Autonomous Virtual Cognitive Assessment through Conversational Agents Leveraging Natural Language Processing Techniques

Bahar Karimi

A Thesis

in

The Department

of

Concordia Institute for Information System Engineering (CIISE)

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Information Systems Security) at

Concordia University

Montréal, Québec, Canada

April 2023

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Bahar Karimi**

Entitled: **Autonomous Virtual Cognitive Assessment through Conversational Agents Leveraging Natural Language Processing Techniques**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Information Systems Security)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Mohsen Ghafouri

_____ Examiner
Dr. Farnoosh Naderkhani

_____ Supervisor
Dr. Arash Mohammadi

Approved by _____
Dr. Abdessamad Ben Hamza, Chair
Department of Concordia Institute for Information System Engineering (CIISE)

_____ 2023

_____
Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Autonomous Virtual Cognitive Assessment through Conversational Agents Leveraging Natural Language Processing Techniques

Bahar Karimi

The COVID-19 pandemic has abruptly and undoubtedly changed the world we knew at the end of the second decade of the 21st century. To be prepared for future possible pandemics, special attention should be devoted to the fact that seniors, aged 60 and over, are more vulnerable during the pandemic era. In addition to a higher risk of infections, seniors are also at higher risk of suffering from mental and cognitive issues. Given an expected and alarming population ageing in near future, it is crucial and of significant importance to developing innovative and advanced autonomous cognitive screening systems. Of particular interest to this thesis is the development of autonomous cognitive screening systems via the integration of Signal Processing (SP), Artificial Intelligence (AI), and Machine Learning (ML) models. In particular, the focus is on the development of an AI-empowered avatar that autonomously performs the Neurobehavioral Cognitive Status Examination (Cognistat) assessments. Results obtained from Cognitive screening tests can be used in conjunction with other data sources to perform differential diagnosis of dementia or other cognitive disorders. Although Cognistat is widely utilized in clinical applications, its administration and interpretation of the results solely rely on a well-trained physiologist, which is highly restrictive during a pandemic. Towards addressing this issue, in the thesis, an Automated Virtual Cognitive Assessment (AVCA) framework [1, 2] is proposed that integrates Natural Language Processing (NLP) and hand gesture recognition techniques. The AVCA framework is an autonomous cognitive assessment system that receives audio and video signals in a real-time fashion and performs semantic and synthetic analysis using NLP techniques and DNN models. The proposed framework provides individual scores in the seven major cognitive domains, i.e., orientation, attention, language, contractual ability, memory, calculation, and reasoning. Additionally, we propose an efficient model to facilitate human-machine interactions from speech recognition to text classification. In particular, an unsupervised contrastive learning framework is proposed using Bidirectional Encoder Representations from Transformers (BERT) that outperforms its state-of-the-art unsupervised counterparts.

Abbreviation

Abbreviation	Description
AI	Artificial Intelligence
ANNs	Artificial Neural Networks
ANS	Autonomic Nervous System
ASR	Automatic Speech Recognition
AUC-ROC	Area Under the ROC Curve
AVCA	Autonomous Virtual Cognitive Assessment
BoW	Bag-of-Words
BERT	Bidirectional Encoder Representations from Transformers
CCE	Compositional Coding Embeddings
CNNs	Convolutional Neural Networks
CNS	Central Nervous System
CPC	Contrastive Predictive Coding
CRF	Conditional Random Field
CTC	Connectionist Temporal Classification
DAE	Denoising Autoencoding
DL	Deep Learning
DNN	Deep Neural Network
ECG	Electrocardiogram
EEG	Electroencephalogram
EMG	Electromyography

ELECTRA	Efficiently Learning an Encoder that Classifies Token Replacements Accurately
FFNNs	Feed Forward Neural Networks
FPR	False Positive Rate
GATs	Graph Attention Networks
GCNs	Graph Convolutional Networks
GDA	Generative Data Augmentation
GNNs	Graph Neural Networks
GPT	Generative Pre-trained Transformer
GRNs	Graph Recurrent Networks
GRU	Gated Recurrent Unit
GraphSAGE	Graph Sampling and Aggregation
GSR	Galvanic Skin Response
gTTS	Google Text-to-Speech
HMM	Hidden Markov Model
HMI	Human Machine Interface
LD	Levenshtein Distance
LSTM	Long Short-Term Memory
MEMM	Maximum Entropy Markov Model
MLM	Masked Language Modeling
MMSE	Mini-Mental Status Examination
MoCA	Montreal Cognitive Assessment
MNLI	Multi-Genre Natural Language Inference
MSA	Multi-head Self Attention
MCI	Mild Cognitive Impairment
NLP	Natural Language Processing
NLU	Natural Language Understanding
NLG	Natural Language Generation
NCSE	Neurobehavioral Cognitive Status Examination

NER	Named Entity Recognition
NSP	Next Sentence Prediction
ORM	Object Relational Mapper
PD	Parkinson's Disease
POS	Part-of-speech
PNS	Parasympathetic Nervous System
RoBERTa	Robustly Optimized BERT Pretraining Approach
RNNs	Recurrent Neural Networks
SE	Sentence Embedding
SimCSE	Similarity-based Contrastive Self-supervised learning for Embeddings
SNS	Sympathetic Nervous System
SNLI	Stanford Natural Language Inference
SP	Signal Processing
STS	Semantic Textual Similarity
SVMs	Support Vector Machines
T5	Text-to-Text Transfer Transformer
TF-IDF	Term Frequency-Inverse Document Frequency
TTS	Text-to-Speech
TPR	True Positive Rate
VA	Virtual Assistant

Acknowledgments

I would like to express my deepest gratitude to my thesis advisor for his guidance and support. My family, friends, and colleagues also deserve my thanks for their unwavering encouragement and motivation throughout this journey. I appreciate everyone who contributed to my success.

Contents

List of Figures

List of Tables

Chapter 1

Thesis Introduction

The COVID-19 pandemic has had a profound impact on various aspects of modern life, necessitating attention to the fact that seniors aged 60 and over are especially vulnerable during pandemics. This population not only is at a higher risk of contracting infections but also at higher risk of experiencing mental and cognitive issues, making it essential to prioritize their health and well-being in the face of present and future pandemics. As the aging population is expected to increase significantly in the near future, it is imperative to develop innovative and advanced autonomous cognitive screening systems. Such systems would not only aid in the identification and prevention of health risks, but also contribute to the early detection of cognitive impairments and neurological conditions in seniors.

In line with the above discussion, the thesis focuses on developing an Artificial Intelligence (AI)-empowered avatar capable of autonomously performing Neurobehavioral Cognitive Status Examination (Cognistat) assessments. These assessments are crucial in evaluating brain functionality during pandemics, providing a quick estimate of a patient's mental functioning. Designed by neurologists, Cognistat can help detect cognitive impairments and neurological conditions in seniors and provide valuable information for differential diagnosis. The results of Cognitive screening tests are commonly used in conjunction with other data sources to perform a thorough assessment of cognitive function and determine whether further neuropsychological assessment is necessary. By leveraging advanced technologies such as Signal Processing (SP), Machine Learning (ML), and Deep Learning (DL) models, autonomous screening systems can be developed to accurately and

efficiently identify potential health risks in seniors.

Although various cognitive assessment procedures, including Cognistat, are developed and are widely utilized in clinical applications, administration of each assessment and interpreting the results solely relies on presence of human examiner and a well-trained physiologist. However, during a pandemic, such as COVID-19, and mission critical endeavors, such as battlefields and spaceships, access to the above-mentioned resources to perform a cognitive test is largely restricted. Through this thesis, an AI-empowered avatar designed that autonomously performs the Cognistat assessments, i.e., conduct the test autonomously, understand participants inputs (e.g., speech, drawings), perform the inference task and provide the evaluation results. Overall, the development of such an advanced autonomous screening system is of significant importance, especially given the alarming population aging expected in the future. Such systems have the potential to improve the quality of life of seniors and contribute to a more proactive and effective approach to health care during pandemics.

1.1 Autonomous Cognitive Assessment (ACA)

As stated previously, this thesis aims to automate the process of performing the Cognistat test. The thesis is grounded on replacing the human interventions in the assessment process with AI-based techniques. To be more specific, in the proposed solution, the examiner will be replaced by a chatbot, which was built based on the most recent breakthrough in the field of human language understanding. The expertise of the examiner is also be replaced by various DL/SP modules, which will be trained over datasets collected from historical and anonymized Cognistat tests.

More specifically, to perform cognitive assessment in clinics, parapsychologists or experts in other domains, e.g., neurologists, neurosurgeons, physiatrists, and/or psychiatrists, would follow the instructions and simultaneously report user's responses to the questionnaire. The examiner also grades the performance of the user in each cognitive domain and reports the associated grades in user's Cognistat assessment profile. In order to automate the whole process, we envisioned development of an Natural Language Processing (NLP) -based Autonomous Cognitive Assessment (ACA) framework, which consists of the following modules:

1. *Speech Recognition Unit:* Speech processing module transforms the user's speech into text via Automatic Speech Recognition (ASR) and mining speech information. Afterwards, the resulting text can be processed to extract information contents of the speech. Such a module should be able to report the exact answers to each question, and if required, translate user's language to the system's default language for further analysis.

2. *Natural Language Processing (NLP) Engine:* NLP is a collection of techniques used to extract grammatical structure and infer from speech to perform a certain task or feed another module in the processing pipeline. The design and selection of the NLP engine is crucially important but is a challenging step towards determining the overall performance of the ACA system. The challenge comes from multidisciplinary nature of this research requiring expertise related to linguistics, cognitive science, psychology, philosophy and engineering (especially signal processing). In this thesis, the goal of the NLP engine is to capture the meaning of the spoken language to correctly fill out the Cognistat form.

3. *Cognitive Scoring:* Potentially, the most challenging component for development of ACA platforms is the process of cognitive scoring, where the judgment of an examiner needs to be replaced by AI models. In other words, the wide range of inputs that may be received for a question in the test and also the variance of interpretations and judgments among examiners are major challenges that need to be addressed.

With regards to Item 3 above, the focus is to categorize the existing questions in the Cognistat test into different main classes and then take different policies in scoring questions associated with the identified class. For development of AI-enabled scoring methodologies, the following three classes are initially considered:

- *Type I:* This category contains questions with self-expressive answers and the ones with obvious answers, such as mathematical questions or construction metrics. The questions in this category do not require any judgment from the examiner (or the AI agent) and can be easily filled and scored.

- *Type II:* This category contains questions that require the person to draw an image, either

replicating an illustrated image in the test or based on a description of an image given in the test. The answer to this type of questions could be evaluated by designing Deep Neural Network (DNN) models, which are intrinsically developed based on measuring a similarity score. For the questions that the person is asked to draw an image based on a given description, one can consider the following approach: Pair each of the underlying questions with a number of potentially correct answers and then use techniques to measure the similarity between the drawing and the potential answers, and;

- *Type III:* Answers to the questions in this category are mainly communicated in spoken language and scoring is based on the correctness of the content, speed of speaking, and the ability in remembering a sequence of words. The judgment of examiner on the answers to questions in this category is extremely challenging to be learned and mimicked. Potential type of questions that investigate the correctness of spelling and the speed of talking in the answers, can be scored through various computational methods that are specialized in assessing the correctness of speech and are widely employed in language learning platforms.

Next, contributions of the thesis to address the aforementioned challenges are briefly outlined.

1.2 Contributions

The main objective of the research presented in this thesis is to develop an ACA that can aid healthcare providers in diagnosing and treating senior patients. The ACA utilizes ML/DL and NLP techniques to analyze patient data and accurately diagnose medical conditions. Throughout the thesis research, several contributions were made to advance the development of the ACA. These contributions include the integration of DL algorithms to enable accurate medical diagnoses, the incorporation of NLP techniques to enhance the system's ability to communicate with patients, and the development of a user-friendly interface to facilitate ease of use. These advancements represent significant strides in the field of healthcare technology, with the potential to transform the way medical professionals diagnose and treat patients. In summary, the main contributions [1, 2] of my thesis research work are as follows:

(1) **The AVCA Framework:** We proposed the Autonomous Virtual Cognitive Assessment (AVCA) framework, which represents a robust and scalable solution for cognitive assessment. This innovative system is built using a containerized approach, which includes three primary containers: the frontend, web app, and database. The use of containers allows each component of the application to be isolated and scaled independently, providing unparalleled flexibility and adaptability. One of the unique features of the AVCA is its ability to gather information from patients using only a microphone and camera. This approach makes the testing process non-intrusive and eliminates the need for any special equipment. Furthermore, patients do not have to write or click on anything during the test, as the entire process is autonomous. This approach maximizes patient comfort and minimizes the potential for errors. In short, the AVCA is a highly adaptable and innovative cognitive assessment solution that is built using a containerized approach. This design ensures that each component of the application can be scaled independently, while the use of microphone and camera eliminates the need for special equipment and maximizes patient comfort. Overall, the proposed AVCA framework represents a significant advancement in cognitive assessment technology, with the potential to transform the way cognitive assessments are conducted in healthcare settings.

(2) **The Sentence Embedding Model:** The AVCA's Sentence Embedding (SE) model is subjected to a thorough assessment using seven standard Semantic Textual Similarity (STS) tasks, including STS 2012-2016, STS Benchmark, and SICK-Relatedness. The SE model is compared to several of its counterparts, and the results demonstrate a significant improvement in almost all STS tasks, particularly those related to semantic similarity. The utilization of data augmentation techniques during the production of training data has enabled the AVCA to more accurately evaluate the similarity of sentences with similar semantic meaning, despite variations in vocabulary, by extracting more critical elements from the underlying text. It is important to note that these findings are based on unsupervised models, and it is expected that the implementation of supervised models with data augmentation techniques will lead to even more significant outcomes. Ultimately, the contribution of the SE model is to enhance sentence embeddings and improve the performance of NLP tasks that rely on semantic

similarity.

(3) **Graph Neural Network (GNN)-based Text-Classification**: We have implemented an accurate evaluation system capable of automating the assessment of patients' comprehension of the consequences of specific actions via a set of pre-determined questions with correct answers. The system achieves this by conducting text classification on a proprietary dataset using various ML models, such as Support Vector Machines (SVM), Convolutional Neural Network (CNN), and GNN. Among these models, inductive GNNs are found to be particularly effective in text classification tasks, owing to their ability to manage the underlying graph structure of textual data and generalize to new data. By leveraging the graph structure of language and the ability to generalize to new data, inductive GNNs achieve higher accuracy in text classification tasks compared to other DL models. This is particularly important for AVCA's evaluation system, which must accurately determine patient understanding of specific concepts to facilitate proper diagnosis and treatment. Overall, AVCA's evaluation system represents a significant improvement in the field of patient assessment, providing an efficient and effective method for automating the evaluation of patient comprehension.

1.3 Thesis Organization

The rest of the thesis is organized as follows:

- Chapter 2 provides a comprehensive overview of various NLP techniques and their applications in cognitive assessment systems.
- Chapter 3 provides a detailed overview of the AVCA application, and its implementation.
- Chapter 4 discusses the use of cognitive assessments in healthcare systems and presents the required modules. It also introduces a novel contrastive learning framework for sentence representations.
- Chapter 5 concludes the thesis and explains some directions for future research.

Chapter 2

Literature Review and Background

Natural language processing (NLP), also known as computational linguistics, an artificial intelligence technique that enables computers to comprehend human speech, is used by medical chatbots to communicate with patients. The three iterative tasks that make up the chatbots' functionality include comprehending a patient's response, selecting the subsequent piece of information to get based on the prior piece of information, generating a corresponding question [3].

The field of NLP can be broadly categorized into two areas: core areas and applications, though it is not always straightforward to determine which area a particular issue falls into [4]. The core areas focus on fundamental challenges such as language modeling, which involves identifying associations between words; morphological processing, which deals with breaking down words into meaningful components and identifying their parts of speech; syntactic processing, which constructs sentence structures as a foundation for semantic processing; and semantic processing, which aims to extract meaning from text at various levels. On the other hand, the application areas include tasks such as extracting relevant information (e.g. named entities and relationships), translating text between different languages, summarizing written works, answering questions automatically by inference, and classifying and grouping documents. Often, it is necessary to address one or more core issues and use the ideas and techniques to tackle practical problems in the application areas.

2.1 Overview of NLP techniques

NLP techniques have been used in a wide range of evaluation systems, such as sentiment analysis, opinion mining, and text classification. These techniques involve preprocessing, tokenization, part-of-speech tagging, named entity recognition, and intent recognition.

2.1.1 Preprocessing

Preprocessing is an essential step in Natural Language Processing (NLP) that involves cleaning and transforming raw text data before it is analyzed [5]. The main purpose of preprocessing is to ensure that the text data is in a format that can be easily analyzed by NLP algorithms [10]. In this section, we will discuss the most common preprocessing techniques used in NLP.

Tokenization Tokenization is the process of breaking text into individual words or tokens. It is a critical step in NLP since most of the algorithms used in NLP require text data to be broken down into smaller units of meaning. Tokenization involves separating punctuation marks, whitespace, and other special characters to create a list of individual words that can be easily analyzed. For example, the sentence "I love natural language processing!" can be tokenized into ["I", "love", "natural", "language", "processing", "!"].

Lemmatization Lemmatization is the process of reducing words to their base form or lemma. This technique is used to reduce the dimensionality of text data by grouping together words that have the same meaning. For example, the words "am", "is", and "are" can be reduced to their base form "be". This technique helps NLP algorithms to better understand the meaning of the text and improve accuracy. For example, the sentence "I am running in the park" can be lemmatized to "I be run in the park".

Stemming Stemming is a technique used to reduce words to their root form by removing prefixes and suffixes. This technique helps to reduce the dimensionality of text data by grouping together words that have the same meaning. However, stemming can sometimes lead to incorrect results since some words may have the same stem but different meanings. For example, the words "plays", "played", and "playing" can be reduced to the stem "play". However, the

word "playing" may have a different meaning than the word "play" in certain contexts.

Noise Removal Noise removal is a preprocessing technique that involves removing unnecessary elements from the text data that do not contribute to the meaning of the text. These elements can include stopwords, punctuation, and special characters.

Stopword removal Stopwords are common words in a language, such as "the," "a," "an," and "in," that do not carry much meaning on their own and can be safely removed from the text data without affecting its overall meaning. Stopword removal is a simple and effective technique to reduce the dimensionality of the text data and improve the accuracy of downstream NLP tasks. One of the popular stopword lists is the NLTK stopwords corpus, which contains a list of 179 stopwords for English language.

Punctuation removal Punctuation marks, such as commas, periods, and question marks, are used to add structure and meaning to the text data. However, in some cases, they can add noise to the data and can be safely removed without affecting the meaning of the text. Punctuation removal is a simple and effective technique to reduce the noise in the text data and improve the accuracy of downstream NLP tasks.

Text Cleaning and Correction Text cleaning and correction is a preprocessing technique that involves fixing common errors and inconsistencies in the text data. These errors can include spelling mistakes, grammar errors, and inconsistent capitalization.

Spelling correction Spelling errors are common in text data, and they can negatively affect the accuracy of downstream NLP tasks. Spelling correction is a preprocessing technique that involves identifying and correcting spelling mistakes in the text data. One of the popular spelling correction algorithms is the Levenshtein distance algorithm, which measures the distance between two words based on the number of insertions, deletions, and substitutions required to transform one word into the other.

Grammar correction Grammar errors, such as incorrect verb tense, subject-verb agreement, and sentence structure, can affect the readability and clarity of the text data.

Grammar correction is a preprocessing technique that involves identifying and correcting grammar errors in the text data. One of the popular grammar correction tools is the LanguageTool, which uses rule-based and statistical methods to detect and correct grammar errors in text data.

Lowercasing Lowercasing is the process of converting all letters in text data to lowercase. This technique is used to ensure that the text data is consistent and uniform, making it easier to analyze. By converting all text to lowercase, NLP algorithms will not distinguish between words that are capitalized or not, reducing the dimensionality of the data. For example, the sentence "Natural Language Processing is amazing!" can be transformed into "natural language processing is amazing!".

2.1.2 Part-of-speech tagging

Part-of-speech (POS) tagging is a fundamental task in natural language processing (NLP), which aims to assign each word in a sentence with its corresponding part-of-speech tag, such as noun, verb, adjective, and so on. POS tagging plays an essential role in many NLP applications, including text classification, named entity recognition, and parsing.

There are various models for POS tagging, including rule-based models, Hidden Markov Models (HMMs), Maximum Entropy Models (MEs), Conditional Random Fields (CRFs), and neural network-based models such as Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and Transformers [6].

Rule-based models rely on pre-defined rules and heuristics to assign POS tags to words based on their contextual and syntactic features. For example, a rule-based model might assign the tag "NN" (noun) to words that end in "s," such as "books" or "dogs."

HMMs are probabilistic models that use Markov assumptions to model the probability of a sequence of tags given a sequence of words. HMMs estimate the probabilities of transitions between tags and the probabilities of observing a word given a tag. HMMs are computationally efficient and can handle large datasets, but they suffer from the "label bias problem," where the most frequent label tends to be assigned to rare words.

