ASLBoT – Assistive Sign Language Bot Translator

Gustavo Camero, Luis Hurtado, Michael Loyd, and Jared Spinks

Department of Electrical and Computer Engineering
University of Central Florida

***Abstract*—Human interpreters are relied upon by the sign language community for interpretation of the English language. However, this method of used in order to translate English to the appropriate sign language results in high costs, reliability, and availability issues. With major advances in animation programing and machine translation new avenues may now be taken help this community obtain the reliable and responsive sign language translations. Herein, we provide a novel approach to translating English into the animated sign language version providing a low-cost alternative. This novel approach is composed of using small corpora and our own developed training framework. A neural machine translation (NMT) model is trained to translate from English to ASL-Gloss and a Unity script generates the corresponding sign language animation tied to the ASL-Gloss translation. ASLBoT aims to be the first product on the market that can translate between English to American Sign Language while providing a cost-effective, easy-to-use and reliable translation for people with hearing disabilities.**

***Keywords*—Neural machine translation, American Sign Language, 3D graphics rendering, Unity Engine.**

I. Introduction

Translation provides the channel for maintaining a meaningful conversation between two individuals who speak different languages. Human based translations have been widely applied in many aspects of society including the financial, medical, legal, and travel industries. The process of translating through human interpreters, however, proves to be erratic, difficult to arrange, and expensive. In a hospital setting between a doctor and the patient, a human interpreter listens to the patient speak for several minutes and then provides a summary of the dialogue to the doctors. This abridged response that the interpreter provides the doctor may omit important information of symptoms, medical history, and other details that are critical. Therefore, this lack of information may “lead to misdiagnosis and improper or delayed medical treatment”. This issue has been also seen in classroom settings where deaf students enrolled in a normal classroom environment require a robust translation rather than an abridge translation. Research has shown in a High School environment the performance of Deaf Students vs Hearing Students in terms of grades. The research concluded that Deaf Students performed poorly due to poor quality interpretation services provided by the school. Additionally, there are multiple cases that patients don’t get up to standard interpretation services provided by the Hospital’s translation. This is most evident with deaf people as hospitals have moved to online interpreting services, requiring continuous Internet connectivity and high upkeep due to the contract signings. Since 2011, there have been multiple court cases regarding interpreting services for deaf hospital patients in which some cases have settled in the sum of $70,000. Moreover, in recent years, deaf students have been facing with the issue of fast responsiveness as the request of ASL interpretation services enabled by the Services for Students with Disabilities requires at least a three-day notice. Thus, the need for a more accurate, responsive, read to use, and low-cost translation system is required for such applications.

Neural machine translation (NMT) provides the next generation of real time translation with minimal errors. NMTs have proven to be useful for its flexible deployment and its ease of use. While they require large data sets to function properly, there are open-source NMTs that can be used for product commercialization. However, these NMTs only perform the translations for spoken languages and don’t consider sign languages such as the American Sign Language (ASL). People who are congenitally deaf or have never developed an understanding of spoken language use ASL as their primary language. They don’t develop the same understanding of the language as an individual who is able to listen and speak the language. Thus, an any-to-text translation is not enough for people that have the disability. This is where the NMT algorithms’ primary ability to translate any-to-text falls short of providing the service to ASL users. In this document, we are proposing a real time translator with virtual ASL interpretation. This device will be designed to require minimal overhead, low budget, and be accurate on speech acquisition and translation delivery. In addition, the device will deliver real time translations such that conversations between both users are continuous with no wait on the translation. The functionality of the translator is to provide two translation modes: Speech-to-ASL Translation (SAT) mode, and Speech-to-Text Translation (STT) mode. The SAT mode will focus on translating spoken language to ASL, while the SST mode will focus on conventional translation from Speech to Text to provide further insights on the English to ASL translation.

Our proposed real time translator with ASL interpretation bridges the gap between universal machine translation and physical human translation. This is critical in modern society where the greater population, including those with disabilities, rely on advanced artificial intelligence to enrich their daily lives. The goal of our Assistive Sign Language Bot Translator (ASLBoT) is to provide a user friendly experience, effective sign language rendering and a high-level of accuracy. Ultimately, ASLBoT aims to provide students with hearing disabilities the capability to provide effortless and natural communication within classroom environments.

II. Goals for ASLBoT

Based on our research findings, the main objective and overall goal of the ASLBoT is to be able to translate from English Speech to American Sign Language (ASL). To achieve this, the system will use a neural machine translation model using only small corpora and the proposed training framework found in this paper. The system has to perform Speech-to-Text and Speech-to-ASL language translations. To achieve Speech-to-Text translation, the system requires wireless capabilities to access cloud-based services. The system will indicate the current status of the system through LEDs and LCDs. To provide reliable translations, the system will capture audio recording from the user on real-time. To make the system more interactive, the system allows the user to dictate when to start the recording process and when to stop it. Finally, the system will display real-time rendering animation of the sign language gestures of the user input sentence. Additionally, to show transparency of the system, the system will show the original English sentence and the ASL translated sentence.

Our engineering requirement specifications are based on the goals and objectives set above. The microphone requirement for our system needs to be able to operate in the voice frequency range (300Hz to 3kHz). The memory size of our system needs to around 32GB so that it can hold the animation rendering of the ASL gestures locally. The system will require at maximum 32 W of power. The translation accuracy of the system is 20% on BLEU score. The response of the system should be less than 5 seconds. The vocabulary of the model should be larger than 50 words to add complexity in the sentence. The NMT data set should be less than 1000 entries to signify the use of small corpora.

