Sign Language Recognition and Translation: A Multi-Modal Approach using Computer Vision and Natural Language Processing

Jacky Li, Jaren Gerdes, James Gojit, Austin Tao, Samyak Katke, Kate Nguyen, and **Benyamin Ahmadnia**
Department of Computer Engineering and Computer Science
California State University, Long Beach, United States
jacky.li01@student.csulb.edu, jaren.gerdes01@student.csulb.edu,
james.gojit@student.csulb.edu, austin.tao01@student.csulb.edu,
samyak.katke01@student.csulb.edu, kate.nguyen@student.csulb.edu,
benyamin.ahmadnia@csulb.edu

Abstract

Sign-to-Text (S2T) is a hand gesture recognition program in the American Sign Language (ASL) domain. The primary objective of S2T is to classify standard ASL alphabets and custom signs and convert the classifications into a stream of text using neural networks. This paper addresses the shortcomings of pure Computer Vision techniques and applies Natural Language Processing (NLP) as an additional layer of complexity to increase S2T's robustness.

1 Introduction

Globally, sign language is one of the main languages for those who cannot communicate verbally. Despite its global presence, not many people understand it or use it. In 2020, 48 million people in the United States alone experience some form of hearing loss, with less than 500,000 – about 1% – of them that drive sign language regularly (Lacke, 2020; NIDCD, 2021). The World Health Organization (WHO) estimates that the number of individuals with hearing loss will affect nearly 2.5 billion by 2050 (WHO, 2023). With these setbacks, signers may find it challenging to communicate with other individuals not akin to their mode of communication.

While mild hearing loss can be remedied with hearing aids and rehabilitation, these solutions may often be too expensive. Individuals can alternatively learn sign language. Hand gestures are a form of non-verbal communication used by individuals in conjunction with speech to communicate. With the increasing use of technology, hand-gesture recognition is considered an essential aspect of Human-Machine Interaction (HMI), allowing the machine to capture and interpret the user's intent and respond accordingly. The ability to discriminate between human gestures can help in several applications that range from virtual and augmented reality to healthcare services (Ceolini et al., 2020).

As technology becomes easier to use and accessible, many people can likely perform simple commands with computer devices, such as typing text and video streaming. To address the problem statements, we propose S2T – a solution to close the sign language knowledge gap by translating simple hand gestures into text.

1.1 Sign-to-Text v1

The first Sign-to-Text (S2T) iteration was implemented using Computer Vision to classify the English alphabet and custom gestures for text, such as space and delete. Computer Vision allows for gesture learning and recognition through images or video by identifying repeated patterns. Specific key descriptors can be isolated in a given frame using preprocessing techniques to eliminate noise and allow the neural network to perform on the highest data quality. While this process allows for the appropriate classification of newly introduced data, Computer Vision alone is not accurate enough to classify all ASL signs due to the limitations of Computer Vision and the nuances of ASL.

Classification accuracy in Computer Vision is dependent on the quality of the data. Two key factors that affect performance are image lighting, which affects how much detail can be seen, and image quality, which affects how much detail is retained. These can be seen within the data as qualities such as object luminosity, palm orientation, and hand shape.

The nuances of ASL are due to the limited range of signs. About 10,000 different ASL signs correspond to the English language or about 200,000 words. Some signs differ from others by a slight hand rotation, while others are polysemous. Signs that vary slightly with one another and signs that have multiple meanings make it near-impossible for

Proceedings of Recent Advances in Natural Language Processing, pages 658–665
Varna, Sep 4–6, 2023
https://doi.org/10.26615/978-954-452-092-2_071

Computer Vision alone to classify the signer's entire message with 100% accuracy, especially when trying to sign long sentences. Here we introduce Natural Language Processing (NLP) in conjunction with Computer Vision to overcome ASL nuances and address the weaknesses of Computer Vision as a standalone solution (Klingler, 2021).

1.2 Natural Language Processing

NLP is the computer's ability to understand language in both verbal and written forms. NLP is used in various applications, such as Speech Recognition, Language Translation, and Image Interpretation. In recent scientific research, it is also used to investigate inter-specie communication between humans and whales to understand and better aid them. S2T can improve output results by leveraging specific NLP techniques such as autocorrection and context awareness. S2T can also enhance accessibility by applying Machine Translation (MT).

1.2.1 Autocorrect

Autocorrect is a word processing task that identifies misspelled words and tries to resolve them by providing potentially intended words as a replacement. Autocorrect can be implemented in many ways depending on its use case, but all follow the same foundation to rely on some form of corpus or dictionary (D'Agostino, 2021).

