsoft Azure Cognitive Services have been proven to accurately generate speech to text files, this process always requires additional validation, due to the importance of the alignment between onset and offset times corresponding to the audio, as these are crucial in selecting the correct sEEG sequence [41].

Therefore, the transcripts were validated using Praat, which is a software package designed for the phonetic analysis of speech signals [7]. Utilizing Praat allowed for checking the audio alignment, and the quality of the transcriptions. Furthermore, by using praat, the Word Error Rate (WER) for the transcripts could be computed, which is defined by the number of incorrect words identified during validation divided by the total number of words. The resulting WERs are shown below in figures 2 and 3. Selecting brain activity correspondent to each word could then be performed by utilizing the onset and offset time, at which each word was spoken, derived from the TextGrid file. The specifics of these timings are discussed in a later section, as they depend on the task for which the brain data is used.

Table 2: Word Error Rates (WER) for subject 1

	Word detection count	error detection count	error percentage [%]
run 1	2437	132	5.42
run 2	2375	122	5.15
total	4812	254	5.28

Table 3: Word Error Rates (WER) for subject 1

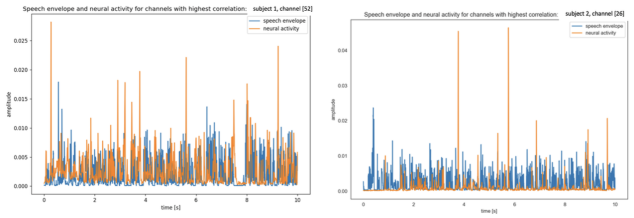
	Word detection count	error detection count	error percentage [%]
run 1	2594	133	5.13
run 2	2051	157	7.65
total	4645	290	6.24

3.5 sEEG feature selection and de-noising through Auto Encoders

Earlier work, conducted at the UMCU, has already shown that at least some of the electrodes in each patient are related to the speech envelope, generated from the audio recorded during the reading task. The speech envelope corresponds to the slow overall amplitude fluctuations of the speech signal over time, with peaks occurring roughly at the syllabic rate, and has been shown to correlate well with activations in the auditory cortex as well as motor processing and continuous speech perception [4] [5] [46].

The speech envelope was computed, utilizing the python library pyNSL, which is the python equivalent of the Matlab NSL toolbox developed by Chi and colleagues, and computes the sound spectrogram following the biological model of sound processing by the cochlea [11]. The spectrogram was extracted at 8 ms frames along 128 logarithmically spaced frequency bins in the range of 180–7,200 Hz. The spectogram data was then averaged over the frequency bins to obtain a 1D spectral sound envelope. The resulting spectral envelope was down-sampled to 100 Hz to match the sampling rate of the sEEG Sequence data. The resulting speech envelope was plotted alongside the HFB sEEG data, as can be seen in figure 16.

in order to assess the relationship between the speech envelope and the HFB sEEG data, the non-parametric Spearman correlation coefficient was calculated for each channel. To account for the time lag between these two quantities, the correlation for each lag between the two signals in a range of



(a) (b)

Figure 16: High gamma activity [60-150 Hz] tracking the speech envelope. Panels a and b depict results for subject 1 and 2, respectively. The speech envelope is depicted in blue, whereas the HFB sEEG data for channels 52 (sub-1) and 26 (sub-2) are shown in orange.

-200ms to 200ms with 10 ms steps was computed. Afterwards, one value of correlation for each electrode was obtained by taking the the maximum correlation over the time lags.

These correlations between the speech envelope and electrode activations were bootstrapped by shuffling the sEEG recordings into 10 second blocks, in order to obtain a random distribution of brain activity. This shuffling was performed 1000 times and the correlations between the shuffled brain data and speech envelope were calculated and saved during every shuffle. In doing so, the actual correlations between electrode activations and the speech envelope could be compared to the correlations from the random distributions, and the p-value for each electrode could be calculated as the ratio between the number of correlations greater than the actual correlation and the number of iterations. Setting the p-value at 0.01 allowed for calculating which of the electrodes showed sadistically significant correlations with the speech envelope, as can be seen in figure 17.

As can be seen from figure 17, however, most of the electrodes for each patient showed either no significant correlation or a very moderate correlation of $0.05 < r < 0.1$, indicating that many of the electrodes would most likely not aid in decoding semantic information and would likely represent some form of noise as far as the decoding models were concerned. Furthermore, having more channels could add complexity to the input space, especially if most of these channels did not contain any decodable information. In order to both reduce the complexity of the input space and to remove some of the noise represented in uncorrelated channels, a special type of neural network architecture introduced in the related work section, called the auto encoder was applied. As was mentioned before, the purpose of such models is mainly to reduce the dimensionality of the input data by aiming to reconstruct the input data from a reduced latent space, also called a bottleneck layer. In doing so only the most important information of the input data could be preserved in the bottleneck layer, which effectively removes noise and decreases the data's complexity.

Since the sEEG recordings were represented as multichannel sequence data, two overarching types of auto encoders cold be applied here, namely sequence auto encoders,

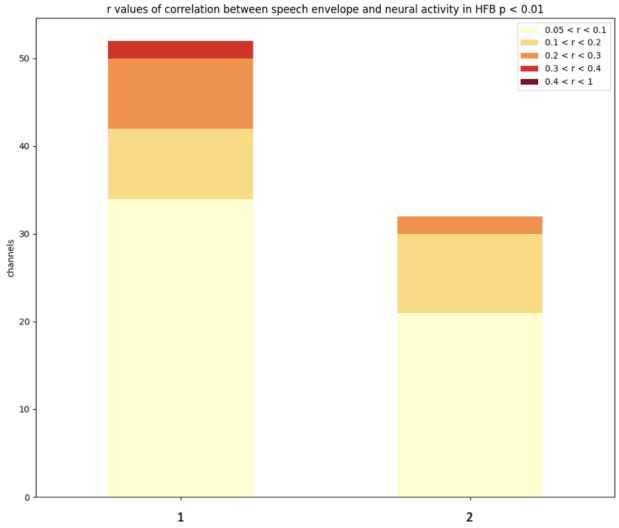


Figure 17: Channels with significant correlation ($p < 0.01$) between speech envelope and neural activity in HFB. The color bar represents different values of correlation. Full number of channels sub 1 = 99, sub 2 = 79.

which take the temporal relationships of the data into account, and feed-forward auto encoders, which simply process one time-step at a time. The benefit of utilizing sequence auto encoders is that they may uncover temporal relationships that could have been missed by feed-forward networks, and may thus better capture semantic patterns which arise over multiple time steps rather than instantly. A benefit of feed-forward auto encoders, on the other hand, is that they are less complex and have more training data to work with, since each sequence consists of multiple time-steps, which can be separately fed into the feed-forward auto encoder. To see which model is better suited for this study's needs, both types of auto encoders were implemented.

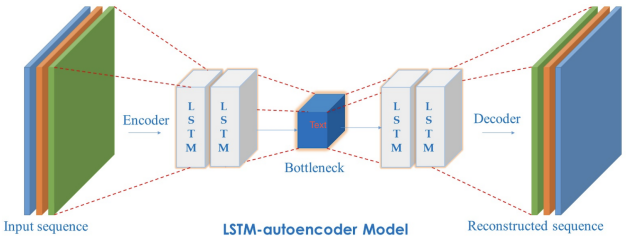


Figure 18: LSTM auto encoder architecture. The input sequence is first encoded into a lower dimensional latent space using a two-layer LSTM encoder, after which it is decoded using a two-layer LSTM decoder. Figure from reference [19].

Furthermore, a variational version was created for both types of auto encoders as well. This study utilized variational auto encoders (VAEs) in a slightly non-traditional way, because such models were applied without the use of the KL

divergence regularization term, discussed in the related work section. This was an intentional choice, because the KL divergence term forces the latent space to follow a Gaussian distribution learned from the combination of all inputs. However, by forcing the latent space to adhere to the Gaussian of all inputs, some of the latent space's differentiability is lost, meaning it can generalize the data too much. By removing the KL divergence term, whilst still sampling from the Gaussian distribution in a way that makes that sampling differentiable (reparameterization), the model could focus on extracting relevant mean and standard deviation parameters, solely for the purpose of improving reconstruction error. Choosing this approach involved a trade-off as such a VAE sacrifices most of its generative capabilities. However, early testing revealed a significant improvement in results by omitting the KL divergence term, and given that generative capabilities were irrelevant for the current study, the decision was straightforward. Therefore the variational auto encoder could better be described as a hybrid between regular auto encoders and variational auto encoders.

In summary, four types of auto encoders were applied with the purpose of applying both feature selection and de-noising to the input sEEG data. More specifically, the feed-forward auto encoders consisted of single layer encoders and decoders with Leaky-ReLu activations. The sequence models, on the other hand, featured dual layer LSTM encoders and decoders, in order to extract more complex temporal patterns from the input data. Since these models have their own activation functions built-in, no additional activation functions were applied. Furthermore, the Sequence models were configured to return the entire input sequence, without compressing the temporal extent of the data, because there is currently no evidence that doing so would increase performance, and because preserving the temporal extent of the data keeps the matching between the sEEG data and the audio recordings intact, which makes it easier to apply the resulting compressed data later on. Lastly, the variational auto encoder variants of each type of auto encoder also featured a separate hidden layer, which represented the Gaussian distribution of the input data, through its associated mean and standard deviation parameters, and contained 256 hidden units. For all models, the bottleneck layers were set at 30 units, meaning that the input sEEG data would always be reduced to 30 channels. The models were trained under the Mean Squared Error (MSE) loss function, and under the Adam $(lr = 0.01)$ optimizer. All models were implemented in PyTorch, which is a commonly used deep learning library. An overview of the differences between each architectures can also be viewed in table 4.

Besides measuring model performance in terms of training and validation loss, a more tangible performance metric presents itself in the form of the electrode correlations to the speech envelope, discussed earlier. The assumption here is that if the reduced sEEG data show better or at least similar correlations to the speech envelope, one can generally assume that the auto encoders were successful in keeping the critical data intact, whilst removing redundant information from the sEEG data. Therefore, channel correlation to the speech envelope constitutes the main performance metric for these auto encoder models, as it serves as a more transparent performance metric than the model losses, which by themselves give no representation of the usefullness of the compressed sEEG data.

Table 4: different Auto Encoder architecture characteristics, AE = auto encoder, VAE = variational auto encoder

	AE	VAE	LSTM AE	LSTM VAE
temporal extent (time steps)	1	1	variable	variable
Gaussian distribution layer	No	Yes	No	Yes
number of encoder/decoder layers	1	1	2	2

3.6 Semantic representations and decoding tasks

3.6.1 Large Language models and word embeddings

As mentioned in the related work section, this study explores two major ways in which decodable semantic representations can be extracted from the validated speech transcripts, generated from the audio recordings. The first of these methods involves generating word embeddings through the application of large language models, and has been used in this study's predecessor as well, through the use of the BERT language model, which generates contextual word embeddings. This means that one word can have multiple embeddings based on the context in which it was used. Beside utilizing contextual word embeddings models, this study also makes use of static embedding models, which learn one fixed word embedding for each unique word. Such models are less complex, because the embedding for each unique word is based on global contexts, rather than individual context occurrences, meaning that all the different contexts in which a single word can occur are squished into one representation for that word. for example, consider the following two sentences:

"The man was accused of robbing a bank."

"The man went fishing by the bank of the river."

A static model would consider both instances of the word *bank* to be identical, whereas a contextual model, like BERT, would consider these words to have different meanings. This makes contextual models better for word sense disambiguation tasks, where the goal is to infer the correct meaning of a word based on the context in which it appears. On the other hand, the increased complexity of contextual word embedding models may increase the difficulty of decoding semantic information from brain data, as contextual models generally represent the semantic information in a much more high dimensional format, when compared to static models [18]. This

is a direct result of the contextual nature of such models, as they need far more expressability in order to differentiate between different word senses. This is the main reason for considering both contextual and static embedding models, as the dimensionality of the input data, corresponding to the number of electrodes implanted in each patient, is far lower than the dimensionality of the word embeddings generated through a contextual embedding model, like BERT, which generates 768 dimensional word embeddings. An in depth overview of how these different model architectures are used is given in the following two subsections, after which this study's alternative to word embedding models is presented.

3.6.2 Contextual embeddings through BERT

As mentioned in the related work section, BERT is a bidirectional encoder transformer model, meaning that it uses the self-attention mechanism, which allows it to handle arbitrarily long sequences, and its bidirectional nature allows it to process context on both the left and the right side of a target word. Its architecture consists of 12 layers and 768 hidden states [**?**]. Transformers need an enormous amount of data in order to be properly trained, which is why BERT is already pre-trained on a vast text corpus, meaning that it has already learned the vector representations of many words. Furthermore, BERT's tokenizer divides a word into the sub-words that make up that word, which allows it to check the sub-words as well, if any of the words it encounters should be outside of its vocabulary. BERT can be further fine-tuned for specific downstream tasks, but this is not necessary for the purposes of this study, as only the embeddings from BERT's hidden state are required for the current decoding tasks. What is required for this study, however, is a Dutch pre-trained version of BERT, as this study's reading task was carried out in Dutch. To that end, a Dutch monolingual version of BERT, called "BERTje", was used as the model of choice, due to it having fewer out of vocabulary words than the other available dutch BERT models. In order to convert the speech transcript words into embeddings, the following steps were carried out in a similar fashion as in the UMCU's earlier work:

- Out of vocabulary words were removed from the word list.
- The BERT model and tokenizer were loaded.
- Each input sentence was marked with start "[CLS]" and separator tokens "[SEP]".
- The words were then tokenized using the tokenizer.
- afterwards, each token was assigned to a unique token ID and an index denoting its position within the sentence.
- each sentence was then given its own segment-id, which allows for distinguishing between different sentences.

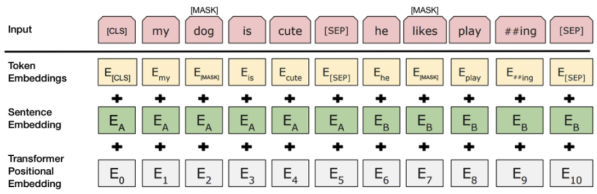


Figure 19: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings [18].

- finally, the BERT model was applied to each sentence, where the last four layers of the model were summed in order to generate the embeddings.

768 dimensional embeddings for all of the words that were in BERT's vocabulary were obtained by following these steps, utilizing the huggingface transformers library. After having generated the word embeddings, stop words and other words with low semantic quality were removed, utilizing NLTK's stop-word package, which includes an extensive list of dutch stop-words. Furthermore, words which did not relate to the task, mostly consisting of sentences spoken before or after reading the chapter, were removed as well. In doing so, only 1889 words out of the original 4812 remained for subject 1, whereas for subject 2 only 1486 of the original 4645 words were left. Note that subject 2 lost much more of its word count, due to the presence of multiple unscripted sentences in subject 2's data.

Table 5: word count before and after stop-word and error removal for subjects 1 and 2

	sub 1	sub 1 post removal	sub 2	sub 2 post removal
run 1	2437	961	2594	754
run 2	2375	928	2051	732
total	4812	1889	4645	1486

A visual representation of some common words in the embedding space can be generated through Sklearn's t-SNE package, which can visualize high dimensional data in a 2D space. Doing so highlights the efficacy of embeddings in keeping related words close together in the semantic space, as can be seen from figure 20. From this figure, one can observe that pairs of similar words such as meneer (sir) and mevrouw (madam), straat (street) and hoek (corner), nek (neck) and snor (moustache), mantel (coat) and laarzen (boats) all appear close to each other in the semantic space.

3.6.3 Static embeddings through Wikipedia2Vec

In contrast to the contextual BERT model, generating word embeddings pre-trained static embedding models such as Word2Vec or GLoVe does not require any special formatting

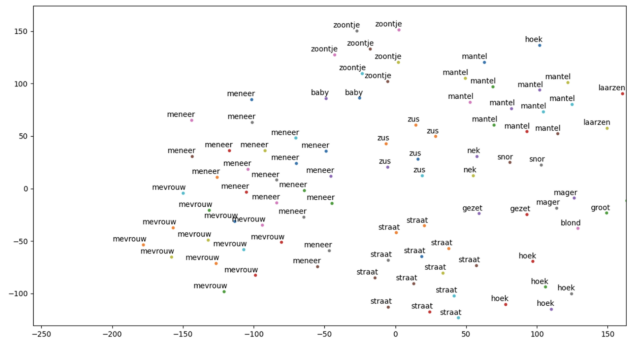


Figure 20: 2D representation of some common words in the BERT embedding space, generated using t-SNE

or pre-processing. A pre-trained model can simply be downloaded and its embeddings can be retrieved by querying the list of embeddings based on their word labels. Finding dutch pre-trained embedding models, however, is more problematic, when compared to the BERT model, since there are no pre-trained GLoVe models for the dutch language. Luckily, there are some pre-trained Word2Vec models available, such as the Wikipedia2Vec model, which contains pre-trained embeddings for twelve languages, including Dutch. Wikipedia2Vec is trained according to the skip-gram model, which is one of the original training models with which Word2Vec was originally introduced [38].

skip-gram works by training a probabilistic classifier that, given a target word and a context window, assigns a probability based on how similar this context window is to the target word. The learning algorithm for skip-gram embeddings is given an initial random embedding vector and learns to maximize the similarity of the target word and the contexts that occur with it in the training corpus, whereas it learns to minimize the similarity between the target word and contexts which do not appear nearby in the training corpus. In doing so it adjusts the embedding vectors, which are the model weights, such that similar words which occur in similar context are associated with similar embedding vectors. After training, the classifier is discarded and the embeddings are extracted from the model's hidden state.

Wikipedia2Vec offers both 100 dimensional as well as 300 pre-trained dimensional embeddings in Dutch. The 100 dimensional embeddings were selected for this study, because the main reason behind using static embeddings, was for them to serve as a less complex alternative to the contextual embedding models, mentioned in the previous subsection. After having downloaded the dutch pre-trained Wikipedia2Vec embeddings, the labels from the bert model were used to select the appropriate Wikipedia2Vec embeddings, since comparing these models requires using the same labels for each model. Afterwards, the electrode activations were mapped to the appropriate embeddings, based on the word they were originally paired with. Due to Wikipedia2Vec's static nature, multiple

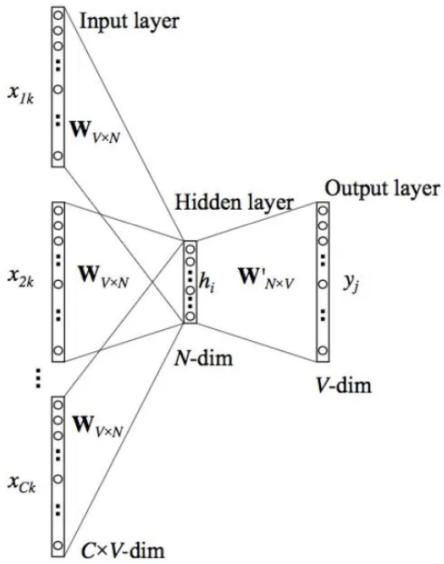


Figure 21: The word2Vec model's skip-gram architecture [56]

electrode activations, referring to the same word in a different context, were mapped to the same static embedding for that word, resulting in a more sparsely populated semantic space. A visual representation containing embeddings for the same words as in the BERT model example can be observed in figure 22. As can be seen, the embedding space is indeed far more sparsely populated with only one embedding per word label. Note that, although the coordinates of each embedding are different, when compared to BERT, the overall similarity patterns are quite similar, with similar words again appearing close together in the semantic space.