III. Project Description

A. *Machine Translation*

For proper execution of interlingual translation, the system must be able to recognize the context of a given sentence or paragraphs. For example, word-by-word translation is not sufficient to produce an accurate translation. Thus, machine learning is required to perform the complex task of text-to-text translation. Machine translation was used for translating English to ASL-Gloss, where ASL-Gloss is a written form of ASL where gestures are marked by their equivalent English word or phrase. In this project, the English text generated by the Watson Speech-to-Text API from an audio file is directly passed into the machine translation system and translated into ASL-Gloss.

Within machine translation, there are several different approaches, with each approach providing a different algorithm for achieving the translation. The most common approaches are rule-based, statistical, and neural machine translation. Each machine learning method aims to translate the source sentence into a target sentence while accounting for the necessary context of the original sentence to produce an accurate translation. In other words, the meaning of the input text in one must be fully realized in the output text in another language **[1]**.

Rule-based machine translation (RBMT) and statistical machine translation (SMT) were initially considered but ultimately not used for the translation in this project from English to ASL-Gloss. RBMT relies on a sizeable amount of built-in linguistic rules and requires the availability of millions of bilingual dictionaries **[1]**. This approach was not feasible because large amounts of time and effort would be required to produce a database of rules for ASL-Gloss. This method would not be able to guarantee a successful translation from English to ASL-Gloss since this approach requires modification of the linguistic rules. SMT also proved to not be feasible since this method relies on creating several statistical models that rely heavily on existing multilingual corpora **[1]**. The computation of statistical models is also known to be CPU-intensive and requires custom hardware configuration. Therefore, creating a large multilingual corpus for English to ASL-Gloss would not be possible using this method.

The last approach was neural machine translation (NMT), and this method proved to be the most appropriate approach for the English to ASL-Gloss required in this project. NMT is a more modern approach to machine translation and large tech companies have already switched from SMT to NMT due to its proffered fluent translations. This method produces a single neural network with weights tuned through the training process rather than developing several models like in SMT **[2]**. This method also has the ability to maintain the context of a given sentence, which is not possible with the RBMT and SMT approaches. NMT achieves this by using sequence-to-sequence (seq2seq) based models. The seq2seq models are designed to have an encoder, an encoder vector, and decoder. Both the encoder and decoder of the seq2seq are made up of recurrent neural networks (RNNs). These RNNs allow the neural network to receive a series of inputs without having a predetermined number of inputs **[3]**, which is critical for machine translation since the amount of words in a given sentence is arbitrary. RNNs also include a hidden state vector that maintains the context of the previous inputs and outputs. This hidden layer is what allows the NMT model to give context to each word that would be translated, allowing for a more natural translation. The seq2seq models have the encoder RNN collect data and pushes it forward. The encoder vector then captures the information from all inputs of the encoder to provide the decoder with additional assistance in translating **[4]**. The decoder then delivers the final translation using the similar RNNs approach as the encoder. Seq2seq models are popular because of their ability to process despite having the input and output at different lengths since they are not associated.

The downside to using an NMT system for the translation is that a GPU is required for training the NMT model. The model also often requires a large training set such that the appropriate weights are reached, and a more natural translation occurs. The large training corpora issue was mitigated in this project through a two-step approach that allowed the NMT model to focus on particular words and cluster of words. This led to a more appropriate translation of English to ASL-Gloss without the use a large training corpus.

IV. System Components

The system is composed of many different components that connect and interface together to form the final product. This section provides technical details for each one of these components.

A. *Microcontroller*

The brain of the printed circuit board is a TI MSP430FR6922IG56R. The MSP430 family was chosen for its universal serial communication (USC) port, high capacity memory, and large number of general-purpose input and output pins. The chip runs at 16 MHz allowing for multiple operations to be done in quick succession, which is a vital quality for our system. The Universal Serial Communication port supports I^2C, SPI, and UART serial communication. The integrated

1. The avatar model, Amane Kisora-chan, was designed by SapphiArt Co. and is used in this project under the Extension Asset License provided by Unity.

development environment, Code Composer Studio, along with an MSP-EXP430FR6989 programmer, was used to develop, debug, and program the chip.

B. *Single Board Computer*

The powerhouse of the system is the UDOO x86 II Advanced Plus, which is a single board computer. This board was chosen for its x86_64 architecture which was required to run Unity-based applications such as ASLBoT, the application used in this project. The board features an Intel Celeron quad-core processor, 4G of RAM, an Intel HD Graphics card, and an Arduino Leonardo MCU that is connected to the processor by an internal USB connection. In addition, the board supports serial communications through the pins with protocols such as I^2C, UART and SPI. The UDOO x86 II has 32 GB of internal storage with a microSD card slot to expand the memory, if desired. Furthermore, the board features several interfacing options such as USB 3.0, HDMI, and Arduino pinouts. The integrated Arduino unit allows for easy communication between the microcontroller unit and the Unity application running on the UDOO board.

C. *Microphone*

The Blue Snowball microphone is used to record audio snippets to be processed and translated. This microphone was chosen because of its easy-to-use digital interface and seamless integration into the system. Its wide frequency range of 40 Hz to 18 KHz ensures that the user's voice could be picked up. It also features different audio detection patterns that allow for this microphone to be used in a variety of situations. The Snowball is connected and powered via USB and is plugged directly into the UDOO. The microphone does not require drivers to use, making setup effortless.