MEs are probabilistic models that use a maximum entropy approach to estimate the conditional

probability of a tag given a word and its context. MEs use a set of features that capture contextual and syntactic information to estimate the probabilities of assigning a tag to a word. MEs are known for their flexibility and ability to handle a large number of features, but they require a large amount of training data to achieve high accuracy.

CRFs are probabilistic models that estimate the probability of a sequence of tags given a sequence of words. CRFs use a set of features that capture contextual and syntactic information to estimate the probability of a tag given a word and its context. CRFs are known for their ability to capture complex dependencies between tags and for their high accuracy, but they can be computationally expensive.

Neural network-based models such as RNNs, CNNs, and Transformers have become increasingly popular in recent years due to their ability to capture complex patterns and long-range dependencies in language. RNNs are a type of neural network that can handle sequential data and are well-suited for POS tagging tasks. CNNs are a type of neural network that can extract local features from text, and have been used for POS tagging with good results. Transformers are a type of neural network that can model long-range dependencies and have been shown to achieve state-of-the-art results in many NLP tasks, including POS tagging.

In terms of the best approach for POS tagging, it depends on the specific task and the amount of training data available. Generally, deep learning and machine learning approaches, such as RNNs, CNNs, and Transformers, have shown to achieve state-of-the-art results in many POS tagging tasks. However, rule-based and probabilistic models such as HMMs, MEs, and CRFs can still be effective in certain contexts, such as when there is limited training data or when interpretability is important.

2.1.3 Named Entity Recognition

Named Entity Recognition (NER) is a subtask of information extraction in Natural Language Processing (NLP) that involves identifying and classifying named entities in text, such as people, organizations, locations, and dates. NER is important in various NLP applications, including information extraction, question answering, and text classification.

NER can be approached using rule-based, statistical, and deep learning methods [7]. Rule-based methods rely on pre-defined rules and heuristics to identify named entities based on their

linguistic features, such as capitalization and context. Statistical methods use machine learning algorithms, such as Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs), to identify named entities based on patterns and structures in text. Deep learning methods, such as Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and Transformers, can automatically learn patterns and structures in text without the need for pre-defined rules or features. These methods can also capture long-range dependencies and contextual information in text, leading to higher accuracy.

Despite the significant progress made in NER, there are still some challenges that need to be addressed. One of the main challenges is handling ambiguous and rare entities. NER models can struggle to accurately identify entities that have different meanings depending on the context, or that appear infrequently in the training data. Another challenge is dealing with noisy data. Text data can contain errors, misspellings, and other types of noise that can make it difficult for NER models to correctly identify entities. Finally, domain adaptation is an important challenge in NER, as models trained on one domain may not perform well on data from a different domain.

To address these challenges, researchers and practitioners can consider using techniques such as data augmentation to increase the diversity of training data, leveraging multi-task learning to improve model performance, and fine-tuning pre-trained models on domain-specific data. They can also incorporate techniques such as ensembling and active learning to improve model accuracy and reduce the impact of noisy data.

2.1.4 Intent Recognition

Intent recognition and slot filling are crucial tasks in Natural Language Understanding (NLU) that involve identifying the intention or goal behind a user's input and extracting relevant parameters, respectively. Slot filling involves identifying and extracting specific pieces of information, or "slots," from a user's input that are required to fulfill the intent. These tasks are often used in conversational AI systems to provide accurate and relevant responses.

There are various approaches to joint intent detection and slot filling [8] including rule-based, statistical, and deep learning methods. Rule-based methods use pre-defined rules and heuristics to

identify intents and slots based on linguistic features, such as keywords and part-of-speech tags. Statistical methods, such as Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs), use machine learning algorithms to learn the patterns and structures in text to identify intents and slots. These methods require a large amount of annotated data for training and can achieve high accuracy.

Deep learning methods, such as Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and Transformers, have shown promising results in intent recognition tasks. These methods can automatically learn the patterns and structures in text without the need for predefined rules or features. They can also capture long-range dependencies and contextual information in text, leading to higher accuracy.

An important approach to intent recognition is supervised learning, where a large dataset of labeled user inputs and their corresponding intents is used to train a model. This requires manual annotation of the dataset, where human annotators label each user input with its corresponding intent. The annotations are typically represented as binary labels, indicating whether the input belongs to a particular intent or not.

Formally, given a set of n user inputs $x_1, x_2, ..., x_n$ and their corresponding intents $y_1, y_2, ..., y_n$, the intent recognition task is to learn a function $f(x)$ that maps an input x to its corresponding intent y. This function is typically learned using a machine learning algorithm that minimizes a loss function over the labeled dataset.

2.2 Applications of NLP in Evaluation Systems

NLP techniques are essential for automatic evaluation systems to effectively process and analyze natural language data. These techniques enable these systems to accurately interpret the meaning of text, extract relevant information, and address the challenges of working with unstructured natural language data. As such, the development and advancement of NLP techniques will continue to be critical in the advancement of automatic evaluation systems.

2.2.1 Sentiment Analysis

Sentiment analysis, also known as opinion mining, is a subfield of Natural Language Processing (NLP) that focuses on extracting subjective information from text data [9]. The main objective of sentiment analysis is to classify the sentiment of the text into positive, negative, or neutral. There are various techniques and approaches to perform sentiment analysis, some of which are discussed below.

Rule-based Approach The rule-based approach is a simple technique that involves defining a set of rules or patterns that match specific sentiment-related words or phrases in the text data. These rules can be created manually by experts or by using machine learning algorithms to learn the rules from labeled data. The approach can work well for specific domains and languages, but it requires significant human effort to define the rules and may not generalize well to other domains.

Lexicon-based Approach The lexicon-based approach is a technique that involves using a pre-built sentiment lexicon or dictionary that contains a list of words with their associated sentiment polarity, such as positive or negative. The approach calculates the sentiment score of the text by summing up the sentiment scores of each word in the lexicon that appears in the text. The sentiment score can be calculated using the following formula

$$S_{text} = \sum_{i=1}^{n} s_i \tag{2.1}$$

where S_{text} is the sentiment score of the text, s_i is the sentiment score of the i^{th} word in the text, and n is the number of words in the text. The sentiment score of each word can be obtained from the sentiment lexicon.

The sentiment lexicon can be created manually or automatically using machine learning algorithms. The approach works well for general domains and languages but may not perform well in specific domains with domain-specific language.

2.2.2 Sentiment Analysis using ML

The machine learning approach for sentiment analysis is a popular technique that involves training a classifier model on labeled data to predict the sentiment of the text data. The approach involves three main steps: feature extraction, model training, and model testing.

In the feature extraction step, the text data is transformed into a set of numerical features that can be used as input to the classifier model [10]. This step involves various techniques such as bag-of-words, TF-IDF, and word embeddings. Let X denote the feature matrix, where each row represents a document and each column represents a feature. Let x_i denote the feature vector for the i^{th} document.

In the model training step, a classifier model is trained on the labeled data to learn the relationship between the input features and the sentiment labels. Let y denote the vector of sentiment labels, where each element represents the sentiment label of the corresponding document in X. Let $f(X)$ denote the classifier model, which maps the input feature matrix X to the predicted sentiment labels $\hat{y}$. The objective of the training step is to minimize the following loss function

$$L(X, y) = \frac{1}{m} \sum_{i=1}^{m} L(f(x_i), y_i) \tag{2.2}$$

where m is the number of documents in the labeled data, L is the loss function, and $f(x_i)$ is the predicted sentiment label for the i^{th} document.

In the model testing step, the trained classifier model is applied to the test dataset to evaluate its performance. The performance of the classifier model is measured using various evaluation metrics, such as accuracy, precision, recall, F1-score, and area under the ROC curve (AUC-ROC). These metrics are defined as follows:

Accuracy: It measures the proportion of correct predictions out of all predictions made by the classifier model. The accuracy is defined as follows

$$\text{Accuracy} = \frac{\text{Number of correctly predicted instances}}{\text{Total number of instances}} \tag{2.3}$$

Precision: It measures the proportion of true positive predictions out of all positive predictions made by the classifier model. The precision is defined as follows

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}} \tag{2.4}$$

Recall: It measures the proportion of true positive predictions out of all actual positive instances in the test dataset. The recall is defined as follows:

$$\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}} \tag{2.5}$$

F1-score: It is the harmonic mean of precision and recall and provides a balanced measure between the two. The F1-score is defined as follows:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.6}$$

Area under the ROC curve (AUC-ROC): It measures the performance of the classifier model across different thresholds. The AUC-ROC is defined as the area under the ROC curve, which is a plot of true positive rate (TPR) versus false positive rate (FPR) at different classification thresholds. The TPR is the same as the recall, while the FPR is defined as follows:

$$\text{FPR} = \frac{\text{False positives}}{\text{False positives} + \text{True negatives}} \tag{2.7}$$

These evaluation metrics provide a quantitative measure of the performance of the sentiment analysis classifier model. The choice of evaluation metric depends on the specific requirements and constraints of the application. For example, if the cost of false positive predictions is high, then precision may be a more important metric than recall.

2.2.3 Feature Extraction For Sentiment Analysis

During the feature extraction process, the textual data is converted into a collection of numerical features that can be utilized as input to the classifier model. Different methods can be applied in this

step, including bag-of-words, TF-IDF, and word embeddings [9].

Bag-of-Words

One common way to represent the input text data is the bag-of-words (BoW) model. In this model, each document is represented as a vector of the counts of the words in the document. Formally, given a set of documents $\mathcal{D} = d_1, d_2, ..., d_n$ and a vocabulary $\mathcal{V} = w_1, w_2, ..., w_m$, the BoW representation of document d_i is a vector $x_i = [x_{i,1}, x_{i,2}, ..., x_{i,m}]$, where $x_{i,j}$ represents the number of occurrences of word w_j in document d_i.

Term Frequency-Inverse Document Frequency

Another common approach is the Term Frequency-Inverse Document Frequency (TF-IDF) model, which takes into account the importance of words in a document with respect to the entire corpus. The TF-IDF score of a word w_j in a document d_i is defined as follows

$$\text{tf-idf}(w_j, d_i) = \text{tf}(w_j, d_i) \times \text{idf}(w_j), \tag{2.8}$$

where $\text{tf}(w_j, d_i)$ is the term frequency of word w_j in document d_i, and $\text{idf}(w_j)$ is the inverse document frequency of word w_j defined as:

$$\text{idf}(w_j) = \log \frac{N}{n_j}, \tag{2.9}$$

where N is the total number of documents in the corpus, and n_j is the number of documents that contain word w_j. The TF-IDF model represents each document as a vector of the TF-IDF scores of the words in the document.

Word Embeddings

Word embeddings are a popular technique for representing words as dense vectors in a high-dimensional space, where words with similar meanings are located close to each other. They have become a fundamental component in natural language processing (NLP) tasks such as sentiment

analysis, machine translation, and named entity recognition. Word embeddings are learned from a large corpus of text using unsupervised learning methods such as Word2Vec, GloVe, and BERT. These models take into account the context in which words appear and generate a dense vector representation for each word in the vocabulary. This vector representation can then be used as input to downstream NLP tasks.

Let V be the vocabulary of all possible words and D be the dimensionality of the word embeddings. Then, a word embedding $w \in \mathbb{R}^D$ can be represented as:

$$w = [w_1, w_2, ..., w_D] \in \mathbb{R}^D \tag{2.10}$$

where w_i represents the i-th dimension of the word embedding.

One recent development in word embeddings is Compositional Coding Embeddings (CCE), which learns embeddings that are not only sensitive to the meaning of individual words but also their composition in a sentence. This is achieved by assigning each word a set of binary codes that represent its position in the sentence and using these codes to generate a unique embedding for each sentence. The resulting embeddings have been shown to outperform traditional word embeddings in several NLP tasks.

Another recent development in word embeddings is the use of BERT (Bidirectional Encoder Representations from Transformers), a language model that generates contextualized word embeddings by taking into account the context in which the word appears. BERT has achieved state-of-the-art results on a wide range of NLP tasks, including question answering, sentiment analysis, and named entity recognition.

Let S be a sequence of words in a sentence and C_i be the context vector for the i-th word in the sentence. Then, the BERT embedding for the i-th word is given by:

$$w_i^{BERT} = [C_i; E_i] \in \mathbb{R}^D \tag{2.11}$$

where E_i is the token embedding for the i-th word and $[\cdot;\cdot]$ represents concatenation. The token embedding E_i is learned from a large corpus of text using an unsupervised learning method, such as the masked language modeling task used in BERT.

2.3 Recent Language Models

In recent years, there have been several language models developed with high accuracy, including GPT-3, T5, ELECTRA, and BERT [11], [12].

GPT-3 (Generative Pre-trained Transformer3) [13] is a language model developed by OpenAI that is pre-trained on a massive corpus of text and can generate natural language text with high quality. GPT-3 uses a Transformer architecture with only a decoder component. The input to GPT-3 is a sequence of tokens, and the model predicts the next token in the sequence based on the context of the previous tokens. The architecture of GPT-3 has 175 billion parameters, making it one of the largest language models to date.

The T5 (Text-to-Text Transfer Transformer) [14] model, developed by Google, is also a transformer-based language model. However, it is unique in that it can perform a variety of tasks by simply fine-tuning a single model on different tasks. T5 uses a pre-trained transformer encoder-decoder architecture, which allows it to be fine-tuned on a variety of NLP tasks by providing task-specific input and output formats.

ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) [15] is a language model developed by Google that uses a novel pre-training method called "discriminative pre-training." In this method, a generator model is trained to replace tokens in the input text with other tokens, and a discriminator model is trained to distinguish between the original input text and the modified text. ELECTRA uses a Transformer architecture with both encoder and decoder components, but the decoder component is only used during fine-tuning for downstream tasks.

BERT (Bidirectional Encoder Representations from Transformers) [63] is a language model developed by Google that uses a Transformer architecture with both encoder and decoder components. BERT is pre-trained on a large corpus of text using two tasks: masked language modeling (MLM) and next sentence prediction (NSP). In MLM, some of the tokens in the input text are randomly masked, and the model is trained to predict the masked tokens based on the context of the surrounding tokens. In NSP, the model is trained to predict whether two input sentences are consecutive in the original text. BERT has achieved state-of-the-art results on a wide range of natural language processing tasks.

The main difference between these models is their architecture and pre-training tasks. GPT-3 is a large-scale transformer-based model that has the ability to generate text, while T5 is designed for transfer learning on a variety of NLP tasks. ELECTRA uses a unique pre-training task to improve accuracy and efficiency, and BERT is a bidirectional model that captures more complex relationships between tokens. Here are the equations used in training some of these models:

The GPT-3 model is trained using a combination of unsupervised and supervised learning, with the following loss function:

$$\mathcal{L}GPT-3 = \mathcal{L}\text{MLM} + \mathcal{L}\text{NSP} + \mathcal{L}\text{QA} \tag{2.12}$$

where $\mathcal{L}$MLM is the masked language modeling loss, $\mathcal{L}$NSP is the next sentence prediction loss, and $\mathcal{L}_{\text{QA}}$ is the question answering loss.

The T5 model is trained using a combination of supervised and unsupervised learning, with the following loss function:

$$\mathcal{L}T5 = \mathcal{L}\text{Supervised} + \mathcal{L}_{\text{Unsupervised}} \tag{2.13}$$

where $\mathcal{L}$Supervised is the supervised learning loss for the specific task being fine-tuned, and $\mathcal{L}$Unsupervised is the unsupervised learning loss used during pre-training.

BERT and ELECTRA are both transformer-based models, but they differ in their pre-training objectives. BERT uses masked language modeling (MLM), where some of the input tokens are masked and the model must predict the original tokens, while ELECTRA uses a discriminative pre-training objective where it trains a generator to replace some of the input tokens with plausible alternatives, and a discriminator is trained to identify which tokens are real and which are generated. The formulas for BERT's MLM objective and ELECTRA's discriminator objective are

BERT:

$$\mathcal{L}\text{BERT-MLM} = \sum_{i=1}^{N}[\text{mask}i]\left(-\log p(x_i|\mathbf{x_i}\text{i}, \mathbf{x}_{i}\text{i}; \theta)\right) \tag{2.14}$$

where N is the number of input tokens, mask_i is a binary variable indicating whether the ith token is masked, x_i is the ith token, and θ are the model parameters. The notation $[\text{mask}_i]$ denotes the Iverson bracket which equals 1 if mask_i is true and 0 otherwise.

ELECTRA:

$$\mathcal{L}\text{ELECTRA} = \mathbb{E}x \sim \mathbb{P}\text{data}\left[\sum i = 1^{N} \log D_{\theta_{\mathrm{D}}}(x_i) + \sum_{i=1}^{N} \log(1 - D_{\theta_{\mathrm{D}}}(\hat{x}_i))\right] \tag{2.15}$$

where x is the input sequence, $\mathbb{P}$data is the distribution of real data, $D\theta_{\mathrm{D}}$ is the discriminator with parameters θ_{D}, $\hat{x}$ is the sequence with some tokens replaced by plausible alternatives generated by a generator $G_{\theta_{\mathrm{G}}}$, and θ_{G} are the generator parameters.

T5 is a transformer-based model that is designed to perform various NLP tasks using a single architecture. It is trained using a combination of supervised and unsupervised learning, where it is first pre-trained on a large corpus using denoising autoencoding (DAE) and then fine-tuned on specific tasks using supervised learning. The architecture of T5 is similar to that of transformer models, but with a few modifications such as the use of a reversible encoder-decoder architecture and a different type of attention mechanism. The formula for T5's DAE objective is:

$$\mathcal{L}\text{T5-DAE} = \sum_{i=1}^{N}[\text{mask}i]\left(-\log p(x_i|\mathbf{x_i}i, \mathbf{x}_{i,i}; \theta)\right) \tag{2.16}$$

where N is the number of input tokens, mask_i is a binary variable indicating whether the ith token is masked, x_i is the ith token, and θ are the model parameters. In summary, GPT-3, T5, ELECTRA, and BERT are all transformer-based language models that have achieved state-of-the-art performance on various NLP tasks. They differ in their architecture, pre-training objectives, and fine-tuning.

2.4 BERT and Sentence Transformers

Transformer-based models have become the state-of-the-art approach in many natural language processing (NLP) tasks, including language translation, sentiment analysis, and text classification. Two popular transformer-based models in NLP are the Bidirectional Encoder Representations from Transformers (BERT) and the Sentence-Transformers.

2.4.1 Bidirectional Encoder Representations from Transformers (BERT)

BERT is a pre-trained transformer-based model developed by Google in 2018 [63]. It utilizes a novel training approach called Masked Language Model (MLM), where random words in a sentence are masked, and the model is trained to predict the masked words based on their context. BERT also uses a technique called Next Sentence Prediction (NSP), where the model is trained to predict if two sentences are consecutive in the original text. This approach allows the model to capture the relationships between different sentences in a text document.

The BERT architecture consists of multiple layers of transformers that process the input sequence in a bidirectional manner, allowing the model to capture both the left and right contexts of a word. The output of the last layer of transformers is then used as input to a task-specific neural network that fine-tunes the model for a specific NLP task.

Mathematical Formulation

Let $x_1, x_2, ..., x_n$ be the sequence of input tokens, where each token x_i is represented as a vector of dimension d. The BERT model takes as input the concatenation of two special tokens, [CLS] and [SEP], which are added to the beginning and end of the input sequence, respectively. The output of the BERT model is the sequence of hidden states $h_1, h_2, ..., h_n$, where each hidden state h_i is a vector of dimension d. The output of the BERT model can be represented mathematically as follows

$$\mathbf{H} = \text{BERT}([\text{CLS}] \oplus \mathbf{X} \oplus [\text{SEP}]), \tag{2.17}$$

where $\mathbf{X} = [x_1, x_2, ..., x_n]$, $\oplus$ denotes the concatenation operator, and $\mathbf{H} = [h_1, h_2, ..., h_n]$.

The BERT model is trained using a multi-task learning approach, where the model is fine-tuned for a specific NLP task by adding a task-specific neural network on top of the output of the last layer of transformers. The fine-tuning process involves minimizing a task-specific loss function, which measures the difference between the predicted outputs and the true labels of the task.

2.4.2 Robustly Optimized BERT Pretraining Approach (RoBERTa)

RoBERTa is a pre-trained transformer-based language model that is an extension of BERT and was introduced by Facebook AI in 2019 [59]. RoBERTa also employs a similar pre-training objective as BERT, namely MLM, where a subset of tokens in a sentence are randomly masked and the model is trained to predict these masked tokens. In addition, RoBERTa utilizes a larger training corpus and training steps than BERT, and removes the NSP pre-training objective used in BERT, which has been shown to be less effective.

RoBERTa employs a similar transformer-based architecture as BERT, consisting of multiple layers of transformers that process the input sequence in a bidirectional manner, allowing the model to capture both the left and right contexts of a word. RoBERTa also utilizes a task-specific fine-tuning process similar to BERT, where a task-specific neural network is added on top of the output of the last layer of transformers, and the model is fine-tuned for a specific NLP task.

2.4.3 Sentence-Transformers

Sentence-Transformers [65] are a type of models that can generate high-quality embeddings for sentences or paragraphs. These embeddings can be used for various downstream tasks, such as sentence classification, semantic search, and clustering. The Sentence-Transformers model is based on the transformer architecture, similar to BERT and GPT models.