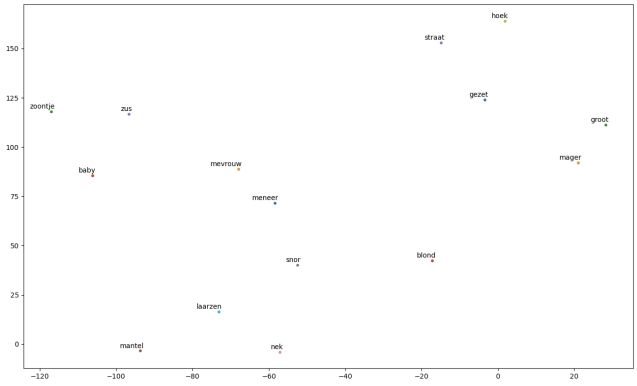


Figure 22: 2D Wikipedia2Vec embedding representation of the same words from the BERT embedding space, generated using t-SNE

3.6.4 Clustering word embeddings

Besides using the embeddings generated through BERT and Word2Vec for decoding directly, they can also be used to generate semantic categories, the decoding of which can be seen as a classification task, which is much simpler than trying to reconstruct the vectors themselves. The clustering was performed according to a distance function, which measured how far apart different vectors, or embeddings in this case, were from one another in the vector space. The distance function utilized in this study for this task, and for almost any other task that follows, is based on the cosine similarity function, described in the equation below, which knows many applications in the fields of Natural language processing and information retrieval [82] [25].

$$cos(\theta) = \frac{A \cdot B}{||A||||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3}$$

As can be seen from equation 1, cosine similarity is essentially the normalized dot product of two vectors. Its appeal, when compared to other similarity functions and other distance functions, is that it only considers the angle between the two target vectors and is not influenced by the vector's absolute values , meaning that it is scale-invariant, which is very helpful when dealing with high-dimensional data, where the importance of individual dimensions may vary, and scale-invariance becomes crucial. Furthermore, in high-dimensional data, the impact of outliers in one dimension can be disproportionate. Cosine similarity is less sensitive to outliers, making it more robust in the presence of noisy or outlying data points. Due to these advantages in cases of high-dimensional data, Cosine similarity and its complementary measure, cosine distance, defined to be $1 - cos(\theta)$, form the basis for most of the comparison metrics utilized in this study. Utilizing NLTK's clustering package with cosine-distance as its distance function allowed for generating clusters of both the static and contextual embedding spaces.

The number of clusters for each embeddings space was set to 5, resulting in the generation of 5 semantic categories for the purpose of decoding. This number was chosen in order to keep the complexity of the decoding task within a feasible range, and matched the number of categories that were used in the motor decoding task that is discussed in the last subsection of the tasks. The semantic content of the clusters and the cluster distributions are presented along with the main results of this study, as the differences between the contents of each cluster could be related to differences in model performance. These differences themselves can also be considered as part of the results as they indicate how semantic encoding may differ between different models.

All in all, by utilizing clustering techniques, the complexity of the decoding task could be greatly reduced from trying to reconstruct high dimensional vectors, to a standard classification problem. Do note, however, that cluster classification is a less precise task than semantic vector reconstruction, meaning that some of the differentiability between the different words is lost, due to the more overarching nature of clustering, but this trade-off seemed more than worth it, when taking into account this study's low data availability, combined with the original complexity of the decoding task.

3.6.5 semantic space dimensionality reduction using Auto encoders

As can be observed from the previous sections, one of the major goals of this study was to simplify the tasks, both from the input data as well as the semantic representation perspectives. For the semantic representations, this trend could already be seen in the application of a static embedding model, which is inherently less complex than a contextual embedding model, and through the application of clustering methods in order to further simplify the representations. Besides selecting models with less complexity and utilizing clustering, one can also aim to reduce the complexity of an existing model by reducing the complexity of the semantic space in which the word embeddings reside. Doing so keeps the differentiability of different words intact, unlike clustering methods, whilst still providing simpler representations of each word.

Earlier work conducted at the UMCU has already been conducted towards this end, in which Principal Component Analysis (PCA) was applied in order to reduce the 768 dimensional BERT vectors to 50 dimensional vectors. The dimensionality reduction performance of PCA, however, was found to be incapable of preserving the semantic qualities of the original BERT embeddings, with many related embeddings losing their similarity, after having been reduced through PCA.

An alternative method for dimensionality reduction was already mentioned in the sEEG feature selection section, where auto encoders were utilized in order to reduce the number of sEEG channels. A similar approach can be adopted for the word embeddings generated by the BERT and Word2Vec models, in the sense that the original 768 and 100 dimensional vectors can be reduced to a fraction of their dimensionality, through the use of auto encoders. Note, however, that unlike the sEEG sequence data, there exists no temporal extent in the word embeddings, meaning that the sequence auto encoders, introduced earlier, serve no purpose in compressing the word embeddings. The feed-forward auto encoders, on the other hand, could be implemented in a similar manner as was done when applying them to the sEEG data. The only differences between these models and the ones used before can be found in the dimensions of the bottleneck layer, the number of layers used in the encoders and decoders, and the loss function used to train the models.

Whereas the sEEG data auto encoders used a bottleneck layer with 30 hidden units, the auto encoders for the word embeddings instead utilized bottleneck layers with 50 hidden units, meaning that the embeddings were always reduced

to 50 dimensional embeddings, compared to 768 and 100 dimensional vectors for the original BERT and Word2Vec models, respectively. Additionally, since embedding compression represents a complex task, given BERT's original 768 dimensional nature, the auto encoder structure was updated in order to feature hierarchical dual layer encoders and decoders. In this hierarchy, the first layer would have half of the embedding's original dimensionality, and the second layer would have the final 50 units, which would constitute the reduced embedding space. Furthermore, When compressing the Wikipedia2Vec embeddings, the first layer would consist of $\frac{3}{4}$ of the input space, since the compressed dimensions would already be half the size of the original Wikipedia2Vec vectors. The variational auto encoder, however, was not granted a second hierarchical layer, because its sampling layer already introduced enough complexity on its own. Instead its sampling layer was increased from 256 hidden units to 384 units, when reducing the BERT embeddings, in order to handle the higher dimensional embeddings. For Wikipedia2Vec such alterations were not necessary.

Additionally, whereas the sEEG auto encoders relied on the MSE loss function for comparing the reconstructed vectors to the original input data, the embedding auto encoders relied on cosine similarity as the basis of their loss function. This metric, and the resulting cosine distance loss function, defined in the previous subsection, was chosen because here the auto encoders aimed to preserve word similarity in their embeddings, and the method of choice for this task was cosine similarity, due to reasons mentioned in the clustering subsection. Aside from these three differences, the auto encoders were similar to the ones mentioned in the sEEG auto encoder section. Performance assessment of these auto encoders was conducted according to the models' train and validation losses, as well as by testing the difference in similarity levels between words before and after dimensionality reduction, the results of which are shown together with the main results of this study.

3.6.6 Semantics and lexical databases

The other major way in which decodable semantic representations can be extracted is through the use of lexical databases that link words semantically, such as the WordNet database which is utilized in the current study. WordNet semantically links different nouns, verbs, and adjectives according to a hierarchical class system, in which each each parent class is more general than its sub-classes. The resulting knowledge base thus hierarchically builds up from very specific concepts to the most general classes they belong to, such as people and dogs both belonging to the entity super class. An example of this hierarchy can be observed in figure 23.

besides using this class based structure, WordNet also groups words with similar meanings into groups called synsets, and captures different meanings or word senses for each concept in the same synset as well, meaning that the

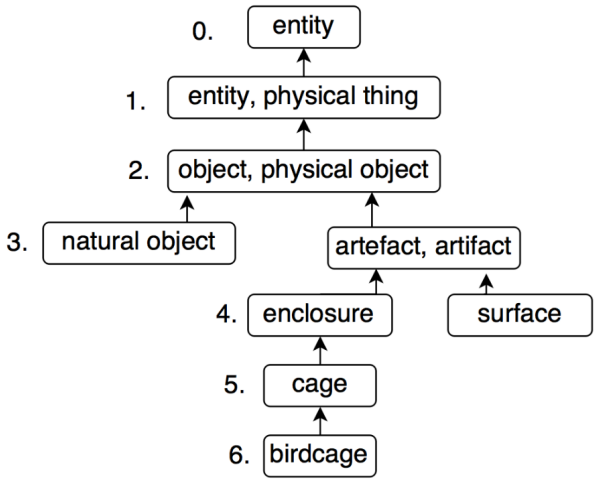


Figure 23: WordNet hierarchy example. The most general classes are on top, whereas specific concepts are found further down in the graph [85].

polysemy of words is preserved in its knowledge base. This concept of polysemy relates back to the different word senses between which the previously discussed BERT model can differentiate, meaning that the word bank has multiple meanings in WordNet just as it did under the BERT model. This implies that a given synset includes multiple instances of the same word, each having a different word sense, as well as words that are closely related to the target word, from which the synset was generated. In this study, WordNet is utilized as an alternative to embedding models. In doing so the aim is to create semantic categories based on the similarity between words according to the positions of the words in WordNet's hierarchy. The idea behind this is that words with similar meanings have shorter paths to their closest parent class than words with less similar meanings. This similarity measure is called Wu-Palmer similarity and is commonly used for calculating similarity at both the word and sentence level in Natural language processing, Informational Retrieval, Text-Mining and Q&A systems [85]. It is also one of the few similarity measures which allows for the comparison of words with different Part Of Speech tags such as words and nouns [85]. Wu-Palmer similarity is calculated according to the following formula.

$$Wu-Palmer = 2 * \frac{Depth(LCS(W1,W2))}{Depth(W1)+Depth(W2)} \quad (4)$$

in this formula, W1 and W2 refer to the words between which the similarity is calculated, whereas depth represents the depth of a concept in the hierarchy, or in other words, the length of the path from the concept to the root of the hierarchy. Lastly, LCS refers to the least common subsumer

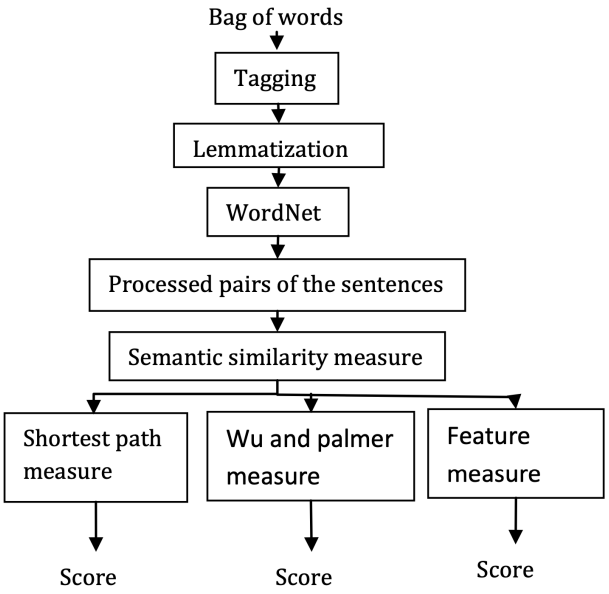


Figure 24: general WordNet pipeline for semantic similarity computation [85].

of the two words, meaning the concept at the lowest level in the hierarchy that is an ancestor to both W1 and W2. The resulting similarity ranges from 0 to 1, where 0 indicates no similarity, and 1 indicates that the concepts are identical. The measure takes into account both the depth of the LCS and the depths of the individual concepts to provide a normalized similarity score.

3.6.7 WordNet implementation

WordNet is integrated into the python NLTK package where it can be used to look up the different synsets and hypernyms or parent classes of each concept that is fed into it. The general similarity score generation pipeline can be viewed in figure 24. Note that WordNet is designed to be used with the English language, and although multi-lingual versions of WordNet do exist, they are far less complete than their native English counterpart. The implications of this limitation became apparent when trying to generate synsets for the different labels that were used in the BERT model, where more than half of the words were simply not recognized under the Dutch version of WordNet. Therefore, the different labels had to be manually translated from dutch into English, as using automatic translation packages, such as python's google translate API, proved to be inadequate for the task of keeping the semantics of each label intact after translation. After having translated the words from Dutch to English, the synsets for each word were generated, which could subsequently be used to compare different words for semantic relatedness.

Also note that WordNet does not automatically distinguish

$$\begin{bmatrix} & W_1 & W_2 & \dots & W_N \\ W_1 & sim & sim & sim & sim \\ W_2 & sim & sim & sim & sim \\ \dots & sim & sim & sim & sim \\ W_N & sim & sim & sim & sim \end{bmatrix}$$

Figure 25: Word by Word similarity matrix, where N refers to the total number of unique labels, and sim = WuP, ranging from 0 to 1

between different inflections or word senses of a given word, it simply returns the synset of all word senses and all related words. Because POS tagging is a time consuming task given the current scattered format of the word labels, a different approach was selected in order to differentiate between the existing synset entries of each target concept. First, Lemmatization was intentionally skipped, as the English version of WordNet can distinguish between different conjugations of a given verb. Keeping the conjugations in tact allowed WordNet to more easily distinguish verbs from nouns, in cases where the Lemma of a word could be interpreted as both. Furthermore, synset selection was performed according to the simple but effective approach of simply selecting the first synset, as this represented the most common entry with respect to how the verbs were conjugated. For nouns the first entry would also represent the most common entry with regard to its meaning. Subsequently, similarity between two word pairs was calculated using Wu-Palmer similarity. In doing so, similarity scores for each of the possible word pairs were obtained and were subsequently stored in a matrix of size $[labels * labels]$, as can be seen in figure 25. This similarity matrix was then used to cluster the different words using NLTk's clustering package utilizing cosine-distance, in a similar manner as was done for the embedding clustering, mentioned earlier, which resulted in the generation of 5 semantic categories. The resulting clusters are discussed in the results section of this study, for similar reasons as discussed in the word embeddings clustering section.

3.6.8 Syllable quantity classification

Besides aiming to decode semantic information from the sEEG recordings, a preliminary motor decoding task was also included as a way to check whether the current data contained any decodable information in the first place. Furthermore, interpreting motor decoding results was much more straightforward, when compared to semantic decoding, as these results have a transparently measurable outcome. Additionally, obtaining above chance motor decoding performance could aid in the discussion of semantic decoding performance, as it can serve as a sanity check for the existing pipeline. More specifically,if motor decoding proved to be a feasible task for the current data and decoding models, the suitability of the decoding models could be validated even if the semantic tasks yielded inconclusive results. Simply put, any lack of perfor-

mance could in that case not be attributed to the decoding models themselves, because their efficacy would have been proved beforehand. Lastly, because earlier work conducted at the UMCU has shown that at least some of the electrodes were correlated to the speech envelope, which peaks roughly at the syllable rate, as mentioned in the sEEG auto encoder section, including a motor task should be at least be a somewhat feasible task, given the data's relationship to the speech envelope as can be seen in figure 17.

Therefore, this study included a motor decoding task, which presented itself in the form of a syllable quantity classification task, information which should be reflected in the peaks of the speech envelope. Syllable counts of each word were obtained by utilizing the Pyphen package, which has the ability to Hyphenate words and includes a distribution for the Dutch language. In doing so the words were split into 5 categories, corresponding to the number of syllables in each word, ranging from 1 to 5. The distribution of the different categories can be viewed below in tables 6 and 7, for subjects 1 and 2, respectively. As can be seen from the distribution of the number of syllables, the different classes are quite imbalanced, with one and two syllable words making up more than 80% of all the words used for decoding. Ways in which this class imbalance was dealt with are discussed in the deoding model section of the methodology.

Table 6: distribution of number of syllables for all word used for decoding, subject 1

Syllables	1	2	3	4	5	total
Count	798	771	274	41	5	1889
percentage	42.24%	40.82%	14.51%	2.17%	0.26%	100%

Table 7: distribution of number of syllables for all word used for decoding, subject 2

Syllables	1	2	3	4	5	total
Count	631	590	227	34	4	1486
percentage	42.46%	39.70%	15.28%	2.29%	0.27%	100%

3.6.9 Applied tasks

This subsection provides a final overview of how the different decoding tasks, discussed in the previous subsections, were used for decoding. The first of these tasks is based on the embeddings generated through BERT and Word2Vec, as mentioned before, and the task here is to reconstruct the embeddings from the sEEG recordings directly. Therefore, this task is essentially a vector reconstruction task, in which the aim is to make the embeddings generated by the decoding models as similar as possible to the original word embeddings. If the reconstructed embeddings are highly similar to the original embeddings, one can conclude that the network is able to decode semantic information from brain data. Similarity was measured in terms of cosine similarity for reasons that were discussed in the clustering section earlier, including its ability to deal with high dimensional data, and its decreased sensitivity to outliers. Cosine similarity therefore served as the loss function for the networks that are introduced in the next section. In order to define which results are meaningful, a strict baseline score denoting random chance level performance is also required. To that end, random chance level performance was defined to be the performance of a decoding model under the condition that the brain data and embeddings are shuffled, such that the embeddings and brain data no longer match with each other. This condition was chosen due to the fact that a model could simply learn the embedding structure of a given semantic space, rather than the actual semantic differences between different brain activations, and shuffling the labels and input data keeps the embedding structure intact, whereas it removes the semantic connection between the brain data and the embeddings. In doing so, one can clearly distinguish between a model that has learned to decode actual semantic information from brain data and a model that is simply reconstructing the overall structure of the embedding space, which would correspond to the average vector of the embedding space.

For the clustering based tasks, including the clustered contextual and static embeddings, the WordNet clusters, and the syllable clusters, more standard comparison measures could be utilized. This is the case because these tasks all represented more straightforward classification tasks, where accuracy could simply be defined as the number of correctly classified cluster labels divided by all the predictions made by the model. In these tasks, random chance level performance was set to the prevalence of the majority class, as the best strategy for a model that has not learned anything useful, is to always pick the most frequently occurring category. Random chance level performance would thus be dependant on the distribution of the different clusters and is therefore mentioned seperately for each task and training scenario.

3.7 Decoding models

This section gives an operational description of all the models that were used for the different decoding tasks. Note that the conceptual workings of each model are omitted here, since they have already been discussed extensively in the related work section. The type of decoding models used in this study mainly included the sequence models discussed in the related work section, as the sparse distribution of sEEG electrodes did not allow for exploiting the spatial extent of the data, meaning that spatial models, such as 2D convolutional neural networks and their derivatives such as Convolutional LSTMs could provide no added benefit over standard sequence models, given the current data. Also note that, due to the low number of available training samples (1889 for sub 1, 1486 for sub 2), most advanced models provided no increased benefit over the

standard sequence models either, since increased complexity leads to an increase in a model's number of trainable parameters, which in turn require more training data to provide any benefits. With little training data, complex models were unlikely to reach convergence which made them unsuitable for the task at hand. On the other hand, semantic decoding is a complex task, with no clear relationships between brain data and semantic representations, meaning that the simplest of models are also unlikely to properly capture any semantic relationships. Therefore, the focus of decoding models is placed upon models with as little complexity as possible, whilst keeping enough complexity intact such that the models are able to properly map the brain data to the sEEG data. What this entails for specific model details is made clear in the next subsections which the discusses the different aspects of the decoding models individually.