D. *LCD Display*

The LCD display for this project is a 2-line, 16-character display with blue backlighting. It takes a 5-V power supply and is used in 4-bit mode. Built into the module is a HD44780 controller, which allows for the microcontroller unit to program it. The display is used to indicate the status of the microphone to the user.

E. *IR Sensor*

The IR sensor for this project is a TSOP38238. This sensor can be powered at any voltage between 3 and 5 V and is tuned to 38 KHz making it perfect for the product. Also featured is improved ambient light and noise immunity, improving the final product's use in classroom settings. Furthermore, the IR codes are automatically demodulated as they arrive at an IR diode, allowing the signals to be much easier for the microcontroller unit to decode.

V. PCB Design & Utilization

Figure 1 shows the hardware block diagram of the final product. The system was powered by a 12-V, 3-A DC power source that plugs directly into the SBC board through a banana plug. Connected to the SBC is the display, microphone, and the printed circuit board which was used as a control board. The system display was powered by its own power supply, while the microphone was powered through its USB port. Furthermore, audio for the system was sent to the display's integrated speakers via an HDMI cable. The control board was connected to the single board computer by a 2-wire I^2C connection as well as 5-V power and ground connections for the board.

The 5-V power line from the UDOO was sent to the 16-character 2-line LCD display and IR sensor connected to control board. Furthermore, the 5-V power line was connected to an AMS1117 3.3 V regulator that stepped down the voltage from 5 V to 3.3 V for powering the MCU located on the control board.

The MCU was programmed to perform key functions for the system. Inputs were taken from the user in the form of button presses and an IR receiver. When a button on the IR remote was pressed, the MCU transmitted a series of codes over the I^2C connection that waked the system from sleep and automatically signed into Windows. Furthermore, when the start button on the PCB was pressed, the MCU sent a signal to the SBC that was translated as a keyboard press and made the system begin recording. A similar signal was sent when the stop button was pressed on the PCB, but instead signals for the game to stop the recording and begin translating the captured audio file. After these signals were sent to the SBC, an acknowledge signal will have been sent back and read by the control board. The final input was a reset button that is used in the case that an unexpected state was encountered, and the control board becomes stuck. The reset button cleared the LCD and was wired to the reset pin on the MCU.

Outputs on the control board included the LCD display along with status LEDs. The LCD display showed the current status of the microphone as either recording or ready. The LEDs show multiple statuses such as power, recording, transmitting, and receiving, which were all updated in real time.

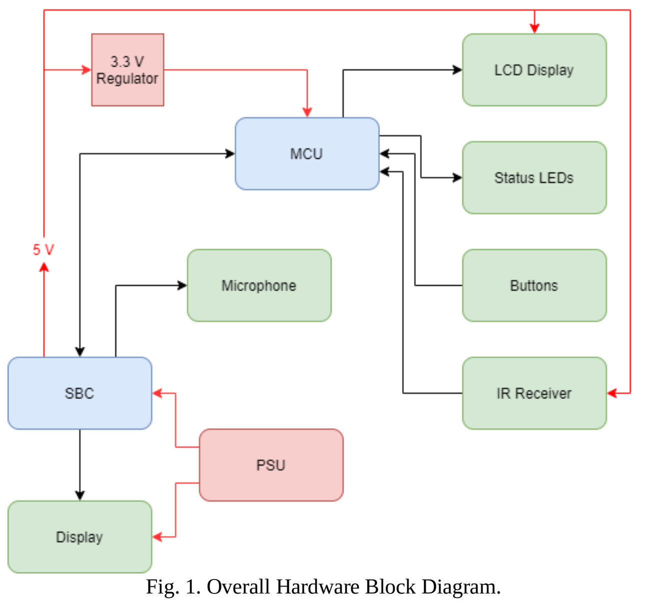


Fig. 1. Overall Hardware Block Diagram.

1. The avatar model, Amane Kisora-chan, was designed by SapphiArt Co. and is used in this project under the Extension Asset License provided by Unity.

Shown in Figure 2 is the finalized PCB board. Included on the board are 7 status LEDs, 3 push buttons, 5 male pin headers, a voltage regulator, and one MSP430FR6922 MCU. Control of the contrast of the LCD display was achieved using a 10 kΩ trimmer resistor that allowed for control of the voltage that was passed to the V0 pin. Due to shortcomings in the microphone chosen during the development phase of the PCB, the I2S microphone was abandoned; this decision resulted in leaving the 'MIC1' header left unused. Furthermore, the micro USB connector that was planned to power the PCB was not able to be soldered properly to the board due to its connectors being too small and brittle. Due to the COVID-19 pandemic, no new USB connectors for the board could be ordered. However, the board was designed robustly enough, and the system was able to be powered through the 'SBW_conn' header. Therefore, the initial board design was able to be used in the final product.

