School of Information Technology and
Engineering at the ADA University

School of Engineering and Applied Science
at the George Washington University

SPELLING CORRECTION FOR AZERBAIJANI LANGUAGE USING SEQUENCE TO SEQUENCE MODEL

A Thesis
Presented to the Graduate Program of Computer Science and Data Analytics
of the School of Information Technology and Engineering
ADA University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Computer Science and Data Analytics
ADA University

By
Asad Dashdamirli

April, 2023

THESIS ACCEPTANCE

This Thesis by: Asad Dashdamirli
Entitled: *Spelling correction for Azerbaijani language using sequence to sequence model*

has been approved as meeting the requirement for the Degree of Master of Science in Computer Science and Data Analytics of the School of Information Technology and Engineering, ADA University.

Approved:

(Adviser)	(Date)
(Program Director)	(Date)
(Dean)	(Date)

ABSTRACT

In natural language processing (NLP), spelling correction is an essential task which seeks to automatically correct misspelled words in text documents,. This thesis focuses on Azerbaijani language spelling correction, which presents unique challenges due to its rich morphology and complex orthographic norms.

Beginning with a comprehensive literature review covering the extant approaches and techniques for spelling correction in various languages, the thesis then proceeds to its methodology. We identify the limitations of existing methods and propose a novel approach for Azerbaijani orthography correction based on a sequence-to-sequence (seq2seq) deep neural network.

Our proposed method makes use of seq2seq models, which have demonstrated great success in a variety of NLP tasks, to discover the mapping between misspelled words and their right counterparts. In addition, we introduce techniques for generating artificial noise to augment the training data and enhance the model's ability to manage various types of misspellings.

We conduct extensive experiments on a large corpus of Azerbaijani text data in order to evaluate the performance of our approach. We evaluate the results in terms of character error rate, word error rate and sequence error rate by comparing our method to several other methods. Our experiments demonstrate that our seq2seq-based approach reaches adequate results, 5.3% character error rate and 25.77% word error rate in text from news which shows potential to enhance the accuracy of Azerbaijani text spelling correction.

In addition, we analyze the effect of artificial noise generation techniques on the performance of our model and provide insights into how effective they are in managing various misspelling types. In addition, we discuss the limitations of our methodology and possible future directions for further development.

This thesis concludes with a novel approach to Azerbaijani spelling correction using deep neural networks, specifically the seq2seq model, along with artificial noise generation techniques. The experimental results demonstrate the viability of our method for enhancing the precision of Azerbaijani text documents in real-world settings.

ACADEMIC INTEGRITY STATEMENT

"I affirm that this is my own work, I attributed where I used the work of others, I did not facilitate academic dishonesty for myself or others, and I used only authorized resources for my Thesis, per the ADA University Academic Integrity requirements. If I failed to comply with this statement, I understand consequences will follow my actions. Consequences may range from failing the course to expulsion from the program/university and may include a transcript notation."

Asad Dashdamirli		24.04.2023
(Full Name)	(Signature)	(Date:DD.MM.YY)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Explanation
NLP	Natural Language Processing
ASC	Automatic Spelling Correction
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Units
CER	Character Error Rate
SER	Sequence Error Rate
WER	Word Error Rate

1 Introduction

While humans are able to communicate using natural languages such as English, Spanish, and Mandarin, the majority of people are unable to comprehend machine code, also known as machine language, which is the computer's native language. Instead of words and sentences, machine code consists of millions of zeros and ones that indicate logical operations. This lets computers to process and execute extremely basic commands, but it is difficult for people to read and comprehend. As a result, humans often communicate with computers using programming languages that are closer to natural language, such as Python or Java.

Due to the complexity and diversity of languages, it is challenging for computers to comprehend and generate natural language. There are numerous languages, dialects, and modes of self-expression, and each has its own set of laws and traditions. Nevertheless, languages are frequently unstructured and contain mistakes, omissions, and other anomalies. While machine learning techniques such as deep learning and supervised learning can assist computers in mimicking human language, they do not always capture the full range or underlying structure of human language. Natural language processing provides a framework for addressing the complexity and ambiguity of human language. NLP is essential for applications such as speech recognition and text analytics, where it may help structure and make sense of vast amounts of unstructured data.

NLP has numerous applications, including enabling computers to speak with humans in their native tongue. NLP can be used, for instance, to enable computers to read text, listen to voice, comprehend the meaning and tone of the language, and extract significant points. With the rising availability of massive datasets and sophisticated computational resources, computers can now analyze language-based data on a scale that would be difficult for people to accomplish manually. This is especially crucial given the everyday generation of enormous quantities of unstructured data, such as social media posts and medical records, which require rapid and objective analysis.

Spelling correction is one of the most important and influential applications of natural language processing. Researchers have developed and polished a variety of methods for automatically detecting and fixing misspellings in text over the years. These approaches are implemented in a variety of commonplace applications, including word prediction in text messaging, spell-checkers in word processing software, and query prediction in search engines.

1.1 Definition of the Problem

Spelling errors in the Azerbaijani language are a widespread problem that affects both native and non-native speakers. Azerbaijani has a complex orthography, with numerous spelling rules and exceptions that can be challenging to comprehend. This complexity can lead to frequent errors in written texts, which can result in ambiguity, confusion, and misunderstanding.

In sectors where precision and clarity are crucial, such as academia, business, and government, the prevalence of spelling errors in Azerbaijani is particularly pronounced. In academic writing, for instance, misspellings can result in a reduced grade or even the paper's rejection. Such errors in business and government documents can result in misunderstandings and even legal disputes.

Also detrimental to the credibility and professionalism of written communication are misspellings. This is especially true in the age of digital communication, where written texts are used frequently in email, social media, and online forums.

Developing an effective spelling correction system for the Azerbaijani language is essential for enhancing the quality and precision of communication in a variety of disciplines. This system can aid in the reduction of misspellings, the improvement of legibility, and the enhancement of the overall professionalism of written communication.

Although there are a great number of complex models for the English language, this is not the case for languages with limited resources, one of which is Azerbaijani. Because of this problem, I propose a method for the problem of correcting language that makes use of neural networks. The fundamental component of this method is a recurrent neural network equipped with an attention mechanism that functions as an encoder-decoder. By focusing on the level of the character, the network is able to circumvent the problem of terms that are not in its lexicon.

1.2 Objective of the study

The purpose of this research is to develop an efficient Azerbaijani language orthography correction system. The system will be founded on machine learning techniques and will take the unique characteristics of the Azerbaijani language into account.

The development of an efficient spelling correction system necessitates a comprehensive comprehension of the Azerbaijani language's specific spelling rules and patterns, as well as the most frequent errors made by speakers. Therefore, the study will include a comprehensive examination of the orthography rules and patterns of the Azerbaijani language, as well as a review of the most frequent errors made by speakers in written texts.

Based on this analysis, the study will design and construct a machine learning model capable of accurately detecting and correcting Azerbaijani spelling errors. The model will be trained on a large corpus of text data in Azerbaijani, which will include data from new agencies. The system will identify and remedy spelling errors using machine learning models.

To evaluate the performance of the spelling correction system, a series of experiments will be conducted to measure the system's precision and efficacy. Experiments will entail testing the system on a variety of text data types, including news and data from the economic domain.

The ultimate objective of this research is to develop a spell-checking system that will enhance the precision and clarity of written Azerbaijani communication. A wide variety of individuals and organizations, including students, academics, business professionals, and government agencies, can utilize such a system.

1.3 Significance of the problem

The problem of mistakes in spelling in the Azerbaijani language is significant and has far-reaching repercussions in a variety of communication contexts, including academia, business, and government.

Spelling errors can result in a reduced grade or even the rejection of a paper in academic settings. Academic papers are expected to satisfy stringent requirements for precision and clarity, and misspellings can undermine the credibility of the research and the author. Incorrect spelling can also make it difficult for readers to comprehend the content, which can hinder the spread of knowledge.

In business, misspelled words can negatively affect a company's reputation. Professional and error-free business documents, such as emails, reports, and contracts, are expected. Spelling mistakes can create the impression of sloppiness or a lack of attention to detail, which can negatively affect the company's relationships with clients, partners, and investors.

In the government, misspellings can contribute to confusion and even legal disputes. It is expected that government documents, such as contracts, policies, and regulations, are plain and unambiguous. Errors in spelling can lead to confusion and ambiguity, which can have severe consequences, especially in areas like law and public policy.

Moreover, in the age of digital communication, written texts are frequently used in email, social media, and online forums. In such contexts, misspellings can generate a negative impression of the author and reduce the effectiveness of communication.

Creating an effective spelling correction system for the Azerbaijani language is essential for enhancing the quality and precision of written communication in a variety of disciplines. This system can aid in the reduction of misspellings, the improvement of legibility, and the enhancement of the overall professionalism of written communication. It can also help promote the use of Azerbaijani in academic, professional, and government contexts, which can have positive effects on the language's preservation and promotion.

1.4 Review of significant research

There has been a significant amount of research on spelling correction for English. However, there has been relatively little research on spelling correction for other languages, such as Azerbaijani.

In [59] The paper presents a neural network-based spelling correction system for the Azerbaijani language. The system is trained on a large corpus of Azerbaijani text, and it is able to correct a variety of spelling errors, including single-letter errors, transposition errors, insertion errors, deletion errors and replacement errors. The system is evaluated on a test set of Azerbaijani text, and it is shown to be effective at correcting spelling errors. The system is also compared to a rule-based spelling correction system, and it is shown to be more accurate. Some of the key findings of the paper are that a neural network-based spelling correction system can be effective at correcting spelling errors in the Azerbaijani language and that the system is more accurate than a rule-based spelling correction system.

In general, dictionary-based spell checking systems utilize exhaustive dictionaries that include all conceivable words and their respective frequencies or probability of being spelled properly in a particular context. In the event that a term does not match any entry in the dictionary, the system’s algorithm flags it as an error for additional review by humans or artificial intelligence agents. The key advantage of this method is its simplicity, as it involves only simple lookup operations and does not require sophisticated algorithms. Due to their reliance on predefined dictionaries, however, these

systems may not be able to recognize new words or slang expressions that have not yet been added to the databases, resulting in erroneous or insufficient spell checking.

Deep learning models offer a significantly higher level of precision than more traditional approaches, while at the same time requiring significantly less manual labor during the training phases. These models make use of neural networks that are trained on enormous datasets that contain millions of input/output pairs that represent correctly spelled words and words that have been misspelled. The more times neural networks are exposed to these datasets, the more adept they become at determining whether a particular sequence should remain constant based solely on what has been observed in the past. Neural networks are able to do this without heavily relying on predetermined rule-sets, such as those that can be found in static dictionaries. Even though natural language processing is still a fairly new technology, it has a tremendous amount of potential for future applications. One example of this potential is the creation of automatic proofreading systems that are run entirely by artificial intelligence rather than requiring human involvement.