3.7.1 Traditional sequence models

The main models used for decoding were the traditional sequence models introduced in the related work section, including RNNs GRUs and LSTMs. These models have in common that they all process entire sequences of data and have the ability to classify or reconstruct a single instance of data from that sequence. As mentioned in the related work section, RNNs are the simplest sequence models, followed by GRUs and finally LSTMs, which are the most complex vanilla sequence models. For the current decoding tasks, all models were similarly implemented, and the differences in implementations can be categorized as belonging to the following design choices:

- **Layer type**: The type of layer used in a neural network constitutes the defining difference between the different model types. As mentioned, both RNNs, GRUs and LSTMs are used for the current decoding tasks.
- **Number of hidden layers:** Utilizing a single hidden layer may prove to be too simplistic in order to capture any semantic patterns in the data, which is why architectures with a varying number of layers are tested, ranging from 1 hidden layer to 4 hidden layers, with 2 hidden layers most likely striking the proper balance between complexity and trainability.
- **Post sequence fully connected layer:** Both architectures with and without another fully connected layer after the last sequential layer were tested, in case additional processing after the sequence layer activations should prove beneficial. Note that in case a hidden layer was used, the last sequential layer would not return the entire sequence of its hidden states, but would only return the last hidden state, such that the information could be processed by a feed-forward hidden layer.
- **Activation functions:** All sequence models have built in activation functions, with RNNs using the Tanh activation function, and GRUs and LSTMs using both the Tanh activation function for the recurrent activation as well as the sigmoid activation function for the layer activation. In case an additional hidden layer was used, both the Tanh and ReLU activations were implemented and tested.
- **Output layer:** the output layer of the model is the final layer, whose number of nodes correspond to the dimensionality of the semantic representation used for decoding. Therefore, this layer would consist of 5 nodes for any of the cluster classification tasks, whereas it would consist of 50, 100, or 768 nodes for the vector reconstruction task, depending on which embedding model was used, and whether dimensionality reduction through the use of auto encoders was applied.
- **Number of hidden units:** A general rule of thumb for choosing an appropriate number of hidden units, is to take the average of the input and output dimensions and adjust from there. Furthermore, because having too much or too little complexity is unwanted due to reasons mentioned earlier, the number of hidden units per layer would never exceed the minimum or maximum boundaries set by the input and output dimensions of the decoding models. Therefore, the number of hidden units would range from 30 to 768, as these represent the minimum and maximum values that the inputs and outputs adhere to. The exact number of hidden units would naturally depend on the decoding task and the input data that was used, but would generally be close to the lower end of the aforementioned spectrum, in order to curb complexity.
- **Output layer activation:** Activation functions are generally not used for vector reconstruction tasks, meaning that the output layer for these tasks had no activations and could be viewed as a simple linear layer. For the cluster classification tasks, on the other hand, the softmax activation function was applied, in order to transform the raw network outputs into probabilities for each node of the layer, corresponding to the number of clusters in the task. These probabilities sum up to one and serve as a common way to have a network choose between different output classes, as is the case with the current cluster classification task.
- **Loss function:** The loss function used for all of the vector reconstruction tasks is the cosine similarity loss function, for reasons mentioned in the semantic representations section of the methodology. Shortly, cosine similarity is one of the most commonly used metrics for assessing vector similarity, because of its scale invariance, outlier insensitivity, and due to its ability to properly deal with high dimensional vectors. For the classification tasks on the other hand, the loss function of choice was categorical cross entropy loss, which is

explicitly designed to handle multi-class classification problems, and constitutes one of the industry standards for these kinds of problems. When utilizing cosine similarity as a loss function, the loss function is represented as the cosine distance, defined as $1 - cos(\theta)$. Utilizing cosine distance, rather than cosine similarity directly, is necessary in order for the model's optimizer to minimize the loss function.

- **Optimizers:** Early model architecture selection, including choosing the number of layers and activation functions was usually done under the Adam optimizer, due to its lower sensitivity to differences in hyperparameter choices, when compared to other optimizers [100]. After these had been determined, however, all models were changed to use the stochastic gradient descent (SGD) algorithm, because it has been shown to better generalize than Adam, which is most likely due to its simplicity [100].

- **Learning rate:** The learning rate dictates the step size in updating the model's weights. Larger learning rate values may inhibit the model from reaching convergence, as it might skip the global minimum in the loss function of the model. Therefore, the learning rate was kept at lower values to ensure reliable training and reaching convergence. learning rates of 0.001 were compared to both higher values, with a maximum of 0.01, as well as smaller values, with a minimum of 0.00001.

- **Regularization techniques:** Different regularization techniques were included in order to curb potential over-fitting problems, although such methods were only implemented in cases where over-fitting actually did occur. These techniques included adding dropout between different model layers, applying weight decay to penalize large model weights, and applying gradient clipping in order to keep the gradients in manageable ranges during training. The regularization strength of each technique varied, with dropout probability ranging from 0 to 25%, weight decay being kept in the smaller range between 0 if not applied and between $1e-5$ and $1e-3$ in cases where it was applied, and finally gradient clipping ranging from 0.5 to 0.25, meaning that the gradients were clipped to be in the range of $\lambda \leq \nabla \leq \lambda$, where λ denotes the clip value.

- **Batch size:** Multi batch training was implemented in various levels in order to significantly speed up training times. Commonly used batches were 1, 8, 16, and 32. Larger batches were not included as these have been shown to negatively impact model generalization capabilities. When a batch size larger than one was used, the sEEG sequences were zero padded to be the same size, since every instance in a single batch must have an equal number of time steps. Padding was performed in a pre-padding fashion, meaning that padding would be applied for the start of the sequence. Furthermore, a masking layer was used in order to make the models ignore the padded values during training.

- **Training iterations:** The number of training iterations, sometimes called the number of training epochs, refers to how many times the model makes a pass through all of the training data. This maximum number of training iterations was determined based on observing the loss curves of each model, and would typically fall between 50 and 500 iterations, with a lower number usually being applied in cases of over-fitting.

An overview of all these different model implementations and hyperparameter choices can be observed in table 8.

Table 8: different sequence model architectures and hyperparameters

Parameter	Values
Layer type	RNN, GRU, LSTM
Number of hidden layers	1, 2, 3, 4
Post sequence fully connected layer	Yes or No
Activation function	Sigmoid, Tanh, ReLU
Output layer dimensionality	5, 50, 100, 768
Hidden units	30-768
Output layer activation	Linear or Softmax
Loss function	Cosine Similarity or Categorical Cross Entropy
Optimizers	SGD or Adam
Learning rate	$1e-5, 1e-4, 1e-3, 0.01$
Dropout	$0-25\%$
Weight decay	0, or $1e-5$, $1e-3$
Gradient clipping	No clipping, or 0.25 - 0.5
Batch size	1, 8, 16, 32
Training iterations	50, 100, 250, 500

3.7.2 Convolutional decoding models

Despite earlier having mentioned that spatial models are unsuitable for the current decoding task, given the sequential and sparsely distributed nature of the sEEG data, convolutional models can still be successfully applied, if one shifts their attention from spatial properties in the data towards temporal properties, through the use of 1D convolutional neural networks, as discussed in the related work section. Therefore, 1 dimensional convolutional neural networks were implemented alongside the traditional sequence models, where the following different architectural choices were taken into account.

- **Number of convolutional layers:** The 1D CNN was exclusively implemented with two convolutional layers, as this proved to serve as a proper balance between complexity and trainability.

- **Number of filters:** the number of filters were stacked hierarchically between the two layers, with the first layer containing either 8 or 16 filters and the second layer containing either 16 or 32 filters.

- **Kernel size:** the kernel size of each filter varied from 3 to 10 time steps, and would always span only the temporal dimension. This means that filters would be applied to each channel separately.
- **Max pooling:** pooling layers were utilized in order to reduce the computational complexity of the models, with pooling kernel sizes varying from 2 to 4 over the temporal dimension.
- **Flattening:** the resulting feature maps resulting from the convolutional layers were flattened using the standard flattening layer or using Global Max Pooling.
- **fully connected layer:** a fully connected layer with hidden units ranging from 32 to 128, depending on the task, were utilized for post-processing, after the flattening layer.
- **Activation function:** convolutional layers exclusively used the ReLU activation function, which is common practice for CNNs. The fully connected layer was implemented with either the ReLu or Tanh activation function

Aside from these architectural differences, the 1 dimensional convolutional models were implemented according to the architecture and hyperparameter choices, defined in the traditional sequence model section, and visualized in figure 8.

3.7.3 Decoding model data input formats

As mentioned before, the input data for the decoding models is represented as multichannel sequence data. In order to properly feed this data into the decoding models, spectrum data corresponding to the onset and offset times of each word were selected. For the motor decoding task, 50ms of additional activity before and after the onset and offset times were selected as well, because earlier work conducted at the UMCU has shown that correlations to the speech envelope were maximal when taking these time lags into account, as can be observed in 26. For the semantic decoding tasks on the other hand, a time lag of 500ms before the onset time of each word was taken into account when selecting sEEG samples, because semantic processing starts before word pronunciation [99]. In doing so, samples for each word of size $[timesteps * channels]$ were selected and were stacked into a 3D matrix after having been zero padded, such that all sequences had the same size. The resulting 3D matrix had a shape of $[words * timesteps * channels]$. The output representations were stacked into a 2d matrix with shape $[words * features]$, where the number of features would differ based on the decoding task of choice, ranging from 5 to 768 features. The resulting pairs of input and output data were randomly split into training and validation data with an 80/20 split, and stratified sampling in case of the classification tasks.

The training and validation sets were subsequently normalized using Min-Max scaling, which is common practice in deep learning and scales all values between 0 and 1. Note that for each channel, values larger or smaller than 2 standard deviations were already clipped, as mentioned earlier, in order to prevent outliers from influencing the normalization. subsequently, the training and test sets were divided into batches and fed into the decoding models. The models were implemented in both PyTorch as well as TensorFlow in order to verify correct implementation. Lastly, if one of the models would achieve above chance level performance, its final performance would be dictated by splitting the data for that model again, but with the use of sklearn's stratified Kfold function, with the number of splits set to the default of 5, thus providing an 80/20 training-test split for each fold. Performance would then be the average performance achieved over these 5 folds, and would represent a more accurate representation of how the models performed. A visual overview of the input and output data applied to the decoding models can be observed in figure 27.

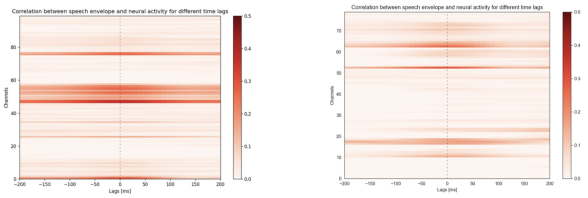


Figure 26: Individual correlation profiles per channel over different time lags, for subjects 1 (a) and 2 (b). The x-axis represents cross-correlation lags (in ms), where negative lags indicate that the neural activity precedes audio. The maximum correlation is achieved using a time lag between -50 to 50 ms.

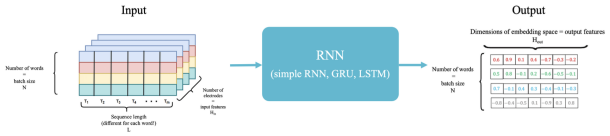


Figure 27: Block diagram of input and output data applied to the decoding model.

3.7.4 Imbalanced classes

As was seen when discussing the syllable classification task, the different classes were not equally distributed, which could impact model performance. Furthermore, the other clustering techniques are likely to contain imbalanced classes as well, as clustering distributions are rarely equal. In order to deal with imbalanced class data, a technique called class weighting was

introduced, which aims to make a model to pay more attention to the minority class by penalizing misclassifications of the minority classes more severely compared to the majority classes [42]. Towards that end, Sklearn's utility package was used to assign class weights, which estimates the class weights by means of the following formula.

$$\frac{samples}{classes * occurences(class)} \tag{5}$$

In this formula, the term samples refers to the total number of samples comprising all of the classes, whereas the classes refer to the number of different classes and the occurrences refer to the instances of the particular class for which one wants to calculate the desired class weight. applying class weighting in combination with the stratified sampling method, mentioned before, allowed for properly handling the unbalanced nature of the different class distribution, and was applied when necessary.

3.8 Multi patient training

One of the benefits of having multiple subjects perform the same task is that their sEEG brain data corresponds to the same semantic representations, as the two subjects in this study mostly did pronounce the same words during the task. The final part of this study's methodology therefore pertains to how data from the two subjects discussed so far can be used in conjunction, with the hope of increasing decoding performance on all the tasks mentioned before. Note that the data from the two different subjects is not used in order to increase the number of available training samples. Doing so would constitute an infeasible task, because the electrodes in each patient were placed in vastly different locations, as could be seen in figure 14, and the activations from the different locations between the two subjects would likely only interfere with each other, as the same semantic representation would then be paired with vastly different electrode activations. Instead, an alternative manner of utilizing the sEEG data from both patients was adopted in which the different electrode channels between two channels would be concatenated, meaning that each semantic representation would be paired with the electrode activations from both patients at the same time. This constituted a far simpler approach than the one discussed in the related work section, but when considering the fact that data from only two subjects was available in this study, structuring the combined electrodes into a grid would still result in a very sparse grid, which would require a lot of interpolation in order to form a dense grid. Therefore, forsaking the spatial extent of the sEEG data and opting for simple concatenation was deemed the only feasible approach towards combining the data these two subjects

A counterargument for adopting this combination strategy posits that increasing the dimensionality of the input data only increases the complexity of the task, without generating additional training data to account for this increased complexity. This marks a valid concern, as the dimensionality of the input space would almost be doubled when concatenating the sEEG channels from both patients. This is where auto encoders yet again proved their usefullness, as they could be applied to reduce the dimensionality of both subjects' sEEG data before concatenation. When reducing the sEEG data from the two subjects to just 30 channels, as discussed earlier, the input dimensionality of the concatenated sEEG data would still contain fewer channels (60) than the original data for either of the two subjects (99 and 79), with the added benefit of having the most important activations from each subject, without most of the noise these signals originally contained.

To that end, the different electrode activations from the two patients were concatenated, after having been compressed, by aligning the timings of each word pronounced by both subjects, and by selecting the corresponding brain data from both patients. This matching was done in order to ensure that each word pair between the two patients was the exact same word, mentioned in the same context, as the BERT model differentiates between different instances of the same word based on the context in which that word was used. Furthermore, since the time it took to pronounce a word, differed between the two subjects, the longest instance of each word was chosen and the shorter instance was zero padded to match the longer instance.

However, due to differences in the words that were actually spoken by each subject during the reading task, subject 1's total quantity of decodable semantic representations did suffer, and was reduced from 1889 to only 1486, which matched the number of words spoken by subject 2. Despite this reduction in decodable semantic representations, it could be interesting to see whether the increase in correlated electrode activations could potentially enrich any semantically encoded content of the input data.

4 Results

This section provides an overview of the results pertaining to both the encoding and decoding tasks discussed in the Methodology section. The sEEG data dimensionality reduction techniques are discussed first, as these are utilized in all other results, after which the motor decoding task is discussed. Afterwards, the decoding results are presented. Note that due to the quantity of model parameters and different architectures, only the best versions of relevant models are discussed for the decoding tasks.

4.1 sEEG data dimensionality reduction performance

The training results of the different auto encoder architectures for subjects 1 and 2 can be observed from figures 9 and 10, respectively. Note that the differences in training times were obtained by letting each model train until over-fitting started to occur. As can be seen from these figures, the non-temporal variational auto encoder consistently presented with the lowest validation loss of all models. These results were within general expectations, given the probabilistic nature of variational auto encoders, which aids in curbing over-fitting, meaning that these models are generally able to achieve higher validation scores. The other architectures generated higher losses, with the LSTM auto encoder being the second best performing architecture, followed by the LSTM variational auto encoder and the regular auto encoder, which presented the overall highest validation losses.

Table 9: Training results for the different auto encoder architectures, subject 1

Architecture	Epochs	loss (MSE)	val. loss (MSE)
AE	25	0.139	0.156
VAE	25	0.013	0.014
LSTM AE	25	0.047	0.092
LSTM VAE	50	0.053	0.102

Table 10: Training results for the different auto encoder architectures, subject 2

Architecture	Epochs	loss (MSE)	val. loss (MSE)
AE	25	0.109	0.104
VAE	50	0.006	0.006
LSTM AE	50	0.041	0.031
LSTM VAE	50	0.052	0.041

Aside from the differences between individual model architectures, it is important to note that the losses for subject 2 were consistently lower than they were for subject 1. However, this does not necessarily imply that the models performed better on the data from subject 2, which became clear after observing the significant correlations ($p < 0.01$) between the compressed sEEG HFB data's channels and the speech envelopes from both subjects, depicted in figures 28-31. Note that the channels pertaining to the reduced input space were not actual channels, but abstract input nodes, which incorporated data from multiple original channels. They are still referred to as channels from here on, for the sake of convenience.

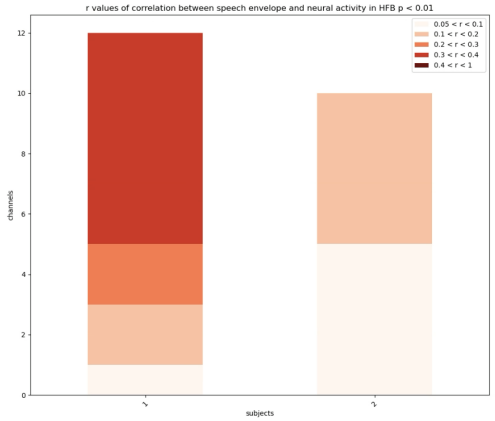


Figure 28: **AE** Channels with significant correlation (p < 0.01) between speech envelope and neural activity in HFB. The color bar represents different values of correlation. Full number of channels sub 1 =30, sub 2 = 30, *sig. channels: sub 1 = 12, sub 2 = 10*

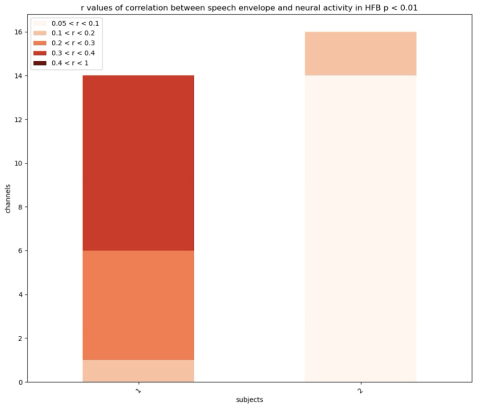


Figure 29: **VAE** Channels with significant correlation (p < 0.01) between speech envelope and neural activity in HFB. The color bar represents different values of correlation. Full number of channels sub 1 =30, sub 2 = 30, *sig. channels: sub 1 = 14, sub 2 = 16*

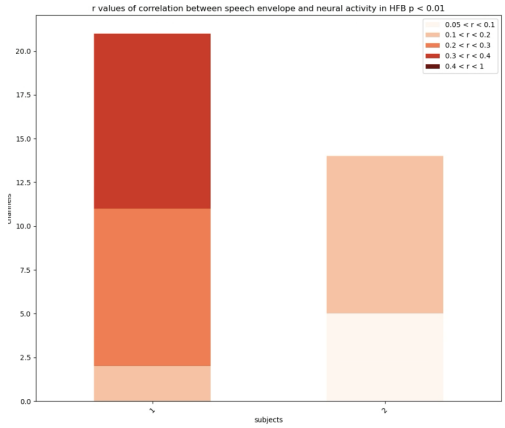


Figure 30: **LSTM-AE** Channels with significant correlation ($p < 0.01$) between speech envelope and neural activity in HFB. The color bar represents different values of correlation. Full number of channels sub 1 =30, sub 2 = 30, *sig. channels: sub 1 = 21, sub 2 = 14*

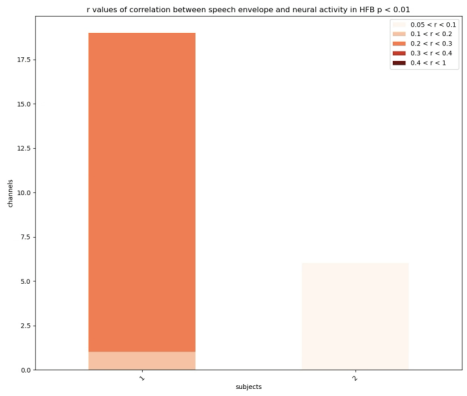


Figure 31: **LSTM-VAE** Channels with significant correlation ($p < 0.01$) between speech envelope and neural activity in HFB. The color bar represents different values of correlation. Full number of channels sub 1 =30, sub 2 = 30, *sig. channels: sub 1 = 19, sub 2 = 6*

The different correlational plots indicate that, in terms of keeping correlations to the speech envelope in tact, the LSTM auto encoder demonstrated the best results (*21 channels sub 1, 14 channels sub 2*), and the non-temporal variational auto encoder was relatively close in terms of the number of correlations (*14 channels sub 1, 16 channels sub 2*). The other two models still demonstrated several correlations, but generally showed fewer correlations, when compared to the other two models. Besides the number of significant channels the strengths of the correlations were also the highest for the LSTM auto encoder, with many of the correlations exceeding the correlations of the original input data, with respect to the speech envelope.

Furthermore, the correlations were consistently higher for subject 1, when compared to subject 2, which was contradictory to the results pertaining to model training. This, combined with the fact that the LSTM auto encoder performed best in terms of correlations to the speech envelope, but not in terms of general model performance, implies that good model performance is not necessarily indicative of an increased ability to keep the encoded data intact. However, it is important to note that there were far fewer strongly correlated channels in subject 2's sEEG data to begin with as could be observed in figure 17, which could explain why the significant correlations in the reduced input space for subject 2 were lower than those of subject 1.