The first iteration of S2T can correctly classify hand gestures with 82.76% accuracy. S2T can benefit from autocorrect by identifying misclassified alphabet gestures and replacing them with candidate words. This may help improve S2T's accuracy in achieving the desired final output.

1.2.2 Context Awareness

Simple autocorrection may not fully capture the user's intent in their sentences. Simple algorithms such as the Levenshtein distance would compare misspelled words too closely similar based on the number of edits from each word. This type of algorithm may often time alter and lose the original context, making it hardly usable for regular conversation language processing. Due to the complexity of languages, context awareness can be used to help retain the original context and convey user intent. Context awareness can be implemented in many ways, including part-of-speech tagging and attention mechanisms. The main idea behind context awareness is to analyze the sentence and extract key terms. These terms will then determine the best word to replace a target word (autocorrect), provide insight, and suggest the following word (autocomplete). When context awareness is used with autocorrect, it is more likely to retain the context of a given sentence and less likely to veer off (Wood, 2014).

1.2.3 Machine Translation

MT is an NLP technique that translates one language into another without the help of humans. There are four main types of MT techniques – Rule-Based Machine Translation (RBMT), Statistical Machine Translation (SMT), Neural Machine Translation (NMT), and Hybrid Machine Translation (HMT). Early iterations of MT use the rule-based approach to extrapolate grammatical rules as the basis for building sentences. However, this approach poses several limitations, such as the inability to process complex sentence structures and idioms. SMT is another approach where the system uses extensive bilingual data and statistical models to determine the most probable output. Like RBMT, SMT can also not process complex sentences and idioms (Martin et al., 2011).

NMT is a more recent approach that utilizes deep learning models. NMT takes advantage of being trained over large amounts of data, enabling it to process complex sentences and idioms as opposed to RMBT and SMT. Depending on how the data and model are prepared, these single-network approaches may not catch all translations. HMTs can be used to combat this by combining translation models to improve the output further (Brownlee, 2019; Torregrosa et al., 2019; Aulamo et al., 2021).

This paper is organized as follows; Section 2 reviews the previous related work. Section 3 details the proposed methodology. Section 4 outlines the experimental design. Section 5 describes and analyzes the experimental results, and finally, Section 6 concludes the paper and provides our future directions.

2 Related Work

This research will explore NLP techniques and apply them to S2T to enhance the translation quality after making prior classifications in Computer Vision. We know that the research field combining NLP and ASL is limited. However, it is noted that NLP can be applied to ASL applications when provided with some consumable input, such as text. In the field of NLP, immense research has been

put into autocorrect, context awareness, and machine translation. Since S2T can be broken into two parts (autocorrect and machine translation), we treat each part as an individual entity.

Autocorrect algorithms can vary in performance depending on their use case. However, they all follow a similar pattern by cross-referencing an accurate corpus to identify misspelled words. *TextBlob* is a standard open-source library launched in 2013 and has been widely used as a standard autocorrect tool (TextBlob, 2013). A study on *TextBlob* shows that it can correct 54.6875% of the mistakes in a given prompt. This low score can be due to *TextBlob's* over-correcting behavior and lack of information to correct it to the target word (Popovic, 2023).

There are also many machine translation algorithms and architectures that each perform best depending on the specific application. Transformer models commonly show great success and have been a standard in many NLP tasks since Google introduced them in 2017 (Caswell and Liang, 2020).

3 Sign-to-Text v2

S2T is equipped with computer vision techniques to translate sign language into text. We propose NLP as a second layer of data processing to enhance translation accuracy and introduce an extra translation feature to make the program more accessible. This additional layer will address the main drawbacks of Computer Vision as a standalone solution.

3.1 Classification Improvement

One major flaw of S2T-v1 has its low classification accuracy of 82.76%. Given the letter-by-letter translation nature of S2T, a letter-by-letter classification will most likely result in typos in a given text. To reduce the number of typos based on gesture classification, autocorrect can be used to detect and fix them. Traditionally, autocorrect can identify misspelled words by comparing the target words against a known dictionary or corpus. Advanced autocorrect features must be utilized, such as context awareness, due to the nature of how misspellings are created. With context awareness, it can further analyze the text stream to provide a closer and more appropriate approximation to the user's intended sentence.