1.5 Assumptions and Limitations

The spelling correction methods that are now available have a number of drawbacks. One of the primary obstacles is the fact that the majority of the algorithms used for correcting spelling are based on probabilistic models that are trained using vast bodies of text. This indicates that the accuracy of the correction suggestions may differ depending on the quality and relevance of the training data. Consequently: In addition, the vast majority of spelling correction algorithms make assumptions regarding the dispersion of words and errors throughout the text. Examples of common spelling errors that can be corrected by algorithms include letters being switched around or vowels being left out. On the other hand, these algorithms might not be able to deal with errors that are more uncommon or unique, such as made-up words or misspellings that do not adhere to the typical patterns.

Traditional word processors do not include a context verification feature, which is another drawback of these programs. In most cases, these methods examine each word on its own, without taking into account how the word functions within the phrase or the paragraph as a whole. Homonyms are words that sound the same but have various meanings and pronunciations. This can lead to problems with homonyms, which are words that sound the same but have different meanings. For instance, the spell checker might show that a statement is correct despite the fact that it utilizes the incorrect homonym, such as "Their to proud to admit they were wrong" rather than "They’re too proud to admit they were wrong."

Context verification is something that has to be integrated into the system so that spell checkers can become more accurate. The application of natural language processing methods, such as part-ofspeech tagging and syntactic analysis, can be utilized to accomplish this goal. These methods provide assistance to the system in determining the intended meaning and application of each word contained inside a phrase. The spell checker is able to deliver more accurate and helpful suggestions for repairs because it takes into account the context in which a word is used.

It is necessary to possess strong spelling and reading skills in order to make efficient use of a spell checker. Whenever a term is misspelled, the spell checker will often provide a list of alternatives for the misspelled word; however, these suggestions may not be helpful unless the initial effort is somewhat near to the proper spelling. Even if the user is given ideas that are logical, they still need to

be able to accurately comprehend those suggestions. In addition to this, it is essential to have an understanding of the constraints that are imposed by spell checkers. For instance, a spell checker might not detect a word that is written correctly but is used in the incorrect context. This could occur if the word is typed "super" instead of "supper" or if "nece" rather than "necə" is used when writing in Azerbaijani. In addition, some words, such as proper nouns or specialized terminology, may not be recognized by spell checkers even though they are entirely legitimate. Because of these restrictions, it is essential for users to have a solid grasp of the language in order to make efficient use of spell checkers.

In addition, the vast majority of programs that are designed to correct spelling focus on correcting individual words rather than modifying entire sentences or chunks of text. Because of this, it is possible that they will not be able to handle problems that are more complicated, such as those including erroneous punctuation, missing words, or incorrect sentence structure. Also, it is possible that these algorithms will not be able to make ideas for alternate phrasings or methods of expressing an idea, which can be helpful for increasing the overall clarity and quality of written work.

There are numerous constraints that can influence how accurate and efficient spelling correction systems are. These constraints can include: Because NLP algorithms are trained on enormous corpora of text, one of the primary issues is that their accuracy and performance might vary depending on the quality and relevancy of the training material. This presents a significant difficulty that must be overcome. An NLP system that has been trained on formal, academic writing may not be able to deliver correct suggestions when applied to text that is more colloquial or casual.

In general, while the currently available approaches for correcting spelling can be beneficial for locating and fixing specific instances of misspelled words, these methods also have drawbacks that can compromise their accuracy and their applicability in certain contexts. It is necessary to conduct additional research and development in order to enhance the capabilities of these algorithms and find solutions for their limits.

2 Literature review

In 1964, Damerau conducted one of the earliest formal investigations into spelling correction (1). In his research, he introduced a fundamental strategy for correcting spelling mistakes, which relied on the concept that every word not present in the dictionary must contain at least one error that falls into one of four distinct categories. This seminal work established the basis for many of the techniques currently utilized in modern natural language processing systems for spelling correction. The four categories outlined by Damerau include:

- A word may be incorrect
- A word may contain an additional letter
- A word may be missing a letter
- A word may contain a single transposition, whereby a pair of adjacent letters in the word areinterchanged.

The author notes that the four categories of spelling errors identified in Damerau's study are believed to account for roughly 80% of all spelling mistakes. To implement the proposed strategy, Damerau utilized a bit encoding system to represent characters in a 28-bit register, with the last two bits reserved for special characters and numbers. The author highlights that Damerau's approach relied on a heuristic comparison of strings of similar length to the query string to detect any errors. This method is one of the earliest examples of using machine learning techniques for automatic spelling correction in natural language processing systems.

The vast majority of spell checkers available on the today locate misspelled words within a given text by consulting a dictionary that contains only correctly spelled terms. These kinds of systems frequently resort to effective lookup strategies like hash tables or search trees in order to attain high levels of performance. This method is utilized by a large number of prominent open-source spelling correction systems, such as Aspell and Hunspell, to recognize misspelled words, propose probable repairs, and score these candidates based on the likelihood that they are correct. In order to do spelling correction, these techniques are utilized extensively across a wide variety of natural language processing applications.

By doing a semantic context analysis on the text being checked, automated spellcheckers are able to identify real-world spelling errors. This is a task that involves a deeper comprehension of the meaning as well as the structure of the language, making it more difficult than the identification of non-word mistakes. Researchers have been able to discover real-word problems by using language models, such as homophonic faults in the Bangla language (3). These algorithms are able to identify words that do not work well inside the context and offer suggestions for terms that are a better fit.

Boytsov (4) examined different methods for approximating matching when indexing dictionaries for spelling correction. Deorowicz and Ciura (5) noted that a dictionary consisting solely of correct words can be overly large, leading to false positives or difficulties identifying unusual spellings. For languages such as Chinese, where words lack separation by spaces, distinct techniques are required to identify spelling errors. For instance, the researchers in (6) employed a blend of character embeddings and conditional random field classification to identify spelling errors in Chinese text. This approach can deal with the absence of word boundaries in Chinese and identify spelling errors in context.

Once a spelling error has been identified, automatic spelling correction (ASC) systems typically generate a list of potential corrections from a lexicon of correct words. While it is technically possible to include every correct word as a candidate, it is more efficient to limit the search to words that are related to the detected error. This can help to reduce the number of false positives and make the correction process more efficient.

In addition to edit distance, candidate generation also makes use of n-grams, Soundex, Speedcop systems, semantic feature vectors, and a variety of other phonetic indexing methods. A phonetic algorithm known as Soundex is used to index names according to the sound they make. This ensures that homophones are encoded to the same representation, allowing them to match despite slight variations in the spelling of the names. It is mostly used to encode consonants and will only encode a vowel if it is the initial letter. When it comes time to generate candidates, it will only filter out words that have the same Soundex key as the word that was supplied.

Moreover, keys could be created using the Speedcop systems. It begins with the first letter, then moves on to the consonant letters of the word, and finally moves on to the vowel letters, with each letter being recorded a single time. The keys were arranged in the order that they appeared in the word. When the system was given an incorrect spelling, it immediately began calculating the key to the incorrect word and searching for candidates. Both the Soundex and the Speedcop key are methods that can be utilized to bring the section of the vocabulary that needs to be considered down to a more reasonable size. But, if the term that is required is not included in the set of words that are being considered, the corrector will not locate it. Another phonetic technique called Metaphone (7) is used by GNU Aspell to represent words by their approximate pronunciation (that is, phonetic information). This approach makes it possible to fix words that appear to be orthographically different from one another.

Nonetheless, compute power and speed continue to be a challenge when dealing with enormous datasets. The utilization of Bloom Filters, trie, smart caching, or n-gram indexing techniques are some of the ways that efficiency can be increased. Bloom Filters are utilized by a piece of software by the name of STAVA, which was developed by Viggo Kann (8) and is responsible for locating and correcting misspellings in Swedish. To achieve a higher level of productivity in (9), Mawardi et al. make use of the trie algorithm and the word bigram index. Increasing one’s productivity can be done in a number of ways, some of which have been mentioned above.

Whitelaw and his colleagues (10) from Google proposed that for each token in the input text, potential proposals be selected from the word list and rated using an error model. These choices are re-ranked after being examined contextually through the use of a language model. We utilize a classifier for each token to establish our level of certainty regarding whether or not a word has been misspelled and, if it has, whether or not it ought to be autocorrected to the proposal that has the highest possible score.

The method (11) developed by Peter Norvig involves deriving all feasible terms that are located within an edit distance of two or fewer steps away from the query term and then looking those terms up in the dictionary. For a word with a length of n, an alphabet size of a, and an edit distance of 1, there will be n deletions, n-1 transpositions, a*n alterations, and a*(n+1) insertions. This will result in a total of 2n+2an+a-1 terms being found during the search, which is a slow process but provides a higher level of precision. On the other hand, Wolf Garbe (12) utilized a symmetric delete algorithm,

which is three orders of magnitude faster than the one that Norving took. It only derives deletes with an edit distance that is either less than two or equal to two, and that applies to both the query word and each dictionary term. There will be simply n deletions for a total of n terms at search time for a word that has a length of n, an alphabet size of a, and an edit distance of 1. This will make it possible for Peter Norvig's approach to work more quickly.

The method developed by Garbe additionally precomputes all deletes for each and every phrase in the dictionary. If the dictionary is very large, performing the necessary precalculations of time and storage space may be inefficient. The noisy channel model (13) is yet another well-known model that is utilized for the purpose of spelling correction. The interpretation is probabilistic, which means that you assume that the first version has some probability (prior), and that the addition of specific noise has some probability as well (conditional). Using these probabilities, one can determine the original version that is most likely to have occurred given the version that was actually observed.

The development of a statistical model for spelling errors by Brill and colleagues (14) was the first step toward the creation of an autonomously trainable spelling system. This was accomplished through the use of a training corpus that was manually generated and consisted of misspelled words. An expansion of the noisy channel model was presented here.

There are several different strategies that make use of n-gram models. It might be a character n-gram, or it might be a word n-gram. N-grams are a useful tool for integrating information into context. This methodology (15) presented correction ideas by picking the most promising choices from a prioritized list of correction candidates that were created based on character n-gram statistics and lexical resources. The authors of this methodology are Ahmed, et.al.

The researchers Bassil and Alwani (16) used a character-based 2-gram model and the Google Web 1T unigram data set to generate a list of candidate spellings for every detected error in the text. They then used the Google Web 1T 5-gram data set to perform context-sensitive error correction and select the most appropriate spelling candidates.