In terms of overall performance, the LSTM auto encoder was able to reduce the sEEG input space, whilst extracting important features from the original input space, with respect to the speech envelope, as demonstrated by the higher correlations with respect to the speech envelope as, depicted in figure 30, when compared to the original correlations as depicted in figure 17. These results demonstrate the efficacy of utilizing an auto encoder approach for dimensionality reduction, as well as feature selection purposes. Lastly, the LSTM auto encoder was chosen as the main auto encoder architecture for the remaining results, due to its aforementioned performance.

4.2 Syllable decoding task performance

Classifying the number of syllables in each task was the sole motor decoding task applied in this study and served as a verification of the usability of the decoding models used later on. More specifically, the motor decoding results served as a test to see whether any information could be decoded from the currently available data, since the current motor task is more easily verifiable and has a more transparent relationship with the speech envelope, whose relationship with the current sEEG HFB data had already been verified to a certain extent. The models used for classification included the RNN, GRU, LSTM and 1D CNN, which were used both with the original sEEG HFB data as well as with the compressed input data generated by the LSTM auto encoder model. Each model was trained on the data from subjects 1 and 2 separately, was well as on the combined data for both subjects, after this data had been compressed to 30 channels per subject. The parameters for the best performing variants of each model can be observed in tables 11 and 12, for the full sEEG data and the compressed sEEG data, respectively, whereas relevant model training curves and accuracy statistics can be observed in figures 32- 36 and table 13, respectively.

Table 11: Motor decoding model parameters original sEEG data (99-channel,79-channel for sub 1 and 2) and combined sEEG data (60 channel, 30 channels from both patients after compression)

parameter	RNN	GRU	LSTM	1D-CNN
layers	2	2	2	2
neurons per layer	50	50	50	-
filters per layer	-	-	-	16, 32
kernel size	-	-	-	6
max pooling size	-	-	-	2
Dense layer size	128	128	128	128
Dense layer activation	ReLU	ReLU	ReLU	ReLU
learning rate	$5e-4$	$5e-4$	$5e-4$	$5e-4$
dropout	0	0	0	15%
weight decay	0	0	0	$1e-5$
clip value	0.5	0.5	0.5	0.5
batch-size	16	16	16	16

Table 12: Motor decoding model parameters compressed sEEG data (30 channels for both patients

parameter	RNN	GRU	LSTM	1D-CNN
layers	2	2	2	2
neurons per layer	20	20	20	-
filters per layer	-	-	-	8, 16
kernel size	-	-	-	6
max pooling size	-	-	-	2
Dense layer size	128	128	128	128
Dense layer activation	ReLU	ReLU	ReLU	ReLU
learning rate	$5e-4$	$5e-4$	$5e-4$	$5e-4$
dropout	0	0	0	15%
weight decay	0	0	0	$1e-5$
clip value	0.5	0.5	0.5	0.5
batch-size	16	16	16	16

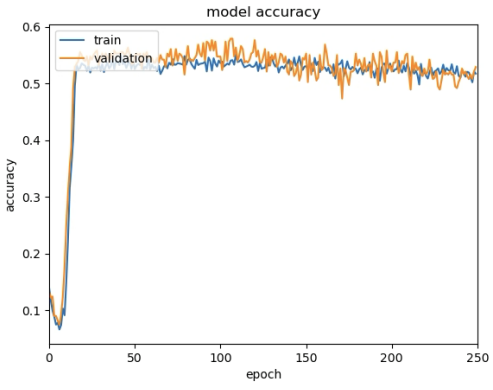


Figure 32: Training and validation accuracy of 1D-CNN, trained on Subject 1's full 99-channel sEEG Data.

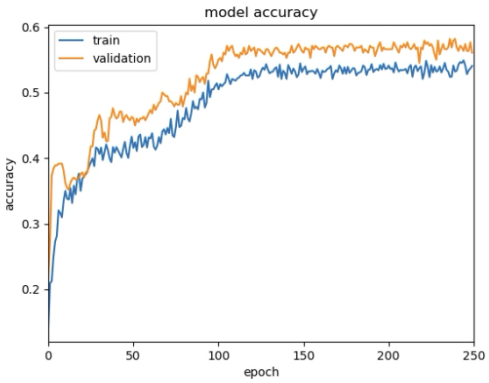


Figure 33: Training and validation accuracy of 1D-CNN, trained on Subject 1's compressed 30-channel sEEG Data.

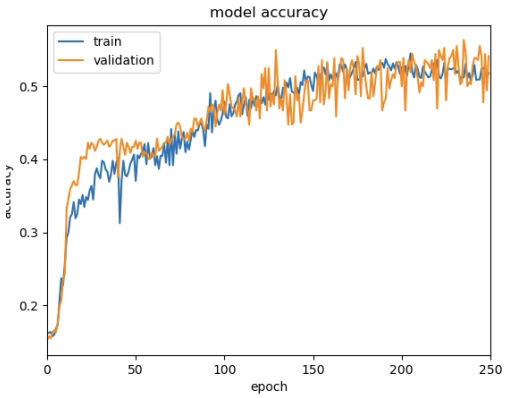


Figure 34: Training and validation accuracy of 1D-CNN, trained on Subject 2's full 79-channel sEEG Data.

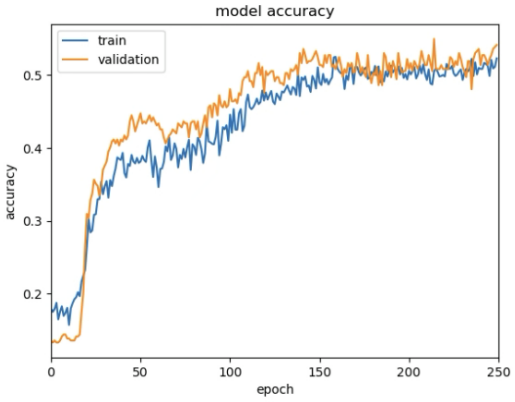


Figure 35: Training and validation accuracy of 1D-CNN, trained on Subject 2's compressed 30-channel sEEG Data.

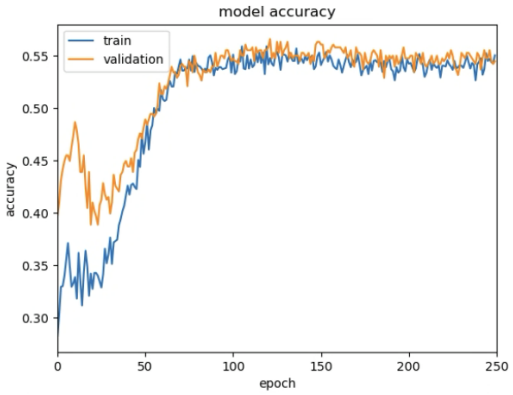


Figure 36: Training and validation accuracy of 1D-CNN, trained on the combined compressed 60-channel sEEG, obtained from the compressed sEEG data for subjects 1 and 2.

Table 13: Motor decoding accuracy for the different training scenarios, *accuracy averaged over 5 folds*.

	accuracy	val. accuracy	*majority class accuracy*
subject 1 full	54.14%	56.35%	*42.24%*
subject 1 compressed	54.07%	57.68%	*42.24%*
subject 2 full	52.31%	53.20%	*42.46%*
subject 2 compressed	51.05%	53.54%	*42.46%*
combined subjects compressed	54.84%	55.56%	*42.46%*

Among the models considered, only the 1D-CNN model consistently achieved performance surpassing the majority class prediction level. Consequently, the depicted accuracy levels and plots for this task exclusively focused on the 1D-CNN models, as can be observed from table 13 and figures 32- 36, respectively. This trend persisted across all decoding scenarios, and attempts to enhance the performance of the traditional sequence models proved unsuccessful, as adjusting parameters did not increase performance past the majority class threshold. Furthermore, increasing model complexity resulted in over-fitting and diminished validation accuracy. Therefore, the 1D-CNN models emerged as the sole relevant candidate for the task of syllable quantity classification.

As can be seen from the training and validation plots, the 1D-CNN performed best under the scenarios which employed auto encoders to compress the input sEEG data, where the model displayed decreased volatility in later stages of training, when compared to the same model trained on the uncompressed input data. More specifically, validation accuracy proved to be at its maximum, when the model was trained on the compressed input sEEG data from subject 1 (*validation accuracy=57.68%*). Note, however, that this level of performance was not substantially higher than compared to the performance levels for the combined data from subjects 1 and 2, (*validation accuracy=55.56*) and the compressed data for subject 2 (*validation accuracy = 53.54*.

Moreover, it's important to observe that the majority of the plots indicate a degree of under-fitting, a phenomenon likely induced by the incorporation of dropout between the convolutional layers. This inclusion was necessary as convolutional filters have a tendency to rapidly over-fit. Without dropout, the training and validation plots would exhibit more smoothing, resulting in closer alignment between validation and model accuracy. However, this would come at the expense of a decrease in overall validation accuracy.

Furthermore, note that performance for the model trained on the combined compressed data from both subjects performed better than the model trained on subject 2's isolated data, but did not perform as well as the model trained on subject 1's isolated data. This discrepancy is likely attributed to the fact that not all electrodes in the original space correlated with the speech envelope. Consequently, adding input data from both subjects, even after compression, introduced some form of noise to the dataset. More importantly, since the speech envelopes between the two subjects did not exactly match as could be observed in figure 16, the data that did correspond to each patient's respective speech envelope could still have been different in the sense that it correlated to a different speech envelope. Adding these data together may therefore have inhibited the model from learning more refined patterns and made it focus on the patterns which were similar between the two patients, causing performance to be roughly in between the individual performance levels of each subject.

Finally, it is crucial to emphasize that, while performance levels surpassed majority class predictions, their robustness remains limited, particularly given the substantial class imbalance evident in tables 6 and 7. The majority of instances were concentrated within two of the five classes. However, it is essential to underscore that the primary objective of this task was to assess the feasibility of extracting meaningful information from the available sEEG data. The earlier-presented results indicate that motor information is indeed encoded to a certain extent in the current dataset. Additionally, the fact that combining data from both patients did not result in a significant performance decline suggests that such aggregation could still be beneficial for decoding performance later on. Notably, the models trained on the compressed data consistently performed at or above the level achieved by models trained on each patient's full data, underscoring the effectiveness of utilizing auto encoders to both enhance and compress the input data. The following subsection delves into whether the ability to decode motor information from sEEG brain data can be extended to decoding semantic information from the same dataset, aligning with the primary objective of this study.

4.3 Word embedding decoding performance

4.3.1 BERT decoding

As mentioned in the related work section, the first and most computationally intensive semantic decoding task consisted of reconstructing the word embeddings, generated through either BERT or Wikipedia2Vec, from the sEEG HFB data obtained from each subject. Towards that end, the decoding model parameters introduced in table 11, were adjusted based on these 2 new tasks, which led to the final testing parameters for each version of this task as indicated in figures 14 and 16. Note that the level of complexity of each model differed between the BERT and Wikipedia2Vec decoding tasks. This was a direct result of the different levels of dimensionality these different embeddding models introduced, with BERT utilizing 768 dimensional vectors, whereas Wikipedia2Vec consisted of 100 dimensional vectors. Because the input data almost always consisted of a lower dimensional input (99 channels subject 1, 79 channels subject 2), the mapping from input to output would almost always be from a lower to a higher dimensional space. Therefore, the number of nodes in the hidden sequence layers and fully connected layers of each model would differ between the BERT and Wikipedia2Vec embedding tasks, in order to facilitate this mapping of input and output, as the relationship between input, model, and output should not have extreme differences in dimensionality. Do note, however, that in order to keep model complexity within reasonable parameters, given the low training sample size, the models for the BERT task could not be made too complex, as this would simply cause over-fitting on what little training data was available.

Table 14: BERT Semantic decoding model parameters

parameter	RNN	GRU	LSTM	1D-CNN
layers	2	2	2	2
neurons per layer	100	100	100	-
filters per layer	-	-	-	16, 32
kernel size	-	-	-	6
max pooling size	-	-	-	2
Dense layer size	192	192	192	192
Dense layer activation	Tanh	Tanh	Tanh	Tanh
learning rate	$5e-4$	$5e-4$	$5e-4$	$5e-4$
dropout	0	0	0	15%
weight decay	0	0	0	0
clip value	0.5	0.5	0.5	0.5
batch-size	16	16	16	16

In terms of the BERT model, the training and validation accuracies of the LSTM architecture can be observed in figure 37. At first it seemed the model achieved decent performance here, with a validation cosine similarity of 0.69. However, analyzing the performance of the same model, when trained under a scenario in which the training samples and labels had been shuffled, depicted in figure 38 demonstrated the opposite. More specifically, it became clear that the model could not achieve a validation cosine similarity higher than the same model when trained on this shuffled data, which also achieved a validation cosine similarity of 0.69. This in turn indicated that model performance did not exceed random chance level performance. Checking the differences between the predictions for each model further confirmed the model's inability to successfully decode any semantic information from the sEEG input data, as the predictions would have an inter-cosine similarity of 0.99, meaning that the average cosine similarity between all predictions was 0.99. Additionally, the cosine similarity between the predictions and the average vector of the BERT embedding space was 0.99 as well, indicating that the model had simply learned to always predict the average BERT vector, which demonstrates no relationship between the current input and output data.

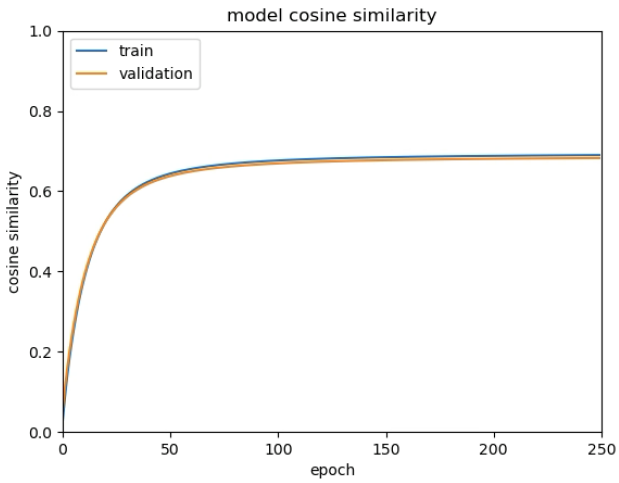


Figure 37: BERT 768D, LSTM training and validation accuracy, trained on Subject 1's full 99-channel sEEG Data.

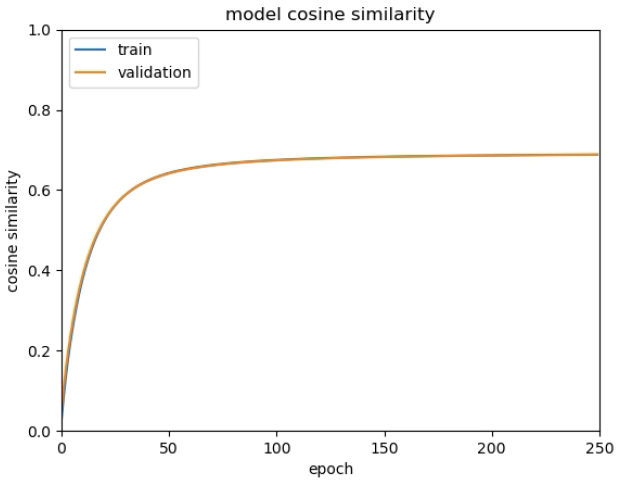


Figure 38: BERT 768D, LSTM training and validation accuracy, trained on Subject 1's full 99-channel *randomized* sEEG Data.

Table 15: BERT 768D, LSTM decoding performance under different training scenarios

	subject 1	subject 2	combined subjects
cosine similarity	0.70	0.69	0.69
val. cosine similarity	0.69	0.69	0.70
control cosine similarity	*0.69*	*0.70*	*0.70*

As can be observed in table 15, model performance did not increase under the different training scenarios created by training on the data from subject 2 and the combined data from subjects 1 and 2. As can be seen from the table, control cosine similarity is annotated for each training scenario individually, rather than for all scenarios, which is the case because subjects 1 and 2 had a different number of samples, which influences the distributions of their respective semantic spaces. Therefore, control cosine similarity was calculated for each scenario individually, by running each model under the randomization constraint, mentioned earlier. In this case, the spaces were not significantly different from one another, as signified by each space having roughly the same control cosine similarity, but these differences could theoretically be significant, which is why thee values were still measured individually, for the sake of completeness. Furthermore, note that the different scenarios did not include the use of the compressed sEEG data from separate subjects, discussed earlier in the results section. These data were not used for decoding the full embedding models, since compressing the input data without compressing the word embeddings would make the difference between input and output dimensionality too great, which would likely not facilitate increased performance. Instead the compressed data is used when the embeddings are also compressed, the performance of which is discussed in a later subsection. Additionally, training curves and performance tables are only shown for the LSTM model, which is due to the fact that none of the models discussed so far, including RNNs, GRUs and 1D-CNNs were able to achieve cosine similarities which were above the randomized control level of cosine similarity. Additionally, changing model parameters for any of the architectures, would not increase performance either, with increased complexity only leading to over-fitting, as discussed earlier, and regularization techniques being unable to reduce over-fitting whilst also improving validation cosine similarity. All in all these results demonstrate that none of the models were able to successfully decode any semantic information, when utilizing the BERT embedding model. These insignificant performance levels could have been caused by a combination of low data availability and BERT's highly complex semantic space.

4.3.2 Wikipedia2Vec decoding

Given the issues mentioned in at the end of the BERT decoding section, performanc might ameliorate when using a more simple embedding model, like the Wikipedia2Vec model. Because this model is less complex with only 100 dimensional embeddings, the resulting models used for decoding could be less complex as well, which might positively modulate performance. The simplified models used for reconstructing the Wikipedia2Vec embeddings can be observed in table 16, and are indeed less complex, with fewer neurons in each sequential layer as well as in the fully connected layer. These models were trained under the same scenarios as the models used for decoding the BERT embeddings. Once again, the training and validation accuracies for the LSTM version of these models can be observed in figures 39 and 40, displaying results for the normal and randomized control scenarios, respectively.

Table 16: Wikipedia2Vec Semantic decoding model parameters

parameter	RNN	GRU	LSTM	1D-CNN
layers	2	2	2	2
neurons per layer	50	50	50	-
filters per layer	-	-	-	8, 16
kernel size	-	-	-	6
max pooling size	-	-	-	2
Dense layer size	50	50	50	50
Dense layer activation	Tanh	Tanh	Tanh	Tanh
learning rate	$5e-4$	$5e-4$	$5e-4$	$5e-4$
dropout	0	0	0	15%
weight decay	0	0	0	0
clip value	0.5	0.5	0.5	0.5
batch-size	16	16	16	16

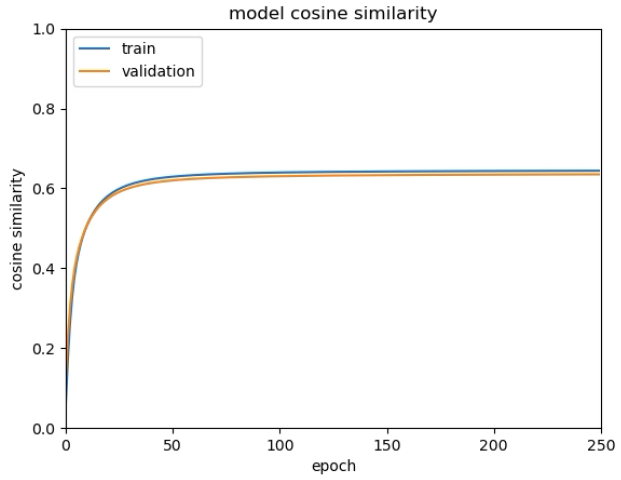


Figure 39: Wikipedia2Vec 100D, LSTM training and validation accuracy, trained on Subject 1's full 99-channel sEEG Data.