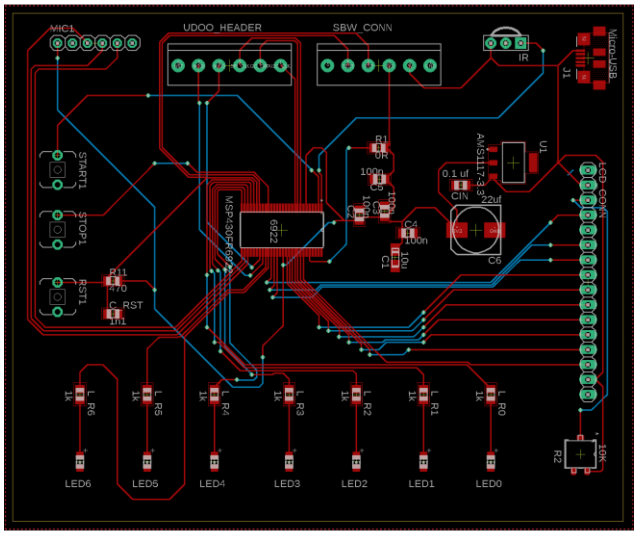


Fig. 1. PCB Layout.

VI. Software Details

A. OpenNMT

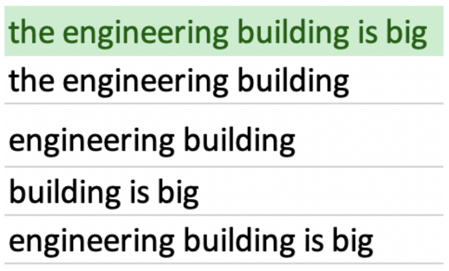
After deciding to use an NMT as the default machine translation approach for this project, which NMT system would be most viable for the purposes of this project needed to be decided. NMT systems were offered from the larger tech companies; however, these systems remain strictly proprietary which would not prevent modifications the system and train a suitable model to meet the specifications for this project. Open-source NMT systems were also available, and this style of an NMT was the main focus when deciding on which system to choose. After considering various open-source systems, OpenNMT was chosen due to its necessary provided support that other software lacked. Other software that were mainly used for research code did not provide substantial support. OpenNMT also provided support for the PyTorch and TensorFlow frameworks, which provide the ability to train and validate the NMT model from a high-level programming interface. Due to the arbitrary choice between the PyTorch and TensorFlow backbones in the scope of this project, the PyTorch framework was chosen for this project. This OpenNMT software was also based on the aforementioned seq2seq models.

The NMT presented the challenge of finding a GPU capable of training the model in a short amount of time. Training with a CPU would exceed 24 hours for each session and therefore would not be replicable when several models would need to be tweaked and re-trained. Google Research's Colaboratory (abbr. Colab) was used to overcome the issue of lacking a capable GPU to train the model. Using Colab allowed code to be written to and executed on Google's cloud servers and also use their available GPU's at no cost. Colab also provides a Python environment and therefore allows for simple integration of the PyTorch framework. Training the NMT model with Google's GPU took about 40 minutes per session.

Train the OpenNMT model requires four different text files. Two of the text files correspond to the English source corpus, and the other two correspond to the ASL-Gloss target corpus. Both the source and target corpora contained 516 entries within each corpus, whereas most training sets for other research teams contain up to hundreds of thousands of entries. A smaller set of corpora was used for the validation files in both English and ASL-Gloss. The validation files had 38 entries for each corpus, and the model used these files to evaluate the convergence of its training.

As aforementioned, NMTs require large training corpora in order to achieve a respectable translation. The unobtainability of large multilingual corpora for English to ASL-Gloss translations resulted in the 516-entry corpora. To overcome the lack of such a large training corpus, a two-step approach was utilized. The two-step approach included keeping the training corpora to a specific domain and employing redundancy among the sentences.

For the first step, the corpora were limited to contain sentences that were related to the school domain. Therefore, this restricted the number of words available throughout both corpora. Choosing the school setting as the primary domain also allowed for the use of a simpler vocabulary set and more commonly used sentences. After implementing the first approach, there showed improvements in the translations; however, the ASL-Gloss translations were still not acceptable for the purposes of this project. Two translation examples English to ASL-Gloss are shown in Figure 3. These two examples demonstrate that some words such as "Me" and "Need" are correctly translated but not the rest of the sentence.


the engineering building is big

the engineering building

engineering building

building is big

engineering building is big

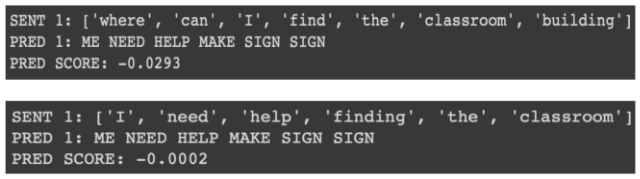
Fig.3. Example of breaking down a sentence from the corpus into smaller phrases for redundancy.

1. The avatar model, Amane Kisora-chan, was designed by SapphiArt Co. and is used in this project under the Extension Asset License provided by Unity.

The second step was to add to the training corpora redundancy. This was accomplished by splitting up sentences already in the corpora into smaller phrases as shown in Figure 3. Sentences similar to existing sentences were also introduced to the corpora. This introduction of redundancy greatly improved the performance of the resulting NMT model. This amelioration can be attributed to the neural network needing to force itself to adjust its weights and focus more on the limited and repeated words within the corpora. An example of the vast improvement after implementation of the second approach can be seen from Figure 4. Figure 5A is the previous example still using the first approach and Figure 5B is the same example using the second approach. Between these two figures, the addition of approach two to the training corpora providing a much more accurate and more acceptable translation can be shown. A similar improvement can similarly be seen between Figures 5C and 5D.