A system was developed by Wint et al. (17) that uses dictionary techniques to check words, twelve different types of error checking and correction approaches to correct non-word errors (the names of these approaches are Plural to singular, Removing End Characters, Removing Middle Characters, Reordering, Similar Shape, Similar Sound, Dividing, Removing Extra Character, Adding Character, Anagram, Nearby Key and Changing Character), and word n-gram and Levenshtein distance In the event that more than one corrected word is obtained from each method, the system employs word n-gram techniques to select the most appropriate and logically consistent corrected word from the words contained in the n-gram database. A careful application of semantic spaces could be useful if you want to take into account the context of the words.

A conceptual expansion, which includes the addition of related words by employing word embedding models such as Word2vec or Glove. The pre-trained word embedding model known as Glove (18) is capable of representing words in the form of feature vectors and may be found here.

Zhang (23) defined similarity joining as the process of identifying all pairs of strings that have a similarity above a certain threshold, as determined by a given distance function. Kernighan et al. (24) suggested restricting the set of candidate corrections to words that differ from the original word by only

one edit operation, such as substitution, insertion, deletion, or replacement of consecutive letters. This approach, which is based on the Damerau-Levenshtein edit distance (25), can help to reduce the number of false positives and make the correction process more efficient.

To generate a list of potential correction candidates for misspelled words, the creators of (26) employed a character-level language model that had been trained on a vocabulary of permissible phrases. Levenshtein distance was used by Reffle (27) to suggest corrections, while Yu et al. (28) offered a summary of approximation searching methods. To expedite approximative dictionary searches, they used an index. A finite-state automaton was utilized by the creators of (5; 29) to expedite their search for relevant content inside the lexicon.

Mixing numerous statistical models is frequently advantageous for context modeling. For instance, Melero et al. (30) presented a spelling system that used numerous language models with varying weights. Language models can concentrate on a variety of characteristics, including lemmas, part-ofspeech tags, uppercase and lowercase words, etc.

The authors of (32) suggested a context model that employs several Bayesian classifiers. The initial component of the model employs trigrams to create candidate words with the most probable part-of-speech tags. The second component is a naive Bayes classifier that use the nearby words and collocations.

The Winnow technique (32; 33) is a statistical classifier that employs numerous models for context modeling. The final ranking is determined by the weighted sum of the outputs from various Winnow classifiers that were trained with varying parameters. This approach employs the same characteristics as the preceding method, including word occurrence in context and collocation of tags with adjacent terms (34). Golding and Roth (32) and Carlson et al. (36) both utilised a large training corpus and a comparable strategy for correcting Chinese spelling.

In their article (37), Banko and Brill proposed a four-classifier voting method. This strategy focuses learning from enormous data sets, employing more than 100 million words. This system uses a basic memory-based learner, a perceptron, a Winnow classifier, and a naive Bayes classifier as classifiers. Each classifier is individually trained and assigned a complementarity score, as stated by Brill et al. (38), which shows the precision of the classifier.

The spelling correction problem can be constructed using the problem of determining the best transcription of a series of characters into another sequence. The techniques employed in machine translation are ideally suited for overcoming this type of issue. Generic string-to-string translation models can be used to a range of problems, including grapheme-to-phoneme conversion, transliteration, and lemmatization (39). These models are not restricted to rectifying spelling mistakes. The machine translation representation of the spelling correction problem can accommodate joined and split words, however effective error model learning requires a large corpus.

Zhou et al. (40) defined the machine-translation method of spelling correction using the following equation:

$$s' = argmax_s P(s|S)$$

where *s* is the potential correct sequence, *s′* is the optimal adjustment, and *S* is the provided wrong sequence. Characters are composed of both "proper" and "incorrect" language. The words in the training database are converted to lowercase sequences of characters, and blank spaces are converted to special characters. A parallel corpus of occurrences of misspelled words and corrections is used to train the machine translation system. The ASC system proposed by Koehn et al. (42) and Sariev et al. (41) employs Moses, a statistical machine translation system.

Neural networks are used in a variety of methods as well. The following are some of the more modern methods: A unique, unsupervised, and distance measure agnostic method for search space reduction in spell correction using neural character embedding was presented by Pande (19). This method was presented by Pande. The embedding is acquired using skip-gram Word2Vec training carried out on sequences that are produced from dictionary words in a phonetically informationretentive fashion. He made an effort to cut down on the amount of time required for correcting spelling errors by narrowing the word pool that needed to be combed through in order to find the appropriate spelling. During the process of training, he made use of the dictionary to ensure that he was spelling terms correctly. Once the search area has been narrowed down, any distance measure of spelling correction can be employed without affecting the results. This property is known as "distance measure agnosticism." His approach is implemented as a filter prior to an algorithm that does spell checking.

A Deep Learning model for the correction of text written in English was recently proposed by Ghosh and Kristensson (20). Using semi-character Recurrent Neural Networks as their method of attack, Sakaguchi et al. (21) attempted to solve the problem of spelling correction. Spelling correction was accomplished through the employment of a character-based sequence to sequence learning neural network technique, as stated by Tal Weiss (22). However, models based on deep learning can be used for a wider variety of applications. A substantial amount of training data is required in the event that it is to perform appreciably better.

The SymSpell algorithm is an innovative approach to spelling correction that is based on the concept of edit distance. The edit-distance measure estimates the lowest number of single-character changes (insertions, deletions, substitutions, or transpositions) required to transform one word into another. This approach combines a quick, memory-efficient, and scalable index structure with a word segmentation procedure to repair spelling problems in real time.

Beginning with a dictionary of correctly spelled words that has been predefined, the method maintains a Deletion Trie data structure. When a word is entered, the algorithm calculates all potential variations of the word using the set edit distance and assesses whether or not these variations are present in the dictionary. If a match is detected, the suggested correction is offered; otherwise, a list of alternatives ranked by probability score is returned. This score is based on the frequency of each recommendation within a large text corpus.

The Symmetric Delete method for correcting misspellings is based on the observation that the majority of misspellings consist of one or two consecutive character deletions, with the likelihood of such deletions increasing as the length of the word rises. The algorithm uses this insight to limit the number of potential correction candidates, hence improving the performance and efficiency of the correction procedure.

3 Research approach or methodology

3.1 Spelling Errors

Understanding where misspellings come from is necessary for the development of a system that can automatically fix them. There are a lot of books out there that are about typos. Mitton (44) wrote a book in which he researched several types of misspellings and presented different approaches to the creation of an automatic spelling correction system.

The authors of the paper by Yannakoudakis and Fawthrop (45) demonstrated that the vast majority of misspelled words are consistent with predetermined rules that are determined by phonological and sequential considerations. In the study (45) the authors established and detailed three categories of misspellings: consonantal, vowel, and sequential. They also published the results of an inquiry into 1377 different spelling error forms. In addition, the authors of Kukich (43), Toutanova and Moore (46), and Pirinen and Lindén (47) split spelling errors into two categories based on their origin: cognitive and typographic errors.

Cognitive errors (also known as orthographic or consistent errors) are produced by the impairments of the author. It's possible that the author is unaware of the proper writing style. It's possible that the author has trouble reading, writing, or dealing with other cognitive difficulties. It is possible that the author is still in the process of learning the language, in which case they may not be aware of correct spelling. This category of errors is user- and language-specific since their occurrence is highly dependent on the correct application of linguistic norms (47). An illustration of a cognitive error would be the statement "The Levenstain distance is among the most common approaches utilized for correcting misspelled words." The word "Levenshtein" has been spelled incorrectly in this instance, which is the problem.

Mistakes in conventional typography (often referred to simply as typography): They are often linked to the technical limits of the input device (physical or virtual keyboard, or OCR system), as well as the conditions of the surrounding environment. When you type quickly, you usually end yourself pressing the wrong combination of keys for the close bracket. Typographic errors that are caused by hurried typing are typically language-agnostic, which means that they are not tied to the author's original language, but they can be affected by local keyboard mapping or a localized OCR system (47). "Bu güN özümü yaxsi hiss etmirəm" for instance, is an example of a typographical error. The improper case of 'gün' and the misspelling of 'yaxşı' are the errors that can be found in this sentence.

There are a few distinct writing systems, including Arabic, Vietnamese, and Slovak, that make use of different character variants that change the meaning of the word. The writers of (49) confirmed that the omission of diacritics is a common type of spelling error in Brazilian Portuguese. Diacritical markings are frequently absent from writing that is done in Modern Standard Arabic (50). A typographical error occurs when the author fails to include necessary character marks and then expects the reader to derive the author's intended meaning from the text. In most cases, the omitted marks designate short vowels or letter changes. They are positioned either above or below the graphemes, depending on the context. Diacritization, often known as vowelization, is the process of adding vowels and other diacritic characters to a text written in Arabic (50). Azmi and Almajed (51) discussed the topic of Arabic diacritization, which

refers to the process of adding missing diacritical markings to Arabic letters, and gave an evaluation metric. On the other hand, Asahiah et al. (52) published a study of Arabic diacritization processes.

3.2 Deep Learning

Deep learning is a subfield of machine learning that models complex data relationships by employing neural networks that have multiple layers of connectivity. The high number of layers that are contained within these neural networks is what is meant by the term "deep." Deep learning algorithms, in contrast to traditional machine learning methods that rely on feature engineering, learn to extract useful features from raw data via multiple layers of non-linear transformations. This is in contrast to traditional machine learning methods, which are based on the concept of feature engineering.

It primarily revolves around the creation of artificial neural networks, the structure and operation of which are modeled after those of the human brain. A neuron is the fundamental building block of a neural network. Neurons are responsible for taking input from other neurons and producing output, which is then distributed to other neurons in the network. During the training phase of a neural network, weights are used to represent the connections that are made between neurons. These weights are then modified in order to improve the network's overall performance on a specific task.

Neurons, which are typically responsible for processing incoming data and producing outputs, make up each layer of a typical deep neural network. The results of one layer are used as inputs for the next layer, and so on, right up until the output of the entire process is generated. The number of layers and the number of neurons contained within each layer can change according to the degree of difficulty of the issue that needs to be resolved.

Adjusting the weights between neurons is required when training a deep neural network. This is done with the goal of minimizing a loss function that measures the difference between the predicted output and the actual output of the network. In most cases, one can accomplish this goal by utilizing an optimization algorithm such as stochastic gradient descent.

One of the most significant benefits of deep learning is that it can automatically learn useful features from raw data without the need for any kind of manual feature engineering. This is one of deep learning's primary advantages. Because of this, deep learning is particularly well suited for applications that require processing of high-dimensional and complex raw data, such as speech and image recognition.

In recent years, deep learning has been able to achieve state-of-the-art performance on a wide variety of tasks, such as image classification, natural language processing, and speech recognition. However, training deep learning models is typically computationally intensive and complex, requiring large amounts of data and computational resources. This is because deep learning models are designed to learn from examples rather than being taught from scratch. In order to overcome these challenges, the current research is concentrating on the creation of deeper learning architectures and training algorithms that are more efficient.