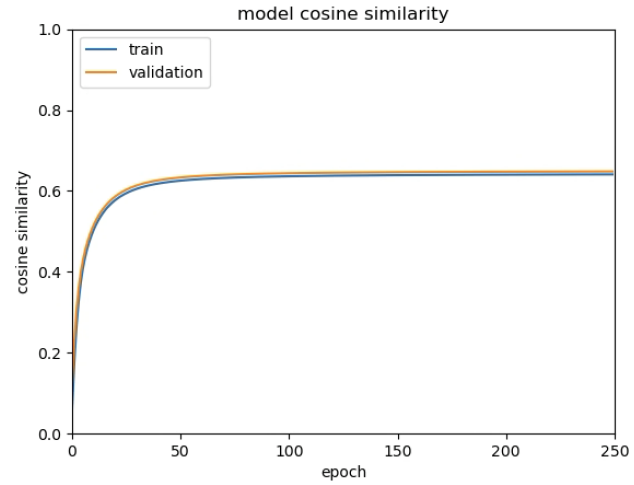


Figure 40: Wkipedia2Vec 100D, LSTM training and validation accuracy, trained on Subject 1's full 99-channel *randomized* sEEG Data.

unfortunately, these results display the exact same trend that could be observed when aiming to reconstruct the BERT embeddings, in the sense that the model was not able to achieve a validation cosine similarity higher than the same model, trained under the scenario of shuffling the training samples and labels. The predictions once again demonstrated an average inter-cosine similarity value of 0.99, and the predictions shared this same similarity with the average vector of the Wikipedia2Vec embeddings space.

Note that the validation cosine similarity for the Wikipedia2Vec embeddings was lower ($cos(\theta) = 0.65$), when compared to the validation cosine similarity achieved when reconstructing the BERT embeddings ($cos(\theta) = 0.69$). This is likely caused by the fact that the BERT embeddings were overall more similar to each other, which resulted in the average vector being more similar to the predictions as well. This, in turn, caused the predictions, which were in essence the average vector of the embedding space, to have a higher cosine similarity to the test set under the BERT model. This, however, had no actual implications for decoding performance, as the performance for both models is almost exactly equal to their randomized control counterparts, meaning that there was no difference in the extent to which the decoding models were able to extract any semantic information from the brain data.

Furthermore, when looking at the validation cosine similarities obtained by training the model on the different input data scenarios (subject 1 vs subject 2 vs combined compressed sEEG data), as shown in table 17, it once more became clear that the model was unable to achieve above chance validation cosine similarity performance, under any of the training scenarios. As was the case for the BERT model, applying different model architectures such as the RNN, GRU or 1D-CNN, changing hyperparameters, or increasing model complexity did not yield performance gains either, suggesting that reconstructing the original BERT and Wikipedia2Vec models represents an infeasible task, given the current input data. The next section, therefore aims to simplify both embedding spaces, through the application of auto encoder architectures, in a similar manner as was utilized when compressing the sEEG data itself.

Table 17: Wikipedia2Vec 100D, LSTM decoding performance under different training scenarios

	subject 1	subject 2	combined subjects
cosine similarity	0.65	0.65	0.65
val. cosine similarity	0.64	0.65	0.64
control cosine similarity	*0.65*	*0.65*	*0.65*

4.4 Word embedding dimensionality reduction performance

4.4.1 Semantic space dimensionality reduction performance

The training results of the different auto encoder architectures for the BERT and Wikipedia2Vec embedding spaces can be observed in table 18. The differences in training times were obtained by letting each model train until over-fitting started to occur. Note that the PCA (principal component analysis) technique was also included in these results, since earlier work conducted at the UMCU has concluded that PCA was unable to keep the semantic structure of the embeddings intact after compression. Therefore, PCA was included as a baseline performance metric, in order to see how the vanilla and variational auto encoder models, discussed in the methodology, would perform in comparison.

Table 18: Performance of dimensionality reduction on BERT (768d) and word2vec (100d) embeddings. All methods reduced the dimensionality of the embeddings to 50 dimensions. *AE = vanilla auto encoder, VAE = variational auto encoder, PCA = prinicipal component analysis.*

Embedding	Epochs	reduction method	(AE / VAE) val. Cosine Similarity	(PCA) Explained variance
BERT	-	PCA	-	40.35%
BERT	400	AE	0.77	-
BERT	500	VAE	0.84	-
Word2vec	-	PCA	-	82.67%
Word2vec	450	AE	0.89	-
Word2vec	500	VAE	0.94	-

Even though these different classes of methods utilize different performance metrics with auto encoders utilizing cosine similarity as their performance metric, and PCA relying on the percentage of variance in the data its principal components can explain, it still seemed that the auto encoders were vastly superior when only looking at these performance metrics. Whereas the auto encoders seemed to generate quite similar vectors, with high levels of cosine similarity for both BERT (val. $cos(AE) = 0.77$, $cos(VAE) = 0.84$) and Wikipedia2Vec (val. $cos(AE) = 0.89$, $cos(VAE) = 0.94$), PCA seemed to especially struggle in compressing the BERT semantic space (*Explained Variance= 40.35%*), whereas it seemed to perform decently when compressing the Wikipedia2Vec semantic space (*Explained Variance= 82.67%*). However, auto encoders aim to keep vectors similar, whereas PCA aims to model capture the differences between the different embeddings, which makes these different performance metrics difficult to interpret, in relation to one another.

Therefore, differences in performance could be compared in a more coherent manner by measuring the cosine similarity between different pairs of word, after compression. More specifically, both word pairs with high cosine similarity in the original semantic space of each embedding model, as well as pairs of words with low similarity in these original spaces, were used to evaluate differences between the different techniques. The results of these comparisons can be observed in tables 20, 21 and 22, 23, for the BERT and Wikipedia2Vec models, respectively. Table 19 also depicts a legend with translations for all the words that were used for comparison.

Table 19: Translations of comparison words

Dutch	English	Dutch	English
zoontje	son	zus	sister
jongetje	boy	baby	baby
meneer	sir	kind	child
achteruit	reverse	idee	idea
cijfers	numbers	middag	afternoon
aansteker	lighter	tijd	time
raam	window	zonsopgang	sunrise

Table 20: BERT cosine similarity between words with **high** cosine similarity

Word 1	Word 2	**cosine sim. original**	cosine sim. AE	cosine sim. VAE	cosine sim. PCA
zoontje	zoontje	**1**	1	1	1
zoontje	zus	**0.68**	0.85	0.77	0.49
zoontje	kind	**0.73**	0.85	0.84	0.62
zoontje	jongetje	**0.79**	0.89	0.83	0.71
zoontje	baby	**0.75**	0.84	0.87	0.77
zoontje	meneer	**0.68**	0.74	0.63	0.37
zus	zus	**1**	1	1	1
zus	kind	**0.60**	0.83	0.66	0.27
zus	jongetje	**0.64**	0.84	0.65	0.35
zus	baby	**0.63**	0.85	0.74	0.48
zus	meneer	**0.63**	0.75	0.62	0.34

Table 21: BERT cosine similarity between words with **low** cosine similarity

Word 1	Word 2	**cosine sim. original**	cosine sim. AE	cosine sim. VAE	cosine sim. PCA
zoontje	achteruit	**0.43**	0.79	0.20	-0.14
zoontje	idee	**0.37**	0.76	0.41	0.03
zoontje	zakje	**0.47**	0.86	0.59	0.19
zoontje	cijfers	**0.44**	0.82	0.51	-0.13
zoontje	middag	**0.36**	0.71	0.31	-0.21
zoontje	grond	**0.35**	0.71	0.27	-0.12
zus	aansteker	**0.45**	0.81	0.49	0.07
zus	tijd	**0.35**	0.65	0.18	-0.27
zus	raam	**0.42**	0.74	0.30	-0.10
zus	zonsopgang	**0.38**	0.73	0.37	-0.14

Table 22: Wikipedia2Vec cosine similarity between words with **high** cosine similarity

Word 1	Word 2	**cosine sim. original**	cosine sim. AE	cosine sim. VAE	cosine sim. PCA
zoontje	zoontje	**1**	1	1	1
zoontje	zus	**0.80**	0.90	0.84	0.75
zoontje	kind	**0.75**	0.92	0.83	0.68
zoontje	jongetje	**0.73**	0.92	0.80	0.63
zoontje	baby	**0.58**	0.83	0.59	0.40
zoontje	meneer	**0.54**	0.76	0.52	0.27
zus	zus	**1**	1	1	1
zus	kind	**0.70**	0.85	0.79	0.63
zus	jongetje	**0.53**	0.77	0.63	0.33
zus	baby	**0.55**	0.71	0.56	0.33
zus	meneer	**0.48**	0.65	0.50	0.21

Table 23: Wikipedia2Vec cosine similarity between words with **low** cosine similarity

Word 1	Word 2	**cosine sim. original**	cosine sim. AE	cosine sim. VAE	cosine sim. PCA
zoontje	achteruit	**0.28**	0.74	0.25	-0.07
zoontje	idee	**0.48**	0.80	0.41	0.06
zoontje	zakje	**0.42**	0.83	0.53	0.08
zoontje	cijfers	**0.23**	0.70	0.23	0.01
zoontje	middag	**0.35**	0.72	0.42	-0.05
zoontje	grond	**0.36**	0.78	0.34	-0.09
zus	aansteker	**0.34**	0.63	0.40	0.01
zus	tijd	**0.43**	0.69	0.38	0.10
zus	raam	**0.36**	0.71	0.49	0.04
zus	zonsopgang	**0.24**	0.67	0.33	-0.17

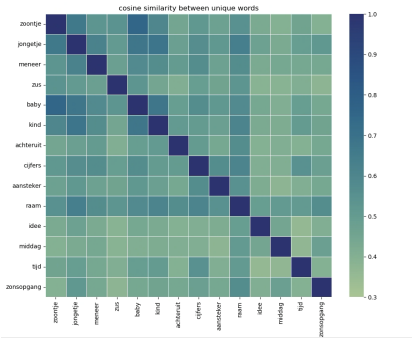
From these results, it became clear that the vanilla auto encoder was not outperforming the PCA technique, in the same way as when simply looking at their respective performance metrics. The major trend observed for the vanilla auto encoder was that it would constantly yield very high cosine similarities, even for the words that were supposed to be dissimilar. For such dissimilar word pairs, the cosine similarity was lower, when compared to the more similar

word pairs, but would still be far higher than the other two techniques. These results are in stark contrast to the relatively high cosine similarity the model achieved during training, but one has to keep in mind that the cosine similarity of all the words in the different embedding models is already quite high, implying that the cosine similarities achieved by the vanilla auto encoder (val. $cos(AE) = 0.77$ BERT, $cos(AE) = 0.89$ WIKipedia2Vec) were not high enough in order to adequately distinguish between the different words.

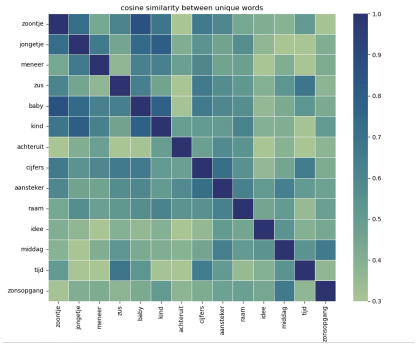
PCA on the other hand proved to be inadequate for the exact opposite reason. From the comparison tables, one can observe that for certain words with high cosine similarity in both original semantic spaces, PCA's reduced embeddings were actually the most similar in terms of cosine similarity (*BERT: zoontje-kind, zoontje-baby, Wikipedia2Vec: zoontje-kind, zus-kind*) . However, there were also several instances in which similar words in the original words lost most of their similarity after applying PCA reduction (*BERT: zus-kind, Wikipedia2Vec: zus-jongetje, zus-baby, zus-meneer*). Furthermore, for all dissimilar words, PCA would simply remove all granularity in the embeddings and make them completely dissimilar. One could argue that making dissimilar words a lot more dissimilar in the reduced space is not detrimental, as doing so can lead to enhanced discrimination between dissimilar words, making it easier for the decoding models to distinguish between semantically dissimilar words. In doing so, however, exaggerating dissimilarity might lead to a loss of nuanced information, making it harder for the model to capture subtle semantic relationships between words. Furthermore, exaggerating these differences can lead to semantic drift, where words that are related in meaning end up being treated as more dissimilar than they actually are. This could lead to a decrease in the model's ability to generalize across related concepts. Therefore it would be best if the reduced embeddings simply capture the semantics of the original space as closely as possible, without introducing unwanted extra similarity or dissimilarity.

lastly, the variational auto encoder seemed to display none of the issues that the other two reduction techniques presented with. It had both the highest model performance of the two embedding models and was also able to keep the reduced embeddings closest to those of both of the original spaces, when compared to PCA and the vanilla auto encoder. In cases where it did not yield the highest cosine similarities between words pairs, when compared to the original spaces, it was still never far off from the target values. Importantly, it was able to keep both the proper similarities for similar as well as dissimilar word pairs intact, meaning that it could adequately discriminate between different semantic meanings, while keeping the nuanced balance of both embedding spaces intact.

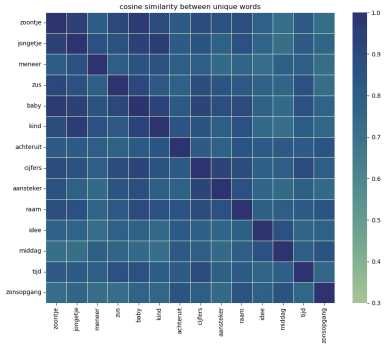
In order to further support these results, heat-maps of the cosine similarities between all the combinations of the words discussed previously, were generated for each reduction method under each embedding model. These heat-maps can be observed in figures 41 and 42, for the BERT and Wikipedia2Vec models, respectively. As can be observed from these heat-maps as well, the variational auto encoder managed to represent the similarities for each word combination in a much more convincing manner, especially for the Wikipedia2Vec embeddings, when compared to the normal auto encoder and PCA. This was to be expected as Wikipedia2Vec is a much simpler model than BERT, but the VAE results for BERT were still impressive, when compared to the other two reduction methods, which were not able to keep the similarities intact at all. Therefore, the variational auto encoder constituted the best performing dimensionality reduction tool for embedding compression, and it represented the only reduction technique that was applied for the decoding tasks mentioned in the following subsection.



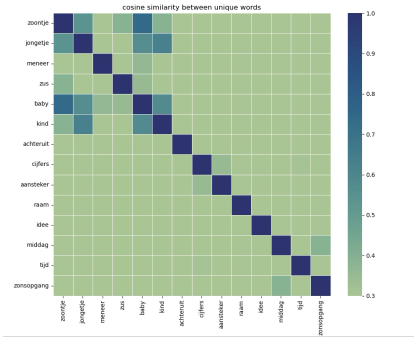
(a) Original space



(b) 50D Variational Auto Encoder

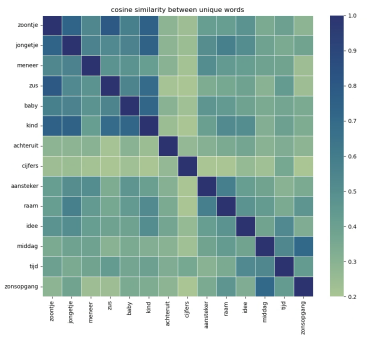


(c) 50D vanilla Auto Encoder

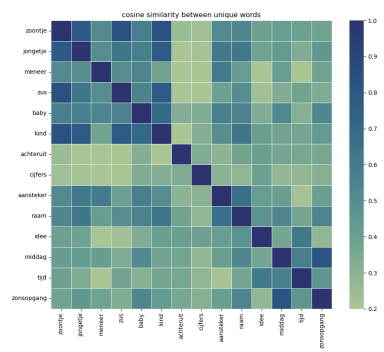


(d) 50D Principal Component Analysis

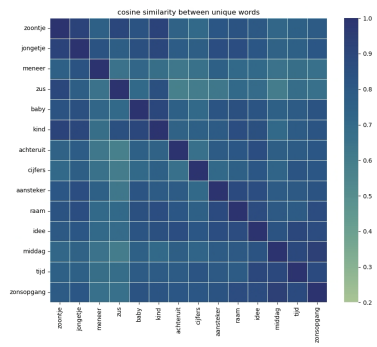
Figure 41: **BERT** heatmap of cosine similarities between differently related words



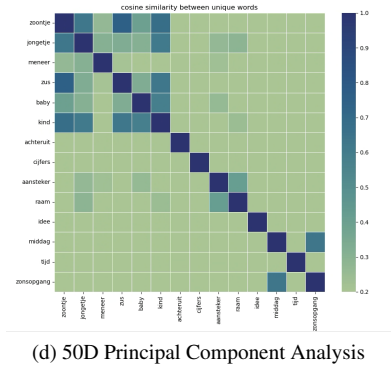
(a) Original space



(b) 50D Variational Auto Encoder



(c) 50D vanilla Auto Encoder



(d) 50D Principal Component Analysis

Figure 42: **Wikpedia2Vec** heatmap of cosine similarities between differently related words

4.4.2 Compressed embedding decoding performance

After applying the variational auto encoder to compress both the BERT and Wikipedia2Vec semantic embedding spaces, models with similar levels of complexity as when decoding the original Wikipedia2Vec embeddings were utilized to reconstruct these new compressed embedding spaces. This level of complexity was chosen because the dimensionality of the Wikipedia2Vec embedding space was already not much more complex than the new reduced spaces and because further reducing complexity did not yield any improvements in performance, likely indicating that further reducing complexity makes the models too simple for the task of semantic vector Reconstruction. Therefore, the same models as described in table 16 were utilized for reconstructing the compressed embeddings of both the BERT and Wikipedia2Vec semantic spaces. Furthermore, since the semantic spaces were now small enough to use the compressed sEEG data from individual subjects, the model training and validation curves, depicted in figures 43, 44, were based on training the LSTM model on the compressed sEEG data from subject 1. However, when looking at these curves and the associated performance table for the model under all training scenarios, it again became clear that the model could still not achieve above chance validation cosine similarity performance, when compared against the control validation cosine similarity values. This was the case for all of the training scenarios, which now included both the full and compressed data from subjects 1 and 2 individually, as well as the combined compressed data from both subjects together, as can be observed in table 24.

4.4.3 BERT compressed semantic space decoding results

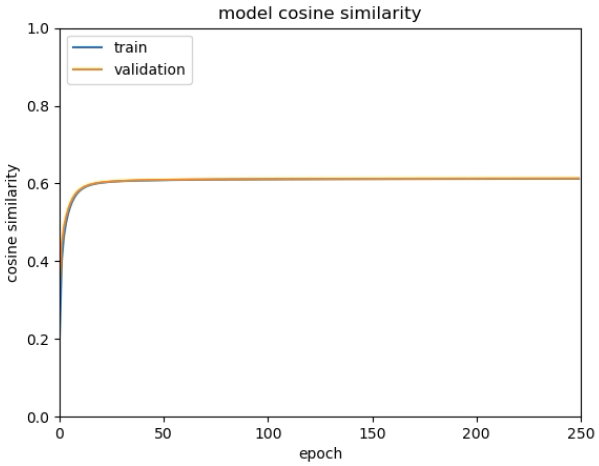


Figure 43: BERT 50D, LSTM training and validation accuracy, trained on Subject 1's reduced 30-channel sEEG Data.

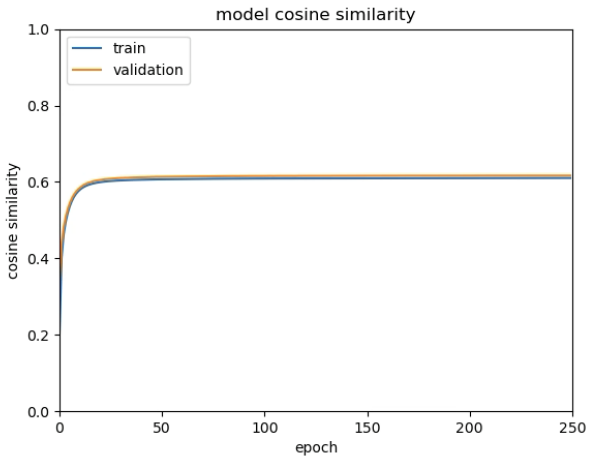


Figure 44: BERT 50D, LSTM training and validation accuracy, trained on Subject 1's reduced 30-channel *randomized* sEEG Data.