Evaluation of the performance of the model could have been done through several available machine translation scoring techniques. The BLEU score was used to calculate a score for the English to ASL-Gloss translation model. The BLEU score is known to be solution that is simple to implement to retrieve a score; however, this scoring method also include its draw backs [5]. The main objective of the BLEU score is to compare the n-grams of both the human-translated reference sentence and the machine-translated reference sentence [6]. The term "n-gram" refers to a consecutive string of words within a sentence; for example, a unigram is one word, a bigram is a set of two consecutive words, etc. within a sentence. The problem with using the BLEU score arises from the fact that the overall BLEU score for one sentence is calculated as the geometric mean of the all the n-grams calculated for this, where $n \in [1,4]$ for this project's scoring method. Therefore, for shorter sentences whose word count are less than the highest n-gram (in this case 4-gram), the resulting geometric mean is zero.

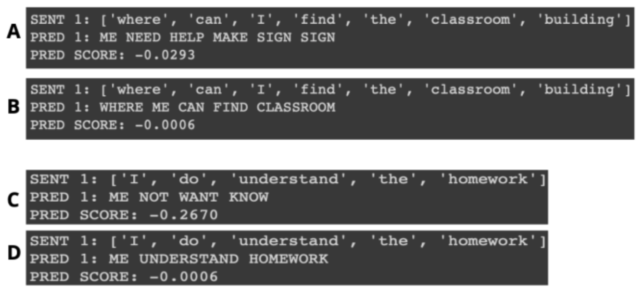
Evaluate the BLEU score more accurately was done by utilizing an open-source script, SacreBLEU. The script provides additional smoothing techniques that mitigate the issue when a higher n-gram precision resulted in zero [7]. For the calculation of the bleu score, a human-translation reference corpus and a machine-translated corpus were needed. For this project, 40 entries were used from the available corpora. The overall BLEU score and the subsequent scores for each n-gram match are shown in Table I, where BLEU-1 refers to the unigram-based score, BLEU-2 for digram scores, etc. The BLEU scores shown were multiplied by 100 from their original scores, since the original scores were given between 0 to 1. A score of 100 implies that the entire set of corpora completely match each other. Table I shows that the scores for BLEU-1 and BLEU-2 are much higher than the scores for BLEU-3 and BLEU-4. Since the ASL-Gloss translation is a truncation of English, and shorter English sentences were used to simplify the training, the resulting ASL-Gloss corpus was composed of several sentences containing less than 4 words. Therefore, evaluating the 4-gram BLEU score resulted in a very small number (nonzero as a result of the smoothing provided by SacreBLEU). There is a higher general probability for a unigram match than a digram match, or a digram match compared to a trigram match, and so on; this is why the n-gram BLEU scores show a marked decrease as the n-gram increases. Based on how English to ASL-Gloss translations function for the purposes of these training sets, the BLEU-1, BLEU-2, and BLEU-3 scores provide a more significant overall score than when combined with the calculated BLEU-4 score.



```
SENT 1: ['where', 'can', 'I', 'find', 'the', 'classroom', 'building']
PRED 1: ME NEED HELP MAKE SIGN SIGN
PRED SCORE: -0.0293
```

```
SENT 1: ['I', 'need', 'help', 'finding', 'the', 'classroom']
PRED 1: ME NEED HELP MAKE SIGN SIGN
PRED SCORE: -0.0002
```

Fig. 4. These are two translation examples from English to Gloss using the first training corpora modification approach. Neither of the attempts were satisfactory translations.



A
```
SENT 1: ['where', 'can', 'I', 'find', 'the', 'classroom', 'building']
PRED 1: ME NEED HELP MAKE SIGN SIGN
PRED SCORE: -0.0293
```

B
```
SENT 1: ['where', 'can', 'I', 'find', 'the', 'classroom', 'building']
PRED 1: WHERE ME CAN FIND CLASSROOM
PRED SCORE: -0.0006
```

C
```
SENT 1: ['I', 'do', 'understand', 'the', 'homework']
PRED 1: ME NOT WANT KNOW
PRED SCORE: -0.2670
```

D
```
SENT 1: ['I', 'do', 'understand', 'the', 'homework']
PRED 1: ME UNDERSTAND HOMEWORK
PRED SCORE: -0.0006
```

Fig. 5. Here the first and second approach compared for two translation examples of English to Gloss. Fig. 5A is translated using a corpora that only has aproach one, and Fig. 5B was translated with the addition of approach two. Similarly, Fig. 5C is translated using a corpora with approach one and Fig. 5D was translated with the addition of approach .two.

TABLE I. BLUE SCORE

Bleu Score – Corpora of 40 Entries				
Bleu Total	***Bleu-1***	***Bleu-2***	***Bleu-3***	***Bleu-4***
13.81	62.0	30.2	14.1	1.6

B. *Unity Engine*

The graphics rendering and user interface was created using a game built using the Unity 4 Engine. Included with this game are the game objects, which include a 3D avatar and a text-displaying canvas. An animation controller was also included to handle animation transitions and flow. C# scripts were attached to specific game objects according to the child objects referenced within each script. These scripts also communicated with the animator controller and between each other to handle timing of events within the game.

The game was built as a standard Unity game folder. Included with this pre-built game was a file folder titled "ASL" which included two files. The first was a vocabulary text file, which included a list of line-separated words that would be used during startup to develop a hash table based on the dictionary entries available to the OpenNMT model. The second was a file which included an OpenNMT model for translating text.