3.3 Long short-term memory

There are numerous neural network variants, each suited to a specific type of data input. Standard and convolutional neural networks perform well with static data, such as images that are analyzed in their entirety at once. Recurrent Neural Network (RNN) is utilized when data is dynamic and sequentially organized, such as video frames or stock market values. RNNs include LSTM (Long Short-Term Memory) as a subset.

As the name suggests, LSTM networks "remember" previous data states. This memory is selectively programmed to remember only specific portions of past data, even for extended periods of time. LSTM is highly relevant in applications where predictions depend on previous data values. Keras and TensorFlow implementations of LSTM are widely utilized in sequential prediction applications such as auto-response suggestions in email, stock value predictions, and speech/writing recognition.

In the Fig.2, darkness of the shades represents the sensitivity of nodes. As additional inputs overwrite the activation of a hidden unit and the network "forgets" the initial input, the sensitivity degrades exponentially over time.

In conventional feed-forward NN, all data values are given equal weight, regardless of whether they are one day old or several months old. This is effective for reading image data sets to distinguish between a cat and a dog. However, when dealing with time series data in which each value is time-stamped, the relevance of the data diminishes over time. RNN is capable of handling data dependencies, but only over brief time intervals. This is accomplished by feeding the output of the hidden layer h(t1) back into the input of the hidden layer via a conceptual delay block. As a result of the vanishing gradient problem, however, RNNs are ineffective with time-sensitive data.

LSTM networks are a type of RNN that contain a "memory cell" that retains information in memory for extended periods of time. A set of gates is used to control when information is input, output, and forgotten. This architecture allows them to gain knowledge of longer-term dependencies.

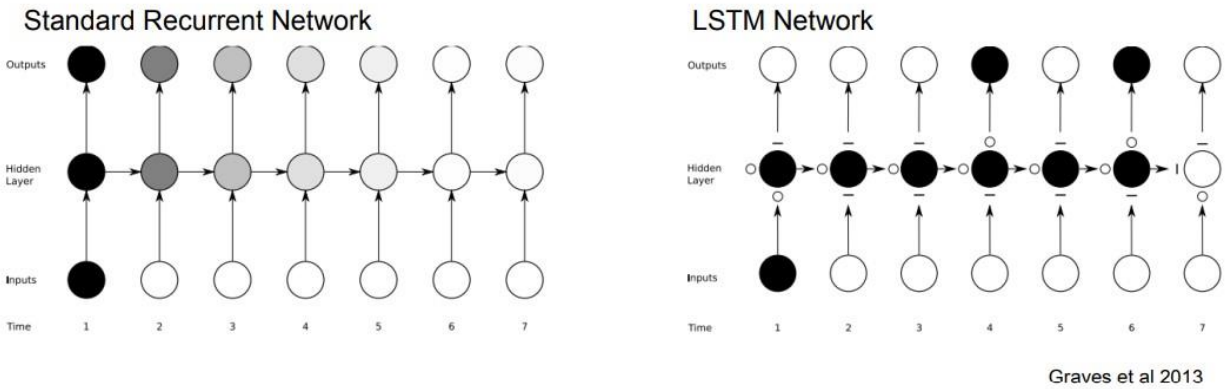

Figure 1: RNN vs LSTM - darkness of the shades represents the sensitivity of nodes. As additional inputs overwrite the activation of a hidden unit and the network "forgets" the initial input, the sensitivity degrades exponentially over time. (53)

To eliminate the vanishing (or exploding) gradient problem, the LSTM cell has a unique feature called Forget Gate. This helps reduce the multiplicative effect of small gradients significantly.

In the hidden layers of all neural networks, a series of repeating nodes is present. Standard RNN nodes may contain an input, an output, and a straightforward tanh function in the middle. The hidden layer nodes of LSTM contain three interacting functions or "gates". These gates safeguard and regulate the ’memory’ - the information stored in the cell state.

- Cell-State functions like a network-spanning conveyor belt. Node after node, the value fluctuatesbased on the information added/removed by the gates.
- Hidden-State: The node’s actual output for a given input. However, it is always concealedbecause it is not input until the subsequent time step.
- Input-Gate: Open exclusively at time step t. Here, the node determines which inputs will updatethe current cell state and which will be added as new candidates.
- Forget-Gate: Determines what data to discard from the cell state. It examines the input, previoushidden state, and returns a value between 0 and 1 for each cell state number. 1 = "Completely remember", 0 = "Completely forget".
- Output-Gate: Sends out a filtered version of the cell state as output

Suppose mathematically that there are h hidden units, n batch size, and d number of inputs. Thus, the input is $\mathbf{X}_t \in R^{n\times d}$ and the previous time step’s hidden state is $\mathbf{H}_{t-1} \in R^{n\times h}$. The gates at time step t are defined as follows: the input gate is denoted by $\mathbf{I}_t \in R^{n\times h}$, the forget gate by $\mathbf{F}_t \in R^{n\times h}$, and the output gate by $\mathbf{O}_t \in R^{n\times h}$. The calculation is as follows:

$$\mathbf{I}_t = \sigma(\mathbf{X}_t\mathbf{W}_{xi} + \mathbf{H}_{t-1}\mathbf{W}_{hi} + \mathbf{b}_i),$$

$$\mathbf{F}_t = \sigma(\mathbf{X}_t\mathbf{W}_{xf} + \mathbf{H}_{t-1}\mathbf{W}_{hf} + \mathbf{b}_f),$$

$$\mathbf{O}_t = \sigma(\mathbf{X}_t\mathbf{W}_{xo} + \mathbf{H}_{t-1}\mathbf{W}_{ho} + \mathbf{b}_o),$$

Where $\mathbf{W}_{xi},\mathbf{W}_{xf},\mathbf{W}_{xo} \in R^{d\times h}$ and $\mathbf{W}_{hi},\mathbf{W}_{hf},\mathbf{W}_{ho} \in R^{h\times h}$ are weight parameters and $\mathbf{b}_i,\mathbf{b}_f,\mathbf{b}_o \in R^{1\times h}$ are bias parameters.

In LSTM, the cell state is maintained as a rolling value until it leaves all hidden layers and reaches the output. The 3-gate structure controls and protects the cell value by permitting information to pass through voluntarily. They consist of a layer of sigmoid neural networks and a tanh operation vector for point-wise multiplication.

Data processing occurs in batches. For a new batch, the input layer receives inputs from the previous batch’s concealed state (output layer). To facilitate manipulation, even textual contents are encoded and stored as integers in the cell state.

Each iteration consists of the following stages (at time step t):

- The node receives the hidden state value from t-1 and the input value at t
- The Forget gate eliminates unneeded information from the cell’s state based on its inputs

- The input gate determines which old information to retain and which new information to add tothe cell state
- Cell state value is also saved as a hidden state at time t
- Output gate might either send a filtered version of cell state out of the hidden layers or backinto a recurrent loop

Networks using simple RNNs likewise have just a short-term memory capacity. Following a predetermined amount of time, the output of a hidden layer node may be reused as an input. In order for each node to be able to "remember" the value of the cell for one time step (short term). As the loop is iterative, in theory, the value ought to be stable even over lengthy spans of time given that it is repetitive. Yet, the effect of this lessens as mistake correction by back propagation makes its way to the early tiers of the network. After a certain number of cycles, this will cause the values of the buried layer nodes to deviate from the output that is desired, and the memory will be lost.

The integrity of the cell state is maintained by the LSTM's three gates. Because of this, the node with short-term memory is able to keep its cell state consistent throughout all of the hidden layers of the network (long time). The 'forget gate' is a vital element that stands between numerous time steps in the hidden layer and offers control and precision of the cell state at all times. This gate is referred to as the 'forget gate'.

3.4 Gated recurrent unit

Gated Recurrent Units (GRUs) (54) are a type of recurrent neural network (RNN) architecture that, like LSTMs, are designed to model temporal sequences with long-term dependencies. GRUs were introduced to address some of the limitations of LSTMs, including their computational complexity and difficulty of training.

GRUs are composed of a single gating mechanism that controls the flow of information into and out of the hidden state. This gating mechanism is simpler than the three gates used in LSTMs and consists of a reset gate and an update gate.

The reset gate determines how much of the previous hidden state should be forgotten, while the update gate determines how much of the new input should be retained. The gates are implemented using sigmoid and tanh activation functions, similar to LSTMs, to control the flow of information.

During the current time step *t*, the reset gate defines how much of the prior concealed state should be ignored. It accepts the current input x_t and the previous hidden state h_{t-1} as inputs and returns a number between 0 and 1 that reflects the percentage of the previous hidden state to be reset. It is calculate with the following equation:

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \quad (1)$$

The update gate determines how much of the new input should be retained at the current time step t. It takes as input the current input x_t and the previous hidden state h_{t-1}, and produces a value between

0 and 1 that represents the proportion of the new input to be retained. The update gate is calculate with the following equation:

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{2}$$

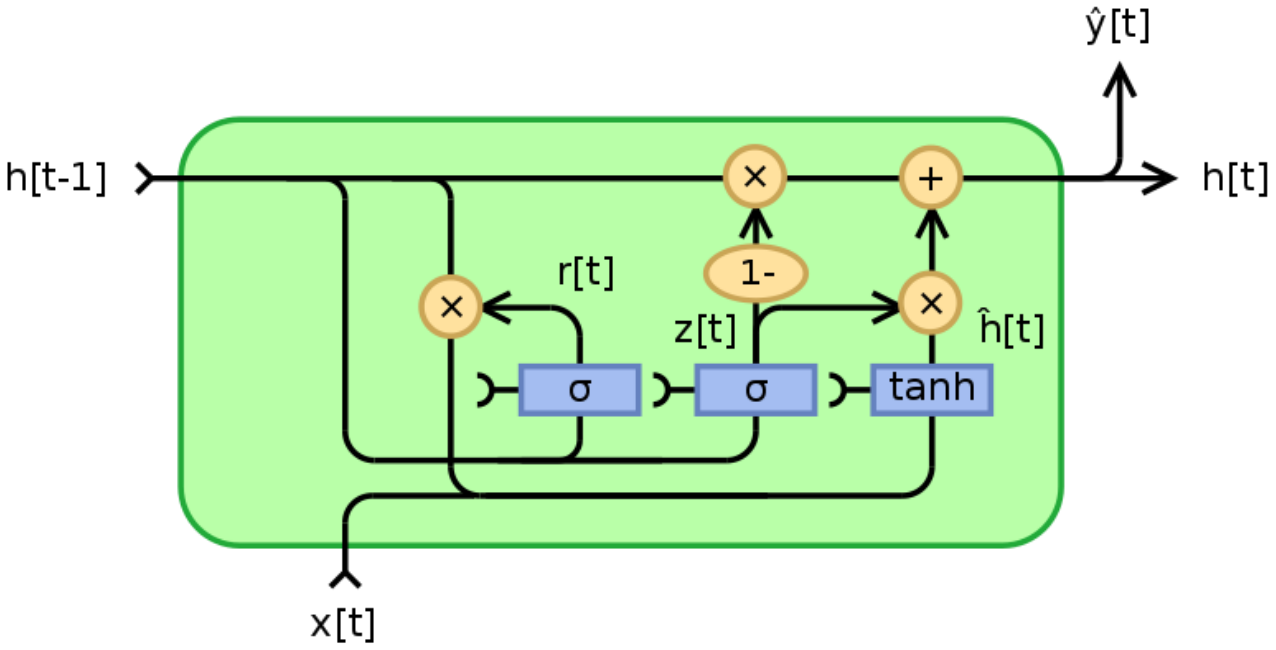

Figure 2: Gated Recurrent Unit [54]b

The candidate activation is used to update the hidden state at the current time step t. It takes as input the current input x_t and the reset gate r_t applied to the previous hidden state h_{t-1}, and produces a new hidden state candidateh'_t:

$$h'_t = \tanh(W_h x_t + U_h(r_t * ht - 1)) \tag{3}$$

The hidden state is the output of the GRU at each time step t. It is computed based on the update gate z_t, the previous hidden state h_{t-1}, and the candidate activationh'_t:

$$h_t = (1 - z_t) * h_{t-1} + z_t * h_t \tag{4}$$

GRUs are computationally inexpensive and easier to train than LSTMs because of the simplicity of their design, yet they can still capture long-term relationships in sequential data. GRUs have been demonstrated to be effective in a range of sequence modeling applications, including language modeling, speech recognition, and machine translation.

3.5 Attention mechanism

The encoder-decoder model for machine translation benefited from the addition of an attention mechanism so that it could operate more effectively. By combining all encoded input vectors with

weights, with the most relevant vectors having the highest weights, the attention mechanism's goal was to give the decoder the ability to utilize the most important parts of the input sequence in a flexible manner. This was accomplished by combining the weights, with the most important vectors having the highest weights.

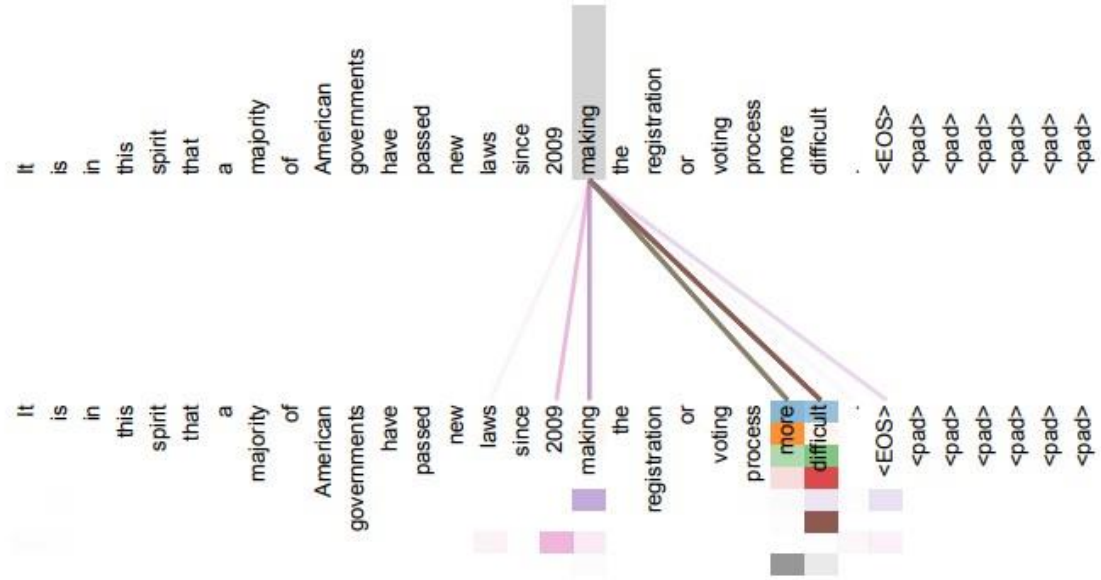

Figure 3: An illustration of the attention mechanism that follows long-distance dependencies in the encoder self-attention in layer 5 of 6. Several of the attention heads attend to a remote dependence of the verb 'making', so completing the phrase 'making...more difficult'. Emphasis is focused solely on the word 'making' in this instance. Various colors depict various types of heads.(58)

Bahdanau et al. (55) established the attention technique to overcome the bottleneck issue that emerges with the usage of a fixed-length encoding vector, where the decoder would have limited access to the input's information. This is believed to be especially problematic for lengthy and/or complex sequences, whose representations would be required to have the same dimensions as shorter or simpler sequences.

Notice that Bahdanau et al.'s attention mechanism is composed of the sequential computations of alignment scores, weights, and context vector:

1. Alignment scores: Using the encoded hidden states, $\mathbf{h}_i$, and the previous decoder output, $\mathbf{s}_{t-1}$, the alignment model calculates a score, $e_{t,i}$, that shows how well the components of the input sequence fit with the current output at the location, **t**. A function, $a(.)$, represents the alignment model and can be implemented by a feedforward neural network:

$$e_{t,i} = a(\mathbf{s}_{t-1}, \mathbf{h}_i) \tag{5}$$

2. Weights: Applying a softmax operation to the previously computed alignment scores yields theweights, $\alpha_{t,i}$:

$$\alpha_{t,i} = \text{softmax}(e_{t,i}) \quad (6)$$

3. Context vector: At each time step, a unique context vector, $\mathbf{c}_t$, is provided to the decoder. It is derived from the weighted sum of all encoder hidden states, **T**:

$$\mathbf{c}_t = \sum_{i=1}^{T} \alpha_{t,i}\mathbf{h}_i \quad (7)$$

For both the encoder and decoder, Bahdanau et al. implemented an RNN. However, the attention mechanism can be reformulated into a universal form applicable to any sequence-to-sequence (seq2seq) work in which the information is not necessarily related sequentially.

The Luong attention (56) which is also referred to as Multiplicative attention tried to build upon the Bahdanau model for neural machine translation by incorporating two additional kinds of attention mechanisms: a global approach that attends to all source words and a local approach that only pays to a subset of words in predicting the target phrase.

The similarities between global and local focus:

- Both the global attention and the local attention technique employ the hidden state $\mathbf{h_t}$ at the top layer of a stacking LSTM as their starting input during the decoding phase. This happens at each time step *t* in the process.
- The objective of both approaches is to derive a context vector $\mathbf{c_t}$ that captures source-side information pertinent to predicting the current target word $\mathbf{y_t}$.
- Attentional vectors are provided to subsequent time steps as inputs to inform the model aboutprior alignment decisions.

The main difference between global and local attention models is in how they derive the context vector $\mathbf{c_t}$.

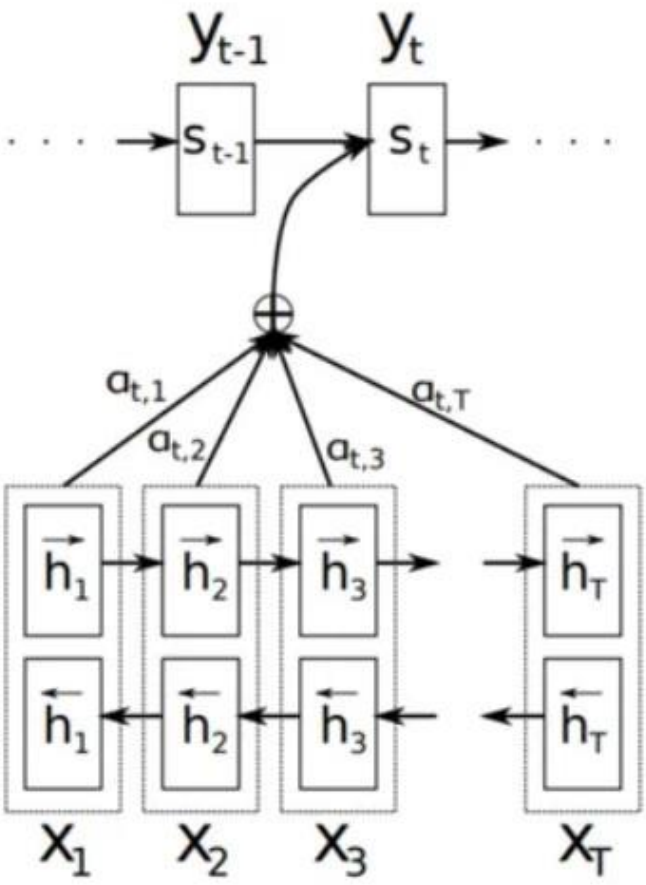

Figure 4: Bahdanau Attention Mechanism [55]

The components depicted in Figure 4 are the following:

- s_{t-1} is the hidden decoder state at the previous time step t-1.
- c_t is the context vector at time step, t which is generated at each decoder step to find target word, y_t.
- h_i is an annotation that encapsulates the information contained in the words comprising the entire input sentence, $\{x_1, x_2, \dots, x_T\}$, with particular emphasis on the i-th word out of T total words.
- a_i represents the weight assigned to each annotation, h_i, at the current time step, t.
- $e_{t,i}$ is an attention score derived by an alignment model, $a(.)$, which measures the degree to which s_{t-1} and h_i match.

3.6 Encoder–Decoder Neural Network Model

The encoder–decoder architecture, also known as Sequence to sequence (Seq2seq) (60; 61), includes two consecutive Recurrent Neural Networks. (RNNs). The first RNN takes variable-length symbol

sequences and converts them into a fixed-length representation. The second RNN, on the other hand, takes the fixed-length representation and turns it back into variable-length symbol sequences.

The fact that a neural network needs to be able to compress all of the information necessary to decode an input sentence is one of the potential problems with the typical Seq2seq technique. Because of this, it may be difficult for the model to process very long sentences, particularly ones that are longer than the sentences that were used for training (55). According to (55), the Attention mechanism enables the model to learn how to generate a context vector that best portrays the input sentence at each stage of decoding; this means that the model learns to concentrate and pay more attention to the pertinent areas of the input sequence.