Table 24: BERT 50D, LSTM decoding performance under different training scenarios

	cosine similarity	val. cosine similarity	*control cosine similarity*
subject 1 compressed	0.61	0.61	*0.62*
subject 1 full	0.61	0.61	*0.62*
subject 2 compressed	0.61	0.62	*0.62*
subject 2 full	0.61	0.61	*0.62*
combined subjects compressed	0.61	0.62	*0.62*

When looking at the training and validation curves, as well as the performance results, as indicated in table 24, it also became clear that the cosine similarities were lower, when compared to reconstructing the full embedding spaces. This is a direct result of compressing BERT's semantic space, and indicates that the semantic vectors became less similar after compression, which is not strange, as some nuance is always lost when compressing a 768 dimensional space into a 50 dimensions, and differentiating between vectors has to be done slightly more coarsely when having fewer available dimensions. This was not detrimental for the feasibility of the vector reconstruction task, as it simply implied that the baseline similarity level of the embedding spaces was lowered, which was reflected in the randomized control cosine similarity values as well. Nonetheless, none of the models were able to achieve significant performance levels under these circumstances either, and changing model parameters or model complexity had no effect on performance either, in a similar manner as was observed when aiming to reconstruct the uncompressed semantic spaces.

4.4.4 Wikipedia2Vec compressed semantic space decoding results

Model performance for reconstructing the reduced Wikipedia2Vec semantic space followed a similar trend as in the BERT case, with performance again not exceeding the

randomized control threshold, as can be observed in figures 45, and 46, for the unshuffled and shuffled training scenario, trained on the compressed sEEG data from subject 1. This also again held true under all training scenarios, as can be observed in table 25

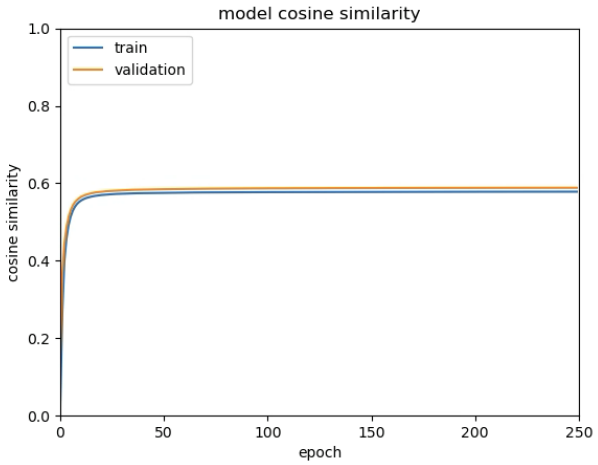


Figure 45: Wikipedia2Vec 50D, LSTM training and validation accuracy, trained on Subject 1's reduced 30-channel sEEG Data.

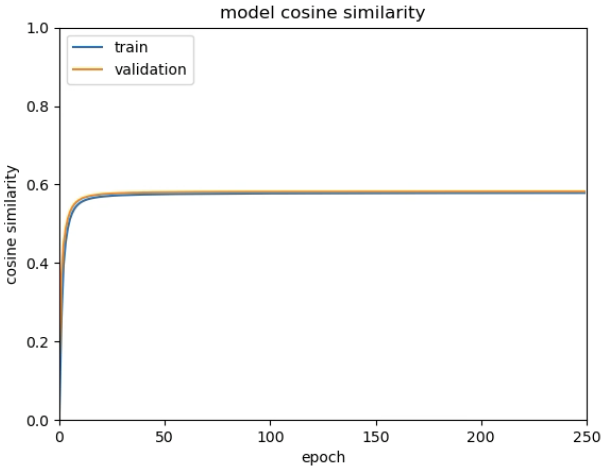


Figure 46: Wikipedia2Vec 50D, LSTM training and validation accuracy, trained on Subject 1's reduced 30-channel *randomized* sEEG Data.

Table 25: Wikipedia2Vec 50D, LSTM decoding performance under different training scenarios

	cosine similarity	val. cosine similarity	*control cosine similarity*
subject 1 compressed	0.58	**0.59**	*0.58*
subject 1 full	0.58	0.58	*0.58*
subject 2 compressed	0.58	0.57	*0.59*
subject 2 full	0.58	0.57	*0.59*
combined subjects compressed	0.58	0.59	*0.59*

Furthermore, in a similar fashion as was observed when reconstructing the reduced BERT semantic vectors, the control, training and validation cosine similarities were all lower than when reconstructing the original Wikipedia2Vec embeddings, although the difference was smaller, which is likely a result of the fact that less nuance was lost when compressing a 100 dimensional space into a 50 dimensional space, when compared to compressing from 768 dimensions to 50 dimensions.

All in all, the results for all of the vector reconstruction tasks, whether the full embedding space or the reduced embedding space was used as the target for reconstruction, have shown no signs of the feasibility of successfully reconstructing such semantic information, when utilizing the sEEG data from subjects 1 and 2. There were also no performance differences between utilizing the full sEEG data from each subject and between utilizing the compressed versions of their data, through the application of LSTM auto encoders. Lastly, combining the compressed data from both subjects did not yield any performance gains either, suggesting that semantic vector reconstruction might simply be a task too complex, for the currently available data. Therefore, a final effort towards reducing the complexity of the semantic information was employed, which is discussed in the following and last section of the Results.

4.5 Clustering results and classification performance

This study's final results section pertains to the usage of clustering methods, which were used in order to further simplify both the BERT and Wikipedia2Vec semantic spaces. Furthermore, Semantic clusters were also obtained through the use of WordNet, as described in the methodology section. in doing so, 5 semantic clusters were obtained from each semantic space, which are discussed in separate sections, both in terms of cluster contents and cluster classification performance. Note that in a similar fashion as in the semantic vector reconstruction tasks, only the best performing iterations of the models under each scenarios are discussed. Lastly, since the task contained the same input and output dimensions as the motor decoding task, the model configurations used for this task were similar to the ones as described in table 11 and 12 for the full input data for subjects 1 and 2 and combined reduced data from both subjects, as well as for the compressed input data for both subjects individually, respectively.

4.5.1 BERT clusters

The different cluster labels, obtained by visually inspecting the contents of each cluster, as well as the distributions of the clustered space, for subject 1, and subject 2 and the combined case, can be observed from tables 26 - 28. Note that the distributions for subject 2 and the combined case are the same due to the fact that the combined case only consisted of words which were present in the data from both subjects, which

restricted the word count, and the resulting distributions to be the same as for subject 2's individual data.

Table 26: BERT Cluster Legend

Cluster	theme
1	Everyday life
2	Communication and Actions
3	Emotions and states of being
4	physical environments and objects
5	Expressions and reactions

Table 27: BERT distribution of different clusters for all word used for decoding, subject 1

Cluster	1	2	3	4	5	total
Count	410	364	348	549	218	1889
percentage	21.70%	19.27%	18.42%	29.07%	11.54%	100%

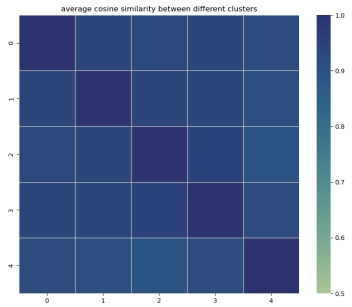
Table 28: BERT distribution of different clusters for all word used for decoding, subject 2 and combined subjects

Cluster	1	2	3	4	5	total
Count	314	283	331	373	185	1486
percentage	21.13%	19.05%	22.27%	25.10%	12.45%	100%

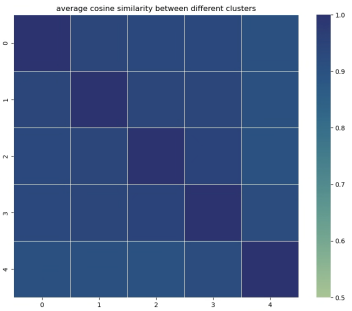
The overarching trends for each clusters are elaborated on below. Note however, that the themes are quite general, which is to be expected when only considering 5 clusters.

- **Cluster 1: Everyday Life**. This cluster mostly contained words related to everyday life, conversations, and various situations one might encounter during the day. It included common words, names, and actions that might occur in regular day-to-day scenarios.
- **Cluster 2: Communication and Actions**. This cluster mostly focused on words related to communication, actions, and interactions. It included verbs and expressions associated with talking, expressing, and doing things.
- **Cluster 3: Emotions**. This cluster appeared to be centered around emotions and various mental states. It included words related to feelings, descriptions of situations, and emotional states.
- **Cluster 4: Physical environments and entities**. This cluster seems to describe settings, scenes, and various objects. It includes words related to locations, physical descriptions, and words denoting physical objects, as well as living entities, such as people or animals.
- **Cluster 5: Expressions and Reactions.** This cluster contained words related to expressions, reactions, and responses. It included words associated with how people react to situations, express themselves, and interact with others.

As can be concluded from the descriptions of each clusters, not only were the clusters very general in terms of their contents, there was also a considerable amount of overlap between the clusters as can be seen in figure 47, which demonstrates that all clusters were significantly similar in terms of their cosine similarities. This lack of separation between different clusters was most likely caused by the fact that differences between words were too nuanced to encompass in 5 broad clusters. Furthermore, not all words followed the topics mentioned in the cluster descriptions above. It is likely that many more clusters would have been needed in order to maintain proper semantic separation, as BERT is highly nuanced in how it differentiates between different words. Doing so, however, was outside the scope of this study, as the goal of the clustering methods was to maintain complexity on par with the motor decoding task.



(a) Subject 1



1pt

(b) Subject 2

Figure 47: **BERT:** heat-map of cosine similarities between the different clusters generated by BERT for subjects 1 and 2.

Classifying these clusters, therefore represented a difficult task to begin with, which is exemplified by the levels of performance as indicated in table 29, as well as the training and validation accuracy curves depicted in figure 48, all pertaining to the LSTM model.

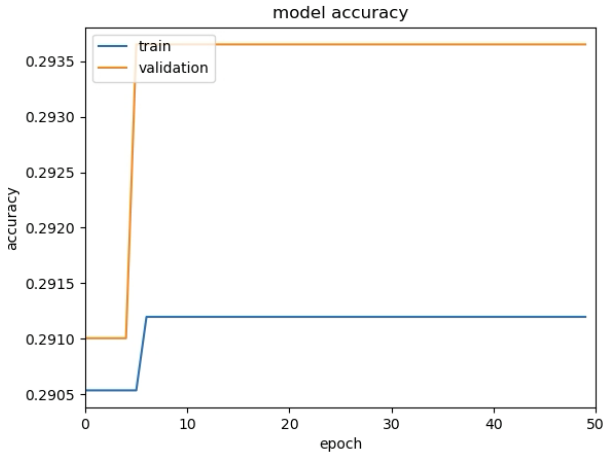


Figure 48: BERT cluster training results, utilizing an LSTM trained on the reduced 30 channel data from subject 1.

Table 29: BERT Clusters classification performance utilizing an LSTM

	accuracy	val. accuracy	*majority class accuracy*
subject 1 compressed	29.12%	29.37%	*29.07%*
subject 1 full	29.19%	29.10%	*29.07%*
subject 2 compressed	25.06%	25.25%	*25.10%*
subject 2 full	25.06%	25.25%	*25.10%*
combined subjects compressed	25.15%	24.91%	*25.10%*

As can be seen from the training and validation curves, the model almost instantly converges to picking the majority class, meaning it has not learned anything about the clusters. This constituted the best case scenario as well, as different versions of the model would simply alternate between picking other classes, without ever even reaching majority class performance. Note that the validation accuracy was ever so slightly above the majority class level, due to the fact that the distributions were slightly different after dividing the data into train and test samples, which is inevitable, even when balancing classes, due to the different total number of samples in the training and validation data after the split. Furthermore, when looking at the performance table, it became clear that the LSTM model was unable to reach above majority level performance under any of the training scenarios. As was observed during the other tasks as well, increasing model complexity or changing model architecture had no effect on model performance other than causing the alternating behavior described above, indicating that decoding the BERT clusters represented an infeasible task, given the current data and the number of clusters that were used.

4.5.2 Wikipedia2Vec clusters

The different cluster labels for the Wikipedia2Vec semantic space as well as their distributions for all scenarios are depicted in tables 30-32.

Table 30: Wikipedia2Vec Cluster Legend

Cluster	theme
1	Physical environment and appearance
2	Emotions and social interactions
3	Quantities and descriptions
4	Actions and events
5	Living entities

Table 31: Wikipedia2Vec distribution of different clusters for all word used for decoding, subject 1

Cluster	1	2	3	4	5	total
Count	366	485	536	283	219	1889
percentage	19.37%	25.68%	28.38%	14.98%	11.59%	100%

Table 32: Wikipedia2Vec distribution of different clusters for all word used for decoding, subject 2 and combined subjects

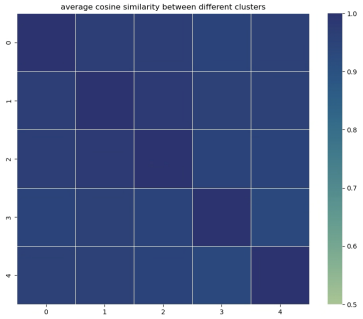
Cluster	1	2	3	4	5	total
Count	287	388	422	222	167	1486
percentage	19.31%	26.11%	28.40%	14.94%	11.24%	100%

The identified clusters could be described as shown below.

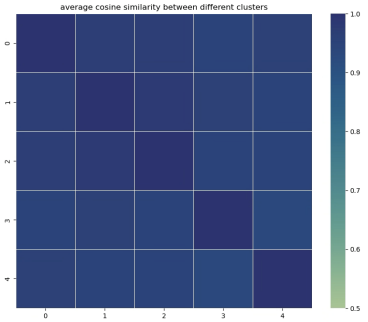
- **Cluster 1: Physical environment and appearance**. This cluster mostly contained words related to physical appearance, clothing, and outdoor settings.
- **Cluster 2: Emotions and social interactions**. The words in this cluster were mostly related to emotions, communication, and social interactions.
- **Cluster 3: Quantities and descriptions**. This cluster consisted of words related to quantity, size, and general descriptors for objects.
- **Cluster 4: Actions and events**. The words in this cluster were mostly related to actions and events, with many verbs and words pertaining to activities.
- **Cluster 5: Living entities** This cluster appeared to mostly contain words related to people, animals, and their descriptions.

When examining the different cluster labels, descriptions and separation heat-maps depicted in figure 49, it became evident that the Wikipedia2Vec clusters exhibited similarly unimpressive levels of separation, featuring equally vague themes among the clusters. Furthermore, similar to the BERT clusters, not all words within each cluster accurately aligned with the identified labels, signifying sub-optimal cluster quality. In parallel, classification performance consistently never exceeded

majority class performance across all training scenarios, as illustrated in Table 33. These outcomes underscore the impracticality of cluster classification for the Wikipedia2Vec clusters, given the existing data.



(a) Subject 1



(b) Subject 2

Figure 49: **Wikipedia2Vec** heat-map of cosine similarities between the different clusters generated by Wikipedia2Vec for subjects 1 and 2.

Table 33: Wikipedia2Vec Clusters classification performance utilizing an LSTM

	accuracy	val. accuracy	*majority class accuracy*
subject 1 compressed	28.39%	28.31%	*28.38%*
subject 1 full	28.39%	28.31%	*28.38%*
subject 2 compressed	28.34%	28.28%	*28.40%*
subject 2 full	28.43%	28.62%	*28.40%*
combined subjects compressed	28.43%	28.62%	*28.40%*

4.5.3 WordNet clusters

The final clusters were obtained utilizing the WordNet similarity matrix, which contained similarity scores for all word combinations. Clustering based on this similarity matrix yielded the semantic clusters, with accompanying labels, depicted in tables 34-36.

Table 34: WordNet Cluster Legend

Cluster	theme
1	Physical environment
2	living entities and their descriptions
3	Time and events
4	Actions and thoughts
5	Emotions and states of mind

Table 35: WordNet distribution of different clusters for all word used for decoding, subject 1

Cluster	1	2	3	4	5	total
Count	204	425	473	310	477	1889
percentage	10.80%	22.50%	25.04%	16.41%	25.25%	100%

Table 36: WordNet distribution of different clusters for all word used for decoding, subject 2 and combined subjects

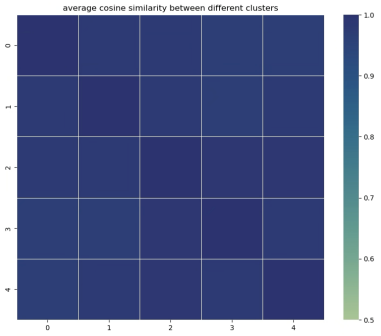
Cluster	1	2	3	4	5	total
Count	170	322	369	241	384	1486
percentage	11.44%	21.67%	24.83%	16.22%	25.84%	100%

The overarching trends for each of these clusters are explained in more detail below.

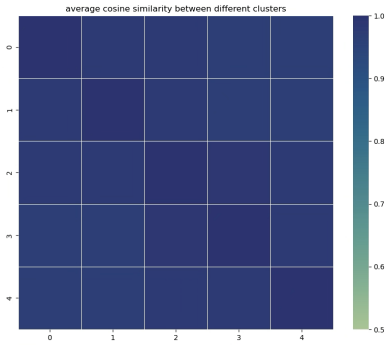
- **Cluster 1: Physical environment**. Cluster 1 mostly contained words describing the physical environment, in a similar fashion as with the Wikipedia2Vec cluster.
- **Cluster 2: Living entities and their descriptions**. This cluster consisted of words relating to living entities and their descriptions, in a similar fashion as was seen with other clustering methods.
- **Cluster 3: Time and events**. Cluster 3 focused on time related concepts, such as the time of day, or day of the week. It also contained more general event descriptions, pertaining to everyday activities.
- **Cluster 4: Actions and thoughts**. Cluster 4 mostly consisted of words pertaining to actions and thoughts and included mostly verbs and adverbs pertaining to verbs.
- **Cluster 5: Emotions and states of mind** Finally, cluster 5 focused on emotions and other words describing one's state of mind, and included literal emotions as well as words pertaining to emotional states with words such as fantastic or cruel, which have a clear emotional charge.

In general, the WordNet Clusters exhibited the weakest separation among the three discussed clustering techniques, as can be observed in figure 50. In a similar fashion as with the BERT and Wikipedia2Vec clusters, none of the clusters

showed proper separation, meaning that WordNet was incapable of grouping the words into the 5 overarching clusters . The results were likely worse in case of WordNet because the concepts it used for comparison had to be manually translated. Furthermore, its lack of contextual information could have made this task even more demanding, as words can become ambiguous without context. This could have been further exacerbated by its lack of part of speech tagging, which normally helps to disambiguate words.



(a) Subject 1



(b) Subject 2

Figure 50: **WordNet** heat-map of cosine similarities between the different clusters generated by WordNet for subjects 1 and 2.

Table 37: WordNet Clusters classification performance utilizing an LSTM

	accuracy	val. accuracy	*majority class accuracy*
subject 1 compressed	25.28%	25.13%	*25.25%*
subject 1 full	24.95%	24.87%	*25.25%*
subject 2 compressed	25.82%	25.59%	*25.84%*
subject 2 full	25.90%	25.93%	*25.84%*
combined subjects compressed	25.82%	25.59%	*25.84%*

The performance figures for the WordNet cluster classification task, as depicted in table 50, once more demonstrate that above majority class level performance proved to be unattainable under any of the training scenarios. As before, changing model parameters or increasing complexity did not increase performance above the respective thresholds of each scenario.

In summary, none of the three discussed clustering methods surpassed random chance performance. These findings confirmed that attempting to extract broader semantic information by clustering words—whether based on contextual meanings (as demonstrated by BERT and Wikipedia2Vec) or on hierarchical semantic meanings derived from WordNet—represented an unattainable task given the available data and the number of clusters that were employed. Furthermore, the clustering task served as the concluding decoding task in the study, leading to the overall conclusion that semantic decoding, using the current techniques and data, was deemed infeasible.

5 Discussion

The main purpose of this section is to discuss the results from the previous section and put them in a broader context with regards to the main research question of this study, pertaining to the feasibility of decoding semantic information from sEEG data. Furthermore, the discussion touches on some of the limitations inherent to the current approaches. Towards that end, the results are discussed in a similar order as they were presented in, meaning that the sEEG compression and motor decoding results are discussed first, after which focus shifts towards the main semantic decoding tasks and the results pertaining to the simplification of these tasks.