The architecture of the Sentence-Transformers model is based on the transformer encoder, which takes the sequence of tokens and outputs the embedding for each token. Then, a pooling operation is applied to the embeddings to generate the final sentence or paragraph embedding. The pooling operation can be performed in various ways, such as mean pooling or max pooling.

One of the unique features of the Sentence-Transformers model is the use of siamese and triplet network structures. The siamese network takes two sentences and generates their embeddings, which can be used to compare their similarity. The triplet network takes three sentences, an anchor, a positive, and a negative, and generates embeddings for each of them. The goal of the triplet network is to ensure that the positive sentence is closer to the anchor than the negative sentence.

The loss function used in the Sentence-Transformers model is the triplet loss function, defined

as follows

$$\mathcal{L}\text{triplet} = \sum i = 1^n \max(0, \cos(\mathbf{a}_i, \mathbf{p}_i) - \cos(\mathbf{a}_i, \mathbf{n}_i) + \text{margin}) \tag{2.18}$$

where n is the number of triplets, $\mathbf{a}_i$, $\mathbf{p}_i$, and $\mathbf{n}_i$ are the embeddings of the anchor, positive, and negative sentences, respectively, and $\cos(\mathbf{a}_i, \mathbf{p}_i)$ and $\cos(\mathbf{a}_i, \mathbf{n}_i)$ are the cosine similarities between the anchor and positive, and anchor and negative embeddings, respectively. The margin parameter is used to ensure that the positive and negative embeddings are separated by a certain distance.

Sentence-Transformers can be fine-tuned on various downstream tasks using transfer learning. For example, a pre-trained Sentence-Transformers model can be fine-tuned on the STS benchmark dataset for semantic textual similarity. The fine-tuned model can then be used for various applications, such as semantic search and document clustering.

2.4.4 Text Classification with Transformer-based Models

Transformer-based models have shown remarkable success in text classification tasks due to their ability to model long-term dependencies and capture the contextual meaning of words. In this section, we will explore how these models can be used for text classification.

BERT for Text Classification

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained transformer-based language model that has achieved state-of-the-art performance in various natural language processing tasks, including text classification. BERT uses a bi-directional transformer encoder to produce a contextualized representation of words in a sentence.

Given a sequence of tokens $w_1, w_2, ..., w_n$, BERT embeds each token into a high-dimensional vector space using an embedding matrix $E \in \mathbb{R}^{d \times V}$, where d is the embedding dimension and V is the vocabulary size. Then, the sequence is passed through L transformer encoder layers, where each layer applies a self-attention mechanism to capture the context of each token in the sequence. The output of the final encoder layer corresponding to the [CLS] token is then used as a fixed-length representation of the entire sequence. This representation can be used as input to a classifier model [16].

Formally, let S be the input sentence and s_i be the i-th token in S. The output of BERT for text classification is given by

$$BERT_{cls}(S) = f\left(\sum_{i=1}^{n} \alpha_i \cdot BERT(s_i)\right) \tag{2.19}$$

where α_i is an attention weight assigned to the i-th token, $BERT(s_i)$ is the output of the final transformer layer corresponding to the i-th token, and f is a classifier model that maps the output of BERT to the target label space.

Sentence-Transformers for Text Classification

Sentence-Transformers are a family of pre-trained transformer-based models that generate fixed-length sentence embeddings for various natural language processing tasks, including text classification. These models use a siamese architecture with two identical transformer encoders that share weights to generate embeddings for two input sentences. The distance between these embeddings can be used as a measure of similarity between the two sentences [65].

Given a sentence S, a Sentence-Transformer model generates a fixed-length vector representation v_S of the sentence using a single transformer encoder. This representation can be used as input to a classifier model.

Formally, let S be the input sentence, and v_S be the fixed-length representation generated by a Sentence-Transformer model. The output of the model for text classification is given by

$$SentenceTransformer(S) = f(v_S) \tag{2.20}$$

where f is a classifier model that maps the output of the Sentence-Transformer to the target label space.

Binary Classification

In binary classification tasks, the goal is to predict one of two possible classes, usually represented by 0 and 1. The output of the classifier model can be interpreted as a probability score for the

positive class. The decision boundary can be set to 0.5, where any score above 0.5 is classified as positive, and any score below 0.5 is classified as negative.

Let y be the ground truth label for a binary classification task, where $y \in 0, 1$, and let p be the predicted probability score for the positive class. The binary cross-entropy loss function can be used to train the model

$$J = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1 - y_i)\log(1 - p_i)) \tag{2.21}$$

where N is the number of training examples. To calculate the gradient of the loss function with respect to the model parameters, we can use the chain rule

$$\frac{\partial J}{\partial \theta} = \frac{\partial J}{\partial p} \cdot \frac{\partial p}{\partial \theta} \tag{2.22}$$

where θ are the model parameters.

The partial derivative of the loss function with respect to the predicted probability score is

$$\frac{\partial J}{\partial p} = -\frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i}{p_i} - \frac{1 - y_i}{1 - p_i}\right) \tag{2.23}$$

The partial derivative of the predicted probability score with respect to the model parameters can be calculated using backpropagation.

In summary, for binary classification using transformer-based models, we first encode the input text using a transformer-based language model such as BERT or a sentence-transformer. The output of the language model is then fed into a classifier model, which maps the output of the language model to a probability score for the positive class. The model is trained using the binary cross-entropy loss function, and the gradient of the loss function is calculated using the chain rule and backpropagation.

Multi-class Classification

In multi-class classification, the goal is to classify an input sentence into one of several possible classes. Transformer-based models can be easily adapted to perform multi-class classification by adding a final classification layer on top of the output of the transformer encoder.

Let $C = c_1, c_2, ..., c_k$ be the set of k classes, and S be the input sentence. The goal is to predict the most likely class c_i given S.

For BERT, the output of the final encoder layer corresponding to the [CLS] token is used as input to a linear classification layer. The output of this layer is a vector of size k, representing the probability distribution over the k classes. This can be formalized as follows

$$p(c_i|S) = softmax(W_c h_{[CLS]} + b_c) \tag{2.24}$$

where $h_{[CLS]}$ is the output of the final encoder layer corresponding to the [CLS] token, W_c and b_c are the weight matrix and bias vector of the linear classification layer, and softmax is the softmax function.

For Sentence-Transformers, the fixed-length vector representation v_S is passed through a linear classification layer to obtain the probability distribution over the k classes. This can be formalized as follows

$$p(c_i|S) = softmax(W_c v_S + b_c) \tag{2.25}$$

where v_S is the fixed-length representation generated by the Sentence-Transformer model, and W_c and b_c are the weight matrix and bias vector of the linear classification layer, and softmax is the softmax function.

The class with the highest probability is chosen as the predicted class for the input sentence. Transformer-based models have shown impressive performance in multi-class classification tasks, often outperforming traditional machine learning algorithms and achieving state-of-the-art results on various benchmark datasets.

2.5 Deep Learning Models for NLP

Deep Learning has significantly impacted Natural Language Processing (NLP) and has been widely used in various applications such as sentiment analysis, text classification, and machine translation. The use of deep learning models in NLP has provided significant improvements in performance compared to traditional machine learning models [17], [18].

There are several types of deep learning models that are commonly used in NLP, including Artificial Neural Networks (ANNs), Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and Transformers.

Artificial Neural Networks (ANNs)

Artificial Neural Networks are feedforward networks that can be used for text classification tasks. ANNs consist of multiple layers of artificial neurons that can learn to extract meaningful features from input data. Mathematically, an ANN can be represented as follows

$$y = f(WX + b) \tag{2.26}$$

where X is the input data, W is the weight matrix, b is the bias vector, f is the activation function, and y is the output.

Feed Forward Neural Networks

Feed Forward Neural Networks (FFNNs) are one of the most basic types of neural networks, consisting of one or more layers of interconnected neurons, where each neuron applies a non-linear transformation to the weighted sum of its inputs. FFNNs are widely used in various applications, including image recognition, speech processing, and natural language processing.

Let $x \in \mathbb{R}^d$ be the input vector, where d is the dimensionality of the input space, and $y \in \mathbb{R}^c$ be the output vector, where c is the number of classes in the classification problem. An FFNN with L hidden layers can be defined as follows

$$h^{(l)} = \sigma(W^{(l)}h^{(l-1)} + b^{(l)}), \quad l = 1, 2, ..., L \tag{2.27}$$

$$f(x) = \text{softmax}(W^{(L+1)}h^{(L)} + b^{(L+1)}) \tag{2.28}$$

where $h^{(l)} \in \mathbb{R}^{h_l}$ is the output of the l-th hidden layer, $W^{(l)} \in \mathbb{R}^{h_l \times h_{l-1}}$ and $b^{(l)} \in \mathbb{R}^{h_l}$ are the weight matrix and bias vector of the l-th layer, respectively, σ is a non-linear activation function, such as ReLU or sigmoid, and $f(x)$ is the output vector of the FFNN. The softmax function ensures

that the output vector $f(x)$ is a probability distribution over the c classes.

The parameters of the FFNN, i.e., the weight matrices and bias vectors, are learned from labeled training data using backpropagation and stochastic gradient descent. The loss function used to train the FFNN depends on the specific task, but it is typically a cross-entropy loss for classification problems

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{c} y_{ij}\log(f(x_i)_j) \tag{2.29}$$

where N is the number of training samples, y_{ij} is the binary indicator of whether the i-th sample belongs to the j-th class, and $f(x_i)_j$ is the predicted probability of the i-th sample belonging to the j-th class.

Recurrent Neural Networks (RNNs)

Recurrent Neural Networks are networks that can process sequential data, such as text. RNNs use a feedback mechanism that allows them to pass information from previous time steps to the current time step, making them well-suited for tasks such as language modeling and machine translation. Mathematically, an RNN can be represented as follows

$$h_t = f(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \tag{2.30}$$

$$y_t = g(W_{hy}h_t + b_y) \tag{2.31}$$

where x_t is the input at time step t, h_t is the hidden state at time step t, f and g are activation functions, and W and b are weight matrices and bias vectors.

Convolutional Neural Networks (CNNs)

Convolutional Neural Networks are networks that are commonly used for image processing tasks but have also been applied to NLP. CNNs can be used for text classification tasks, where they are typically used to extract local features from text. Mathematically, a CNN can be represented as follows

$$y = f(W * x + b) \tag{2.32}$$

where x is the input data, W is the filter, * denotes the convolution operation, b is the bias vector, and f is the activation function.

Transformers

Transformers are a recent development in deep learning that have been particularly successful in NLP tasks such as machine translation and language modeling. Transformers use self-attention mechanisms to enable the model to focus on relevant parts of the input sequence, making them well-suited for handling long sequences of text. Mathematically, a Transformer can be represented as follows:

$$z_i = \sum_{j=1}^{n} a_{ij} h_j \tag{2.33}$$

$$a_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{n} exp(e_{ik})} \tag{2.34}$$

where z_i is the output at position i, h_j is the hidden state at position j, a_ij is the attention weight between positions i and j, and e_ij is the attention score between positions i and j.

In NLP, each of these models has its strengths and weaknesses, and the choice of model often depends on the specific task at hand. For example, CNNs are particularly useful for text classification tasks, while RNNs are better suited for sequential data.

2.5.1 Graph Neural Networks

Graph Neural Networks (GNNs) are a type of deep learning model that can be used for tasks involving structured data, such as graph classification, link prediction, and node classification. GNNs extend the traditional neural network architecture to work with graph-structured data by using message passing between nodes in the graph to propagate information. Inductive and transductive are two approaches for learning with graph neural networks (GNNs).

Inductive Learning [19] refers to the process of learning a model on a smaller subgraph of a larger graph and then generalizing the learned model to a larger graph. In other words, an inductive GNN can learn to make predictions on unseen nodes or graphs that were not part of the original training set. This is achieved by learning a node embedding that captures the local structure of the subgraph

and can be used to make predictions for new nodes.

Transductive learning refers to the process of learning a model that can make predictions for the nodes in the same graph that was used for training. In other words, a transductive GNN learns to make predictions for a specific graph and cannot generalize to new graphs. This is because the model is tailored to the specific graph and cannot capture the variations that may exist in new graphs.

There exist multiple variants of GNNs that can be leveraged for the purpose of text classification. The followings are some of these variations.

Graph Convolutional Networks (GCNs)

Graph Convolutional Networks (GCNs) [20] are a type of GNN that use convolutional operations to aggregate information from a node's neighbors and update the node's representation. Specifically, the convolution operation is defined as follows

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in N(i)} \frac{1}{c_{ij}} W^{(l)} h_j^{(l)}\right) \tag{2.35}$$

where $h_i^{(l)}$ represents the hidden state of node i in layer l, σ is the activation function, $W^{(l)}$ is the weight matrix for layer l, $N(i)$ is the set of neighbors of node i, and c_{ij} is a normalization constant to account for the different numbers of neighbors for each node.

The equation 2.35 defines a single message-passing step in a GCN. The node features from the previous layer, $h^{(l)}$, are updated by aggregating information from neighboring nodes, and the updated features are passed to the next layer.

Graph Attention Networks (GATs)

Graph Attention Networks (GATs) [21] are another type of GNN that use attention mechanisms to selectively focus on different parts of the graph when aggregating information from neighbors. Specifically, the attention mechanism is defined as follows

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top [Wh_i | Wh_j]\right)\right)}{\sum_{k \in N(i)} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top [Wh_i | Wh_k]\right)\right)} \tag{2.36}$$

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in N(i)} \alpha_{ij} W^{(l)} h_j^{(l)}\right) \tag{2.37}$$

where $h_i^{(l)}$ represents the hidden state of node i in layer l, σ is the activation function, $W^{(l)}$ is the weight matrix for layer l, $N(i)$ is the set of neighbors of node i, $\mathbf{a}$ is a learned attention vector, and $\|$ denotes concatenation.

Equation 2.36 calculates an attention coefficient for each neighbor j of node i, based on the similarity between the hidden states of i and j. Equation 2.37 uses these attention coefficients to aggregate information from the neighbors of node i.

Graph Recurrent Networks (GRNs)

Graph Recurrent Networks (GRNs) [22] are a type of GNN that uses recurrent neural network (RNN) architectures to model temporal dependencies in sequential data represented as a graph. In other words, GRNs allow for information to flow through the graph in a time-dependent manner, similar to how information flows through a sequence in a traditional RNN.

The main building block of a GRN is the graph-level RNN, which takes as input the hidden states of all nodes in the graph at a given time step, and produces a single output vector that summarizes the information contained in the graph at that time step. The output vector can then be fed into a fully connected layer for classification or regression.

The message passing function in a GRN is typically defined using a Gated Recurrent Unit (GRU) or Long Short-Term Memory (LSTM) architecture, which allows the model to selectively update and forget information based on the current input and previous hidden state. The output of the message passing function is a hidden state vector for each node in the graph at the current time step. The update equation for a GRU-based message passing function can be represented as follows

$$h_i^{(l)} = \mathrm{GRU}(\sum_{j \in N(i)} W^{(l)} h_j^{(l-1)} + b^{(l)}, h_i^{(l-1)}) \tag{2.38}$$

where $h_i^{(l)}$ represents the hidden state of node i in layer l, GRU is the gated recurrent unit, $W^{(l)}$ is the weight matrix for layer l, $N(i)$ is the set of neighbors of node i, $b^{(l)}$ is the bias term for layer l,

and $h_i^{(l-1)}$ is the hidden state of node i in the previous layer.

GRNs have been shown to be effective for a variety of text classification tasks, including text summarization and dialogue generation. However, they can be computationally expensive, especially for large graphs or long sequences, and may require significant amounts of data to train effectively.

GraphSAGE

GraphSAGE (Graph Sampling and Aggregation) [23] is a GNN model that performs inductive learning on large-scale graphs. It uses a neighborhood sampling scheme to enable scalable training on large graphs, while maintaining high performance. In the following, we will describe the architecture of GraphSAGE.

Let $G = (V, E)$ be an undirected graph with $n = |V|$ nodes and $m = |E|$ edges. Each node $v \in V$ is associated with a feature vector $\mathbf{x}v \in \mathbb{R}^d$, where d is the dimension of the feature vector. We assume that the graph is represented by its adjacency matrix $\mathbf{A} \in 0, 1^{n \times n}$, where $\mathbf{A}ij = 1$ if there is an edge between nodes i and j, and $\mathbf{A}_{ij} = 0$ otherwise. We also assume that there is a node labeling function $y : V \to \mathcal{Y}$, where $\mathcal{Y}$ is the set of node labels.

The GraphSAGE model consists of multiple layers, where each layer learns node representations by aggregating information from the node's local neighborhood. At each layer, the neighborhood is sampled and a shared aggregation function is applied to the sampled nodes' feature vectors. Given a node v, its k-hop neighborhood is defined as

$$\mathcal{N}_k(v) = u \mid d(u, v) \leq k \tag{2.39}$$

where $d(u, v)$ is the shortest path distance between nodes u and v. The k-hop neighborhood of a node can be sampled using a random walk or a breadth-first search. The feature vectors of the nodes in the k-hop neighborhood of node v are aggregated using a shared function $f_\theta(\cdot)$

$$\mathbf{h}v^{(k)} = f\theta\left(\mathbf{x}_u \mid u \in \mathcal{N}_k(v)\right) \tag{2.40}$$

where $\mathbf{h}_v^{(k)} \in \mathbb{R}^d$ is the representation of node v at layer k and θ represents the learnable parameters of the aggregation function. The final node representation is obtained by concatenating the representations from all the layers

$$\mathbf{h}_v = \text{concat}(\mathbf{h}_v^{(0)}, \mathbf{h}_v^{(1)}, \ldots, \mathbf{h}_v^{(K)}) \tag{2.41}$$

where K is the number of layers.

The node representations are then used for downstream tasks such as node classification and link prediction. In summary, GraphSAGE is a GNN model that performs inductive learning on large-scale graphs using a neighborhood sampling scheme and a shared aggregation function. It learns node representations by aggregating information from the node's local neighborhood and concatenating the representations from all the layers to obtain the final node representation.

2.5.2 Contrastive Learning

Contrastive Learning is a technique used in unsupervised learning to learn useful representations of data by contrasting similar and dissimilar examples. It has become increasingly popular in the field of deep learning, particularly for computer vision tasks, but has also shown promise for natural language processing. The basic idea behind contrastive learning is to take two examples, a query and a key, and try to maximize the similarity between the query and the key while minimizing the similarity between the query and all other keys in the dataset. This encourages the model to learn representations that capture important features of the data.

Contrastive Predictive Coding: One popular implementation of contrastive learning is the Contrastive Predictive Coding (CPC) model, which was introduced by van den Oord et al. in 2018 [24]. CPC uses an auto-regressive model to predict the future context of a query example, and then contrasts this prediction with a randomly sampled key example from the same dataset. The loss function used in CPC is the InfoNCE loss, which is defined as follows

$$\mathcal{L}\text{CPC} = -\log \frac{\exp(\text{sim}(q,k)/\tau)}{\sum j = 1^N \exp(\text{sim}(q,k_j)/\tau)} \tag{2.42}$$

where q is the query example, k is the key example, k_j is a negative example, $\text{sim}(q, k)$ is the cosine similarity between the representations of q and k, τ is a temperature parameter that controls the sharpness of the distribution, and N is the number of negative examples used.

SimCLR: Another popular implementation of contrastive learning is SimCLR, which was introduced by Chen et al. in 2020 [85]. SimCLR uses a similar approach to CPC, but instead of predicting the future context of the query example, it applies various data augmentations to both the query and key examples and maximizes the similarity between the augmented examples. The loss function used in SimCLR is the NT-Xent loss, which is defined as follows

$$\mathcal{L}\text{SimCLR} = -\frac{1}{2N}\sum i = 1^N \log \frac{\exp(\text{sim}(z_i, z_i^+)/\tau)}{\sum_{j=1,j\neq i}^{2N} \exp(\text{sim}(z_i, z_j)/\tau)} \tag{2.43}$$

where $\mathcal{L}$SimCLR is the SimCLR loss function, N is the number of samples in a batch, z_i is the encoding of the i-th augmented view of an input, z_i^+ is the encoding of the corresponding positive example for z_i, $\text{sim}(\cdot, \cdot)$ is the cosine similarity function, τ is a temperature parameter to control the smoothness of the distribution, and $\sum j = 1, j \neq i^{2N}$ means the sum is taken over all samples in the batch except for z_i.

SimCSE: Similarity-based Contrastive Self-supervised learning for Embeddings) [69] aims to learn sentence embeddings that capture semantic similarity between sentences. The objective of SimCSE is to maximize the similarity between sentence embeddings of the same sentence while minimizing the similarity between sentence embeddings of different sentences. This is achieved using the following loss function

$$\mathcal{L}\text{SimCSE} = -\frac{1}{N}\sum i = 1^N \log \frac{\exp(\text{sim}(z_i, z_i^+)/\tau)}{\sum_{j=1,j\neq i}^{N} \exp(\text{sim}(z_i, z_j)/\tau)} \tag{2.44}$$

where $\mathcal{L}$SimCSE is the SimCSE loss function, N is the number of samples in a batch, z_i is the encoding of the i-th sentence, z_i^+ is the encoding of the corresponding positive example for z_i, $\text{sim}(\cdot, \cdot)$ is the cosine similarity function, τ is a temperature parameter to control the smoothness of the distribution, and $\sum j = 1, j \neq i^N$ means the sum is taken over all sentences in the batch except for z_i.