3.2 Language Translation

Another feature S2T can leverage is transforming the English output into another language. This additional feature does not directly affect the classification accuracy of the original S2T implementation. Instead, language translation makes it more accessible for users to communicate effectively with various language speakers. The primary challenge that S2T will face is retaining context through its text processing transitions. As machine translation is the final layer of S2T, it will face potential inaccuracies in the initial phase of computer vision classification and the autocorrect technique. Our research explores and compares different autocorrect and machine translation methods to ensure the closest possible translation the user intends to convey.

4 Experimental Framework

4.1 Datasets

For autocorrect to perform well, it requires a dataset that contains correctly spelled words as the source of truth (GWICKS, 2018). Without this, the autocorrect would perform erroneous corrections, such as correcting correct words into incorrect words. This dataset must be pruned of any odd words that may be defined, as these words are infrequent in regular conversations. These sparse representations are pruned as it may negatively impact the autocorrect performance in accuracy.

The other dataset required for autocorrection would be a dictionary of words and corresponding frequencies, on which the autocorrect will base its corrections. Additionally, with the prior dataset, we can create a second dataset with words and their corresponding probabilities of appearing in the English language (Tatman, 2017).

Our work serves ASL, which directly transcribes into English. Therefore, it is necessary for any dataset we use to have bilingual alignments with the English language. Tatoeba, an open-source collective for sentences and translations, is our select source for the translation task (Tatoeba, 2006). Phrase pairs in the retrieved data consist of user-provided, collectively evaluated, and approved translations for many languages, including low-resource languages. As this work is not solely extensive into machine translation, our team found that the one-to-many translation mappings at the sentence level are cordial to our application.

In preparation for the NMT and SMT models observed in this work, given that we have chosen not to develop single-model, multilingual support, all bilingual pairs are uniformly processed. All punctuation is stripped, and all characters are lowercase where applicable. For NMT specifically, all tokens are vectorized before model training. We have also limited the vocabulary size for all models to reduce complexity in this iteration.

4.2 Autocorrection

We propose the following autocorrection algorithm in Algorithm (1).

Algorithm 1 Proposed Autocorrect Algorithm

procedure CORRECT(src, fn, ca_flag)
  $tgt \leftarrow$ Ø
  $words \leftarrow src.split()$
  $sgt \leftarrow$ context suggestions dictionary for src
  for $word$ in $words$ **do**
    $w, p \leftarrow$ true word, punctuation from $word$
    $ac\_sgt \leftarrow$ suggestions of w defined by fn
    if ac_sgt exists **then**
      append w to tgt
    else
      if ca_flag, w in $sgt.keys()$ **then**
        skew ac_sgt by an arbitrary amount using sgt as reference
        re-sort ac_sgt by descending similarity, probability
      end if
      append top result of ac_sgt to tgt
    end if
    append p to tgt
    append whitespace char to tgt
  end for
  return tgt as string
end procedure

Our autocorrection algorithm follows a general structure; however, we wanted to experiment with what word distance algorithm would work best for our project domain. Our team considered researching the performance differences between Minimum Edit Distance, Needleman-Wunsch, and Damerau-Levenshtein algorithms. As our baseline, *TextBlob* library's correction function will be used.

The Minimum Edit Distance algorithm (1), known formally as the Levenshtein Distance algorithm, measures the minimum difference between two words, x and y. The algorithm's recurrence is commonly used in dynamic programming (Nam, 2019).

Algorithm 2 Minimum Edit Distance Algorithm

$$D(i,j) = min \begin{cases} D(i-1,j) + del\_cost \\ D(i,j-1) + ins\_cost \\ D(i-1,j-1) + repl\_cost \end{cases} \tag{1}$$

$$repl\_cost = \begin{cases} miss\_cost \text{ if } x[i] \neq y[j] \\ match\_cost \text{ if } x[i] = y[j] \end{cases} \tag{2}$$

Algorithm 3 Needleman-Wunsch Algorithm

$$D(i,j) = max \begin{cases} D(i-1,j) + g \\ D(i,j-1) + g \\ D(i-1,j-1) + s(x_i, y_j) \end{cases} \tag{3}$$

The Minimum Edit Distance algorithm involves the usage of three cost variables: *del_cost*, *ins_cost*, and *repl_cost*, for each deletion, insertion, and replacement of a letter in word x at index i to the letter in word y at index j, respectively. These three variables can be set to whichever value the user wishes, but for our purposes, we set the values of *del_cost* to 1, *ins_cost* to 1, and *repl_cost* to one of two values as described in (2). Namely, if the letter of word x at index i is not equal to that of word y at index j, then *repl_cost* is set to a variable *miss_cost*, which is 2. Otherwise, *repl_cost* is set to another variable *match_cost*, which is 0.