5.1 Auto encoder sEEG compression

The findings pertaining to sEEG data compression revealed that preserving information relevant to the speech envelope while reducing input data dimensionality was achievable. The most successful auto encoder architecture, as indicated by the results, was the LSTM auto encoder model. This model exhibited the highest and most consistent correlations with the speech envelope for each subject.

This finding was slightly contradicting as the variational auto encoder should theoretically constitute a more robust reduction architecture. This unexpected outcome may have been attributed to the independent sampling of the auto encoders latent space, because the variational auto encoder would sample its latent space from a Gaussian distribution, created independently for each individual time step of the input sequence. This issue arises from the inherent structure of such auto encoders and wouldn't be problematic if the LSTM auto encoder compressed in both the temporal and spatial dimensions. In such a scenario, the hidden state of the LSTM encoder would only encapsulate the last time step, effectively compressing the entire sequence into one time step. This would allow for independent sampling (reparameterization) of the entire sequence, as each sequence would be represented by just one time step.

However, the current LSTM auto encoder compressed only the spatial extent of the input data to preserve timing relationships between brain data, speech envelope, and word onset/offset times. While this design choice was necessary, it could have counteracted the temporal patterns extracted by the LSTM encoder. Independently sampling (reparameterization) each time step in the sequence likely interfered with the temporal patterns extracted by the LSTM, explaining the superior performance of the LSTM auto encoder compared to its variational counterpart. However, the benefit of adding a probabilistic sampling component to the latent space could still be observed, when comparing the the non-temporal auto encoders, where the variational auto encoder outperformed its static counterpart. Overall, combining LSTMs with variational auto encoders could still be a feasible approach, but only in cases where the auto encoder aims to compress in the temporal dimension as well as the spatial dimension, as training in a sequence to sequence manner as was performed in the current study proved more favourable to the vanilla LSTM auto encoder.

Generally, however, the results still demonstrated the feasibility of reducing the complexity of the brain data, whilst keeping important motor information intact. This finding corroborates with earlier work pertaining to the usage of auto encoders for the purpose of EEG denoising and artifact removal and could prove valuable towards other brain data decoding efforts [12]

5.2 Motor decoding results

As discussed in syllable decoding results section, decoding motor information in the form of syllable quantity classification, proved to be at least somewhat feasible when using the current sEEG data sets. The only model that proved relevant here was shown to be the 1D-CNN model, which at first seems contradictory, given the fact that sequence models are specifically designed to tackle sequences in which the temporal extent of the data is taken into account. However, it is important to note that sequence models, like RNNs GRUs and LSTMs, are designed to capture patterns pertaining to the timings and the temporal hierarchies in which peaks in each channel occur, which may be different for individual words, even if they contain the same number of syllables. In other words, these sequence models might be too sensitive towards the differences in locations of different peaks in different words and the different spacing between each peak.The 1D CNN on the other hand, is designed to capture local patterns and does not take the exact locations of these patterns into account, which may have benefited its performance, due to the aforementioned possible differences in local peak occurrences. Therefore the 1D-CNN's relative simplicity may be one of the main factors for its increased performance, when compared to traditional sequence models.

Evaluating the models' performance on these tasks posed challenges due to the absence of studies directly quantifying the number of syllables in each word from brain data.

In terms of sEEG studies focused on reconstructing motor speech information, results from Kohler and colleagues demonstrated that sEEG data encoded enough information to reconstruct audio wave-forms that appeared similar to the ground truth to the human eye, and contained certain intelligible speech fragments, corroborating with this study's motor decoding results, in the sense that certain motor related aspects of speech could be decoded to a certain extent in both studies [45].

However, given that the results achieved in this study were only marginally superior ($10-15\%$) to majority class performance and considering the dataset's inherent imbalance, with the majority of words falling into just two out of the

five classes, it appears that these findings did not reflect outstanding performance. These results, combined with information pertaining to the placement of the sEEG electrodes, as depicted in figure 14, and the original correlations of the electrodes to the speech envelope, depicted in figure 17, suggests that not all electrodes in the data encoded information relevant to the motor aspects of speech.

This became particularly apparent when cross referencing these outcomes with studies that targeted the direct decoding of syllables, rather than merely classifying their quantity. In such studies, the results demonstrated the capacity to decode specific syllables from a diverse range of classes, surpassing chance performance as outlined in [81]. It is important to note, however, that such results were obtained utilizing EEG brain recordings, which had far greater spatial coverage than the sparse and locally distributed sEEG electrodes that made up the current data sets, which likely did not all encode motor related speech information. Furthermore, as mentioned before, decoding motor information from the current sEEG recordings only served to establish whether any information pertaining to speech was encoded in the data, and the results thus far demonstrated that such information was at least partly encoded in the current sEEG recordings.

5.3 Semantic decoding

Despite motor decoding results demonstrating the feasibility to extract certain motor features of speech to a certain extent, this did not hold true for any of the semantic decoding tasks. None of the models achieved above chance level performance for any of the decoding tasks that were introduced in this study. It would be quite easy to attribute this lack of performance to the lack of samples that were available for training, and this fact did indeed limit the complexity of the models that could be applied towards decoding semantic information from the brain signals. However, even with limited training data available there should have been some noticeable effect in performance, if the problem was solely described by a lack of training data. Therefore, there must have been other issues, pertaining to either the brain data or the semantic representations that limited the decodability of the semantic representations from the sEEG recordings.

In terms of the semantic representations, there were some limitations and issues to be found within the current tasks. First of all the task of reconstructing either the BERT or the Wikipedia2Vec embeddings constituted a highly complex task. For reference, one of the few studies that managed to decode words based on word embeddings was conducted by Goldstein and colleagues, which was based on ECoG data, but their task was simpler than the current task, in the sense that they only aimed to predict the probability of the next word, given a sequence of previous words, rather than aiming to reconstruct the entire word vector without context [27].

Furthermore, the only public work which has managed to decode semantic vectors directly, involved the use of fMRI, meaning it was reliant on having enough temporal separation between the different words, and sampled words from semantically different clusters, such that every word that was used for decoding was significantly different from comparison words, which simplifies the decoding task [68]. It was therefore unlikely that the current study would be able to reconstruct the full word embeddings, given the complexity of the original embeddings and the task in general.

This constituted the main reason behind reducing the dimensionality of the embeddings, as this could greatly simplify the vector reconstruction task. The results of this study demonstrated that utilizing auto encoders for this purpose was highly effective and could outperform traditional linear methods such as PCA, which corroborates with earlier findings that have shown that auto encoders can be effective for word embedding dimensionality reduction [48]. More specifically, the variational auto encoder that was utilized managed to keep most of the semantic similarities between the words intact, and reached high overall similarity to the original embedding spaces of both BERT and Word2Vec. Unfortunately, even after successful compression of the semantic space, reconstructing the semantic vectors still proved to be infeasible, as none of the models reached above chance performance on this simplified task. These results indicate that semantic vector reconstruction is simply a very complex task, even when reducing the complexity of the semantic vectors.

The last effort towards simplifying the semantic space transformed the task from semantic vector reconstruction into semantic category classification, with the same number of classes as were present during the motor decoding task, in which above random chance level results could be obtained. The three methods for generating these semantic categories, involved clustering both the BERT and Wikipedia2Vec embedding spaces into clusters, as well as utilizing a WordNet derived similarity matrix for clustering. None of the clustering methods were able to lead to above chance results in terms of semantic category classification, however. Here it would be easy to conclude that the data likely did not encode any semantic information because above chance results could be obtained for the motor decoding task, but not for a semantic decoding task of the same complexity.

However, the clustering methods used to separate the semantic representations into semantic categories were far from perfect, with none of the clustering methods having particularly clear boundaries between the semantic categories they generated. In terms of the BERT and Word2Vec clusters, it is important to note that the clusters were derived from semantic spaces which already displayed high similarity between the different concepts they encapsulated. This became evident after observing that even a random chance model would generate vectors with relatively high levels of cosine similarity to both the BERT ($cos(\theta) = 0.69$) and Wikipedia2Vec vectors ($cos(\theta) = 0.65$), which in turn implied that the differences

between the different embeddings were quite nuanced. This nuanced similarity structure could also be observed when looking at the performance of the vanilla auto encoder model, which was unable to keep similarity structures intact, despite achieving high overall cosine similarity to the entire word space (cos(BERT)=0.77, cos(WIKI)= 0.89). These differences appeared to be more nuanced between the BERT embeddings, which is likely caused by the fact that BERT's embeddings are contextual, as research has shown that static embeddings can only explain a small portion of the variance in contextual embeddings, which implies that contextual models are likely far more intricate in how they differentiate between different embeddings [21]. Furthermore, since BERT's embedding are context dependent, similarities in the sentence structure of the reading task might have further increased the embedding similarities. Note that the high levels of similarity in the embeddings spaces could have also further inhibited performance on the direct vector reconstruction task.

The WordNet clusters, on the other hand, likely suffered from the lack of POS tagging, mentioned in the methodology, which may have inhibited WordNet to select the proper synset for each word. This in turn could have impacted the similarity between the different words, which in turn could have impacted cluster quality. Furthermore, the fact that every word had to be manually translated into English, due to WordNet's limited span over the Dutch language likely inhibited proper synset selection as well, as translating the words without proper context was likely not without loss of semantic quality.

All methods could have also been influenced by the fact that the words were drawn from a natural speech task in which all words revolved around a similar general theme, namely harry potter, which could explain why most words would display increased similarity between one another. This is especially plausible when keeping in mind that models such as BERT and Word2Vec contain embeddings that are trained to be differentiable in a space containing an enormous variety of words, which implies that words drawn from a much smaller subspace revolving around a similar topic would naturally display higher similarity. It is likely a combination of the factors mentioned above that may have led the words to display high similarities and nuanced differences between one another. It therefore seems unlikely that categorizing such nuanced differences into 5 broad semantic categories would lead to interpretable results. For reference, similar work by Pereira and colleagues also clustered word embeddings into semantic clusters, but used far more clusters (200) to represent the differences between the different words [68]. The current study intentionally utilized much broader clusters (5), in order to induce similar computational complexity as for the motor decoding task, in order to allow for easy comparison, but as has become evident, this might have limited the informativeness of the derived clusters, when paired with automated clustering methods.

Besides limitations pertaining to the semantic representations discussed above, the lack of semantic decoding capabilities are likely also attributed to a lack of semantic decoding in the sEEG data that was utilized for decoding in this study. This notion does not contradict with the evidence regarding the presence of correlations between the sEEG data and the speech envelope, because the speech envelope mostly pertains to motor related areas of the brain, whereas semantic processing occurs through a widely distributed number of brain regions [72] [75]. Therefore, high correlations to the speech envelope did not necessarily indicate that the sEEG data would be relevant for semantic decoding as well. Furthermore, when looking at the distribution of the different sEEG electrodes, as indicated in figure 14, it becomes evident that the electrodes were quite sparsely located and might also penetrated brain areas with a lot of white matter that might not be relevant for semantic decoding. Lastly, when observing the initial correlations to the speech envelope as indicated in figure 17, one can see that the available correlations to the speech envelope were already quite sparse to begin with, and even after amplifying these correlations through the use of the auto encoders, still not every reduced channel demonstrated significant correlations to the speech envelope. This fact, combined with a lack of evidence that correlations to the speech envelope would directly translate to semantic correlation, and the sparse nature of the electrode placements, could suggest that the current sEEG samples from both subjects did not encode a sufficient amount of semantic information for the purpose of decoding. It is therefore likely a combination of limitations pertaining to the brain data as well as the semantic representations themselves that posed constraints on successfully decoding semantic information from sEEG data.

5.4 Multi-patient training

The sparsity of the electrode placements, as mentioned in the previous subsection, was one of the main reasons behind combining the data from both subjects into one unified sample. By reducing the dimensionality of the brain data from both subjects beforehand, increased complexity could be mitigated, as the dimensionality of the combined data was lower than when utilizing the original data from isolated subjects. The hope, therefore, was that decoding performance could be ameliorated as a result of the additional information provided by this combination. As demonstrated by the results, however, this was not the case. For the motor decoding task, results neither really improved, nor deteriorated, whereas there was no measurable effect whatsoever during the decoding tasks. This inhibits the formation of a clear conclusion towards the efficacy of combining data from multiple patients. What can be said, however, is that it is plausible that the combined brain data from both subjects still did not cover enough brain areas in order to sufficiently represent semantic processing because the combined data only spanned the electrodes from 2 patients.

Furthermore, the method of combination, which consisted of simply concatenating the channels from both patients, was likely too simplistic, in the sense that simple concatenation does not take the location of these different electrodes into account. Note that this was not possible with only two subjects, as the electrodes would still be too sparsely distributed to form a grid, from which spatial relationships could be ascertained. Therefore, no clear conclusions can be made at this time pertaining to the efficacy of combining brain data from multiple subjects, as it is likely that a threshold needs to be exceeded before such combined data could encompass enough brain areas to sufficiently represent semantic processes in the brain.

6 Future Work

Based on the results and discussion of this study, directions towards future work could focus on both the brain data itself, the decoding models, as well as the output representations. In terms of the brain data, an obvious suggestion would be to utilize the sEEG data from different patients in addition to the currently used data. Data from other patients was available but had not been pre-processed to the same extent as the currently used data, which is why they were left unused for the purposes of this study, as this additional pre-processing would have been beyond the time constraints imposed on this project. the addition of data from multiple subjects could potentially enhance the coverage of the electrodes to an extent where they would be able to sufficiently represent semantic processing in the brain. This has implications for the suitability of spatiotemporal decoding models as well as models like convolutional neural networks or convolutional LSTMs could potentially be applied if the electrodes are abundant enough to form a high density 3d grid. Naturally, the inclusion of many more electrodes would increase complexity too, but this is where convolutional auto encoders could successfully be applied in order to reduce the dimensionality of such a grid. It could, therefore, be interesting to see whether semantic decoding performance could be ameliorated under such conditions. In addition to using more sEEG data, it is worth noting that ECoG data for subjects who performed the same reading task is also available. since ECoG already represents a structured grid, attempting semantic decoding whilst utilizing this data could also be worth considering, although it is currently unclear whether ECoG's superficial coverage and lack of penetration into deeper brain areas could sufficiently record semantic processing in the brain.

Furthermore, one of the limitations of considering the brain data's correlations to the speech envelope was the fact that it was unlikely that such correlations would translate into correlations pertaining to semantic processing too. Therefore, a direct method for ascertaining the correlations between the brain data and semantic processing would be a desirable tool for future studies. recent work by Kohler and colleagues has already proposed what form such a tool might assume, as they demonstrated that Word2Vec embeddings could be utilized to reconstruct neural activity from sEEG electrodes during speech production. More specifically, they were able to show which electrodes significantly encoded semantic information by training a linear regression predictor for each electrode and correlating the results to each time point relative to the word presentation [33]. A similar could be adopted in future work in order to ascertain which electrodes hold potential for semantic decoding.

Lastly, in terms of the semantic representations, it has become clear that direct semantic vector reconstruction is likely too complex given the limited sample sizes and available data and this likely holds true when increasing the number of available electrodes too. Whether such a task becomes achievable after dimensionality reduction through auto encoders is currently unclear, so future work ought to focus on tasks with lower complexity instead. The semantic categories utilized in this study were limited too, in the sense that automatic semantic clustering in combination with aiming to generate a few general clusters was not successful. Therefore, future work could instead aim to increase the number of clusters, in order to better separate the different concepts from one another, in a similar fashion as was done in work by Pereira and colleagues [68]. Even with many clusters, such a task would still have a lower complexity than direct vector reconstruction and could therefore yield more favorable results. Furthermore, manual clustering words into well defined categories could also hold potential as doing so would make the learning space more differentiable. Lastly, if vector reconstruction could successfully be applied, it is likely that the best results could be obtained when using the Word2Vec model as this model was yielded more differentiable word embeddings with a lower dimensionality, which in turn also increased the efficacy of efforts towards further reducing its complexity, as became evident by the auto encoder results for Wikipedia2Vec.

7 Conclusion

The main purpose of this study was to investigate the feasibility of decoding semantic representations, in the forms of semantic categories and semantic vectors, from sEEG data. Towards that end, both static and dynamic embedding models, in the forms of Word2Vec and BERT, were applied to generate word embeddings which encapsulated the semantic meaning of words pronounced by two subjects during a natural language reading task. Such models, in addition to the WordNet lexical database, were also utilized in order to generate broad semantic categories, which served as a simplification of the original decoding task of reconstructing the semantic vector representations. The semantic vectors were compressed utilizing auto encoders, in order to reduce the complexity of directly decoding such vectors. In a similar manner, the sEEG brain data itself was compressed utilizing temporal

auto encoders, which served to reduce the complexity of such signals as well as to make features relevant to speech more salient. Furthermore, in order to ascertain whether the data held any decodable information with regards to speech in the first place, a preliminary motor decoding task, in the form of syllable quantity classification was conducted. Finally, the data from both subjects was combined, after compression, in an effort to mitigate sEEG's sparse coverage of the brain.

The results of this study have demonstrated that auto encoders can successfully be applied to reduce the complexity of both the sEEG data as well as the semantic BERT and Word2Vec embeddings, while keeping important information intact. Furthermore, the results pertaining to motor decoding indicated that motor information could, to a certain extent, be decoded from the data. Despite these favorable results, semantic decoding proved infeasible, with predictions never exceeding above chance level performance on any of the semantic decoding tasks, even when combining data from multiple subjects. This lack of decoding performance could be attributed to a combination of limitations pertaining to both the sEEG data as well as the semantic representations in each decoding task. On one hand, the semantic vector reconstruction tasks remained complex, even after reducing the complexity of the embeddings. Furthermore, automatically clustering the semantic representations into broad semantic categories proved unable to maintain separability between different semantic concepts. However, it is also plausible that the sEEG data did not sufficiently encode semantic processing in the brain for the purpose semantic decoding, even after combining data from multiple subjects and it was unclear whether the correlations to the speech envelope, used to asses the quality of the sEEG data, were relevant for semantic decoding too.

Based on these findings, future work could focus on incorporating more data from other sEEG subjects, or ECoG subjects, which could allow for a better representation of semantic processes in the brain. Furthermore, the relevancy of individual electrodes towards semantic processes should be evaluated, rather than focusing on correlations to the motor related aspects of speech, as this could yield important insights into which electrodes would be suitable for decoding semantic information. Lastly, future work should focus on using manually generated semantic categories, which focus on semantic separability. Alternatively, automated clustering could still be used but would require far more clusters in order to appropriately separate different semantic concepts. All in all, this study was one of the first attempts towards decoding semantic information from sEEG data and could spark new research in the area of sEEG semantic decoding.

Bibliography

[1] Hassan Akbari, Bahar Khalighinejad, Jose L Herrero, Ashesh D Mehta, and Nima Mesgarani. Towards reconstructing intelligible speech from the human auditory cortex. *Scientific reports*, 9(1):874, 2019.

[2] Jasem Almotiri, Khaled Elleithy, and Abdelrahman Elleithy. Comparison of autoencoder and principal component analysis followed by neural network for e-learning using handwritten recognition. In *2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, pages 1–5. IEEE, 2017.

[3] Miguel Angrick, Christian Herff, Emily Mugler, Matthew C Tate, Marc W Slutzky, Dean J Krusienski, and Tanja Schultz. Speech synthesis from ecog using densely connected 3d convolutional neural networks. *Journal of neural engineering*, 16(3):036019, 2019.

[4] Maria Clemencia Ortiz Barajas, Ramón Guevara, and Judit Gervain. The origins and development of speech envelope tracking during the first months of life. *Developmental cognitive neuroscience*, 48:100915, 2021.

[5] Julia Berezutskaya, Clarissa Baratin, Zachary V Freudenburg, and Nicolas F Ramsey. High-density intracranial recordings reveal a distinct site in anterior dorsal precentral cortex that tracks perceived speech. *Human brain mapping*, 41(16):4587–4609, 2020.

[6] Nima Bigdely-Shamlo, Tim Mullen, Christian Kothe, Kyung-Min Su, and Kay A Robbins. The prep pipeline: standardized preprocessing for large-scale eeg analysis. *Frontiers in neuroinformatics*, 9:16, 2015.

[7] Paul Boersma. Praat: doing phonetics by computer [computer program]. *http://www. praat. org/*, 2011.