2.5.3 Text Data Augmentation for Deep Learning

Text data augmentation [76] is a technique used to increase the size and diversity of a text dataset by generating new examples through various transformations of existing text samples. This can help improve the performance of deep learning models trained on limited amounts of data. One popular method of text data augmentation is called **back-translation**, which involves translating a text sample from one language to another and then translating it back to the original language. This can introduce variations in the text while preserving the meaning, and has been shown to be effective for tasks such as sentiment analysis and machine translation. The formula for back-translation can be represented as follows

$$T_{\text{en}} \xrightarrow{f_{\text{de}\to\text{en}}} T_{\text{de}} \xrightarrow{f_{\text{en}\to\text{de}}} T'_{\text{en}} \tag{2.45}$$

where T_{en} is the original English text, $f_{\text{de}\to\text{en}}$ is the translation function from German to English, T_{de} is the German translation of T_{en}, $f_{\text{en}\to\text{de}}$ is the translation function from English to German, and T'_{en} is the back-translated English text.

Another method is **word replacement**, where certain words in a sentence are replaced with synonyms or other words that have similar meanings. This can be done using pre-trained word embeddings and word-level semantic similarity metrics. The formula for word replacement can be represented as follows

$$T' = \text{replace}(T, w_i, w'_i) \tag{2.46}$$

where T is the original text, w_i is the word to be replaced, w'_i is the replacement word, and replace is the function that replaces w_i with w'_i.

Text paraphrasing is another technique, where the original text is rephrased to create a new example with similar meaning. This can be done using techniques such as neural machine translation, where the original sentence is translated to another language and then back-translated to the original language with a different wording. The formula for text paraphrasing can be represented as follows

$$T' = f_{\text{para}}(T) \tag{2.47}$$

where T is the original text and f_{para} is the function that generates the paraphrased text.

Other data augmentation techniques include random deletion, random swap, and contextual augmentation. Random deletion involves randomly deleting words from the text, while random swap involves swapping adjacent words. Contextual augmentation involves modifying the text based on its context, for example, by replacing named entities with their synonyms.

2.6 Summary

In this chapter, we have overviewed various NLP techniques and their applications in cognitive screening systems. The review provides a comprehensive overview of NLP techniques, recent developments in language models, and DL models for NLP, which can be helpful for researchers and practitioners working in the field of NLP and evaluation systems. More specifically, we explored different preprocessing steps, such as part-of-speech tagging, named entity recognition, and parsing. We also reviewed the recent advances in language models, particularly BERT and sentence transformers, and their use in text classification. Finally, we discussed DL models for NLP, including GNNs, contrastive learning, and text data augmentation.

Chapter 3

AVCA: Autonomous Virtual Cognitive Assessment Application

As stated previously, cognitive assessment is an essential component of clinical evaluations for various conditions such as dementia, traumatic brain injury, and intellectual dysfunction. The use of clinically validated tools for cognitive assessment has been widely accepted and has been shown to be effective in identifying cognitive impairment. However, the administration of these tests can be time-consuming, and their interpretation requires specialized training. The emergence of AI and NLP techniques has opened up new possibilities for cognitive assessment. In this chapter, we present the Autonomous Virtual Cognitive Assessment (AVCA) application, which utilizes NLP and AI technologies to automate cognitive testing. AVCA is based on the Neurobehavioral Cognitive Status Examination (NCSE) and provides a comprehensive evaluation of cognitive functioning in multiple domains. The application is designed to be easy to use and provides an automated report of the patient's cognitive status. In what follows, we will discuss the development of AVCA, including the design of the application and the algorithms used for cognitive assessment. We will delve into the various components of the AVCA application, such as the Text Classifier, and provide an in-depth explanation of how the application works, its features, and the user interface. Furthermore, we will discuss the containerized approach, data flow, and processing pipeline used in the development of the application.

Figure 3.1: Language Part A (Speech Sample), where the patient is asked to describe the displayed picture. There is no score associated with this section.

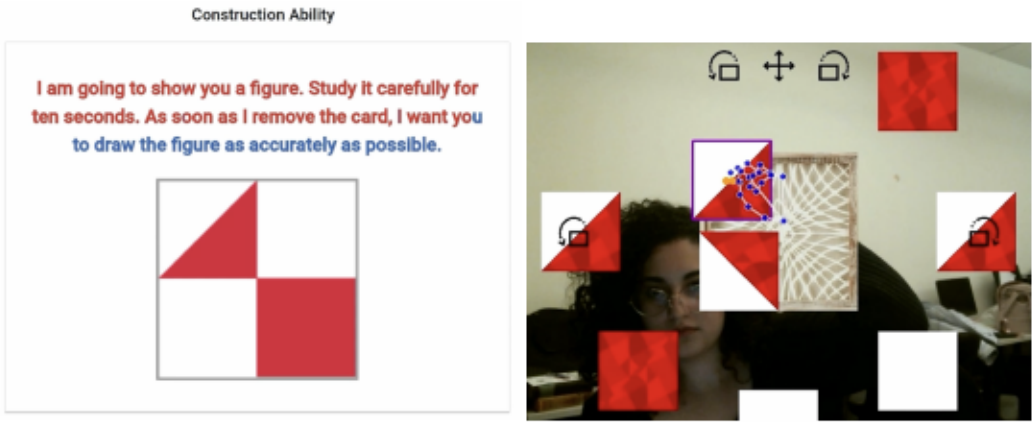
3.1 The NCSE Cognitive Test

The NCSE cognitive test is a comprehensive tool used to assess cognitive function in individuals. It is composed of several sections, each with a specific purpose. Here is a breakdown of the purpose of each section:

- **Orientation:** The purpose of this section is to assess an individual's awareness of their surroundings, including time, place, and person. It tests an individual's ability to recognize and recall basic information about their environment. The examiner may ask the individual questions such as "What is today's date?" or "Where are we right now?" to assess their awareness of time and place.

- **Attention:** This section is designed to assess an individual's ability to sustain attention, shift attention, and divide attention. It measures an individual's ability to focus on a task and maintain concentration. The individual may be asked to repeat a string of numbers forwards and backwards or to cross out specific letters in a string of letters as quickly as possible to assess their attention abilities.

- **Language:** The language section of the NCSE test is divided into four parts: speech sample, comprehension, repetition, and naming. Fig. 3.1 illustrate the first part of the language section. Comprehension tests an individual's ability to understand spoken language. Repetition



(a) (b)

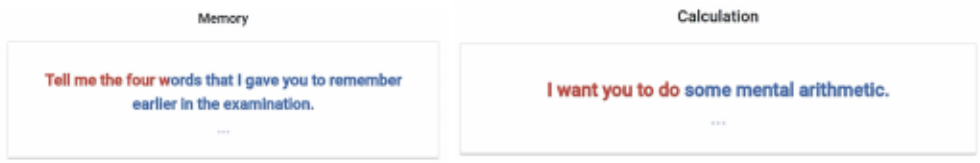
Figure 3.2: Construction Ability: A figure is displayed for a duration of 10 seconds for the purpose of the patients retaining it in their memory. Afterwards, the representation is taken down and red and white tiles are shown for the patient to construct.

tests an individual's ability to repeat words and phrases accurately. Naming tests an individual's ability to recall and name common objects and concepts. The examiner may ask the individual to follow a command such as "point to the ceiling" to assess their ability to understand spoken language. The individual may be asked to repeat a list of words or phrases that the examiner says to assess their ability to accurately repeat spoken language. The individual may be shown pictures of common objects and asked to name them to assess their ability to recall and name common objects and concepts.

- **Construction:** The construction section, shown in Fig. 3.2, assesses an individual's ability to perceive and recreate visual patterns and designs. It measures an individual's spatial perception and visual memory. The individual may be shown a picture or a set of blocks and asked to replicate the design to assess their ability to perceive and recreate visual patterns and designs.

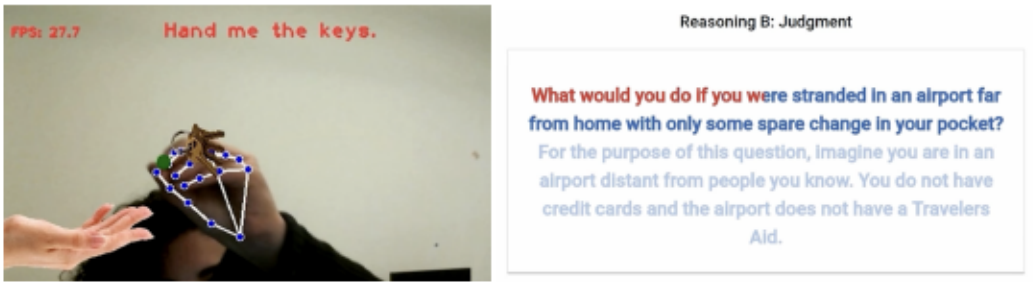
- **Memory:** The memory section is designed to assess an individual's short-term and long-term memory. It measures an individual's ability to recall information presented earlier in the test. The individual may be shown a list of words and asked to recall as many as they can after a certain amount of time has passed to assess their short-term and long-term memory abilities.

- **Calculation:** The calculation section tests an individual's ability to perform basic arithmetic



(a) (b)

Figure 3.3: Memory and Calculation: Further queries are posed using spoken communication.



(a) (b)

Figure 3.4: Language Part B (Comprehension) and Reasoning Part B (Judgment): (a) The patient is asked to do something using displayed virtual objects. (b) The patient is asked questions in the form "what would you do if...?".

operations, including addition, subtraction, multiplication, and division. It measures an individual's ability to perform mental calculations accurately. The individual may be asked to perform simple arithmetic problems such as $4 + 6$ or $9 - 3$ to assess their ability to perform mental calculations accurately. Fig. 3.3 shows the memory and calculation sections.

- **Reasoning/Similarity:** The similarity section assesses an individual's ability to recognize and identify relationships between objects and concepts. It measures an individual's abstract reasoning skills. The individual may be shown pictures or objects and asked to identify the common feature between them to assess their abstract reasoning skills.

- **Reasoning/Judgment:** The judgment section is designed to assess an individual's ability to make decisions and exercise sound judgment. It measures an individual's ability to reason logically and make appropriate decisions in real-life situations. The individual may be presented with hypothetical scenarios and asked to make a decision based on the information provided to assess their ability to make decisions and exercise sound judgment in real-life situations. Fig. 3.4 provides an illustrative example of the Reasoning and Judgment sections of the test.

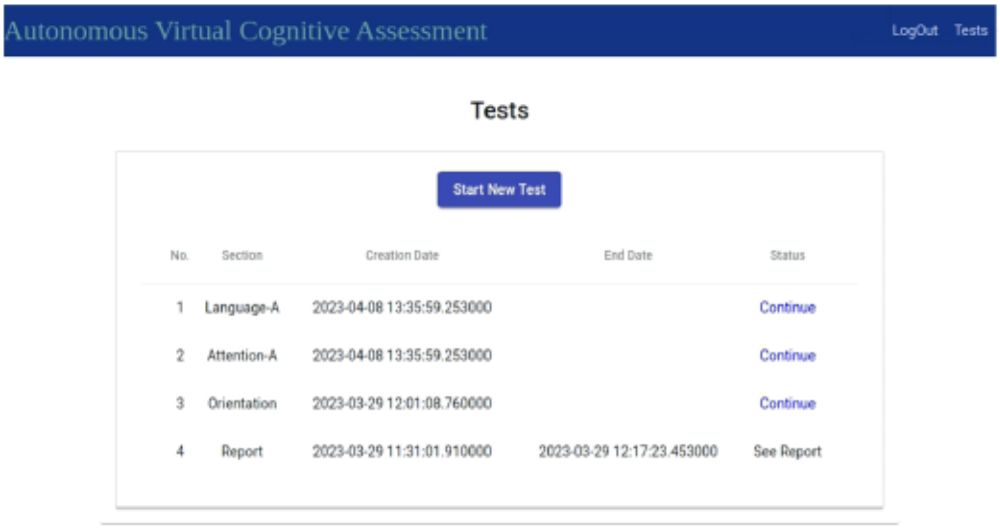


Figure 3.5: The web application's user interface: it allows patients to initiate a new test or resume their previous test from where they left off.

3.2 AVCA App Framework

The AVCA framework is a technology-based system used for cognitive assessments, designed to automate the testing process and provide more objective results. The framework uses ASR to record patient responses, which are then processed and analyzed by various modules, including string matching and semantic similarity, depending on the test section. In the calculation section, a system has been developed to track patient response times automatically. The AVCA framework is a promising solution that reduces the burden on clinicians and allows for more consistent and reliable testing. The framework also includes a section on Reasoning/Judgment, where patients are evaluated based on their understanding of the consequences of certain actions. An evaluation framework is developed to predict the score associated with the patient's response. Fig. 3.5 shows the user interface of the application.

3.2.1 Modules for Assessment

The AVCA framework consists of various modules that are used for the automated assessment of cognitive abilities. Three main modules used in the framework are string matching, phonetic matching, and semantic similarity modules, i.e.,

- **String Matching:** This module measures the difference between the patient's answer and the

correct response using the Levenshtein Distance (LD) algorithm. The LD algorithm calculates the fewest number of symbol substitutions, insertions, and deletions required to change one word into another. If the difference is 30% or less, the response is considered correct.

- **Phonetic Matching:** This module compares words based on their pronunciation, rather than their spelling. This can be useful when dealing with misspellings or different spellings of the same word. One common method for phonetic matching is the Soundex algorithm, which converts each word to a code based on the sound of its consonants. The Soundex code is generated by taking the first letter of the word, followed by three numbers that represent the sound of the remaining consonants. Another popular method is the Metaphone algorithm, which is an improved version of Soundex that also takes into account the pronunciation of vowels. The Metaphone code is generated by analyzing the phonetic patterns of the word and assigning a code based on those patterns.

- **Semantic Similarity:** This module compares the semantic similarity of two input text phrases as opposed to their lexicographical similarity, using the cosine similarity of their embeddings. The text phrases are fed to a language model such as BERT followed by a pooling layer to form the embedding vectors.

The aforementioned modules are implemented as follows:

- For the Orientation, Language/Naming, Calculations, and Reasoning/Similarity sections, the string matching module is used to compare the patient's response with its corresponding correct response stored in the database. In case of inaccuracies in the ASR module, the patient's response is preprocessed using spell correction or phonetic matching.

- For the Attention, Language/Repetition, and Memory sections, the string matching module is used after tokenization. Tokenization refers to breaking down the patient's response into smaller units, such as words or phrases.

- For the Language/SpeechSample and Reasoning sections, the semantic similarity module is

used. The preprocessing includes POS tagging to recognize the number of verbs in the response and noise removal such as stopword and punctuation removal to access the core keywords in the patient's response.

3.2.2 Assessment Enhancements

Further algorithmic enhancements have been made to the calculation and reasoning/judgment sections. In the calculation section of cognitive assessments, patients are asked to perform a series of arithmetic calculations and are given a time slot to respond. Traditionally, the clinician notes the precise moment when the patient provides the right response, but in an automatic version, the timestamp is automatically stored. To enable autonomous calculation assessment using audio, we had to address several challenges. Let us denote the patient's spoken response as S, which is transcribed into text form using our ASR system. Since patients may correct themselves mid-response and begin computing while speaking, we needed to extract each numeral from their response and save them with their respective timestamps. When our ASR system transcribes the patient's spoken response, it represents numbers using their alphabetical names rather than their numerical values. To address this, We define the extracted set of numerals as $N = n_1, n_2, ..., n_k$, where k is the number of numerals extracted. To convert the alphabetic names of the numerals to their numerical values, we developed an algorithm that matches words in the input text against a set of predetermined patterns. We define the converted set of numerals as $C = c_1, c_2, ..., c_k$, where c_i is the numerical value of the i^{th} numeral in N. The final answer provided by the patient is denoted by A, which is the last numeral in N, i.e., $A = n_k$. All other numerals in N, except A, are classified as incorrect responses. We also saved the original alphabetic text response given by the patient as T. Further elaboration on saving the timestamp of each token can be found in Section 4.4.3.

Regarding the Reasoning/Judgment section, an evaluation framework has been developed based on text classification to predict the score associated with the patient's response. In this section, patients are evaluated based on their understanding of the consequences of certain actions.

3.3 Sentence Embedding of AVCA

The proposed AVCA framework uses an unsupervised contrastive learning model in its sentence embedding module. During training, two sentences are treated as positive samples, while the other sentences are negative samples. The model is trained using a contrastive loss objective function to bring representations of the same class closer and separate them from other sentences. Data augmentation is required to minimize overfitting, and various techniques like dropout, weight penalties, and symbolic and neural augmentations have been explored. The proposed method uses Pegasus paraphrasing as a data augmentation technique, which maintains the semantic meaning of each input sample. Pre-trained BERT or RoBERTa models are used to generate embeddings of original and augmented sentences. These embeddings are fed to the encoder with two different dropout masks, and attention probabilities are applied to fully-connected layers in the Transformer's standard training. The unsupervised model is trained on 1 million sentences from English Wikipedia and MNLI and SNLI datasets. Semantically similar neighbors are grouped together, and unrelated ones are separated using cross-entropy target with in-batch negatives while adhering to the contrastive framework. Further elaboration is provided in 4.3.2.

3.4 Text Classifier of AVCA

The Reasoning/Judgment section of AVCA requires the assessment of patients' understanding of the consequences of certain actions through a set of questions with predetermined correct responses. To automate this process, text classification is performed on our in-house dataset using various machine learning models such as SVM, FFNN, CNN, and GNN. The goal is to test the accuracies of these models in predicting the score associated with patients' responses to the set of questions in the reasoning/judgment section. The use of machine learning algorithms improves the efficiency and reliability of cognitive assessments, reducing the burden on clinicians and enhancing patient care. TF-IDF formulation and

3.4.1 Dataset

The proposed framework uses an in-house dataset consisting of 830 patient responses to develop the reasoning module. The responses were collected during in-person cognitive assessments by a physician, and the same judgment task question was asked to each participant. The dataset was partitioned in such a way that 99% of the data was utilized for training while the remaining 1% was reserved for testing, which is different from the test-train separation ratio described in section 4.4.4. To maintain consistency, the same set of training and testing data was used for all the following models described in this section.

3.4.2 Machine Learning Models with TF-IDF

In this subsection, we have evaluated the performance of the AVCA classifier using various ML models. The inputs were preprocessed text retrieved from our in-house-dataset. The best-performing model was the Support Vector Machine (SVM) based classifier, which achieved an accuracy of 92% and 86% *F1* score when trained on the preprocessed and tokenized responses, vectorized using TF-IDF features 4.4. The tokenization process removes stop words, lower-cases the responses, and tokenizes them using NLTK's TweetTokenizer. We experimented with a range of n-grams and found that using unigrams and bigrams gave the best performance for the task.

3.4.3 Deep Learning Models with Sentence Embedding

We implemented classifiers based on Feed Forward Neural Network (FFNN), Convolutional Neural Network (CNN) and Graphical Neural Network (GNN) models. The inputs were preprocessed text retrieved from our in-house-dataset transformed into RoBERTa sentence embedding vectors of length 1024.

Feed-Forward Neural Network

Through experimentation with hyperparameters, including the number of hidden layers, activation functions, batch size, and epochs, the FFNN-based classifier achieved an accuracy of 93% and an F1 score of 89% on in-house data after hyperparameter tuning. Let X be the input layer that takes

the RoBERTa embedding vector of patients' responses. The model contains two hidden layers with ReLU and tanh activation functions, denoted by H1 and H2, respectively. The hidden layers are followed by dropout layers, which are represented by D1 and D2. The final layer of the model is a sigmoid layer, denoted by Y, that produces a floating-point number for each sample. The output of the model is given by

$$Y = \sigma(W_3 D_2(H_2 D_1(H_1 X + b_1) + b_2) + b_3) \tag{3.1}$$

where W_3 is the weight matrix for the output layer, b_1, b_2, and b_3 are bias terms, and $\sigma(\cdot)$ is the sigmoid activation function.

The model is trained by minimizing the focal loss 4.5, and the hyperparameters are optimized using the grid search method with a checkpoint callback to minimize validation loss. The model is trained using batch sizes of 12, 32, 64, and 128 over 100 epochs and evaluated using a hard cutoff threshold of 0.5 for test samples.

Convolutional Neural Network

Let X be the input layer that takes the embeddings of the text data. The model consists of two convolutional layers, denoted by C1 and C2, with 64 and 32 filters respectively. Both convolutional layers have a kernel size of 3 and are followed by max-pooling layers with a pool size of 2 and a dropout rate of 0.2. The output of the second max-pooling layer is then flattened and passed through a dense layer with a sigmoid activation function to produce a binary classification output. The output of the model is given by

$$Y = \sigma(W_3(D_2(\max(0, C_2(D_1(\max(0, C_1(X))))))) \tag{3.2}$$

where W_3 is the weight matrix for the output layer, D_1 and D_2 are dropout layers, and $\sigma(\cdot)$ is the sigmoid activation function.