The Needleman-Wunsch algorithm (3) generalizes the Levenshtein distance and considers global alignment (Kellis, 2021). It functions very similarly to the Minimum Edit Distance algorithm, filling in a similar table of values, but is used primarily in bioinformatics to align protein or nucleotide sequences. Because of this, gaps are punished and given a designated gap penalty in the algorithm's overall calculations.

In the algorithm definition defined in (3), g is the gap penalty, and $s(x_i, y_j)$ is the similarity score between words x and y at indices i and j, respectively. Unlike Minimum Edit Distance, which minimizes the number of edits to convert some word x to another word y, Needleman-Wunsch maximizes the score that an alignment between two sequences

Algorithm 4 Damerau-Levenshtein Algorithm

$$d_{a,b}(i,j) = min \begin{cases} 0 & \text{if } i = j = 0 \\ d_{a,b}(i,j-1) + 1 & \text{if } i > 0 \\ d_{a,b}(i-1,j) + 1 & \text{if } j > 0 \\ d_{a,b}(i-1,j-1) + 1_{(a_i \neq b_i)} & \text{if } i,j > 0 \\ d_{a,b}(i-2,j-2) + 1_{(a_i \neq b_i)} & \text{if } i,j > 1, a_i = b_{j-1}, a_{i-1} = b_j \end{cases} \quad (4)$$

could be.

The Damerau-Levenshtein algorithm (4) calculates the Damerau-Levenshtein distance between two given strings by following the same process as the classical Levenshtein distance but differs from this by including transpositions in its operations calculations (Zhao and Sahni, 2019). This algorithm first determines the optimal string alignment distance and then calculates a distance with adjacent transpositions. The applications of this algorithm include DNA and fraud detection, and the U.S. government uses it in export control.

TextBlob is a Python library for processing textual data. We used our project's $.correct()$ function to identify and correct misspelled words in a given string. This function works by utilizing a dictionary of English words, determining whether a word is correct. If incorrect, a list of possible words based on edit distances is generated, and the word with the least edit distance is selected.

To bolster the accuracy of our autocorrection algorithm, we also considered the implications of context awareness. The context awareness algorithm we used is part of the SpaCy module: the ContextualSpellCheck (Goel, 2020). This module is loaded into a SpaCy pipeline that can then perform on a given sentence string. ContextualSpellCheck will then analyze the entire input, identify misspelled words using an English dictionary, and suggest what each incorrect word should be based on the context of the words around it. The context of each of these words is trained through a model at word-by-word, sentence-by-sentence, and document (entirety) levels. These suggested words were then utilized in our minimum edit distance function to increase the priority of these context-based words being chosen as the ultimate correction. The SpaCy module ContextualSpellCheck was chosen over similar approaches, such as BERT (Bidirectional Encoder Representations from Transformers), due to its compatibility with our code. SpaCy allowed for quick evaluations and gave us the means to increase priority for individually chosen words numerically.

In our proposed autocorrect algorithm (1), we implement the SpaCy-ContextualSpellCheck pipeline as the assignment to sgt using the incorrect corpus src. We then skew the original autocorrect suggestions made by one of the given algorithms above using a word from src and, if context-awareness is allowed and the word is recognized in sgt. This aims to take the contextual suggestions and boost the probabilities of choosing those words. As a result, the words chosen before or after contextual skewing can lead to different words being given as the top result in ac_sgt.

To process the corpus, the algorithm temporarily "removes" directly subsequent punctuation for each word seen. This punctuation is then "returned" once this word is processed. The reason for this particular step results from how each word is processed. The current algorithm can receive an input word with punctuation and output without that punctuation, and the punctuation would get "eaten". If we allowed this to continue for an entire corpus, the corrected corpus could have a different contextual meaning from its original. As such, each word must be sub-processed so that if there is punctuation, that punctuation is saved and returned to its original place.

4.3 Machine Translation

There are many approaches to performing MT, as mentioned in Section 1.2.3. Considering the use cases for our pipeline, we seek methods that can produce quality translations with low overhead in terms of resource usage and increased speed. Initially, we decided to utilize large language models (LLMs) such as T5 or GPT for the end-to-end task. However, to better understand the modern machine translation task from its roots and assess methods built solely for translation, we have chosen to utilize NMT as the base approach, with SMT as a supplement to the outputs of the base model. Choos-

ing these two presents an opportunity to explore an HMT approach, which will be further elaborated in Section 6 as future work.