The approaches Bahdanau's Attention (additive style) (55) and Luong's Attention (multiplicative style) (56) are considered to be the most important ones when it comes to Seq2seq models. Both methods make use of the same application, differing only in the degree to which they customize the

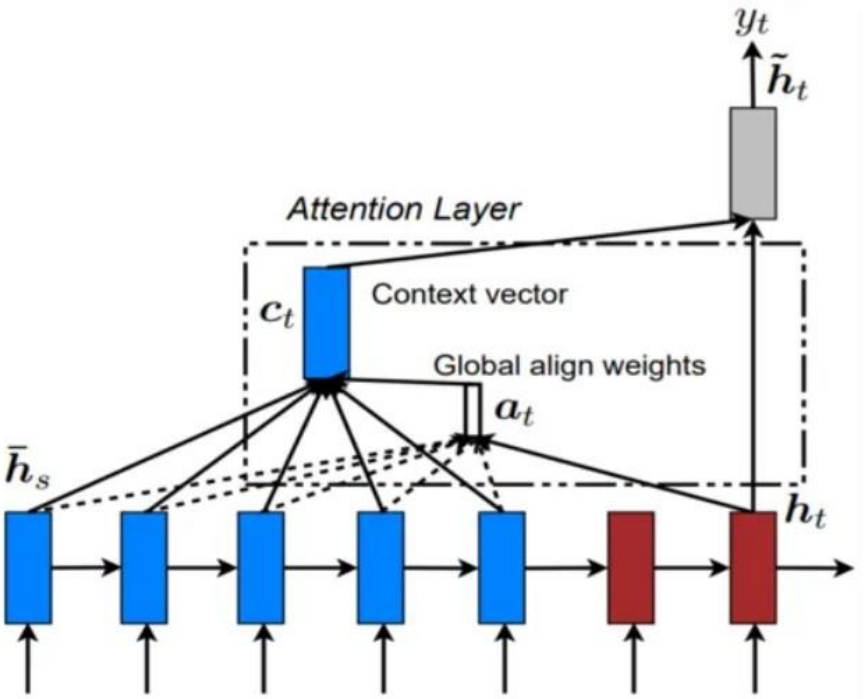

Figure 5: Luong Attention Mechanism [56]

following steps in the scoring process:

$$score(h_t, h_s) = \begin{cases} h_t^T W_a h_s, & Multiplicative \\ v_a^T \tanh(W_a h_t + U_a h_s), & Additive \end{cases}$$

where h_t is the output of the encoder, h_s is the hidden state, and v_a, W_a, and U_a are the matrices corresponding to the weights. Following phases are formed once the encoder has finished generating the hidden states of each element in the input sequence: (i) the alignment scores, also known as attention weights, are computed by comparing the hidden state of the decoder that came before it to the hidden state of each encoder, with T_x standing for the total length of the source:

$$\alpha_{ts} = \frac{\exp(score(h_t, h_s))}{\sum_{k=1}^{T_x} \exp(score(h_t, h_k))} \tag{9}$$

(ii) The context vector is computed by multiplying the alignment scores by the encoder's respective hidden states:

$$c_t = \sum_s \alpha_{ts} h_s \tag{10}$$

(iii) the context vector is concatenated with the previous decoder output (attention vector), sofeeding the decoder at the current time step to generate a new output:

$$a_t = f(c_t, h_t) = \tanh(W_c[c_t; h_t]) \tag{11}$$

All of these processes are performed again at each decoder time step until the decoding process is finished ("<EOS>" special character or maximum length). Recently, a mechanism known as Multi-Head Attention was described in a model architecture called the Transformer model (58) that is made up entirely of attention layers. This mechanism makes use of Scaled Dot-Product Attention in order to compute a representation and correlate distinct places inside a single phrase. The Attention mechanism is able to convert query vectors into a collection of key-value pairs in the output thanks to this method.

The result is calculated as a weighted total of the values, where the weight assigned to each value is decided by a query compatibility function in combination with the appropriate key. The output is computed as a weighted sum of the values. After that, the data that are being taken in are comprised of queries, dimension keys d_k, and dimension values d_v. When calculating the weights of the values, a softmax function is applied to the dot product of the query with all of the keys, and each key is divided by square root of d_k before the calculation is performed. A definition of the output matrix may be found down below:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{12}$$

where the attention function concurrently computes a set of questions and adds them into a Q matrix. In addition, the values and keys are saved in separate matrices K and V, respectively. Then, using an attention head, Multi-Head Attention makes it possible to pay concurrently to information coming from several representation subspaces at various locations, where W_i, W_i^{KQ}, and W_i^V are weight matrices:

$$head_i = Attention(QW_i^Q, KW_i^K, V\,W_i^V) \tag{13}$$

$$MultiHead(Q,K,V) = Concat(head_1,...,head_h)W^O \tag{14}$$

In contrast to the traditional encoder–decoder strategy (58; 60; 61) utilized in the study of linguistics, this method enables the weights to be calculated in parallel while the model is being trained.

Each time an iteration over time steps is performed, a new context vector is generated. This is because the decoding process is completed by projecting subspaces onto the main space (58). Because of this, it moves at a slower pace throughout the inference phase.

3.7 Approach

3.7.1 Model architecture

The encoder–decoder model architecture, specifically Sequence to Sequence (Seq2Seq) approach was used to create the model. The proposed Seq2seq architecture uses GRU layers, with one bidirectional Encoder layer, and the Decoder is another GRU with twice as many hidden units as the Encoder. (to match the number of units in the Encoder). The output of the encoder and decoder are subjected to an application of the Attention technique. In this method, a mixture of the Bahdanau's model, Luong's model and transformer model were used. The mixture includes encoder step with Bidirectional layer, usage of Gated Recurrent layers instead of LSTM layer and after that, using Luong Attention with Layer Normalization, respectively. In addition to this, experiments were conducted with the units value at 256 and 512, and the dropout rate for each GRU is set to 0.2 (which is usual for small models), with batch size of 128 sentences and a normalizing layer is added to the output of both the Attention mechanism and the Decoder. ReduceLROnPlateau and Early Stopping were also applied if there are no improvements at the value of validation loss for 15 and 20 epochs, respectively. Model has total of 5,652,592 trainable parameters. Figure 6 illustrates the summary of the model architecture.

Layer (type)	Output Shape	Param #	Connected to
encoder_inputs (InputLayer)	[(None, None, 112)]	0	[]
encoder_bgru (Bidirectional)	[(None, None, 1024), (None, 512), (None, 512)]	1923072	['encoder_inputs[0][0]']
decoder_inputs (InputLayer)	[(None, None, 112)]	0	[]
concatenate (Concatenate)	(None, 1024)	0	['encoder_bgru[0][1]', 'encoder_bgru[0][2]']
decoder_gru (GRU)	multiple	3495936	['decoder_inputs[0][0]', 'concatenate[0][0]']
attention_layer (Attention)	multiple	0	['decoder_gru[0][0]', 'encoder_bgru[0][0]']
concatenate_1 (Concatenate)	(None, None, 2048)	0	['decoder_gru[0][0]', 'attention_layer[0][0]']
normalization (LayerNormalization)	multiple	4096	['concatenate_1[0][0]']
time_distributed_layer (TimeDistributed)	multiple	229488	['normalization[0][0]']

Figure 6: Model summary

The Adam optimizer with tensorflow's CustomSchedule learning was used in order to reduce the overall loss and improve accuracy. As for the loss, CategoricalCrossentropy with label smoothing of 0.1 was used.

3.7.2 Dataset

The data consists of approximately 4 million sentences scraped from Azerbaijan State News Agency (AZERTAC).

Since the available data consists of sentences without mistakes and could only be used as ground truth, training data was synthetically generated by introducing errors to the data randomly. This process is done only once for the validation data and generated in each iteration for training data. It is a simple function that generates a noisy sentence before encoding.

For each character of every word, one of three different operations are done. These operations consist of insertion, deletion and replacement. The probability of applying one of these modifications is determined by length of the word and maximum text length that was specified for the model. Therefore, the longer the word, the higher the chances of adding errors. Priority among error types is given to the replacement.

In their study (59), social media comments in azerbaijani were scraped and used to examine people's typographical errors. With these analyses, they determined letters and their most misspelled and confused letters (table 1). Since the substitution operation is more rational than random letter selection, it is preferred and employed 80% of the time. By producing synthetic words in this manner, it becomes very near to actual human spelling errors.

Letter	Misspellings
ə	e, i, a, ı
s̩	w, s, sh, i
ç	c, ch
q	g, x, k, ğ
ı	i, l, u, ə, o, s, a
r	t, l, z, x
g̑	g, q
l	d, n, o
d	n, t, x
u	h, i, ı, o
ö	o, i, ü, e
ü	u, i, e
t	d, r
y	g, t, j
s	z, c, d, ss
z	x, s
g	q, y, k, ğ

n	m, v, l
i	u, y, ı
a	i, u, s, e, ə, m, ı, v
k	y, g, q

Table 1: Letters and what they are most mispelled with (59)

In table 2, two pairs of incorrect and correct examples are shown. Here, each synthetic word is generated from the sentence's correct words. In this manner, around 4 million sentence pairs were created.

Original Sentence	Generated Sentence
Saray kompleksində daimi rəsm sərgiləri fəaliyyət göstərir, tarixi sənədlər, milli geyim nümunələri nümayis̨ olunur.	Saruy krompleksində dəimi resm sergileri fəaliyyet ğösterir, tarixı sənedlər , hidli geyim nümunəleri nümayis̨ olunur.
Lakin bu gün bizə bunu əməli olaraq sənət ilə göstərdiniz.	Lakin bu gun bize bunu emeli olaraq senet ile gostərdiniz.

Table 2: Examples for original and generated sentences

In addition to above mentioned methods, some adjustments (such as increasing probabilities of letter 'ə' being replaced) were made depending on the outcome of the experiments as well.

3.7.3 Metrics

For evaluating the performance of the model, either character-level tokens and HTR metrics such as Character Error Rate and Word Error Rate or word-level tokens and GER metrics such as Precision and BLEU could be chosen depending on the approach. In contrast to the Machine Translation and GER measurements, which try to assess the correction more thoroughly and semantically, this statistic evaluates the correction superficially and syntactically. Therefore, Character Error Rate (CER), Word Error Rate (WER), and Sequence Error Rate (SER) were chosen.

CER is calculated using the Levenshtein distance, where we figure out the fewest number of character-level operations needed to change the ground truth text (also called the reference text) into the output. This is expressed by the following formula:

$$CER = \frac{S + D + I}{N} \tag{15}$$

where:

- S is number of substitutions

- D is number of deletions
- I is number of insertions
- N is number of characters in reference text (ground truth)

The result of this equation reflects the proportion of ground truth characters that were wrongly predicted by the model. The better the model's performance, the lower the CER value (with a score of 0 indicating perfection).