The model uses binary focal loss 4.5 function to handle class imbalance by assigning more weight to the minority class. The hyperparameters such as batch size, learning rate, and number of

epochs are fine-tuned using the Adam optimizer with a learning rate of 5e-4 and a batch size of 16, 32, 64, and 128. The model is trained for a maximum of 100 epochs with early stopping to avoid overfitting. After hyperparameter tuning, the CNN classifier achieved an accuracy of 94% and an F1 score of 91% on the in-house data.

Graph Neural Network

We implemented an Inductive Graph Neural Network that is designed for node classification tasks on graph-structured data, where nodes represent sentences and edges represent the similarity between sentence RoBERTa embeddings. Let $G = (V, E)$ be a graph with nodes representing sentences and edges representing the similarity between sentence RoBERTa embeddings. The input features of each node are its corresponding sentence RoBERTa embedding. To generate edges between the nodes, a cosine similarity threshold of 0.7 is set, and a similarity matrix S is computed using the cosine distance metric. A boolean adjacency matrix A is then generated by setting the values in the similarity matrix that exceeded the threshold to 1, and those that did not to 0. The resulting adjacency matrix is used to create a list of edges E', where each edge represents a pair of nodes that are connected by an edge. The resulting graph is represented in fig. 3.6. For each node v in V, its feature vector h_v is computed using the GraphSAGE model

$$h_v^{(k)} = \text{concat}(h_v^{(k-1)}, \text{AGGREGATE}(h_u^{(k-1)}, \forall u \in \mathcal{N}(v))), \tag{3.3}$$

where $\mathcal{N}(v)$ is the set of neighbor nodes of v in the graph, AGGREGATE is a neighborhood aggregation function, and $h_v^{(k)}$ is the feature vector of v after k layers of GraphSAGE.

The output of the model is a binary classification prediction y_v for each node v, which is computed using a fully connected linear layer with one output channel and a sigmoid activation function

$$y_v = \sigma(W_o h_v + b_o), \tag{3.4}$$

where W_o and b_o are learnable weights and bias, respectively, and σ is the sigmoid function.

The model is trained on the training set using the binary focal loss function 4.5 and the Adam

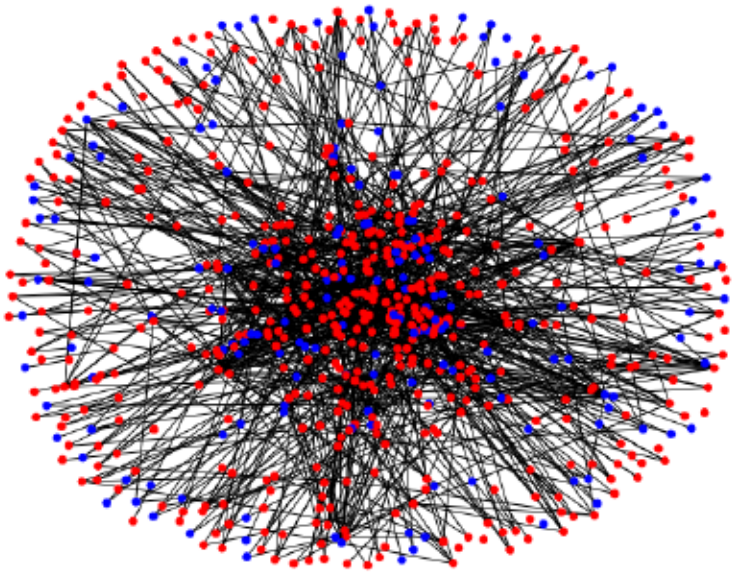


Figure 3.6: A visual display of the resulting graph with nodes colored by a set of colors based on the values of the labels. The blue nodes correspond to the nodes labeled as 1, whereas the red nodes correspond to the nodes labeled as 0.

optimizer. The model is evaluated on the validation set at each epoch, and the training is continued for 300 epochs. The model with the best validation accuracy is chosen for evaluation on the test set. In our study, we employ a 1% split for both the validation and test sets to ensure a representative distribution of correct and incorrect responses in the face of an imbalanced dataset. Due to the limited size of our dataset, we reserve a small portion for validation and testing. Validation data plays a crucial role in the GraphSAGE model by enabling the evaluation of its performance and tuning of its hyperparameters. During training, the model learns to make predictions on the training data it has already seen. However, the ultimate goal is to make accurate predictions on new and unseen data, and validation data helps to simulate this scenario. By assessing the model's performance on validation data, we can detect and address issues such as overfitting or underfitting, thus improving the model's generalization ability. After hyperparameter tuning, the GNN classifier achieved an accuracy of 97% and an F1 score of 95% on the in-house data.

3.4.4 Evaluation

Table 3.1 illustrates results based on different models. Inductive GNNS can outperform other models in text classification tasks due to their ability to effectively handle the underlying graph

Table 3.1: Performance comparison based on different classifiers.

Model	Accuracy	*F1*
ktrain [83] + RoBERTa	88%	83%
SVM + TF-IDF	92%	86%
AVCA FFNN + RoBERTa	93%	89%
AVCA CNN + RoBERTa	94%	91%
AVCA GNN + RoBERTa	**97%**	**95%**

structure of textual data. Inductive GNNs can also generalize to unseen data, which is particularly useful in text classification tasks where the model needs to classify new texts that were not present during the training phase. By utilizing the graph structure of language and the ability to generalize to new data, inductive GNNs can achieve higher accuracy in text classification tasks compared to other deep learning models.

3.5 Design of AVCA

The design of the Autonomous Virtual Cognitive Assessment (AVCA) application has been carefully considered to provide a robust and scalable solution for cognitive assessment. The application is built using a containerized approach, with three main containers: frontend, web app, and database.

3.5.1 Containerized Approach

The AVCA application has been designed using a containerized approach, which involves packaging the software into lightweight and portable containers. This approach provides several benefits, including easier deployment, improved scalability, and better resource utilization. By using containers, each component of the application can be isolated and scaled independently. For example, if the frontend is experiencing high traffic, more instances of the frontend container can be deployed without affecting the other components of the application. Similarly, the database container can be scaled up or down depending on the amount of data being processed. The use of containers

also makes it easier to deploy the application in different environments, such as development, testing, and production. Each environment can have its own set of container instances with different configurations, allowing for easier management and deployment.

3.5.2 Web App and API

The Autonomous Virtual Cognitive Assessment (AVCA) application is built using a web app container that runs on Flask, a micro web framework written in Python. Flask is used to build the RESTful API that the AVCA application uses to communicate with the front-end Angular interface. Flask-RESTful is an extension of Flask that adds support for building RESTful APIs. It allows developers to easily define API endpoints, request handling, and response formatting. Flask-RESTful provides a simple way to build APIs that can be consumed by various clients, including the AVCA application's Angular front-end.

The API is organized into several resources, such as users, patients, and tests, each with its own set of endpoints. For example, the "users" resource has endpoints for creating a new user, logging in, and retrieving user details. The "patients" resource has endpoints for creating a new patient profile, retrieving patient details, and creating and retrieving test results. The API endpoints are accessed using HTTP requests, with responses returned in JSON format. This allows the front-end Angular application to easily communicate with the Flask back-end and retrieve the necessary data to display to the user. The Flask application uses SQLAlchemy, a popular Object Relational Mapper (ORM), to interact with the Microsoft SQL Server database. The database classes are defined using Flask models, which map to the database tables. SQLAlchemy provides an easy way to work with the database and perform complex queries, while also ensuring data consistency and security.

3.5.3 Data Flow and Processing Pipeline

The AVCA application has been designed using a containerized approach with three main containers: frontend, web app, and database. The frontend is built using Angular and Nginx proxy, while the web app container is built using Flask and WSGI. The database used is Microsoft SQL Server, which is connected to the web app container using SQLAlchemy. To connect the database to the web app container, SQLAlchemy, a Python SQL toolkit and Object-Relational Mapping (ORM)

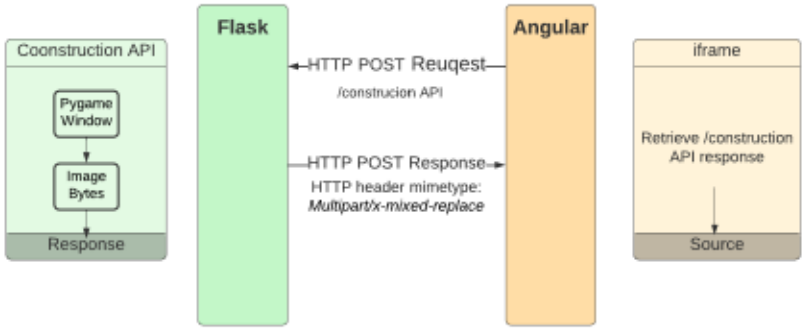


Figure 3.7: Angular-Flask data flow for video streaming.

library, is used. SQLAlchemy provides a high-level SQL abstraction, allowing for efficient and easy database manipulation. It also allows for seamless integration with Flask and other Python web frameworks.

The Flask application uses Flask-RESTful API to provide several resources that can be called from the main app. Users can create their own profile as a patient upon logging in, and the user and patient classes have a one-to-one relationship. Patients can create multiple tests, with each test mapped to each cognitive section (orientation, attention, language, construction, memory, and judgment) using a one-to-one relationship. The data flow in the application follows a processing pipeline, starting from the user logging in and creating a patient profile. The patient can then create a test, which is mapped to each cognitive section, and the test data is stored in the database.

The AVCA application employs the Pygame and OpenCV libraries to facilitate the Language Part B and Construction Ability sections. These sections leverage a camera and associated algorithms to capture and process video, which is subsequently streamed to the frontend as shown in 3.2, 3.4. Specifically, the OpenCV library is utilized to enable the streaming of the Pygame window and its display within the AngularJS-based frontend. To achieve this, the client's request for the Pygame window's content is fulfilled through an HTTP post request. The use of the multipart/x-mixed-replace MIME type in the HTTP header permits the server to transmit updated information dynamically to the web browser. In the frontend, the required inputs are transmitted via the post request utilizing the FormData object. Ultimately, the video stream response is displayed within an iframe element. This approach presents a robust and effective method of video streaming within the AVCA application. The video streaming data flow diagram is shown in 3.7. At the end of each

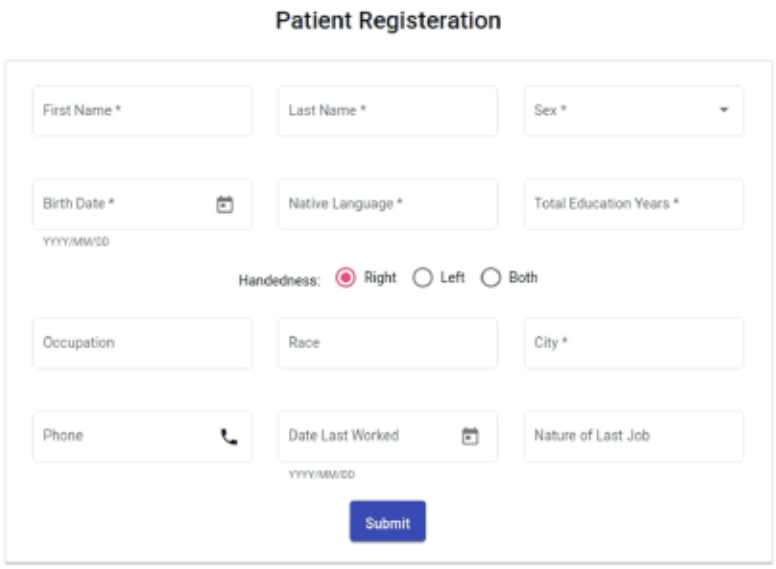


Figure 3.8: Patient Registration User Interface

test, the results are presented to the patient and the clinician, allowing for a detailed analysis of the patient's cognitive functioning.

3.5.4 Description of the User Interface and Features

The Autonomous Virtual Cognitive Assessment (AVCA) application has a user-friendly interface that is designed to guide users through the cognitive assessment process. When a new user logs in, they are prompted to create their patient profile, which includes their personal information and medical history as shown in fig 3.8.

Once the user has created their patient profile, they can begin the cognitive assessment process. The assessment is divided into sections, including orientation, attention, language, construction, memory, and judgment. Each section has a set of questions that are designed to assess the patient's cognitive abilities in that specific area.

During the test, the patient is not required to manually input or click on anything as the entire process is automated. The web application presents inquiries about the patient's personal details, and their responses are recorded. If the patient is prompted to manipulate an object or solve a puzzle, the user interface displays everything needed, and the patient must employ virtual hand movements to complete the task. The web camera records the patient's hand gestures for analysis.

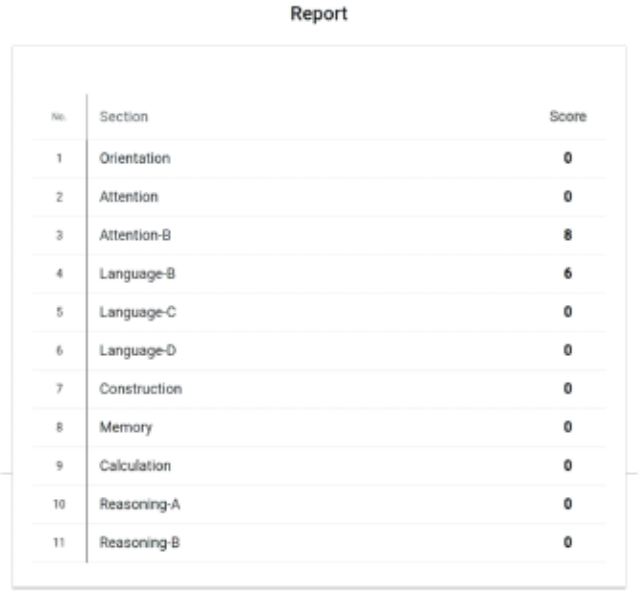


Figure 3.9: The user interface for the Patient's Report Form.

The app takes information from the patient using a microphone and camera only, which ensures that the testing process is non-intrusive and does not require any special equipment. The questions are presented in a clear and concise manner, with the option to repeat a question or skip to the next one if needed. The interface also provides feedback to the patient, indicating if their response was correct or incorrect. At the end of the assessment, the patient can view their results, which include a detailed analysis of their cognitive abilities in each section as shown in 3.9, 3.10. The AVCA application also includes several features to enhance the user experience. For example, the user can save their progress at any point during the assessment and return to it later. Overall, the AVCA application provides an easy-to-use interface and features that make the cognitive assessment process more accessible and efficient.

3.6 Resources

In this section, we describe the main resources in NLP research and development, including software and scientific libraries for running large-scale state-of the art models, focusing on Transformers.

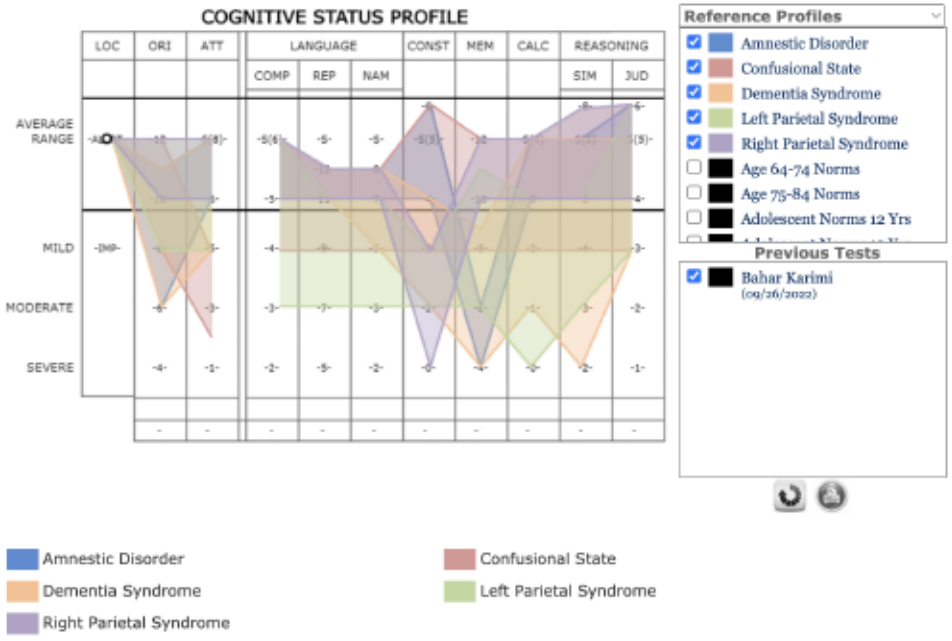


Figure 3.10: The user interface for the Patient's Cognitive Status Profile.

3.6.1 Software

NLP attracted, in the past decades, a consistent number of developers and scientists who have made available a plethora of libraries, tools, and scripts to handle with both low-level NLP modules (e.g. tokenization, PoS tagging) and high-level systems (e.g. document classifiers, models). In the following we provide a brief description of the main tools that we used for performing NLP tasks.

Natural Language Toolkit (NLTK) is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

Transformers provides general-purpose architectures (BERT, GPT-3, RoBERTa, XLNet. . .) for Natural Language Understanding (NLU) and Natural Language Generation (NLG) with over 32+ pretrained models in 100+ languages and deep interoperability between TensorFlow 2.0 and PyTorch. The library is curated by Huggingface, an NLP-focused startup with a large opensource community. The library exposes APIs to use many wellknown pre-trained

transformer architectures described in the previous sections.

Google Text-to-Speech (**gTTS**) is a Python library and command-line tool that provides an interface to Google's Text-to-Speech API. The tool allows users to convert text into a natural-sounding voice in over 200 different languages and dialects. It uses a deep neural network-based text-to-speech (TTS) model that generates high-quality speech with customizable pitch, speed, and volume. The model is based on a combination of concatenative and parametric synthesis, with a neural vocoder for generating speech waveforms. The library also includes support for various audio file formats, including WAV, MP3, and OGG, making it easy to integrate TTS into any project or application.

Fuzzywuzzy is a Python library that provides simple and easy-to-use tools for string matching and fuzzy string searching. It was developed by SeatGeek and released in 2011. The library is built on top of the Levenshtein distance algorithm and uses a combination of various string matching techniques to provide a flexible and efficient matching system.

PyPhonetics is a Python library for phonetic string matching. The library provides a variety of phonetic algorithms such as Soundex, Metaphone, and Double Metaphone that convert words into phonetic codes, allowing for fuzzy string matching.

3.7 Summary

This chapter provides an in-depth exploration of the AVCA application, starting with an introduction to its purpose and framework. The modules required for the NCSE evaluation system are presented along with their usage in each section of the assessment. The chapter also introduces a novel Sentence Embedding module that utilizes unsupervised contrastive learning. Furthermore, a classifier is implemented for the reasoning/judgment section using GraphSage on an in-house dataset, resulting in high accuracy. The design of the AVCA application is described in detail, including its user interface, data flow, and implementation. The chapter concludes with a list of resources used in the development of the AVCA application. The comprehensive and detailed account of the AVCA application presented in this chapter highlights the novelty and effectiveness of

its cognitive assessment approach.

Chapter 4

AVCA: Autonomous Virtual Cognitive Assessment

Fast-paced and ever-growing advances in Artificial Intelligence (AI) and Deep Neural Network (DNN) models have initiated works on autonomous monitoring/screening systems to assess individuals' cognitive state. In conventional cognitive assessment systems, a physician evaluates the mental abilities of the brain by rating the patient's numerical, verbal, and logical responses. Development of an autonomous cognitive assessment system that assist the physician is both of significant importance and a critically challenging task. As a first step towards achieving this objective, in the paper an Automated Virtual Cognitive Assessment (AVCA) framework [1, 2] is proposed that integrates Natural Language Processing (NLP) and hand gesture recognition techniques. The proposed AVCA framework provides individual scores in the seven major cognitive domains, i.e., orientation, attention, language, contractual ability, memory, calculation, and reasoning. More specifically, the AVCA framework is an autonomous cognitive assessment system that receives audio and video signals in a real-time fashion, and performs semantic and synthetic analysis using NLP techniques and DNN models. Real-time video processing engines of the AVCA monitors hand motions and facilitate simpler engagement for visual evaluation. Additionally, we propose an efficient model to facilitate human-machine interactions from speech recognition to text classification. In particular,

an unsupervised contrastive learning framework is proposed using Bidirectional Encoder Representations from Transformers (BERT) that outperforms its state-of-the-art unsupervised counterparts achieving an average of 77.84% Sparsman's correlation on standard Semantic Textual Similarity (STS) tasks.

The remainder of the chapter is organized as follows: Section 4.1 provides a refreshing introduction. Section 4.2 analyzes various language models and speech recognition techniques for building a virtual assistant system. In Section 4.3, the AVCA's NLP module, the suggested sentence embedding framework, the COGNISTAT testing tasks utilizing hand gesture detection are described. Experiments and results are presented in Section 4.4 where various tasks of cognitive assessments are explained along with how we implemented their automated version. Finally, Section 4.5 concludes the paper.

4.1 Introduction

The COVID-19 pandemic has abruptly and undoubtedly changed the world as we knew at the end of the second decade of the 21st century. It is expected for the negative effects of the COVID-19 to continue for years to come. To overcome these long-lasting effects and be prepared for future possible ones, special attention should be devoted to the fact that seniors, aged 60 and over, are more vulnerable during the pandemic area. In addition to higher risk of infections, seniors are also at higher risk of suffering from mental and cognitive issues. Given an expected and alarming population ageing in near future, it is crucial and of significant importance to develop innovative and advanced autonomous screening systems for unconstraint environments. Of particular interest to this chapter is development of advanced autonomous cognitive screening system via integration of Signal Processing (SP), Artificial Intelligence (AI), and Deep Learning (DL) models. In particular, the focus is on development of an AI-empowered avatar that autonomously performs cognitive screening tests.