[8] Jonathan S Brumberg, E Joe Wright, Dinal S Andreasen, Frank H Guenther, and Philip R Kennedy. Classification of intended phoneme production from chronic intracortical microelectrode recordings in speech motor cortex. *Frontiers in neuroscience*, 5: 7880, 2011.

[9] Shreya Chakrabarti, Hilary M Sandberg, Jonathan S Brumberg, and Dean J Krusienski. Progress in speech decoding from the electrocorticogram. *Biomedical Engineering Letters*, 5:10–21, 2015.

[10] Emmanuele Chersoni, Enrico Santus, Chu-Ren Huang, and Alessandro Lenci. Decoding word embeddings with brain-based semantic features. *Computational Linguistics*, 47(3):663–698, 2021.

[11] Taishih Chi, Powen Ru, and Shihab A Shamma. Multiresolution spectrotemporal analysis of complex sounds. *The Journal of the Acoustical Society of America*, 118(2):887–906, 2005.

[12] Arjun Vinayak Chikkankod and Luca Longo. On the dimensionality and utility of convolutional autoencoder's latent space trained with topology-preserving spectral eeg head-maps. *Machine Learning and Knowledge Extraction*, 4(4):1042–1064, 2022.

[13] Stephanie Chua and Narayanan Kulathuramaiyer. Semantic feature selection using wordnet. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI'04)*, pages 166–172. IEEE, 2004.

[14] Debadatta Dash, Paul Ferrari, Angel W Hernandez-Mulero, Daragh Heitzman, Sara G Austin, and Jun Wang. Neural speech decoding for amyotrophic lateral sclerosis. In *INTERSPEECH*, pages 2782–2786, 2020.

[15] Debadatta Dash, Paul Ferrari, and Jun Wang. Decoding imagined and spoken phrases from non-invasive neural (meg) signals. *Frontiers in neuroscience*, 14:290, 2020.

[16] Debadatta Dash, Paul Ferrari, and Jun Wang. Role of brainwaves in neural speech decoding. In *2020 28th European Signal Processing Conference (EUSIPCO)*, pages 1357–1361. IEEE, 2021.

[17] Wendy A de Heer, Alexander G Huth, Thomas L Griffiths, Jack L Gallant, and Frédéric E Theunissen. The hierarchical cortical organization of human speech processing. *Journal of Neuroscience*, 37(27):6539–6557, 2017.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[19] Jae Seok Do, Akeem Bayo Kareem, and Jang-Wook Hur. Lstm-autoencoder for vibration anomaly detection in vertical carousel storage and retrieval system (vcsrs). *Sensors*, 23(2), 2023. ISSN 1424-8220. URL https://www.mdpi.com/1424-8220/23/2/1009.

[20] Jiaju Du, Fanchao Qi, and Maosong Sun. Using bert for word sense disambiguation. *arXiv preprint arXiv:1909.08358*, 2019.

[21] Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*, 2019.

[22] Quentin Fournier and Daniel Aloise. Empirical comparison between autoencoders and traditional dimensionality reduction methods. In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 211–214. IEEE, 2019.

[23] Saskia L Frisby, Ajay D Halai, Christopher R Cox, Matthew A Lambon Ralph, and Timothy T Rogers. Decoding semantic representations in mind and brain. *Trends in Cognitive Sciences*, 2023.

[24] Jeroen Geuze, Jason Farquhar, and Peter Desain. Towards a communication brain computer interface based on semantic relations. *PLoS One*, 9(2):e87511, 2014.

[25] Joydeep Ghosh and Alexander Strehl. Similarity-based text clustering: A comparative study. In *Grouping Multidimensional Data: Recent Advances in Clustering*, pages 73–97. Springer, 2006.

[26] Gary H Glover. Overview of functional magnetic resonance imaging. *Neurosurgery Clinics*, 22(2):133–139, 2011.

[27] Ariel Goldstein, Zaid Zada, Eliav Buchnik, Mariano Schain, Amy Price, Bobbi Aubrey, Samuel A Nastase, Amir Feder, Dotan Emanuel, Alon Cohen, et al. Shared computational principles for language processing in humans and deep language models. *Nature neuroscience*, 25(3):369–380, 2022.

[28] Frank H Guenther, Jonathan S Brumberg, E Joseph Wright, Alfonso Nieto-Castanon, Jason A Tourville, Mikhail Panko, Robert Law, Steven A Siebert, Jess L Bartels, Dinal S Andreasen, et al. A wireless brain-machine interface for real-time speech synthesis. *PloS one*, 4(12):e8218, 2009.

[29] Zengzhi Guo and Fei Chen. Decoding articulation motor imagery using early connectivity information in the motor cortex: A functional near-infrared spectroscopy study. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2022.

[30] Cong Han, James O'Sullivan, Yi Luo, Jose Herrero, Ashesh D Mehta, and Nima Mesgarani. Speaker-independent auditory attention decoding without access to clean speech sources. *Science advances*, 5(5): eaav6134, 2019.

[31] Christian Herff, Dominic Heger, Adriana De Pesters, Dominic Telaar, Peter Brunner, Gerwin Schalk, and Tanja Schultz. Brain-to-text: decoding spoken phrases from phone representations in the brain. *Frontiers in neuroscience*, 9:217, 2015.

[32] Christian Herff, Dean J Krusienski, and Pieter Kubben. The potential of stereotactic-eeg for brain-computer interfaces: current progress and future directions. *Frontiers in neuroscience*, 14:123, 2020.

[33] Christian Herff, Maxime Verwoert, Joaquín Amigó-Vega, and Maarten Ottenhoff. Semantic representations of speech production in intracranial eeg. pages 4764–4769, 10 2023. doi: 10.1109/SMC53992.2023.10394550.

[34] Nora Hollenstein, Cedric Renggli, Benjamin Glaus, Maria Barrett, Marius Troendle, Nicolas Langer, and Ce Zhang. Decoding eeg brain activity for multi-modal natural language processing. *Frontiers in Human Neuroscience*, page 378, 2021.

[35] Alexander G Huth, Wendy A De Heer, Thomas L Griffiths, Frédéric E Theunissen, and Jack L Gallant. Natural speech reveals the semantic maps that tile human cerebral cortex. *Nature*, 532(7600):453–458, 2016.

[36] Koji Iida and Hiroshi Otsubo. Stereoelectroencephalography: indication and efficacy. *Neurologia medico-chirurgica*, 57(8):375–385, 2017.

[37] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

[38] Daniel Jurafsky and James H Martin. Speech and language processing (draft of december 29, 2021), 2021.

[39] Devinder Kaur, Shama Naz Islam, and Md Apel Mahmud. A variational autoencoder-based dimensionality reduction technique for generation forecasting in cyber-physical smart grids. In *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2021.

[40] Matthew C Kiernan, Steve Vucic, Benjamin C Cheah, Martin R Turner, Andrew Eisen, Orla Hardiman, James R Burrell, and Margaret C Zoing. Amyotrophic lateral sclerosis. *The lancet*, 377(9769):942–955, 2011.

[41] Joshua Y Kim, Chunfeng Liu, Rafael A Calvo, Kathryn McCabe, Silas CR Taylor, Björn W Schuller, and Kaihang Wu. A comparison of online automatic speech recognition systems and the nonverbal responses to unintelligible speech. *arXiv preprint arXiv:1904.12403*, 2019.

[42] Gary King and Langche Zeng. Logistic regression in rare events data. *Political analysis*, 9(2):137–163, 2001.

[43] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[44] M Kemal Kıymık, İnan Güler, Alper Dizibüyük, and Mehmet Akın. Comparison of stft and wavelet transform methods in determining epileptic seizure activity in eeg signals for real-time application. *Computers in biology and medicine*, 35(7):603–616, 2005.

[45] Jonas Kohler, Maarten C Ottenhoff, Sophocles Goulis, Miguel Angrick, Albert J Colon, Louis Wagner, Simon Tousseyn, Pieter L Kubben, and Christian Herff. Synthesizing speech from intracranial depth electrodes using an encoder-decoder framework. *arXiv preprint arXiv:2111.01457*, 2021.

[46] Jan Kubanek, Peter Brunner, Aysegul Gunduz, David Poeppel, and Gerwin Schalk. The tracking of speech envelope in the human cortex. *PloS one*, 8(1):e53398, 2013.

[47] Sreejan Kumar, Theodore R Sumers, Takateru Yamakoshi, Ariel Goldstein, Uri Hasson, Kenneth A Norman, Thomas L Griffiths, Robert D Hawkins, and Samuel A Nastase. Reconstructing the cascade of language processing in the brain using the internal computations of a transformer-based language model. *BioRxiv*, pages 2022–06, 2022.

[48] Md Tahmid Rahman Laskar, Cheng Chen, Jonathan Johnston, Xue-Yong Fu, Shashi Bhushan TN, and Simon Corston-Oliver. An auto encoder-based dimensionality reduction technique for efficient entity linking in business phone conversations. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3363–3367, 2022.

[49] Amanda LeBel, Lauren Wagner, Shailee Jain, Aneesh Adhikari-Desai, Bhavin Gupta, Allyson Morgenthal, Jerry Tang, Lixiang Xu, and Alexander G Huth. A natural language fmri dataset for voxelwise encoding models. *Scientific Data*, 10(1):555, 2023.

[50] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*, 2015.

[51] Xuan Li, Tao Zhang, Xin Zhao, and Zhengming Yi. Guided autoencoder for dimensionality reduction of pedestrian features. *Applied Intelligence*, 50:4557–4567, 2020.

[52] Shiyu Luo, Qinwan Rabbani, and Nathan E Crone. Brain-computer interface: applications to speech decoding and synthesis to augment communication. *Neurotherapeutics*, 19(1):263–273, 2022.

[53] Mohammad Sultan Mahmud, Joshua Zhexue Huang, and Xianghua Fu. Variational autoencoder-based dimensionality reduction for high-dimensional small-sample data classification. *International Journal of Computational Intelligence and Applications*, 19(01): 2050002, 2020.

[54] Stéphanie Martin, Peter Brunner, Chris Holdgraf, Hans-Jochen Heinze, Nathan E Crone, Jochem Rieger, Gerwin Schalk, Robert T Knight, and Brian N Pasley. Decoding spectrotemporal features of overt and covert speech from the human cortex. *Frontiers in neuroengineering*, 7:14, 2014.

[55] Scott McDonald and Michael Ramscar. Testing the distributioanl hypothesis: The influence of context on judgements of semantic similarity. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 23, 2001.

[56] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[57] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[58] Mohammad Jalilpour Monesi, Bernd Accou, Jair Montoya-Martinez, Tom Francart, and Hugo Van Hamme. An lstm based architecture to relate speech stimulus to eeg. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 941–945. IEEE, 2020.

[59] David A Moses, Sean L Metzger, Jessie R Liu, Gopala K Anumanchipalli, Joseph G Makin, Pengfei F Sun, Josh Chartier, Maximilian E Dougherty, Patricia M Liu, Gary M Abrams, et al. Neuroprosthesis for decoding speech in a paralyzed person with anarthria. *New England Journal of Medicine*, 385(3):217–227, 2021.

[60] Brian Murphy, Massimo Poesio, Francesca Bovolo, Lorenzo Bruzzone, Michele Dalponte, and Heba Lakany. Eeg decoding of semantic category reveals distributed representations for single concepts. *Brain and language*, 117(1):12–22, 2011.

[61] Elliot Murphy, Kiefer J Forseth, Cristian Donos, Kathryn M Snyder, Patrick S Rollo, and Nitin Tandon. The spatiotemporal dynamics of semantic integration in the human brain. *Nature Communications*, 14(1): 6336, 2023.

[62] Youngmin Na, Inyong Choi, Dong Pyo Jang, Joong Koo Kang, and Jihwan Woo. Semantic-hierarchical model improves classification of spoken-word evoked electrocorticography. *Journal of neuroscience methods*, 311:253–258, 2019.

[63] Salifu Nanga, Ahmed Tijani Bawah, Benjamin Ansah Acquaye, Mac-Issaka Billa, Francis Delali Baeta, Nii Afotey Odai, Samuel Kwaku Obeng, and Ampem Darko Nsiah. Review of dimension reduction methods. *Journal of Data Analysis and Information Processing*, 9(3):189–231, 2021.

[64] Inc. Neurobehavioral Systems. Neurobs presentation, 2024. URL https://www.neurobs.com/menu_presentation/menu_features/features_overview.

[65] Subba Reddy Oota, Naresh Manwani, and Raju S Bapi. fmri semantic category decoding using linguistic encoding of word embeddings. In *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13–16, 2018, Proceedings, Part III 25*, pages 3–15. Springer, 2018.

[66] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[67] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[68] Francisco Pereira, Bin Lou, Brianna Pritchett, Samuel Ritter, Samuel J Gershman, Nancy Kanwisher, Matthew Botvinick, and Evelina Fedorenko. Toward a universal decoder of linguistic meaning from brain activation. *Nature communications*, 9(1):963, 2018.

[69] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://aclanthology.org/N18-1202.

[70] Steven M Peterson, Zoe Steine-Hanson, Nathan Davis, Rajesh PN Rao, and Bingni W Brunton. Generalized neural decoders for transfer learning across participants and recording modalities. *Journal of Neural Engineering*, 18(2):026014, 2021.

[71] Why Python. Python. *Python Releases for Windows*, 24, 2021.

[72] Qinwan Rabbani, Griffin Milsap, and Nathan E Crone. The potential for a speech brain–computer interface using chronic electrocorticography. *Neurotherapeutics*, 16:144–165, 2019.

[73] Anne Bech Risum and Rasmus Bro. Using deep learning to evaluate peaks in chromatographic data. *Talanta*, 204:255–260, 2019.

[74] Ana-Luiza Rusnac and Ovidiu Grigore. Imaginary speech recognition using a convolutional network with long-short memory. *Applied Sciences*, 12(22):11873, 2022.

[75] Milan Rybář and Ian Daly. Neural decoding of semantic concepts: A systematic literature review. *Journal of Neural Engineering*, 2022.

[76] Milan Rybář, Riccardo Poli, and Ian Daly. Decoding of semantic categories of imagined concepts of animals and tools in fnirs. *Journal of Neural Engineering*, 18 (4):046035, 2021.

[77] Pramit Saha and Sidney Fels. Hierarchical deep feature learning for decoding imagined speech from eeg. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 10019–10020, 2019.

[78] Siavash Sakhavi, Cuntai Guan, and Shuicheng Yan. Learning temporal information for brain-computer interface using convolutional neural networks. *IEEE transactions on neural networks and learning systems*, 29(11):5619–5629, 2018.

[79] Robin Tibor Schirrmeister, Lukas Gemein, Katharina Eggensperger, Frank Hutter, and Tonio Ball. Deep learning with convolutional neural networks for decoding and visualization of eeg pathology. *arXiv preprint arXiv:1708.08012*, 2017.

[80] Martin Schrimpf, Idan Asher Blank, Greta Tuckute, Carina Kauf, Eghbal A Hosseini, Nancy Kanwisher, Joshua B Tenenbaum, and Evelina Fedorenko. The neural architecture of language: Integrative modeling converges on predictive processing. *Proceedings of the National Academy of Sciences*, 118(45):e2105646118, 2021.

[81] Rini A Sharon, Shrikanth Narayanan, Mriganka Sur, and Hema A Murthy. An empirical study of speech processing in the brain by analyzing the temporal syllable structure in speech-input induced eeg. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4090–4094. IEEE, 2019.

[82] Grigori Sidorov, Alexander Gelbukh, Helena Gómez-Adorno, and David Pinto. Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas*, 18(3):491–504, 2014.

[83] Bettina Sorger and Rainer Goebel. Real-time fmri for brain-computer interfacing. *Handbook of clinical neurology*, 168:289–302, 2020.

[84] PZ Soroush, C Herff, S Ries, JJ Shih, T Schultz, and DJ Krusienski. Contributions of stereotactic eeg electrodes in grey and white matter to speech activity detection. In *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 4789–4792. IEEE, 2022.

[85] Pantulkar Sravanthi and B Srinivasu. Semantic similarity between sentences. *International Research Journal of Engineering and Technology (IRJET)*, 4(1):156–161, 2017.

[86] Sergey D Stavisky, Paymon Rezaii, Francis R Willett, Leigh R Hochberg, Krishna V Shenoy, and Jaimie M Henderson. Decoding speech from intracortical multielectrode arrays in dorsal "arm/hand areas" of human motor cortex. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 93–97. IEEE, 2018.

[87] Ã Steven Laureys, Frederic Pellas, Philippe Van Eeckhout, Sofiane Ghorbel, Caroline Schnakers, Fabien Perrin, Jacques Berre, Marie-Elisabeth Faymonville, Karl-Heinz Pantke, Francois Damas, et al. The locked-in syndrome: what is it like to be conscious but paralyzed and voiceless? 2005.

[88] Pengfei Sun, Gopala K Anumanchipalli, and Edward F Chang. Brain2char: a deep architecture for decoding text from brain recordings. *Journal of neural engineering*, 17(6):066015, 2020.

[89] Kahoko Takahashi, Zhe Sun, Jordi Solé-Casals, Andrzej Cichocki, Anh Huy Phan, Qibin Zhao, Hui-Hai Zhao, Shangkun Deng, and Ruggero Micheletto. Data augmentation for convolutional lstm based brain computer interface system. *Applied Soft Computing*, 122: 108811, 2022.

[90] Jerry Tang, Amanda LeBel, Shailee Jain, and Alexander G Huth. Semantic reconstruction of continuous language from non-invasive brain recordings. *Nature Neuroscience*, pages 1–9, 2023.

[91] Marcel Van Gerven, Jason Farquhar, Rebecca Schaefer, Rutger Vlek, Jeroen Geuze, Anton Nijholt, Nick Ramsey, Pim Haselager, Louis Vuurpijl, Stan Gielen, et al. The brain–computer interface cycle. *Journal of neural engineering*, 6(4):041001, 2009.

[92] Sarah K Wandelt, Spencer Kellis, David A Bjånes, Kelsie Pejsa, Brian Lee, Charles Liu, and Richard A Andersen. Decoding grasp and speech signals from the cortical grasp circuit in a tetraplegic human. *Neuron*, 110(11):1777–1787, 2022.

[93] Wei Wang, Alan D Degenhart, Gustavo P Sudre, Dean A Pomerleau, and Elizabeth C Tyler-Kabara. Decoding semantic information from human electrocorticographic (ecog) signals. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6294–6298. IEEE, 2011.

[94] Yasi Wang, Hongxun Yao, and Sicheng Zhao. Auto-encoder based dimensionality reduction. *Neurocomputing*, 184:232–242, 2016.

[95] Leila Wehbe, Brian Murphy, Partha Talukdar, Alona Fyshe, Aaditya Ramdas, and Tom Mitchell. Simultaneously uncovering the patterns of brain regions involved in different story reading subprocesses. *PloS one*, 9 (11):e112575, 2014.

[96] Jonathan Wolpaw and Elizabeth Winter Wolpaw. *Brain–Computer Interfaces: Principles and Practice*. Oxford University Press, 01 2012. ISBN 9780195388855. doi: 10.1093/acprof:oso/9780195388855.001.0001. URL https://doi.org/10.1093/acprof:oso/9780195388855.001.0001.

[97] Hemmings Wu, Chengwei Cai, Wenjie Ming, Wangyu Chen, Zhoule Zhu, Chen Feng, Hongjie Jiang, Zhe Zheng, Mohamad Sawan, Ting Wang, et al. Investigation of contributions from cortical and subcortical brain structures for speech decoding. *bioRxiv*, pages 2023–11, 2023.

[98] H Yamaguchi, T Yamazaki, K Yamamoto, S Ueno, A Yamaguchi, T Ito, S Hirose, K Kamijo, H Takayanagi, T Yamanoi, et al. Decoding silent speech in japanese from single trial eegs: Preliminary results. *Journal of Computer Science & Systems Biology*, 2015:285, 2015.

[99] Yuhao Zhao, Yu Chen, Kaiwen Cheng, and Wei Huang. Artificial intelligence based multimodal language decoding from brain activity: A review. *Brain Research Bulletin*, page 110713, 2023.

[100] Difan Zou, Yuan Cao, Yuanzhi Li, and Quanquan Gu. Understanding the generalization of adam in learning neural networks with proper regularization. *arXiv preprint arXiv:2108.11371*, 2021.