For example, if the ground truth is "köçkünlər" and the model has predicted "köçküvət", then several errors require edits in order to transform model's output into the ground truth:

1. v instead of n (at reference text character 6)
2. Missing l (at reference text character 7)
3. t instead of r (at reference text character 9)

Following are the values to plug into the equation:

- Number of Substitutions (S) = 2
- Number of Deletions (D) = 1
- Number of Insertions (I) = 0
- Number of characters in reference text (N) = 9

With the above, we have (2 + 1 + 0) / 9 = 0.3333. When translated to a percentage, the CER equals 33.33 percent. This indicates that every third character in the sequence was written wrongly. We repeat this computation for each pair of transcribed output and ground truth and then use the mean of these values to determine the CER percentage as a whole.

CER values can surpass 100%, especially when there are several insertions. For instance, the CER for the ground truth 'ABC' and the lengthier OCR output 'ABC12345' is 166.67 percent. To prevent that issue, the number of errors is split by the total of edit operations ($i + s + d$) and the number c of correct symbols, which is always more than the numerator. The normalization method presented above ensures that CER values always lie within the range 0–100% and it can be expressed using the following formula:

$$CER_{normalized} = \frac{S + D + I}{S + D + I + C} \tag{16}$$

where C is the number of correct characters.

There is no one standard for determining a suitable CER value, as it depends greatly on the application. Various contexts and levels of complexity might result in varied model performances.

Word Error Rate is more appropriate if the text at hand includes the transcription of meaningful paragraphs and phrases (e.g., pages of books, newspapers) which provides context.

$$WER = \frac{S_w + D_w + I_w}{N_w} \tag{17}$$

WER is calculated using the same technique as CER, however it operates at the word level rather than the character level. It specifies the number of word changes that must be made, such as deletions, insertions, or substitutions, in order to transform one phrase into another.

Although WER and CER are mostly correlated (as long as mistake rates are not excessively high), it is reasonable to assume that the absolute value of WER will be higher than the value of CER.

4 Research Results and Analysis of Results

Experiments were carried out on the dataset, the model architecture, and the development of artificial noise. In each experiment, the dataset was divided into 90% train, 5% test, and 5% validation. Train set is utilized to teach our model how to correct spelling errors. Using the validation set, we modify the model's hyperparameters to improve its learning by observing how well it identifies the validation set. After training is complete, accuracy is determined using the test set and by observing how our trained model performs on unobserved data pertaining to correcting mistakes.

The first dataset that was utilized had neither been processed nor cleaned in any way, and it contained the following information:

- Insignificant sequences of letters like "x x x"
- Proper nouns that were repeated in approximately every fifth sentence
- Punctuation marks
- Roman numerals
- Numbers

In spite of the fact that the data were unclean, the analysis nonetheless produced satisfactory outcomes due to the sheer volume of the data. Examples of results for the model trained on this particular dataset can be seen in Fig. 3.

Input	Prediction	Ground truth
Ozellesdirme davam edecek, bazar iqtisadiyati davam edecekdir	Ozelleşdirmə davam edəcək , bazar iqtisadiyyatı davam edəcəkdir	Özəlləşdirmə davam edəcək, bazar iqtisadiyyatı davam edəcəkdir
Bu gun hava cox gozeldir	Bu gün hava çox gozeldir	Bu gün hava çox gözəldir

Amerikada kecirdiyim muddet cox qisa olsa bele, burda cox sey oyrendim	Amerikada keçirdiyim müddet çox qisa olsa belə , burda çox sey oyrendim	Amerikada keçirdiyim müddət çox qısa olsa belə, burda çox şey öyrəndim
Gələnm idə yeni agaclar əkilecey	Gələn ildə yeni ağaclar əkiləcək	Gələn ildə yeni ağaclar əkiləcək

Table 3: Example predictions for model that was trained on unclean data

As can be seen from the input sentences, errors contain all of the 4 transformations (insertion, deletion, replacement, transposition). The second dataset that was used for experiments did not contain any of the above tokens mentioned above and the model trained on clean dataset performed 9% better than previous model.

After analyzing the results of the model, it was decided that utilizing confusion matrix for artificial noise generation might yield better results, since it would make misspellings closer to what humans would make. However, giving probability to each letter's most confused counterparts resulted in a worse outcome. There were a lot of problems when identifying mistakes, especially in some characters, one of which was the letter "ə".

Another thing that could be noticed from the analysis was that some of the words with length 5 or less had too many changes that they were unrecognizable. On the other hand, words that had more than 8 or more characters had only minuscule changes. The reason for this was that whether there will be changes in the word or not was done in a character level. This meant that length of the words had no affect whatsoever in the number of operations performed on the word. This was fixed by assigning a probability of performing operation to a character that depends on the length of the word:

$$prob = l * (ratio/ml)/4 \tag{18}$$

Where l is word's length, ml is maximum text length and the $ratio$ is for regularizing and it's default value is 0,6.

To fix the other issue where model struggled to correct certain characters, those characters were given higher chances of getting changed during artificial noise generation. This led model to have enough examples of such errors to be able to learn to deal with them. Another change regarding noise generation was in how characters were replaced. Instead of assigning probability to each replacement candidate for every character, Table 1 was used for substituting characters in a meaningful way. This change helped reducing word error rate down to 33% and character error rate to 7%. From the experiments done on the artificial noise generation, it could be concluded that better, smaller error rates can be achieved without changing the model and only working on the dataset.

Since the model has been trained on news data, it contains data that is mostly from political domain and during testing it could be seen that the model was struggling in correcting words from different domains such as financial domain. Model's performance was tested against model from Mirza (https://www.mirza.az/), which uses edit distance and extensive vocabulary to detect misspelled words and suggests multiple words sorted by their probability of correctness as true values. Mirza does not provide functionality to return predictions with highest probability at once but it requires

users to manually select correct version from suggested words for each detected misspelled word. For the experiment, only suggested words with highest probabilities were chosen.

Model	CER	WER
Before correction	20.13%	80.38%
Proposed model	7.38%	33.45%
Mirza	7.29%	30.81%

Table 4: Comparison with Mirza model in financial domain

To evaluate the performance of proposed spelling correction approach for the Azerbaijani language, experiments were conducted on a dataset consisting of 100 pairs of correctly spelled and actual misspelled sentences in financial domain. These sentences were not generated artificially, ensuring that the dataset contains misspellings from the actual people. We compared the performance of our model to that of Mirza, which serves as a baseline in this case.

The proposed model achieved a CER of 7.38% and a WER of 33.45% on the financial domain dataset. In comparison, the Mirza baseline model performed slightly better with a CER of 7.29% and a WER of 30.81%. Before applying any corrections, the CER was 20% and the WER was 80%.

The results indicate that both proposed model and the Mirza baseline model are effective in correcting misspelled sentences in the financial domain, with our model slightly underperforming in terms of CER and WER compared to Mirza. However, both models show significant improvements in terms of error correction, as the CER was reduced from 20% to around 7% and the WER was reduced from 80% to around 30%.

The relatively low CER and WER values achieved by both models demonstrate the potential of deep neural networks for spelling correction in Azerbaijani language, even in domain-specific text such as financial domain that the model was not trained on. The results suggest that proposed approach has the potential to be further improved by exploring other model architectures, incorporating contextual information, or leveraging external resources, as discussed in the future work section. Overall, the experiments demonstrate the effectiveness of our approach in correcting misspelled sentences, albeit with slightly lower performance compared to the Mirza baseline model.

Model	CER	WER
Before correction	20.37%	80.98%
Proposed model	5.30%	25.77%
Mirza	5.58%	27.42%

Table 5: Comparison with Mirza model in news domain

Furthermore, the performance of proposed spelling correction approach for Azerbaijani language was evaluated using a dataset of sentences from the news domain. The dataset consisted of sentences that were collected from various news articles written in Azerbaijani language. Real-world misspelled sentences in the news domain were represented in the dataset, reflecting the challenges of correcting spelling errors in text from a different domain compared to the financial domain.

A CER of 5.30% and a WER of 25.77% were achieved by our proposed model on the news domain dataset. In comparison, a CER of 5.58% and a WER of 27.42% were achieved by the Mirza baseline model. Before applying any corrections, the sentences in the dataset had a CER of 20.37% and a WER of 80.98%.

The results indicated that both the proposed model and the Mirza baseline model were effective in correcting misspelled sentences in the news domain. However, slightly better performance in terms of CER and WER was shown by our proposed model compared to the Mirza baseline model, with lower error rates achieved.

5 Summary and Future Work

In this paper, a novel method for Azerbaijani language spelling correction based on a sequence-tosequence (seq2seq) deep neural network and artificial noise generation techniques are proposed. Our experimental results demonstrate the efficacy of our approach, which outperforms baseline methods and achieves significant improvements in the accuracy of Azerbaijani text spelling correction. The literature review provided a comprehensive overview of existing methods and highlighted the unique challenges of Azerbaijani language spelling correction, while our proposed approach addresses these challenges and demonstrates promising results.

Based on the findings and limitations of our thesis, there are several possible future research directions. First, additional research could be conducted to determine the effect of various seq2seq model architectures, such as attention mechanisms or transformer-based models, on the performance of Azerbaijani spelling correction.

In addition, other types of artificial noise generation techniques, such as phonetic-based or morphological based noise, could be explored to improve the model's ability to manage a variety of misspellings. Using external resources, such as monolingual or bilingual dictionaries, could potentially improve the accuracy of the model by providing additional reference data. Moreover, other than noise generated artificially, it would be preferable to train the model with actual misspelled words from various domains.

Future research may also investigate the efficacy of our approach on specific domains, such as social media text or domain-specific jargon, where spelling errors may have unique characteristics requiring tailored approaches. In addition, undertaking user studies or incorporating human evaluation to assess the quality and readability of the corrected text could shed light on the practical applicability of our approach.

In order to develop a comprehensive language correction system for the Azerbaijani language, it may be possible to expand the scope of the thesis to include tasks such as grammar correction and punctuation correction. Additionally, extending the method to other languages with orthographic complexities comparable to English could be a fruitful avenue for future research.

Future work could enhance the accuracy and applicability of the proposed approach by exploring different model architectures, exploring other artificial noise generation techniques, using real misspelled words by humans, investigating performance in specific domains, conducting user studies, and expanding to related tasks and other applications.