Conventionally, cognitive health monitoring is performed manually by physicians, however, there is an urgent and unmet quest to develop autonomous systems for cognitive (mental) health screening. This is of significant importance as in this point in time we are dealing with long lasting

effects of the COVID-19 pandemic, among which isolation and mental health issues are devastating. Generally speaking, mental state refers to the current set of emotions and psychological states of a person [25]. It was argued [26] that poor mental health, caused due to excessive stress and/or other negative emotions, can lead to an added emotional and/or economic burden on individuals suffering from mental health problems. Different modalities can be used to perform emotion state monitoring. In brief, emotions can be recognized from modalities such as speech, facial expressions, physiological signals, behavior (gesture/posture), and/or online social media textual posts. Reference [27] reviews commonly used recognition methodologies to perform emotion expressions from facial images. Another modality that has been extensively studied for emotion recognition is through Electroencephalogram (EEG) signals [28]. The concept of EEG-based emotion recognition provides a unique solution with application in Human Machine Interfaces (HMIs) and health care. Reference [29] focused on multimodal emotion recognition integrating EEG signals with other modalities. Reference [30] integrated muscular signals obtained from Electromyography (EMG) with other modalities for the task of human emotion recognition. Finally, in [31], user-generated online social media textual posts are used for development of an emotion recognition system. Early detection of deteriorating mental health, e.g., due to excessive stress and/or anxiety, can prevent conditions such as burnout and depression in highly demanding professions such as nurses and surgeons [32]. In what follows and before presenting contributions of the chapter, we briefly outline basics of cognitive state monitoring systems followed by an overview of different cognitive assessment tools.

Cognitive Assessment: Over the years, several solutions have been proposed to monitor mental states , however, mental states have traditionally been monitored using subjective questionnaires. Different questionnaires are developed and applied for various mental states. Such questionnaires, typically, ask about a person's mental health on a regular basis. The response to such questionnaires is typically in rating format (continuous scale) or one of several specified options (discrete scale). Discrete categorization of emotions includes the six fundamental emotions of happiness, sadness, fear, anger, disgust and surprise. In this scale, more complicated emotions (such as exhaustion or anxiety) are viewed as a mixture of these fundamental ones. Other more complex emotions

(such as fatigue or anxiety) are considered to be a combination of these basic emotions under this framework [33].

An alternative approach to mental state monitoring via questionnaires is by using audio-visual data generated by individuals. Humans communicate emotions with one another using speech, facial expressions, and body gestures. Hence, using these modalities has been a popular method for emotion state monitoring. Another approach to evaluate mental state is by using neurophysiological signals. For example, in [34] different algorithms are developed exploring neurophysiological correlates of mental state. Reference [35] focused on mental state of pilots and looked at neurophysiological changes corresponding to the mental state transitions. Finally, authors in [36] focused on evaluating mental state of construction workers via neurophysiological signals. These signals reflect the activity of the central and autonomic nervous systems. Central Nervous System (CNS) activities can be captured by monitoring electrical, magnetic or hemodynamic activities of the brain. Electroencephalography (EEG) is one such method that captures the electrical activity of the brain using electrodes placed on the scalp surface. EEG has high temporal resolution compared to other neuro-imaging approaches and thus can be used for real-time mental state monitoring [37]. The Autonomic Nervous System (ANS) activity, in turn, regulates bodily functions and can be captured using physiological signals such as Electrocardiogram (ECG), respiration signal, skin temperature, and Galvanic Skin Response (GSR), to name a few. The ANS can be further divided into two categories, i.e., (i) The Parasympathetic Nervous System (PNS), which relaxes the body, and; (ii) The Sympathetic Nervous System (SNS), which is associated with the flight-or-fight response. These two systems impact physiological signals differently and can help distinguish between different mental states [38]. Combining the different signals to create a multi-modal system for mental state monitoring has shown improved performance. This is due to the fact that different signals provide complementary information, as well as added robustness to sensor failure and high noise levels that could be present in individual modalities [49].

A mental state examination provides you with a momentary view of a patient's emotions, thoughts, and behaviour as they are observed. In contrast, a mental status test measures present

mental capacity by assessing general appearance, behavior, any unusual or bizarre beliefs and perceptions (e.g., delusions, hallucinations), mood, and all aspects of cognition (e.g., attention, orientation, memory).

There are three major means (types) of assessing a patient's mental status [50]. One type focuses on the patient's capacity to carry out instrumental activities and activities of daily life in order to assess whether or not they have dementia and how severe it is. In the second form of assessment, "screening tests" or "omnibus tests" are used. These quick tests are carried out separately from the patient's history and physical examination. According to [51], the Mini-Mental Status Examination (MMSE) and the Montreal Cognitive Assessment (MoCA) are the most widespread psychometric cognitive screening tests worldwide. It is worth mentioning that MMSE is seemingly more sensitive to severe dysfunctions while the MoCA is more sensitive to mild-to-moderate dysfunctions. Recently, MMSE and MoCA have also been effective in detecting cognition deficits following COVID-19 [52]. The third means of assessing a patient's mental status is by using specific neuropsychological tests that focus on specific domains of cognition. According to [50], these evaluations are performed on patients who exhibit behavioural and cognitive abnormalities, and they should frequently be accompanied by laboratory and neuroimaging tests that can help identify the underlying pathologic process and enable the development of efficient therapeutic and management strategies.

In summary, cognitive assessment is effective to test for cognitive impairment - a deficiency in knowledge, thought process, or judgment. Psychiatrists often perform cognitive testing during the mental status exam, however, when cognitive impairment is suspected, the cognitive assessment can obtain a more detailed analysis by surveying the neuropsychological domains. This detailed investigation of cognition can diagnose major cognitive impairment (i.e., dementia) and mild cognitive impairment, evaluate traumatic brain injuries, help determine decision-making capacity, and survey intellectual dysfunction.

Cognitive Assessment Tools: Cognitive assessment is often performed using clinically validated tools, such as the Mini Mental State Examination (MMSE) [39], the Montreal Cognitive Assessment (MoCA) [40], and Neurobehavioral Cognitive Status Examination (NCSE) or COGNISTAT [41],

which are briefly introduced below:

- *Mini Mental State Examination (MMSE):* Perhaps the most extensively used clinical measure for determining the degree of cognitive impairment is the MMSE. The MMSE was created as a bedside measure to assess older patients' cognitive health in clinical settings; it is frequently used in surveys to test for dementia and cognitive impairment. It is quick and simple to manage, has an excellent track record of reliability, although several restrictions have been noted. The MMSE may not detect focal brain damage or mild dementia, and validity may be poor in samples that contain psychiatric patients. [53]. The MMSE has a maximum score of 30. Normal is defined as a score of 25 or higher. A score of less than 24 is usually considered abnormal, indicating possible cognitive impairment.

- *Montreal Cognitive Assessment (MoCA):* MoCA was created as a screening tool for patients who have mild cognitive complaints and typically score in the normal range on the MMSE [40]. The final version of the MoCA is a one-page 30-point test administered in 10 minutes. Details on the specific MoCA items are as follows: The short-term memory recall task (5points) involves two learning trials of five nouns and delayed recall after approximately 5 minutes. Visuospatial abilities are assessed using a clock-drawing task (3 points) and a three-dimensional cube copy (1 point). Multiple aspects of executive functions are assessed using an alternation task adapted from the Trail Making B task (1 point), a phonemic fluency task (1 point), and a two-item verbal abstraction task (2 points). Attention, concentration, and working memory are evaluated using a sustained attention task (target detection using tapping; 1 point), a serial subtraction task (3 points), and digits forward and backward (1 point each). Language is assessed using a three-item confrontation naming task with low-familiarity animals (lion, camel, rhinoceros; 3 points), repetition of two syntactically complex sentences (2 points), and the aforementioned fluency task. Finally, orientation to time and place is evaluated (6 points).

- *Neurobehavioral Cognitive Status Examination (NCSE)* The NCSE or COGNISTAT [41] is another cognitive screening test of separate subtests assessing major domains of cognitive

functions. These are orientation, attention, language (comprehension, repetition and naming), construction, memory, calculation, similarity and judgement. COGNISTAT was found to be sensitive to the cognitive effects of stroke, although there was little discrimination between left-sided and right-sided strokes [44]. COGNISTAT is more sensitive than MMSE to identify cognitive impairment in geriatric inpatients with a variety of medical conditions as well as individuals with brain tumours and cerebrovascular disease. [42]. The sensitivity of MoCA-J and COGNISTAT is thought to be nearly comparable, making them both suitable for detecting Mild Cognitive Impairment (MCI) in Parkinson's Disease patients (PD). These outcomes concern patients who have previously received a PD diagnosis [43].

The developers of COGNISTAT claim that their test "represents a new method to quick cognitive testing." This evaluation provides the clinician with a unique profile of the patient's cognitive status by independently assessing numerous domains of cognitive functioning. [41]. The NCSE is a tool that employs an initial screening question for each cognitive domain; if the patient is unable to respond, the tool determines the severity of impairment. If the patient passes the screen, the ability involved is assumed normal, and no further testing is done in that section.Instead of using a single overall score, the points given for correct responses are summed within each category of cognitive ability.

Contributions: In this study, we aim to implement an autonomous system for the evaluation of patients with cognitive impairments. In particular, the main objective is implementing the activities associated with the NCSE assessment system in an autonomous fashion. In achieving this objective, we have formulated the following research questions:

- The first targeted research question is what are the key components of the NCSE assessment system that can be implemented autonomously.
- The second research question is what technologies are needed (available) to be utilized for automated implementation of the identified components.
- The third research question is what are the new methodologies that are needed to be developed (or improved) to provide enhanced performance.

To achieve the aforementioned objective, we made the following contributions:

- Our first contribution is designing a Human Machine Interface (HMI) system, referred to as the Autonomous Virtual Cognitive Assessment (AVCA) framework, which automates cognitive assessment tests. The AVCA framework is developed based on the NCSE concept by identifying its main components across orientation, attention, language, contractual ability, memory, calculation and reasoning dimensions.

- The second contribution of the chapter is with regards to the second research question, where advanced NLP methods are identified and integrated with hand gesture recognition techniques to automate cognitive examination.

- The third contribution of the chapter follows the aforementioned third research question. In this regard, we propose a new data augmentation technique using unsupervised contrastive learning that improves performance of recent state-of-the-art sentence embedding models. Along the same direction, a Feed Forward Neural network (FFNN) model is developed that more accurately evaluates patient's responses in the reasoning section of NCSE.

4.2 Materials and Method

Clinically recognized cognitive assessment tools require the presence of trained clinical staff [45] since the presentation of instructions and stimuli must be conducted in a clinically validated, controlled manner for achieving optimal results [46] [47]. This constraint makes the frequent and longitudinal assessment difficult to be achieved in/outside the clinical setting [48]. Development of an autonomous cognitive assessment system that replaces the physician is, however, a critically challenging task. As stated previously, the proposed AVCA framework integrates NLP methods with hand gesture recognition to automate cognitive examination as briefly described below.

4.2.1 Language Modeling

Language Modeling (LM) has been a central task in NLP with the goal of learning a probability distribution over sequences of symbols pertaining to a language. Deep learning models [54], including Long-Short Term Memory networks (LSTM) [55] and Transformer architectures, are state-of-the-art modeling techniques utilized for different NLP tasks, such as sequence labeling [56], Named Entity Recognition (NER) [57] and Part of Speech (POS) Tagging [58]. Language models developed based on attention mechanism and self-training methods [59] such as ELMo [60], GPT [61], Bidirectional Encoder Representations from Transformers (BERT) [63], XLM [71], and XLNet [62] have brought significant performance gains to the NLP domain. Transformer architecture, which constitutes the main component of such models, uses a self-attention mechanism to capture dependencies among various instances of the input sequence. Self-attention is developed based on a scaled dot-product attention function, mapping a query and a set of key-value pairs to an output. The query (Q), keys (K), and values (V) are learnable representative vectors for the instances in the input sequence with dimensions d_k, d_k, and d_v, respectively. The output of the self-attention module is computed as a weighted average of the values, where the weight assigned to each value is computed by a similarity function of the query and the corresponding key after applying a softmax function. More specifically, the attention values on a set of queries are computed simultaneously, packed together into matrix $\boldsymbol{Q}$. The keys and values are similarly represented by matrices $\boldsymbol{K}$ and $\boldsymbol{V}$. The output of the attention function is computed as follows

$$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V}, \tag{4.1}$$

where superscript ^{T} denotes transpose of a given matrix. It is also beneficial to linearly project the queries, keys, and values h times with various learnable linear projections to vectors with d_k, d_k and d_v dimensions, respectively, before applying the attention function. On each of the projected versions of queries, keys, and values, the attention function is performed in parallel, resulting in $d_v - dimensional$ output values. These values are then concatenated and once again linearly projected via a FC layer. This process is called Multi-head Self Attention (MSA), which helps the model to jointly attend to information from different representation subspaces at different positions.

The output of the MSA module is given by

$$\begin{aligned} MSA(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) &= Concat\big(head_1,\cdots,head_h\big)\boldsymbol{W}^O, \\ \text{where} \quad head_i &= Attention(\boldsymbol{Q}\boldsymbol{W}_i^Q,\boldsymbol{K}\boldsymbol{W}_i^K,\boldsymbol{V}\boldsymbol{W}_i^V), \end{aligned} \tag{4.2}$$

where the projections are achieved by parameter matrices $\boldsymbol{W}_i^Q \in \mathbb{R}^{d_{model}\times d_k}$, $\boldsymbol{W}_i^K \in \mathbb{R}^{d_{model}\times d_k}$, $\boldsymbol{W}_i^V \in \mathbb{R}^{d_{model}\times d_v}$, and $\boldsymbol{W}^O \in \mathbb{R}^{hd_v\times d_{model}}$.

Word and Sentence Embedding

To perform autonomous cognitive state analysis, word/sentence embeddings are needed to construct representation of patients' responses (words/sentences) in the form of real-valued vectors to encode the underlying meaning. A contextualized word embedding is a vector representing a word in a special context. Traditional word embeddings such as Word2Vec and GloVe generate one vector for each word, whereas contextualized word embeddings such as ELMo, ULMFiT, and BERT generate a vector for a word depending on the context. Bidirectional LSTM is, typically, used in these contextualised language models. In the ELMo both the forward language model and the backward language model share the same LSTM, therefore, it fails to simultaneously consider the previous and subsequent tokens. Two further contextualised language models are GPT3 and BERT. GPT Transformer also uses constrained self-attention where every token can only attend to context to its left. BERT only uses the encoder blocks from the transformer and employs bidirectional self-attention, whereas GPT3 only employs the decoder blocks from the transformer and employs limited self-attention, where each token can only pay attention to the context to its left. GPT3 is a potent text generation tool, while BERT can be quickly fine-tuned on a particular downstream task with a limited number of labels. In the proposed AVCA framework, we utilized the BERT architecture, which has the following variants:

- *BERT Architecture:* BERT [63] is a multi-layer bidirectional Transformer encoder that relies entirely on attention mechanisms [64], with no recurrence or convolutions. Unlike most other language models, BERT's transformer encoder reads the entire sequence of words at once, randomly masks words in the sentence and predicts them. Bert includes two training

processes; pre-training and fine-tuning. The pre-trained parameters are first used to initialize the BERT model for fine-tuning, and labelled data from the downstream tasks is used to fine-tune each parameter. Despite being initialized with the same pre-trained parameters, each downstream task has its own fine-tuned models.

- *RoBERTa Architecture:* RoBERTa [59] is an enhanced method for training BERT models that can match or even outperform the effectiveness of existing post-BERT techniques. With more data, longer sequences, and larger batches, the model is trained over a longer period of time. Additionally, the objective for predicting the following sentence is dropped, and the masking pattern that was used to mask the training data is dynamically altered.

- *Sentence-BERT Architecture:* To create a fixed-sized sentence embedding, SBERT [65] adds a pooling operation to the output of RoBERTa and BERT. The mean of all output vectors is calculated as the default pooling approach. Siamese and triplet networks [66] are developed to update the weights in BERT/RoBERTa such that the resultant sentence embeddings are semantically meaningful and can be compared with cosine-similarity.

4.2.2 Automatic Speech Recognition

Automatic Speech Recognition (ASR) allows for the transcription of spoken language into text and is intended to facilitate natural human-machine interaction. All experiments are performed based on the two following open-source ASR models including wav2vec2 [80] and HuBERT [81].

- *The Wav2Vec2* is a transformer-based self-supervised model that encodes speech audio using a multi-layer Convolutional Neural Network (CNN) as a feature extractor before masking the spans of the resulting latent speech representations. After that, a Transformer network receives the latent representations and creates contextualised representations. The model is then fine-tuned on labelled data using the Connectionist Temporal Classification (CTC) [68] algorithm loss to be used for downstream speech recognition tasks. The wav2Vec2 [80] base model is Wav2Vec2-Base-960h, which has been fine-tuned over 960 hours of Librispeech [67] on 16 kHz sampled speech audio.

Table 4.1: Accuracy of wav2vec2 and HuBERT on 100 audio files

Wav2vec2 (US accent)	Wav2vec2 (UK accent)	HuBERT (US accent)	HuBERT (UK accent)
94.13%	93.44%	90.60%	91.12%

- *The HuBERT* uses an offline k-means clustering step and learns the structure of spoken input by predicting the right cluster for masked audio segments. The HuBERT [81] progressively improves its learned discrete representations by alternating between clustering and prediction.The predictive loss is applied over only the masked regions, forcing the model to learn good high-level representations of unmasked inputs in order to infer the targets of masked ones correctly.

It is worth mentioning that Hsu *et. al* [81] showed that the HuBERT outperforms wav2vec2 in most scenarios. A test have been conducted on these two ASR models with 100 audio files as input with US and UK accents. Each audio file stated one of the 100 most well-known names in the world. Each text generated by these ASR models were compared to their corresponding source text using string matching. Similarity percentages of at least 60% were deemed to be acceptable answers. This data was chosen because names are prone to more errors in ASR models. The results are illustrated in Table 4.1. Audio recordings of cities were also evaluated. Wav2vec2's average accuracy was around 70% while Hubert's average accuracy was close to 80%.

This approach was also tested on audio files of cities. Hubert's average accuracy was close to 80%, while wav2vec2's average accuracy was about 70%.

4.3 The AVCA Framework

The proposed AVCA framework is inherently a Virtual Assistant (VA) [72], that provides individual scores for major domains of Cognitive Assessments. The proposed AVCA frameworks consists of four major modules, i.e., the Human Machine Interface (HMI), NLP, and Sentence Embedding (SE) modules, together with the gesture recognition module, which are presented below.

4.3.1 Dataset

To train the proposed unsupervised model, we used a dataset consisting of 1 million sentences from English Wikipedia and the combination of MNLI and SNLI datasets [77]. The MNLI and SNLI datasets, which are described below, contain $570K$ and $433K$, respectively, human-written English sentence pairs manually labeled for balanced classification with the labels entailment (one sentence that is absolutely true), contradiction (one sentence that is definitely false), and neutral (one sentence that might be true). The dataset can be denoted as (x_i, x_i^+, x_i^-) where x_i is the premise, x_i^+ and x_i^- are entailment and contradiction sentences.

The Stanford Natural Language Inference (SNLI) Dataset [78], is a collection of labeled sentence pairs, which is created based on image captioning. The SNLI consists of $570K$ pairs of sentences allowing development of deep learning-based lexicalized classifiers for natural language inference tasks. Amazon Mechanical Turk is used for collecting the SNLI dataset, where in each task a premise scene is presented to the human annotator. The annotator's task is to hypothesizes there sentences, i.e., one entailment, one neutral, and one contradiction. The premises are obtained from a pre-existing corpus, i.e., the Flickr30k corpus consisting collection of approximately 160k captions (corresponding to about 30k images). The released dataset, explicitly the train, development, and test sets. The development and the test sets each consists of $10k$ examples, with each original ImageFlickr caption only appearing in one of the aforementioned three sets.

The Multi-Genre Natural Language Inference (MNLI) Dataset [79], is a widely utilized dataset for evaluation and design of deep learning models for the task of sentence understanding. The MNLI consists of $433k$ examples, making it one the largest datasets that is accessible for sentence understanding. More specifically, the MNLI contains data from ten different spoken and written English genres, which allows for evaluation of new models incorporating the full complexity of English language. The approach used for collection of the MNLI dataset is similar in nature to the one used for construction of the SNLI. Each sentence pair is created by first identifying a sentence as the premise from a pre-existing text, which is called the source. Afterwards, a new sentence is composed to be paired with the premise sentence, as a hypothesis, with the help of a human annotator. Based on development examples, it seems that while the collected hypotheses are not

always complete sentences, they are fluent and correctly spelled. The MNLI dataset is available in two formats, i.e., JSON Lines (jsonl) similar to SNLI and tab separated text. Several fields are specified for each example, including unique identifiers for the pair and prompt, and premise and hypothesis strings. The distributed MNLI dataset explicitly provides, train, development, and test sets, with no premise sentence occurring in more than one set. The development and test sets each consists of $20,000$ examples, which include $2,000$ examples randomly selected from each of the ten genres.