1. The avatar model, Amane Kisora-chan, was designed by SapphiArt Co. and is used in this project under the Extension Asset License provided by Unity.

Overall, the user has three available options while interfacing with the game. The first option is the START button on the PCB. Once this button is pressed, the microphone connected to the computer is enabled to begin recording. The second option is the STOP button on the PCB. If this button is pressed while the microphone had been recording, the game will stop recording from the microphone and proceed with its background translation and animation processes.

There are a set of processes that are performed before the game is initialized and another set of process that are performed after a WAV file has been generated. The first set of processes include the microphone initialization, the hash table creation, and component referencing. The microphone connected to the computer is initialized before the game starts, which forces the game to prompt the user to enable microphone privileges. The hash table is created by reading the vocabulary text file and associating each word in this file to the corresponding animation clip in the animation controller.

The overall software diagram for the system is shown in Fig. 6. The program will start in an idle state waiting for the input of the user. The user can input three different requests: the "start recording" request, the "stop recording" request and the "wake-up" request. All of these options are available through the PCB as button presses and IR signals.

Depending on the request, the system will first see if the board is on a sleep state (or off). The program will not start until the system is awaken from its sleep state by sending the wake-up request through an IR signal. After the system is awoken, the program will now wait for the user to request a "start recording" process. The background animation and translation processes include file conversion, command terminal calls, and variable manipulation. The recording interface was performed using the Microphone class and AudioSource object available in the UnityEngine library. The AudioSource object allowed the game to save an instance of an audio file to the game locally, while the Microphone class allowed the game to create a WAV file from an AudioSource object within Unity.

The Watson Speech-to-Text online API service was utilized in the script to handle conversion of the generated WAV file to a transcribed text file in a JSON format. This JSON file contains the transcript string corresponding to what was spoken in the microphone recording. This file was declassified and the transcript string was isolated and written to a text file. This text file was used as an input to the OpenNMT translate command which used the trained translation model. The output text file was read and parsed into a string array, from which the game could display each text consecutively. After the string array had been created, any unnecessary files are deleted to prepare the game for producing the next set of files for a new translation. These unnecessary files include the WAV file, the JSON file, and the two text files. The OpenNMT model, however, is kept as it is required to perform the next translation. Also, the vocabulary text file is kept since the hash table created upon starting the game is lost upon exiting the game and must be initialized using this file at the beginning of every gameplay.

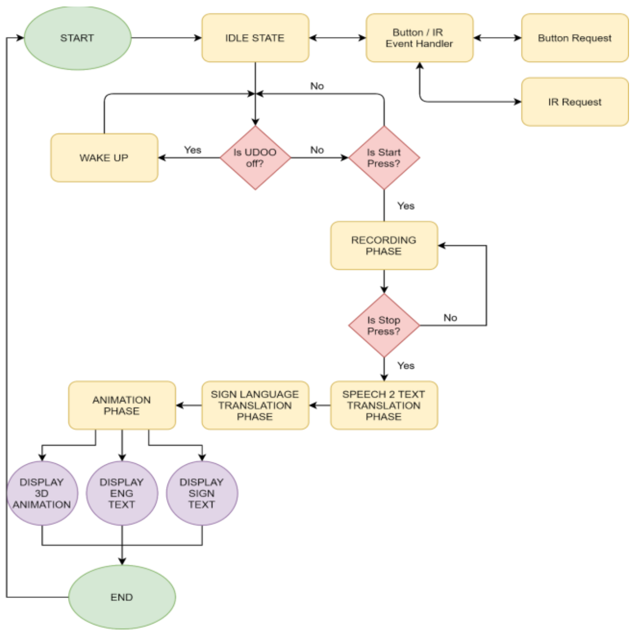


Fig. 6. Overall Software Flow Diagram

The animation controller was utilized by taking the generated string array and evaluating each string in order. The animations consists of an animation controller and a parameter list. The animation controller is composed of transitions to individual states containing a corresponding animation. The parameter list is composed of triggers which will enable a transition to the specified animation clip. A trigger is a special variable type in Unity which operates similar to a Boolean value; a trigger has a true/false nature similar to a Boolean, but once the trigger is set to true and is referenced by the animation controller, the trigger is automatically reset to false.

As the animation controller parses through the string array, if a string exactly matches an entry in the hash table, the string is passed to the animation controller and the corresponding trigger is set. If the string does not return a match, the string is further split into a string array; each letter is passed individually to the animation controller to set the triggers for the individual letter animations; this is referred as fingerspelling in the sign language community and is used for words that lack an applicable gesture in ASL. This hash table mapping is described in Fig. 7.

The animation controller starts with an "Entry" block which immediately transitions to the "IDLE" animation upon starting the game. At this point, any transition is handled using the "Any State" block in the controller. All transitions to animations from this "Any State" block are controlled by a unique trigger associated only with that transition. The animation controller utilizes a FIFO stack while processing triggers; i.e., no two triggers can be set simultaneously despite multiple possible transitions out of the "Any State" block. While the game is processing the audio file in the background, the controller transitions to the "THINKING" state.

1. The avatar model, Amane Kisora-chan, was designed by SapphiArt Co. and is used in this project under the Extension Asset License provided by Unity.