The NMT model utilized in this framework is the ever-familiar Transformer, trained on bilingual pairs. The Transformer is known to be a significant improvement over previous neural architectures like Recurrent Neural Networks (RNNs) and Gated Recurrent Units (GRUs) for sequence transduction (Vaswani et al., 2017). The key feature of the Transformer is the implementation of multi-head attention modules—generally, attention-based methods in artificial neural networks.

Simple word-based SMT was selected to supplement NMT, namely the IBM model series. As an overview, the IBM models consist of several iterations, each aiming to resolve the deficiencies from the previous, that utilize word alignment probabilities to generate tokens. Selective features such as fertility and context are included depending on the model version to improve the model outputs. In our work, we employed IBM Models 1 and 2 from Python's NLTK library, trained on the same bilingual pairs as the Transformer. These early iterations of the IBM series are outdated regarding a well-performing, standalone translation model. Despite this, we have chosen these models as a preliminary mechanism for establishing confidence in the outputs of the NMT model.

5 Result Analysis and Discussion

5.1 Autocorrection Results

Algorithm	% Fixed Errors
Needleman-Wunsch	49.67%
TextBlob	53.31%
Minimum Edit Distance	57.28%
Damerau-Levenshtein	58.28%
Needleman-Wunsch (CA)	59.60%
Damerau-Levenshtein (CA)	60.26%
Minimum Edit Distance (CA)	63.25%

Table 1: Results of each algorithm by the percentage of erroneous words fixed. CA is short for Context Awareness.

We ran each of our algorithms over fifty sentences with randomly distributed incorrect words. We compared these results to the corresponding correct sentence counterparts to determine the percentage of errors that were correctly fixed after being run.

Our findings showed that the Minimum Edit Distance (Levenshtein) algorithm utilizing context awareness performed the best out of all tested algorithms. In contrast, the base Needleman-Wunsch without context awareness performed the poorest. Without context awareness, Damerau-Levenshtein performed the best.

Overall, context awareness improved each algorithm that we tested. Needleman-Wunsch received the most improvement at ten percent but did not outrank the other context-aware options. Damerau-Levenshtein benefited the least from context awareness, and Minimum Edit Distance's percentage of errors fixed increased enough to bump it into first place in the algorithm rankings.

5.2 MT Results

Model	BLEU-4	ROUGE-1
Transformer	31.758	0.534
IBM Model 1	N/A	0.243
IBM Model 2	N/A	0.175

Table 2: Results of each algorithm by BLEU and ROUGE metrics, on Tatoeba EN-FR dataset. IBM Models were not evaluated on BLEU-4.

All three models were trained and evaluated on over 200,000 English-French bilingual pairs provided by Tatoeba (Tatoeba, 2023).

The Bilingual Evaluation Understudy (BLEU) metric is the prominent standard for supervised evaluation of the quality of machine-generated translations. As shown in Table 2, it is used to evaluate the Transformer model to verify that our implementation corresponds with other NMT standards. The IBM Models were not evaluated with BLEU, as we have decided that the purpose of these selected SMT methods would be better suited for unigram overlaps. Hence, we have also evaluated all models with ROUGE-1. Although not used as often as BLEU for judging translation quality, we have selected this metric based on determining each model's efficacy in generating relevant words for a desired translation. These observations drive future work of translation in our pipeline.

To compare, the training and evaluation of the original Transformer on the WMT14 English-to-French dataset scored 38.1 for BLEU. Using the same architecture on the Tatoeba dataset, we have obtained a score of 31.8, a 6.3% decrease.

6 Conclusions and Future Work

This paper proposes a multi-modal approach to improve sign language recognition and translation by combining computer vision and NLP techniques. By applying autocorrect as a fail-safe for computer vision classification, our team was able to fix 63.25% of the errors present in our dataset, which beats the baseline model by 9.94%. This improvement in word correction provides the machine translation layer to perform better as it can retain the context closest to the intended meaning. However, the NMT model implemented in this study performed slightly subpar compared to the original Transformer for English-to-French translation from different datasets. The evaluations conducted for SMT also show poor performance on the selected database. More extensive tuning and training on perhaps another corpus, such as those from past WMT conferences or OPUS, would benefit all methods selected here. This may also align the results of our implementation closer to those of related works utilizing the same architectures. As MT relies on the results of autocorrect, our plan plans to investigate more into improving the implementation of autocorrect. The root of misclassifications primarily comes from the results of computer vision first. While these misclassifications are due to the similarity between each gesture, not all gestures are utterly similar. This suggests that autocorrect can benefit from emphasizing weights for each classification group. By applying an additional bias per classification group, autocorrect can achieve increased correction accuracy overall.