For development of the reasoning module of the proposed framework, an in-house dataset is utilized, which is described below:

Reasoning Dataset: An in-house dataset containing 830 responses from various patients was used to construct the evaluation framework. The responses were collected by a physician in the in-person cognitive assessment test and the same judgment task question was asked to each participant. Using our proposed embedding module we transferred the test data into numerical vectors. For the purpose of training we used 95% of the responses as training data and used the remaining 5% as test data.

4.3.2 HMI, NLP and SE Modules

Patients' audio responses are initially recorded and transformed into text using the ASR Module. Two significant NLP techniques are then employed to process the received text

- *String Matching:* When patients are required to repeat a sentence as part of the cognitive test, this method is employed. The fewest number of symbol substitutions, insertions, and deletions required to change one word into the other is known as Levenshtein Distance (LD). In our implementation, the difference between the patient's answer and the correct response is measured with LD. To adjust for inaccuracies in the speech-to-text module, a difference of 30% or less is regarded as a correct response.

- *Semantic Similarity:* To deal with other test domains where a more complex module is needed to compare phrases semantically as opposed to lexicographical similarity, semantic similarity is used. The semantic similarity of two input text phrases is determined by comparing the cosine similarity of their embeddings. As one of the most popular similarity metrics, the

cosine similarity calculates the angle between two non-zero vectors and measures how similar they are using this angle.

The semantic similarity unit of the AVCA framework receives the audio signal recorded as the patient's response. The pre-processing unit will convert the continuous audio signal to text, which is the fed to a BERT model followed by a pooling layer to form the embedding vector. The NLP model compares the embedding of the patient's response with the embedding of its corresponding correct response stored in the database. A cosine similarity unit will then evaluate the semantic similarity of the two embeddings.

Sentence Embedding Module

The sentence embedding module of the proposed AVCA framework is designed based an unsupervised contrastive learning framework. The contrastive learning model takes two sentences as positive samples in the training phase and treat the other sentences as negative samples. The main idea is to train the representation layer of the model by a contrastive loss objective function [73] to pull closer representations of the same class field (i.e., original sentence and its augmented one(s) called positive samples) and separate them from the rest of the sentences. In this regard, data augmentation is needed, which refers to a series of techniques that create simulated data out of a set of existing data. The predictions of the model ought to be invariant to the modest changes that are frequently present in this simulated data. By randomly rearranging the specific linguistic forms, the regularisation technique known as data augmentation can help prevent over-fitting by minimising the learning rate of erroneous correlations. Numerous regularisation strategies have been explored, including dropout [74] or weight penalties [75]. Augmentations are grouped into symbolic or neural methods [76]. While neural augmentations employ a DNN trained on a separate task to augment data such as Generative Data Augmentation (GDA), symbolic approaches use rules or discrete data structures to produce synthetic instances, such as substituting words or phrases to create augmented examples. Utilizing dropout noise is another method that is regarded as a form of data augmentation. In this method, the positive pairs are one sentence repeated twice, the only difference between their embeddings is in their dropout masks [69]). None of the discrete augmentations outperform dropout noise, and further data augmentation methods like crop, word deletion, and replacement

would have a negative impact on performance. In accordance with the concept we provided before, GDA is a sort of neural data augmentation that entails creating indistinguishable text paragraphs for tasks like summarization. Pegasus [70] is a sequence-to-sequence model designed for abstractive text summarization using gap-sentence generation as a pre-training target. Important sentences are removed/masked from an input document in the PEGASUS, and the remaining phrases are then created as one output sequence, much like an extractive summary.

We propose to use Pegasus paraphrasing as data augmentation technique, which maintains the semantic meaning of each input sample in its process. Pre-trained checkpoints of BERT [63] or RoBERTa [59]) is used to generate the embedding of original and augmented sentences. Consequently, the original and augmented sentences are fed as inputs to the encoder with two different dropout masks (as in SimCSE [69].) Dropout masks and attention probabilities (default $p = 0.1$) are applied to fully-connected layers in the Transformer's standard training [64].

By taking the above representation as the sentence embedding an unsupervised model is trained on 1 Million sentences from English Wikipedia and the combination of MNLI and SNLI datasets [77], which are a collection of $570K$ and $433K$ human-written English sentence pairs manually labeled for balanced classification with the labels entailment (one sentence that is absolutely true), contradiction (one sentence that is definitely false), and neutral (one sentence that might be true). The dataset can be denoted as (x_i, x_i^+, x_i^-) where x_i is the premise, x_i^+ and x_i^- are entailment and contradiction sentences. However, as our approach is in an unsupervised fashion, we only use the first sentences to feed to our framework. Feeding the NLI datasets to our unsupervised contrastive learning framework without performing paraphrase augmentation on the second sentence of the positive pairs results in no appreciable improvement.

By grouping semantically similar neighbours together and separating unrelated ones, contrastive learning seeks to acquire effective representation. We adopt a cross-entropy target with in-batch negatives [84] while adhering to the contrastive framework [85]. Given an unlabelled sentence x_i, its embedding is formed as $h_i^z = f(x_i, z)$ with random dropout mask z. The training objective, therefore, becomes

$$l_i = -\log \frac{e^{sim(h_i^{z_i}, f(p(x_i), \acute{z}))/\tau}}{\sum_{j=1}^{N} e^{sim(h_i^{z_i}, h_j^{\acute{z}_j})}}, \tag{4.3}$$

Table 4.2: Sentence embedding performance (Spearman's correlation) on STS tasks.

Model	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
Unsupervised models								
GloVe embeddings(avg.)	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
$BERT_{base}$ (first-last avg.)	39.70	59.38	49.67	66.03	66.19	53.87	62.06	56.70
$BERT_{base}$-flow	58.40	67.10	60.85	75.16	71.22	68.66	64.47	66.55
$BERT_{base}$-whitening	57.83	66.90	60.90	75.08	71.31	68.24	63.73	66.28
IS-$BERT_{base}$	56.77	69.24	61.21	75.23	70.16	69.21	64.25	66.58
CT-$BERT_{base}$	61.63	76.80	68.47	77.50	76.48	74.31	69.19	72.05
SimCSE-$BERT_{base}$ [69]	68.40	82.41	74.38	80.91	78.56	76.85	72.23	76.25
AVCA-$BERT_{base}$	**71.43**	**82.19**	**76.10**	**83.78**	**78.65**	**79.27**	**73.45**	**77.84**

where $p(x_i)$ is the augmented version of x_i using Pegasus, τ is a temperature hyperparameter and $sim(h_i, h_j)$ is the cosine similarity $\frac{h_1^T h_2}{||h_1||.||h_2||}$.

The proposed model is illustrated in Fig. 4.1. The proposed AVCA's sentence embedding model is thoroughly assessed using seven standard Semantic Textual Similarity (STS) tasks, i.e., STS 2012-2016 [86–90]; STS Benchmark [91], and SICK-Relatedness [92]. SentEval is a toolkit for evaluating sentence representations, which measures the relatedness of two sentences based on the cosine similarity of their two representations. The SICK relatedness (SICK-R) task trains a linear model to output a score from 1 to 5 indicating the relatedness of two sentences. The evaluation metric can be Pearson's or Spearman's correlation. It is claimed [93] that Spearman correlation, which measures ranks rather than actual scores, is more suitable for evaluating sentence embeddings. Table 4.2 provides an overview of the STS task comparison results, where the AVCA model is compared to several of its counterparts. AVCA improves the results on almost all STS tasks. This improvement can be significantly important in semantic similarity related tasks. The use of data augmentation techniques during the production of training data has allowed AVCA to better evaluate the similarity of sentences with similar semantic meaning while having varied vocabularies by extracting more significant elements from the underlying text. Our findings are based on unsupervised models, but we anticipate seeing improved outcomes when supervised models are applied with data augmentation techniques.

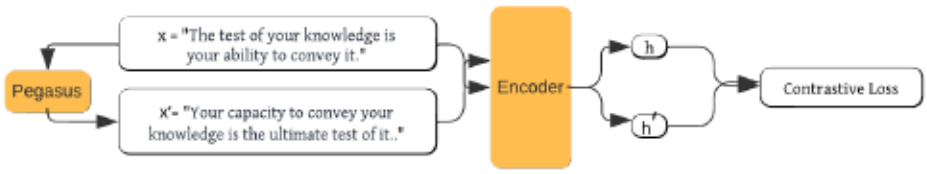


Figure 4.1: Sentence encoder is given each sentence along with its Pegasus-augmented equivalent. The input vectors h and h' are regarded as positive pairs for the model with different dropout masks. While other in-batch vectors are viewed as negative instance.

4.3.3 Gesture Recognition Module

Two domains of NSCE, i.e., language comprehension and constructional ability, involve actual interactions with physical objects. In one case, the participants is asked to create an identified shape using different red and while rectangular and triangular blocks. In the other case, different objects are placed in front of the participant after which basic pre-defined tasks such as picking up a pen are prompted. To autonomously conduct these components of the NSCE, in the proposed AVCA framework, hand gesture recognition is utilized where patients can pick up virtual objects and recreate patterns through a live video stream.

Language Comprehension

In the language comprehension section, patients are asked to use objects in specific ways. An HMI system is developed that allows users to perform the following requested virtual actions: (The ¡Object¿ could be keys, coins, pens, or any other defined object in the HMI system).

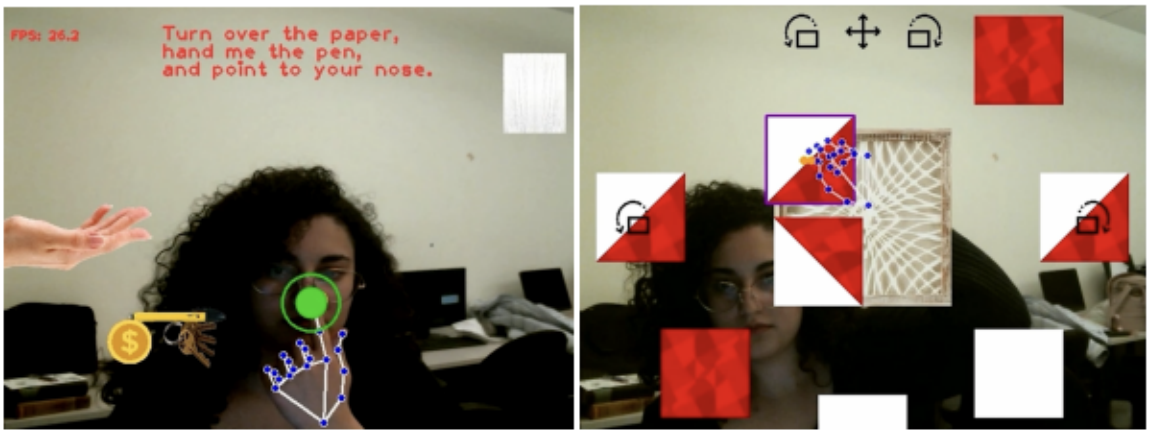
- Hand me the ¡Object¿: Action detection is based on collision detection between the specified object and the static hand on the screen. To achieve success for this implementation, there were many considerations such as validating the collision point between the objects and the static hand.

- Pick up the ¡Object¿: By confirming that the object is correctly grasped and examining its height on the screen, the pick-up motion is recognized.

- Point to the ¡Object¿: While pointing to the floor detection is based on the angle of the index finger on the screen for both the left and right hands, pointing to nose detection uses face detection, as shown in Figure 4.2. The distance between the index finger landmark and the nose landmark is used to identify this action. If this interval is less than the defined threshold for more than the specified time threshold (800 ms), the action is considered complete.

Constructional Ability

For constructional ability section, the patients need to replicate the patterns displayed on the screen using virtual red and white tiles, as shown in Fig. 4.2. All the game control commands are identified as different hand movements. Hand tracking is used to detect hand movements.



(a) (b)

Figure 4.2: Language Comprehension and Constructional Ability (a) Pointing to nose is detected using face detection, (b) Patients can move tiles using hand-tracking system.

4.4 Experiments and Results

We put the proposed AVCA cognitive assessment framework into practice in accordance with NSCE. In particular, the three assessments mentioned above, all contain some form of orientation and attention task. These tasks involve asking the patient to repeat a set of phrases or numbers, or in the case of orientation, to state their name or the time. String matching is the sole module used in these tasks. Other tasks, however, require more advanced algorithms. The descriptions of these

tasks and our proposed algorithms are presented below.

4.4.1 Language - Speech Sample

The patients are given 30 seconds to describe what they observe in a given picture throughout this activity. After preprocessing the response, if the number of words or verbs in the response is not enough, a recording will be played asking the patients to say more. Otherwise, the text's words are compared with a set of keywords. If the similarity percentage of each word and its corresponding keyword is more than or equal to 70 percent, we assume that the patient said the right word. After all the iterations, the similarity of all the responses concatenated and a correct key answer is computed using the AVCA's NLP module. An output percentage of more than or equal to 70 percent is considered a correct answer. The timing of the whole process is also computed and shown.

The speech sample function has been tested with several candidates. The results were promising. All the recorded voices were saying the correct description of the picture, and they all resulted in a semantic similarity percentage of sixty or more with the key sentence. Therefore, 70% is thought to be a suitable threshold for the correct response. Based on the test results, the pronouns of the sentences (e.g., person, boy, girl, etc.) should not be included in the keywords because it would decrease the similarity percentage. Furthermore, three key sentences are added as parameters that are all the same but have three different pronouns (person, boy, girl). In the end, the patient's response will be compared with all the key sentences and the maximum similarity percentage will be selected. One issue with the ASR is that occasionally the converted text sounds correct but the spelling is incorrect. Therefore, the recorded words should also be phonetically compared with the keywords, and if they are phonetically the same, the corresponding word's spelling should be corrected.

4.4.2 Memory

In this task, the patient is asked to say four words that were mentioned to earlier in the assessment. First, the patient is given a time slot to say the four words. The patient's voice is recorded and converted to text using the AVCA's ASR module. Then, each recorded word is compared with its corresponding keyword using string matching. If the word did not pass the string matching step, it

will be compared phonetically. Correct and incorrect responses are stored, and the patient will earn three points for every correct word. For every keyword that was missed by the patient, its category is given to the patient. If the patient remembered that specific keyword or any other unsaid keywords, the patient is given two scores for each of the correct ones. Otherwise, the patient is presented with three words, and one of them is correct. One point is awarded if the patient selects the right option. Otherwise, no score. In the end, the program returns a dictionary, which contains all the keywords, the patient's score for every keyword and the incorrect responses.

4.4.3 Calculation

In this task, the patient is asked a series of arithmetic calculations. The patient is given a 20-second time slot. In the traditional test, the clinician should note the precise moment the patient provided the right response. This time should, however, be automatically stored in the automatic version. Additionally, it should be noted that the patient might respond incorrectly a few times before responding correctly. The timestamps for all incorrect numbers should be kept. As mentioned in Section 4.2, wav2vec2 is fine-tuned on labeled data with the Connectionist Temporal Classification (CTC), which is a character-based algorithm. During the training phase, it can demarcate each character of the transcription in the speech automatically, therefore, the time-frame alignment is not required between the audio signal and transcription. As a result, in order to save a lot of labelling effort, it does not need to know the time-stamp of each word.

To add time-stamps to the ASR's output tokens, we proposed two solutions. One is to create a real-time speech recognition system using wav2vec2. However, the issue is that while wav2vec2 is trained with a 16khz sampling rate, we record patients' voices at 96khz for higher accuracy. We then turned to our second solution in order to get a higher level of precision. To be able to store the time in the process of speech recognition, we intercepted the wav2vec2 model and added a layer before the final transcription to calculate the time-stamp of each word. From there, we compute the time period of each sample, iterate through the samples while the ASR model is processing and calculate the time each letter was said. To compute the time period of each sample, the total number of samples generated by the ASR's processor is divided by the sampling rate (the number of audio samples recorded per second), which in our case is $16,000$. After that, if a letter was detected by

the ASR's model iterating through the samples, the letter and its associated time-stamp are saved. Additionally, wav2vec2 output spells out numerals rather than writing them in a numerical style. We created an algorithm to extract all the numerals from a given text in order to preserve all of the patient's wrong responses prior to saying the proper response.

4.4.4 Reasoning

This task is divided into two sections, i.e., similarity and judgment. In the case of similarities, the patient is asked to state how two items are alike, such as a train and a bicycle. This checks the patients' abstract reasoning. In contrast, judgment assesses the Patients' understanding of the consequences of certain actions, where a set of questions like "What would you do if you found a stamped envelope on the sidewalk?" are asked. As there are a set of correct responses for these questions, an evaluation framework is developed that predicts the score associated with patients' responses as follows:

- *ML-based Evaluation:* Various ML models were used to construct the evaluation framework. The framework based on Support Vector Machine (SVM) produced the highest accuracy of 85.7% compared to logistic regression with 80.9% accuracy. The proposed framework tokenizes the responses after being pre-processed. The pre-processed tokenized responses are then vectorized into n-gram integer vectors using TF-IDF features, to use as the input to the SVM-based classifier model. TFIDF evolved from IDF, which proposes that the frequency of a term in a given document should have an inverse relationship with its associated weight. Eq. 4.4 is the classical formula of used for term weighting [82]. The following TF-IDF formulation is then utilized

$$w_{ij} = tf_{ij} \times \log\left(\frac{N}{df_i}\right), \tag{4.4}$$

  where $w_{i,j}$ is the weight for term i in document j, N is the number of documents in the collection, $tf_{i,j}$ is the term frequency of term i in document j and df_i is the document frequency of term i in the collection.

- *FFNN-based Evaluation:* In this scenario, we used a Feed-forward Neural Network (FFNN) as the classification module of the AVCA to assess patients' responses. The model contains

Table 4.3: Performance comparison based on different classification models.

Model	Accuracy	*F1*
ktrain [83] + RoBERTa	88%	83%
Support Vector Machine + TF-IDF	90%	88%
AVCA FFNN + RoBERTa	**93%**	**92%**

an input layer, which takes the embedding vector of patients' responses, followed by two hidden layers with $ReLU(\cdot)$ and $tanh(\cdot)$ activation functions coupled with dropout layers to mitigate the risk of over-fitting. Sigmoid is used as the last layer that generates a floating-point number for each sample. We used the grid search method to fine tune the hyper parameters of the FFNN. By using the checkpoint callback method, we ensured that the final model is having the minimum validation loss. Hyper parameters were tuned using batch sizes of 12, 32, 64 and 128 with 100 epochs. To evaluate the performance of the final model, a hard cutoff threshold of 0.5 is applied on the model outputs for test samples. Since our data is imbalanced, focal loss is used, which assigns more weights to hard or easily misclassified examples or positive and negative examples. The model is trained by minimizing the focal loss which is given by

$$L(y, \hat{p}) = -\alpha y(1-\hat{p})\gamma \log(\hat{p}) - (1-y)\hat{p}\gamma \log(1-\hat{p}) \tag{4.5}$$

where $y \in 0, 1$ is a binary class label, $\hat{p} \in [0, 1]$ is an estimate of the probability of the positive class, γ is the focusing parameter that specifies how much higher-confidence correct predictions contribute to the overall loss, and α is a hyper-parameter that governs the trade-off between precision and recall. Table 4.3 illustrates results based on different models.

4.5 Summary

In this chapter, we proposed an AI-powered autonomous system, referred to as the AVCA framework, for cognitive assessment based on the NCSE assessment model. The proposed AVCA framework autonomously provides individual scores in the seven major cognitive domains, i.e., orientation, attention, language, contractual ability, memory, calculation, and reasoning, by integrating NLP and gesture recognition techniques. In this context, the AVCA framework incorporates innovative ideas to autonomously perform the "Calculation" and "Reasoning" tasks. For the former category (i.e., the calculation task), we intercepted an advanced ASR system to extract the timestamp of every spoken word as well as the patients' numeral responses. For the latter (i.e., the Reasoning task), a classification module is implemented to evaluate patients' responses. Finally, to improve AVCA's evaluation on patients' responses, a novel unsupervised contrastive learning mechanism is proposed based on the BERT model that outperforms state-of-the-art representation solutions. As a direction for future research, one should target incorporation of more advanced data augmentation techniques to increase the precision of the AVCA-BERT$_{base}$. In order to enhance the performance of other sentence embedding models, such as RoBERTa$_{base}$, our data augmentation technique may be used in the context of unsupervised or supervised contrastive learning. Another fruitful direction for future research is the use of a multi-class classification model to more precisely assess patients' responses in the reasoning component.

Chapter 5

Summary and Future Research Directions

This chapter concludes the thesis with a list of important contributions made in this dissertation and some proposed directions for future work.

5.1 Summary of Thesis Contributions

The central motivation behind this thesis research is to create an Autonomous Virtual Cognitive Assistant (AVCA) that can aid healthcare professionals in the diagnosis and treatment of patients. The system is designed to leverage ML and NLP techniques to analyze patient data, provide accurate diagnoses, and recommend appropriate treatments. The ultimate goal is to improve healthcare outcomes by reducing diagnosis and treatment errors, decreasing healthcare costs, and enhancing patient satisfaction. The thesis made several contributions, which are briefly summarized below.