6 References

[1] Fred J. Damerau. 1964. A technique for computer detection and correction of spelling errors. Commun. ACM 7, 3 (March 1964), 171–176. https://doi.org/10.1145/363958.363994

[2] Pirinen, T.A.; Lindén, K. State-of-the-art in weighted finite-state spell-checking. In Computational Linguistics and Intelligent Text Processing, Proceedings of the CICLing 2014, Kathmandu, Nepal, 6–12 April 2014; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8404, Part 2; pp. 519–532

[3] Mashod Rana, M.; Tipu Sultan, M.; Mridha, M.F.; Eyaseen Arafat Khan, M.; Masud Ahmed, M.; Abdul Hamid, M. Detection and Correction of Real-Word Errors in Bangla Language. In Proceedings of the 2018 International Conference on Bangla Speech and Language Processing, ICBSLP 2018, Sylhet, Bangladesh, 21–22 September 2018; pp. 1–4

[4] Boytsov, L. Indexing methods for approximate dictionary searching. J. Exp. Algorithmics 2011, 16, 11–91

[5] Deorowicz, S.; Ciura, M.G. Correcting spelling errors by modelling their causes. Int. J. Appl. Math. Comput. Sci. 2005, 15, 275–285

[6] Wang, Y.R.; Liao, Y.F. Word vector/conditional random field-based Chinese spelling error detection for SIGHAN-2015 evaluation. In Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing, Beijing, China, 30–31 July 2015; pp. 46–49.

[7] Kevin Atkinson. Lawrence Philips' Metaphone Algorithm. url: http://aspell.net/.

[8] Viggo Kann et al. "Implementation aspects and applications of a spelling correction algorithm". In: Text as a Linguistic Paradigm: Levels, Constituents, Constructs. Festschrift in honour of Ludek Hrebicek 60 (2001), pp. 108–123.

[9] Viny Christanti Mawardi, Rudy Rudy, and Dali S. Naga. "Fast and Accurate Spelling Correction Using Trie and Bigram". In: TELKOMNIKA (Telecommunication Computing Electronics and Control) 16 (Apr. 2018), p. 827.

[10] Casey Whitelaw et al. "Using the Web for Language Independent Spellchecking and Autocorrection". In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2

[11] Peter Norvig. How to Write a Spelling Corrector. url: http://norvig.com/spell-correct.html.

[12] Wolf Garbe. 1000x faster Spelling Correction. url:https://towardsdatascience.com/symspellcompound10ec8f467c9b.

[13] Dan Jurafsky and James H. Martin. Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd Edition. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International, 2009.

[14] Eric Brill and Robert C. Moore. “An Improved Error Model for Noisy Channel Spelling Correction”. In: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics. ACL ’00. Hong Kong: Association for Computational Linguistics, 2000, pp. 286–293.

[15] Farag Ahmed, Ernesto William De Luca, and Andreas NÃ. “Revised NGram based Automatic Spelling Correction Tool to Improve Retrieval Effectiveness”. en. In: Polibits(Dec. 2009), pp. 39–48. issn: 1870-9044.

[16] Youssef Bassil and Mohammad Alwani. “Context-sensitive Spelling Correction Using Google Web 1T 5-Gram Information”. In: CoRR abs/1204.5852 (2012).

[17] Zar Wint, Theo Ducros, and Masayoshi Aritsugi. “Non-words Spell Corrector of Social Media Data in Message Filtering Systems”. In: Journal of Digital Information Management 16 (Apr. 2018).

[18] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. “Glove: Global vectors for word representation”. In: In EMNLP. 2014.

[19] Harshit Pande. “Effective search space reduction for spell correction using character neural embeddings”. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2

[20] Shaona Ghosh and Per Ola Kristensson. “Neural Networks for Text Correction and Completion in Keyboard Decoding”.

[21] Keisuke Sakaguchi et al. “Robsut Wrod Reocginiton via semi-Character Recurrent Neural Network”.

[22] Tal Weiss. Deep Spelling - Rethinking spelling correction in the 21st century.

[23] Zhang, H.; Zhang, Q. EmbedJoin: Efficient edit similarity joins via embeddings. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and lData Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 585–594

[24] Kernighan, M.D.; Church, K.W.; Gale, W.A. A spelling correction program based on a noisy channel model. In Proceedings of the 13th Conference on Computational Linguistics, Helsinki, Finland, 20–25 August 1990; pp. 205–210.

[25] Jurafsky, D.; Martin, J.H. Speech and Language Processing; Prentice Hall: Upper Saddle River, NJ, USA, 2014

[26] Kinaci, A.C. Spelling Correction Using Recurrent Neural Networks and Character Level Ngram. In Proceedings of the 2018 International Conference on Artificial Intelligence and Data Processing, IDAP 2018, Malatya, Turkey, 28–30 September 2018.

[27] Reffle, U. Efficiently generating correction suggestions for garbled tokens of historical language. Nat. Lang. Eng. 2011, 17, 265–282

[28] Yu, M.; Li, G.; Deng, D.; Feng, J. String similarity search and join: A survey. Front. Comput. Sci. 2016, 10, 399–417.

[29] Vilares, M.; Otero, J.; Barcala, F.M.; Domínguez, E.; Dominguez, E. Automatic spelling correction in Galician. In Advances in Natural Language Processing; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3230, pp. 45–57.

[30] Melero, M.; Costa-Jussà, M.; Lambert, P.; Quixal, M. Selection of correction candidates for the normalization of Spanish user-generated content. Nat. Lang. Eng. 2016, 22, 135–161.

[31] Golding, A.R.; Schabes, Y. Combining Trigram-based and feature-based methods for contextsensitive spelling correction. In Proceedings of the 34th annual meeting on Association for Computational Linguistics, Santa Cruz, CA, USA, 23–28 June 1996; pp. 71–78

[32] Golding, A.R.; Roth, D. A Winnow-Based Approach to Context-Sensitive Spelling Correction. Mach. Learn. 1999, 34, 107–130.

[33] Littlestone, N. Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm. Mach. Learn. 1988, 2, 285–318.

[34] Golding, A.R.; Schabes, Y. Combining Trigram-based and feature-based methods for contextsensitive spelling correction. In Proceedings of the 34th annual meeting on Association for Computational Linguistics, Santa Cruz, CA, USA, 23–28 June 1996; pp. 71–78

[35] Li, J.; Wang, X. Combining trigram and automatic weight distribution in Chinese spelling error correction. J. Comput. Sci. Technol. 2002, 17, 915–923

[36] Carlson, A.J.; Rosen, J.; Roth, D. Scaling Up Context-Sensitive Text Correction. In Proceedings of the Thirteenth Conference on Innovative Applications of Artificial Intelligence Conference, Seattle, WA, USA, 7–9 August 2001; AAAI Press: Palo Alto, CA, USA, 2001; Volume 51, pp. 45–50

[37] Banko, M.; Brill, E. Scaling to Very Very Large Corpora for Natural Language Disambiguation. In Proceedings of the 39th Annual Meeting on Association for Computational Linguistics, Toulouse, France, 5–10 July 2011; Association for Computational Linguistics: Stroudsburg, PA, USA, 2001; pp. 26–33

[38] Brill, E.; Brill, E.; Wu, J.; Wu, J. Classifier Combination for Improved Lexical Disambiguation. In Proceedings of the 17th International Conference on Computational Linguistics, Montreal, QC, Canada, 10–14 August 1998; Association for Computational Linguistics: Stroudsburg, PA, USA, 1998; Volume 1, pp. 191–195

[39] Eger, S.; vor der Brück, T.; Mehler, A. A Comparison of Four Character-Level String-to-String Translation Models for (OCR) Spelling Error Correction. Prague Bull. Math. Linguist. 2016

[40] Zhou, Y.; Porwal, U.; Konow, R. Spelling correction as a foreign language. In 2019 SIGIR Workshop on eCommerce, eCOM 2019; CEUR-WS: Aachen, Germany, 2019; Volume 2410.

[41] Sariev, A.; Nenchev, V.; Gerdjikov, S.; Mitankin, P.; Ganchev, H.; Mihov, S.; Tinchev, T. Flexible Noisy Text Correction. In Proceedings of the 11th IAPR International Workshop on Document Analysis Systems, DAS 2014, Tours-Loire Valley, France, 7–10 April 2014; pp. 31–35

[42] Koehn, P.; Hoang, H.; Birch, A.; Callison-Burch, C.; Federico, M.; Bertoldi, N.; Cowan, B. Moses: Open source toolkit for statistical machine translation. In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions, Prague, Czech Republic, 23–30 June 2007; Association for Computational Linguistics: Stroudsburg, PA, USA, 2007; pp. 177–180

[43] Kukich, K. Techniques for automatically correcting words in text. Acm Comput. Surv. 1992, 24, 377–439

[44] Mitton, R. English Spelling and the Computer; Longman Group: Harlow, Essex, UK, 1996; p. 214.

[45] Yannakoudakis, E.J.; Fawthrop, D. The rules of spelling errors. Inf. Process. Manag. 1983, 19, 87–99.

[46] Toutanova, K.; Moore, R.C. Pronunciation Modeling for Improved Spelling Correction. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, PA, USA

[47] Pirinen, T.A.; Lindén, K. State-of-the-art in weighted finite-state spell-checking. In Computational Linguistics and Intelligent Text Processing, Proceedings of the CICLing 2014, Kathmandu, Nepal, 6–12 April 2014; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2014

[48] Kantor, P.B.; Voorhees, E.M. The TREC-5 Confusion Track: Comparing Retrieval Methods for Scanned Text. Inf. Retr. 2000, 2, 165–176

[49] Gimenes, P.A.; Roman, N.T. Spelling error patterns in Brazilian Portuguese. Comput. Linguist. 2015, 41, 175–184

[50] Zitouni, I.; Sarikaya, R. Arabic diacritic restoration approach based on maximum entropy models. Comput. Speech Lang. 2009, 23, 257–276

[51] Azmi, A.M.; Almajed, R.S. A survey of automatic Arabic diacritization techniques. Nat. Lang. Eng. 2015, 21, 477–495

[52] Asahiah, F.O.; Odéjobi, O.A.; Adagunodo, E.R. A survey of diacritic restoration in abjad and alphabet writing systems. Nat. Lang. Eng. 2018, 24, 123–154

[53] Cheung, Brian. 2018. "Long Short Term Memory Networks." Accessed 2019-09-17.

[54] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

[55] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).

[56] Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." arXiv preprint arXiv:1508.04025 (2015).

[57] https://www.bmc.com/blogs/deep-neural-network/

[58] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

[59] Ahmadzade Ahmad, and Saber Malekzadeh. "Spell correction for azerbaijani language using deep neural networks." arXiv preprint arXiv:2102.03218 (2021).

[60] Sutskever, I.; Vinyals, O.; Le, Q. Sequence to Sequence Learning with Neural Networks. Adv. Neural Inf. Process. Syst. 2014, 4

[61] Schmaltz, A.; Kim, Y.; Rush, A.; Shieber, S. Sentence-Level Grammatical Error Identification as Sequence-to-Sequence Correction. In Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications (BEA@NAACL-HLT 2016), The Association for Computer Linguistics, San Diego, CA, USA, 12–17 June 2016; pp. 242–251