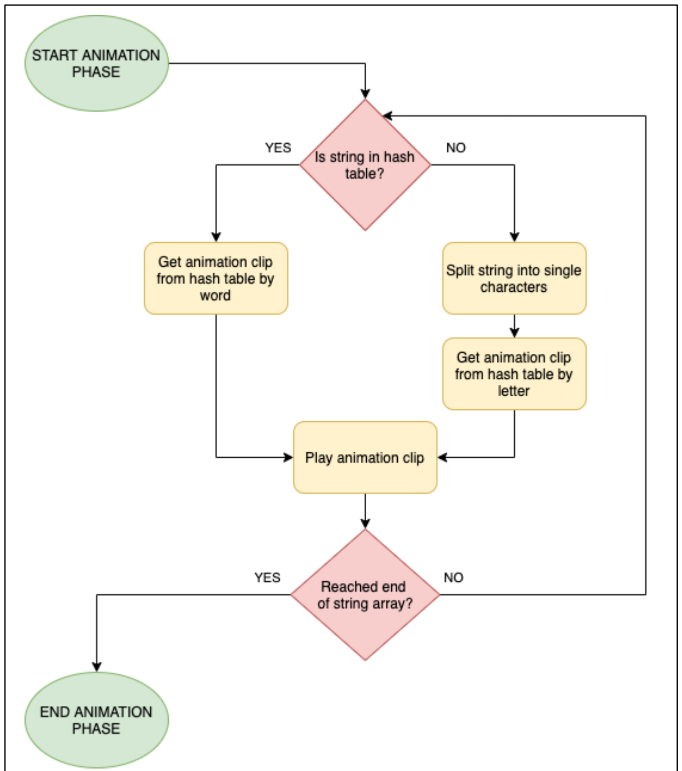

START ANIMATION PHASE
Is string in hash table?
YES
NO
Get animation clip from hash table by word
Split string into single characters
Get animation clip from hash table by letter
Play animation clip
Reached end of string array?
YES
NO
END ANIMATION PHASE

Fig. 7. Animation Flow Diagram.

After a string array is made available, the animator controller sets the corresponding trigger for an animation with respect to the order of the string array. After iterating through the string array, the controller transitions back to the “IDLE” animation. A fragment of the animator controller showing the “Entry”, “Any State”, “IDLE”, and “THINKING” animations is shown in Figure 8.

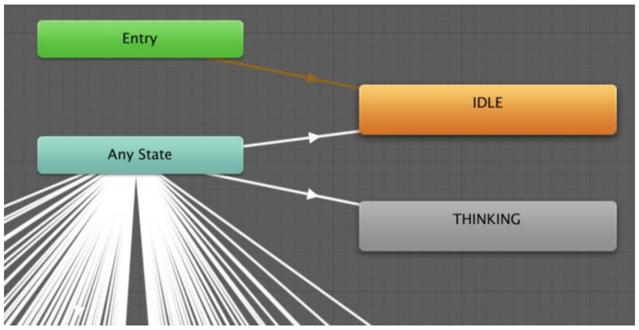


Fig. 8. Animator controller fragment.

Every individual animation was created using the animation clip editor within Unity. Animation clips were created by manipulating the 3D rotations of specific joints in the avatar. The bones required for animation of sign language gestures are shown in Figure 9 under the “Chest” directory. For the purposes of sign language gesture animations, the only joints utilized in this bone/joint map are the neck and the entire upper extremities from the fingers to the shoulders. The model, designed by SapphiArt Co.[1], contains a fairly standard pelvis-to-head bone/joint structure. Each bone in the bone/joint map is characterized by its root bone, followed by its child bones which are directly connected to it. A three-dimensional, global position, rotation, and size are also attributed to each bone.

During the recording of a sign language animation, the starting pose of the gesture was recorded at 0.00 seconds and the ending pose was recorded at 0.30 seconds temporarily. Key intermediate poses were also recorded between 0.00 and 0.30. For a more realistic animation, the timestamps of the final and intermediate poses were rearranged. Complex gestures, like ones requiring individual finger rotations that start and end consecutively, were recorded with an overall starting and ending pose, but intermediate poses were replaced with intermediate animation fragments that are blended together.

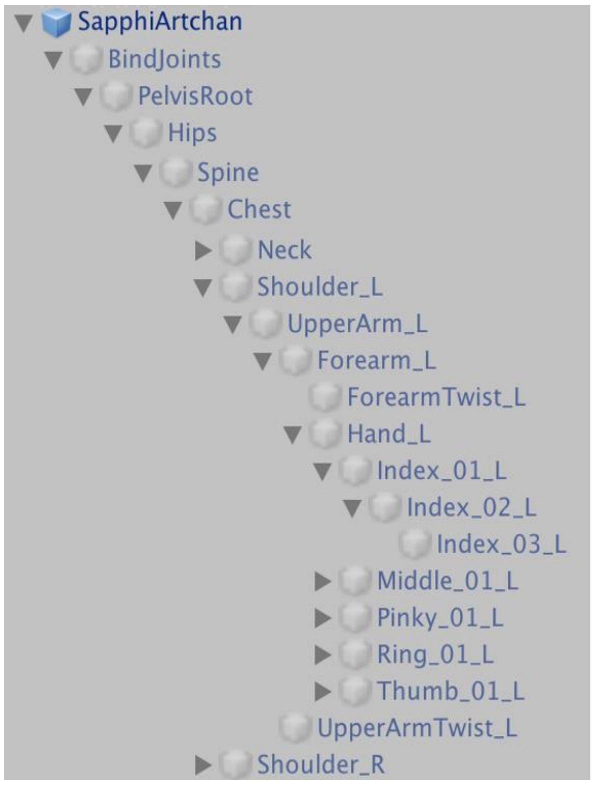


Fig. 9s. Bone/joint structure of Unity avatar, Amane Kisora-chan.