(1) **The AVCA Application:** The Autonomous Virtual Cognitive Assessment (AVCA) application is a robust and scalable solution for cognitive assessment. It is built using a containerized approach, with three main containers: frontend, web app, and database. By using containers, each component of the application can be isolated and scaled independently. For example, if the frontend is experiencing high traffic, more instances of the frontend container can be deployed without affecting the other components of the application. Similarly, the database

container can be scaled up or down depending on the amount of data being processed. The app takes information from the patient using a microphone and camera only, which ensures that the testing process is non-intrusive and does not require any special equipment. The patient does not have to write or click on anything during the test, as everything is autonomous.

(2) **The Sentence Embedding Model:** AVCA's sentence embedding model is thoroughly assessed using seven standard STS tasks, including STS 2012-2016, STS Benchmark, and SICK-Relatedness. The AVCA's SE model was compared to several of its counterparts, and it was found to improve the results on almost all STS tasks, which is particularly significant in semantic similarity related tasks. The use of data augmentation techniques during the production of training data has allowed AVCA to better evaluate the similarity of sentences with similar semantic meaning while having varied vocabulary by extracting more significant elements from the underlying text. These findings are based on unsupervised models, but it is anticipated that the application of supervised models with data augmentation techniques will lead to even better outcomes. Overall, the contribution of the AVCA model is to improve sentence embeddings and enhance the performance of tasks in natural language processing that rely on semantic similarity.

(3) **Evaluation System:** AVCA has implemented an evaluation system with high accuracy that has the ability to automate the assessment of patients' understanding of the consequences of certain actions through a set of questions with predetermined correct responses. To achieve this, text classification is performed on an in-house dataset using various machine learning models such as SVM, FNN, CNN, and GNN. Inductive GNNs are found to be particularly effective in text classification tasks due to their ability to handle the underlying graph structure of textual data and generalize to unseen data. By utilizing the graph structure of language and the ability to generalize to new data, inductive GNNs achieve higher accuracy in text classification tasks compared to other deep learning models.

5.2 Future Research

In terms of future works, there are several avenues that can be explored to further improve the contributions of this thesis, as outlined below:

- One area that could be explored is the contrastive learning framework proposed for sentence embeddings. While we showed the effectiveness of our framework using the $\text{BERT}_{\text{base}}$ model, it would be interesting to see how it performs on more complex models such as $\text{BERT}_{\text{large}}$, $\text{RoBERTa}_{\text{base}}$ or $\text{RoBERTa}_{\text{large}}$. Conducting experiments on these language models would require greater computing power than was available to us during this research. Additionally, other data augmentation techniques can be explored for both the unsupervised and supervised versions of the framework.

- Regarding the evaluation system, one potential improvement could be to incorporate named-entity recognition for a more accurate evaluation of the patient's responses. For instance, in the orientation section of cognitive assessment, we can locate and classify named entities mentioned in unstructured text into pre-defined categories such as time expressions. This would allow for a more precise assessment of the patient's performance in this section.

- In order to evaluate a patient's understanding of a particular concept or topic, intent recognition can be utilized. It can determine whether the patient comprehends the overall meaning of a question or scenario, which is particularly useful in the reasoning section. Incorporating intent recognition would require data with intent labels, which were not available in our study. Obtaining data with intent labels could be a promising direction for future research.

- To improve the accuracy of the AVCA app in diagnosing and treating patients, more precise ASR and NLP techniques can be explored. By incorporating these techniques, we can obtain a more accurate understanding of patient responses and improve the diagnosis and treatment process.

- Finally, a crucial step towards the implementation of the AVCA app in a real-world setting would be to conduct clinical trials with a group of patients and compare the performance of

our app with that of traditional healthcare providers. This would provide a more comprehensive evaluation of the effectiveness and feasibility of our proposed system.

Bibliography

[1] B. Karimi, S. Zabihi, A. Keynia, A. Montazami, A. Mohammadi "AVCA: Autonomous Virtual Cognitive Assessment," Accepted in *Transactions on Computational Science*, 2023.

[2] B. Karimi, S. Zabihi, M. Salimibeni, A. Mohammadi, "Autonomous Virtual Cognitive Assessment via NLP and Hand Gesture Recognition," *56th Asilomar Conference on Signals, Systems, and Computers*, pp. 460-464, 2022.

[3] Blanc, C. et al. "Flaubert vs. camembert: Understanding patient's answers by a french medical chatbot." *Artif. Intell. Med. 2022,* 102264 (2022).

[4] D. W. Otter, J. R. Medina, and J. K. Kalita. "A survey of the usages of deep learning in natural language processing." *arXiv preprint arXiv:1807.10854,* 2018.

[5] S. Vijayarani, M. J. Ilamathi, and M. Nithya, "Preprocessing techniques for text mining-an overview," *International Journal of Computer Science & Communication Networks, vol. 5, pp. 7 16*, 2015.

[6] A. Chiche and B. Yitagesu, "Part of speech tagging: A systematic review 1190 of deep learning and machine learning approaches," *J. Big Data, vol. 9, 1191 no. 1, pp. 1–25,* Dec. 2022.

[7] X. Liu, H. Chen, W. Xia "Overview of named entity recognition technology," *Radio Commun. Technol., vol. 46, no. 3, pp. 251–260,* 2020.

[8] H. Weld, X. Huang, S. Long, J. Poon, and S. Han, "A survey of joint intent detection and slot filling models in natural language understanding," *arXiv preprint, vol. arXiv:2101.08091,* 2021.

[9] M. Birjali, M. Kasri, A. Beni-Hssane "A comprehensive survey on sentiment analysis: Approaches, challenges and trends." *Knowledge-Based Systems, 227, Article 107134*, 2021.

[10] K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. E. Barnes, and D. E. Brown, "Text classification algorithms: A survey," *Information, vol. 10, no. 4, p. 150,* 2019.

[11] B. Min, H. Ross, E. Sulem, *et al.*, "Recent advances in natural language processing via large pre-trained language models: A survey." *arXiv preprint arXiv:2111.01243,* 2021.

[12] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *CoRR, vol. abs/2003.08271,* 2020.

[13] T, Brown, B. Mann, N. Ryder,*et al.*, "Language models are few-shot learners" *In Advances in Neural Information Processing Systems, volume 33, pages 1877–1901. Curran Associates, Inc.*, 202.

[14] C. Raffel, N. Shazeer, A. Roberts, *et al.*, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." *In Journal of Machine Learning Research.*, 2020.

[15] K. Clark, M. Luong, Q. V Le, and C. D, "Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators." *In ICLR,* 2020.

[16] Q. Chen, Z. Zhuo, and W. Wang, "Bert for joint intent classification and slot filling." *arXiv preprint arXiv:1902.10909,* 2019.

[17] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao. 2021. "Deep learning-based text classification." *ACM Comput. Surv. 54, 3*, 2021.

[18] Q. Li, H. Peng, J. Li, C. Xia, R. Yang, L. Sun, P. S. Yu, and L. He, "A survey on text classification: From traditional to deep learning," *ACM Transactions on Intelligent Systems and Technology (TIST), vol. 13, no. 2, pp. 1–41,* 2022.

[19] Y. Zhang, X. Yu, Z. Cui, S. Wu, Z. Wen, and L. Wang, "Every document owns its structure: Inductive text classification via graph neural networks," *in Proc. ACL, 2020, pp. 334–339,* 2020.

[20] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," *in Proc. AAAI, 2019, pp. 7370–7377,* 2019.

[21] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *in Proc. ICLR*, 2018.

[22] X. Huang, Q. Song, Y. Li, and X. Hu. "Graph recurrent networks with attributed random walks." *In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 732–740,* 2019.

[23] W. Hamilton, Z. Ying, and J. Leskovec. "Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems." *in Proc. NIPS, 2017, pp. 1024–1034.*, 2017.

[24] A. Oord, Y. Li, and O. Vinyals. "Representation learning with contrastive predictive coding." *arXiv:1807.03748*, 2018.

[25] H.T. Jung, *et al.*, "Remote Assessment of Cognitive Impairment Level based on Serious Mobile Game Performance: An Initial Proof of Concept." *IEEE J. Biom. & Health Inf.*, vol. 23, no. 3, pp. 1269-1277, 2019.

[26] P. E. Greenberg, A.-A. Fournier, T. Sisitsky, C. T. Pike, and R. C. Kessler, "The economic burden of adults with major depressive disorder in the united states (2005 and 2010)," *The Journal of clinical psychiatry, vol. 76, no. 2, pp. 155–162, 2015.*

[27] F.Z. Canal, et al., "A Survey on Facial Emotion Recognition Techniques: A State-of-the-art Literature Review," Information Sciences, *vol. 582, pp. 593-617, 2022.*

[28] X. Li, et al., "EEG based Emotion Recognition: A Tutorial and Review," ACM Computing Surveys, *vol. 55, no. 4, pp. 1-57, 2023.*

[29] J. Zhang, Z. Yin, P. Chen, S. Nichele, "Emotion Recognition using Multi-Modal Data and Machine Learning Techniques: A Tutorial and Review," Information Fusion, *vol. 59, pp.103-126, 2020.*

[30] M.M Hassan, et al., "Human Emotion Recognition using Deep Belief Network Architecture,," Information Fusion, *vol. 51, pp. 10-18, 20219.*

[31] A. Yousaf et al.,. "Emotion Recognition by Textual Tweets Classification Using Voting Classifier (LR-SGD)," IEEE Access, *vol. 9, pp. 6286-6295, 2021.*

[32] F. M. Dimou, D. Eckelbarger, and T. S. Riall, "Surgeon burnout: A Systematic Review," Journal of the American College of Surgeons, *vol. 222, no. 6, p. 1230, 2016.*

[33] P. Ekman, W. V. Friesen, M. O'sullivan, A. Chan, I. Diacoyanni-Tarlatzis, K. Heider, R. Krause, W. A. LeCompte, T. Pitcairn, P. E. Ricci-Bitti et al. "Universals and cultural differences in the judgments of facial expressions of emotion." Journal of personality and social psychology, vol. 53, no. 4, p. 712, 1987.

[34] S. Ahn, *et al.*, "Exploring Neuro-Physiological Correlates of Drivers' Mental Fatigue Caused by Sleep Deprivation using Simultaneous EEG, ECG, and fNIRS Data," *Frontiers in Neuroscience*, vol. 10, pp. 1-14, 2016.

[35] , S.Y. Han, N.S. Kwak, T. Oh, S.W. Lee, "Classification of Pilots' Mental States using a Multimodal Deep Learning Network," *Biocybernetics and Biomedical Engineering*, vol. 40, no. 1, pp. 324-336, 2020.

[36] B. Cheng, *et al.*, "Measuring and Computing Cognitive Statuses of Construction Workers Based on Electroencephalogram: A Critical Review," *IEEE Transactions on Computational Social Systems*, vol. 9, no. 6, pp. 1644-1659, 2022.

[37] G. Borghini, L. Astolfi, G. Vecchiato, D. Mattia, and F. Babiloni, "Measuring neurophysiological signals in aircraft pilots and car drivers for the assessment of mental workload, fatigue and drowsiness," *Neuroscience & Biobehavioral Reviews*, vol. 44, pp. 58–75, 2014.

[38] R. Castaldo, P. Melillo, U. Bracale, M. Caserta, M. Triassi, and L. Pecchia, "Acute mental stress assessment via short term HRV analysis in healthy adults: A systematic review with meta-analysis," *Biomedical Signal Processing and Control*, vol. 18, pp. 370–377, 2015.

[39] M. F. Folstein and S. E. Folstein, "Mini-Mental State" a practical method for grading the cognitive state of patients for the clinician," *J. Psychiatric Res., vol. 12, no. 3, pp. 189–198, 1975.*

[40] Z. S. Nasreddine et al., "The montreal cognitive assessment, MoCA: A brief screening tool for mild cognitive impairment," J. Amer. Geriatrics Soc., vol. 53, pp. 695–699, 2005.

[41] Kiernan RJ, Mueller J, Langston JW, Van Dyke C. "The neurobehavioral cognitive status examination: a brief but differentiated approach to cognitive assessment." *Ann Intern Med. 1987;107:481-5, 1987.*

[42] Fields SD, Fulop G, Sachs CJ, Strain J, Fillit H. "Usefulness of the Neurobehavioral Cognitive Status Examination in the hospitalized elderly." Int. Psychogeriatr. 1992; 4:93-102.

[43] Murakami H, Fujita K, Futamura A et al. "The Montreal Cognitive Assessment (MoCA) and Neurobehavioral Cognitive Status Examination (COGNISTAT) are useful for screening mild cognitive impairment in Japanese patients with Parkinson's disease." *Neurol. Clin. Neurosci. 2013; 1: 103–8.*

[44] Osmon DC, Smet IC, Winegarden B, Gandhavadi B. "Neurobehavioral Cognitive Status Examination: its use with unilateral stroke patients in a rehabilitation setting." Arch Phys Med Rehabil.1992;73:414-418.

[45] E. G. Tangalos et al., "The mini-mental state examination in general medical practice: Clinical utility and acceptance." *Mayo Clinic Proc., vol. 71, no. 9, pp. 829–837, 1996.*

[46] R. M. Brouillette et al., "Feasibility, reliability, and validity of a smartphone-based application for the assessment of cognitive function in the elderly," PLoS One, vol. 8, no. 6, 2013, Art. no e65925.

[47] C. Timmers, A. Maeghs, M. Vestjens, C. Bonnemayer, H. Hamers, and A. Blokland, "Ambulant cognitive assessment using a smartphone," Appl. Neuropsychol., vol. 21, no. 2, 2013, Art. no. e112197.

[48] P. Schweitzer et al., "Feasibility and validity of mobile cognitive testing in the investigation of age-related cognitive decline," Int. J. Methods Psychiatric Res., vol. 26, no. 3, 2017, Art. no. e1521

[49] S. Koelstra, C. Muhl, M. Soleymani, J.-S. Lee, A. Yazdani, T. Ebrahimi, T. Pun, A. Nijholt, and I. Patras, "Deap: A database for emotion analysis; using physiological signals," IEEE transactions on affective computing, vol. 3, no. 1, pp. 18–31, 2011.

[50] Finney GR, Minagar A, Heilman KM. "Assessment of mental Status." Neurol Clin. 34(1): 1–16, 2016.

[51] K.K. Tsoi, et al. "Cognitive Tests to Detect Dementia: A Systematic Review and Meta-Analysis," JAMA Internal Medicine, vol. 175, pp. 1450-1458.

[52] R. Daroische, et al., "Cognitive Impairment after COVID-19 - A Review on Objective Test Data," Frontiers in Neurology, vol. 12:1238, 2021.

[53] T. N. Tombaugh and N. J. McIntyre, "The mini-mental state examination: A comprehensive review," J. Amer. Geriatrics Soc., vol. 40, no. 9, pp. 922–935, 1992.

[54] A. K. Goyal, A. Metallinou, and S. Matsoukas. "Fast and Scalable Expansion of Natural Language Understanding Functionality for Intelligent Agents." In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers). Association for Computational Linguistics, 2018.

[55] Felix A Gers, Jurgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with lstm", 2000.

[56] Chung, C. Gulcehre, K. Cho, and Y. Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modelling". In NIPS 2014 Workshop on Deep Learning,2014.

[57] Jason PC Chiu and Eric Nichols. "Named entity recognition with bidirectional lstm-cnns" arXiv preprint arXiv:1511.08308, 2015

[58] Z. Huang, W. Xu, and K. Yu. 2015. "Bidirectional lstm-crf models for sequence tagging". arXiv preprint arXiv:1508.0199, 2015.

[59] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692, 2019.

[60] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. "Deep contextualized word representations. In North American Association for Computational Linguistics."(NAACL), 2018.

[61] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. "Improving language understanding with unsupervised learning." Technical report, OpenAI.

[62] Z. Yang, Z.Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le. "Xlnet: Generalized autoregressive pretraining for language understanding." arXiv preprint arXiv:1906.08237, 2019.

[63] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova "BERT: pre-training of deep bidirectional Transformers for language understanding." Proc. 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol., vol. 1, NAACL-HLT 2019, Minneapolis, MN, USA (2019), pp. 4171-4186. Google-AI Language, June 2-7, 2019.

[64] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N Gomez, L. Kaiser, and I. Polosukhin. "Attention is all you need." *In Advances in Neural Information Processing Systems, pages 6000–6010, 2017.*

[65] N. Reimers and I. Gurevych. "Sentence-BERT: Sentence embeddings using siamese BERT-Networks." DOI, 2019.

[66] F. Schroff, D. Kalenichenko, and J. Philbin. 2015. "FaceNet: A Unified Embedding for Face Recognition and Clustering."

[67] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech: an asr corpus based on public domain audio books. *In Proc. of ICASSP, pages 5206–5210. IEEE, 2015.*

[68] A. Graves, S. Fernández, and F. Gomez. "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks." In Proc. of ICML, 2006.

[69] T. Gao, X. Yao, and D. Chen, "SimCSE: Simple contrastive learning of sentence embeddings," *International Conference on Empirical Methods in Natural Language Processing*, pp. 6894-6910, 2021.

[70] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu. "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization." *ICML, 2020.*

[71] G. Lample and A. Conneau. "Crosslingual language model pretraining." arXiv preprint arXiv:1901.07291, 2019.

[72] S. Gondala, L. Verwimp, E. Pusateri, M. Tsagkias, and C. Van Gysel, "Error-Driven Pruning of Language Models for Virtual Assistants." arXiv:2102.07219, *2021.*

[73] S. Chopra, R. Hadsell, and Y. LeCun. "Learning a similarity metric discriminatively, with application to face verification." In CVPR, 2005.

[74] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR), 15(1):1929–1958, 2014.*

[75] J. Kukacka, V. Golkov, and D. Cremers. "Regularization for deep learning: A taxonomy." arXiv preprint arXiv:1710.10686, 2017.

[76] C. Shorten, T. M. Khoshgoftaar, and B. Furht, "Text data augmentation for deep learning," *Journal of Big Data, vol. 8, no. 1, pp. 1–34, 2021.*

[77] Y. Nie, A. Williams, E. Dinan, M. Bansal, J. Weston, and D. Kiela, "Adversarial NLI: A New Benchmark for Natural Language Understanding," Association for Computational Linguistics (ACL), pp 4885-4901, 2020.

[78] S.R. Bowman, G. Angeli, C. Potts, C.D. Manning, "A Large Annotated Corpus for Learning Natural Language Inference," *Conference on Empirical Methods in Natural Language Processing*, pp. 632-642, 2015.

[79] A. Williams, N. Nangia, S. Bowman "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference," *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, pp. 1112–1122, 2018.

[80] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli. "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations.", 2020.

[81] W.-N. Hsu, B. Bolte, Y.-H. Hunert Tsai, K. Lakhotia, R.Salakhutdinov, and A. Mohamed, "HuBERT: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Trans. Audio Speech Lang. Process., vol. 29, pp. 3451–3460, 2021.*

[82] S. Jones, K. "IDF term weighting and IR research lessons." Journal of Documentation, 60(6), 521–523, 2004.

[83] A.S. Maiya. "ktrain: A Low-code Library for Augmented Machine Learning," *arXiv:2004.10703*, 2020.

[84] T. Chen, Y. Sun, Y. Shi, and L. Hong. "On sampling strategies for neural networkbased collaborative filtering." *In ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 767–776, 2017.

[85] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. "A simple framework for contrastive learning of visual representations." *In International Conference on Machine Learning (ICML)*, pages 1597–1607, 2020.

[86] E. Agirre, D. Cer, M. Diab, and A. Gonzalez-Agirre. "SemEval-2012 task 6: A pilot on semantic textual similarity." *In *SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393, 2012.

[87] E. Agirre, D. Cer, M. Diab, A. Gonzalez- Agirre, and W. Guo. "*SEM 2013 shared task: Semantic textual similarity." *In Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, 2013.

[88] E. Agirre, C. Banea, C. Cardie, D. Cer, M. Diab, A. Gonzalez Agirre, W. Guo, R. Mihalcea, G. Rigau, and J. Wiebe. "SemEval-2014 task 10: Multilingual semantic textual similarity." *In Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91, 2014.

[89] E. Agirre, C. Banea, C. Cardie, D. Cer, M. Diab, A. Gonzalez-Agirre, W. Guo, I. Lopez-Gazpio, M. Maritxalar, R. Mihalcea, German Rigau, L. Uria, and J. Wiebe. "SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability." *In Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263, 2015.

[90] E. Agirre, C. Banea, D. Cer, M. Diab, A. Gonzalez-Agirre, R. Mihalcea, G. Rigau, and J. Wiebe. "SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. *In Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511. Association for Computational Linguistics, 2016.

[91] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. *In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017),* pages 1–14, 2017.

[92] M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, and R. Zamparelli. "A SICK cure for the evaluation of compositional distributional semantic models." *In International Conference on Language Resources and Evaluation (LREC),* pages 216–223, 2014.

[93] N. Reimers, P. Beyer, and I. Gurevych. "Task-oriented intrinsic evaluation of semantic textual similarity." *In International Conference on Computational Linguistics (COLING),* pages 87–96.