Further improvements to autocorrect focus on an improved method of context awareness. The current implementation uses the SpaCy-ContextualSpellCheck pipeline. While it already improves upon standard autocorrect algorithms, the overall performance is still not substantial enough to be reliably used. Our team researched using the Viterbi algorithm to improve SpaCy by better determining the best corrections using part-of-speech tagging and hidden Markov models. We can further enhance SpaCy by directly implementing a BERT model step into the pipeline, allowing for more accurate predictions. Despite MT results in this work underperforming, we are looking to merge the sequencing capabilities of the attention-based neural network and the purely linguistic nature of the statistical approach to improve translation quality. Our future work seeks to leverage these approaches into a confidence-driven hybrid approach - justifying NMT outputs and resolving tokens estimated to have high uncertainty through SMT (Wang et al., 2016).

Acknowledgments

The authors thank the CSULB College of Engineering and the CSULB Department of Computer Engineering and Computer Science for their support.

References

Mikko Aulamo, Sami Virpioja, Yves Scherrer, and Jörg Tiedemann. 2021. Boosting neural machine translation from Finnish to Northern Sámi with rule-based backtranslation. In *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 351–356, Reykjavik, Iceland (Online). Linköping University Electronic Press, Sweden.

Jason Brownlee. 2019. A gentle introduction to neural machine translation. *Machine Learning Mastery*.

Isaac Caswell and Bowen Liang. 2020. Recent advances in google translate. *Google AI Blog*.

Enea Ceolini, Charlotte Frenkel, Sumit Bam Shrestha, Gemma Taverni, Lyes Khacef, Melika Payvand, and Elisa Donati. 2020. Hand-gesture recognition based on emg and event-based camera sensor fusion: A benchmark in neuromorphic computing. *Frontiers*.

Andrea D'Agostino. 2021. Nlp - how does an autocorrect model work? *Medium*.

Rajat Goel. 2020. Contextual spell check · spacy universe.

GWICKS. 2018. Dictionary dataset.

Manolis Kellis. 2021. The needleman-wunsch algorithm. *Biology LibreTexts*.

Nico Klingler. 2021. Why computer vision is difficult to implement? (and how to overcome).

Susan Lacke. 2020. Do all deaf people use sign language? *Accessibility.com: Empowering digital accessibility for businesses*.

Eric Martin, Samuel Kaski, Fei Zheng, Geoffrey I. Webb, Xiaojin Zhu, Ion Muslea, Kai Ming Ting, Michail Vlachos, Risto Miikkulainen, Alan Fern, and et al. 2011. Statistical machine translation. *Encyclopedia of Machine Learning*, page 912–915.

Ethan Nam. 2019. Understanding the levenshtein distance equation for beginners. *Medium*.

NIDCD. 2021. Quick statistics about hearing. *National Institute of Deafness and Other Communication Disorders*.

Kristina Popovic. 2023. Spelling correction in python with textblob. *StackAbuse*.

Rachael Tatman. 2017. English word frequency. *Kaggle*.

Tatoeba. 2006. Collection of sentences and translations.

Anki Tatoeba. 2023. Tab-delimited bilingual sentence pairs these are selected sentence pairs from the tatoeba project.

TextBlob. 2013. Simplified text processing.

Daniel Torregrosa, Nivranshu Pasricha, Maraim Masoud, Bharathi Raja Chakravarthi, Juan Alonso, Noe Casas, and Mihael Arcan. 2019. Leveraging rule-based machine translation knowledge for under-resourced neural machine translation models. In *Proceedings of Machine Translation Summit XVII: Translator, Project and User Tracks*, pages 125–133, Dublin, Ireland. European Association for Machine Translation.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv.org*.

Xing Wang, Zhengdong Lu, Zhaopeng Tu, Hang Li, Deyi Xiong, and Min Zhang. 2016. Neural machine translation advised by statistical machine translation. *arXiv.org*.

WHO. 2023. Deafness and hearing loss. *World Health Organization*.

Nicola Wood. 2014. Autocorrect awareness: Categorizing autocorrect changes and measuring authorial perceptions. *Florida State University*.

Chunchun Zhao and Sartaj Sahni. 2019. String correction using the damerau-levenshtein distance. *BMC Bioinformatics*, 20(S11).