VII. User Utilization

Upon starting the game, the user is presented with the avatar in the foreground and the canvas in the background. The avatar will continue to display an idle animation until the microphone starts and finishes a recording. When the user presses the START button on the PCB, the canvas displays “Recording…” which prompts the user to speak a sentence into the microphone. The game waits until the user presses the STOP button on the PCB, but only the first 30 seconds of recorded audio would be kept. Once the STOP button is pressed, the avatar displays a thinking animation while the audio processing occurs. Once these background processes have finished, the canvas refreshes to display the English text generated from the Watson Speech-to-Text API and the sign-language gloss text generated from the OpenNMT model. While this text is being displayed, the avatar will perform the corresponding sign-language gestures. Once

1. The avatar model, Amane Kisora-chan, was designed by SapphiArt Co. and is used in this project under the Extension Asset License provided by Unity.

the avatar finishes signing the sentence, she returns to an idle animation state.

There are two warning messages that will be displayed in the top-left corner. The first warning message is displayed if the user attempts to quit the game while the avatar is not in the idle state. The second warning message is displayed if the user presses the STOP button without having first pressed the START button; without the START button being pressed, the microphone does not start recording.

VIII. Conclusion

The novel approach of training an NMT model based on a much smaller corpora by defining the domain and utilizing redundancy allowed for the realization of a machine-based interpreter that can potentially replace the need of human translators. Due to its cost-effectiveness, ease of use, reliability, and response time, ASLBoT is the first marketable out-of-the-box English-to-ASL machine translator solution that delivers results with no required installation time. The techniques used in the training framework allowed for the product to be deployed in different scenarios under specific domains, including but not limited to hospitals or classrooms. Moreover, the flexibility of this device allowed for any user to create their own smaller, personalized corpora rather than using extensive large corpora that require expensive technological equipment such as high-end GPUs. This means that product can be adapted to the needs of the user for a variety of situations. With the current growing demand for human interpreters in sign language translation being answered with poor-quality interpretation services, ASLBoT provides a crucial bridge to fulfill these shortages.

Acknowledgment

The group would like to extend our appreciation to Dr. Chung-Yong Chan. His guidance and input on the project while providing extensive feedback from the design stages to development stages of the project were priceless. Moreover, his willingness to always meet with the group and go over the progress of the project allowed for the group to succeed in the project. The group would also like to acknowledge Dr. Samuel Richie for his assistance on the project.

References

[1] "What Is Machine Translation? Rule Based Machine Translation vs. Statistical Machine Translation." *SYSTRAN*, www.systransoft.com/systran/translation-technology/what-is-machine-translation/.

[2] Bahdaau, Dzmitry, et al. *NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE*, 19 May 2016.

[3] Venkatachalam, Mahendran. "Recurrent Neural Networks." *Medium*, Towards Data Science, 22 June 2019, towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce.

[4] Kostadinov, Simeon. "Understanding Encoder-Decoder Sequence to Sequence Model." *Medium*, Towards Data Science, 10 Nov. 2019, towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346.

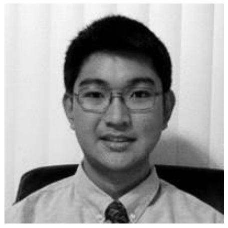
[5] Brownlee, Jason. "A Gentle Introduction to Calculating the BLEU Score for Text in Python." *Machine Learning Mastery*, 18 Dec. 2019, machinelearningmastery.com/calculate-bleu-score-for-text-python/

[6] Papineni, Kishore, et al. "Bleu." *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, 2001, doi:10.3115/1073083.1073135.

[7] Chen, Boxing, and Colin Cherry. "A Systematic Comparison of Smoothing Techniques for Sentence-Level BLEU." *Proceedings of the Ninth Workshop on Statistical Machine Translation*, 2014, doi:10.3115/v1/w14-3346.

Gustavo Camero is senior undergraduate student currently pursuing a BSCpE degree at the University of Central Florida. An experienced Research Fellow in the research areas of Computer Architecture, Emerging Devices and Education with two first-author publications. Currently set to pursuit a PhD in Computer Engineering at Carnegie Mellon University in Fall 2020. His main goal is to become a Professor to guide students from all sectors into achieving their dreams and to conduct research in the area of In-Memory Computing.

Luis Hurtado is a senior at the University of Central Florida pursuing a BSEE degree. During his time at UCF he worked as a research assistant from Prof. Yeonwoong Jung's lab, conducting research on unconventional semiconductor materials for next generation electronics. He will be attending Carnegie Mellon University in Fall 2020 where he will pursue a PhD in the department of Electrical and Computer Engineering.



Michael Loyd is a senior currently pursuing a BSEE degree and Bioengineering minor at the University of Central Florida. He is currently applying to medical schools with the goal of designing medical robotics and prosthetics.

Jared Spinks is a 21-year old graduating Computer Engineering student. He will be taking a job with L3Harris after graduation, as a FPGA Engineer, designing satellite payloads.

1. The avatar model, Amane Kisora-chan, was designed by SapphiArt Co. and is used in this project under the Extension Asset License provided by Unity.