WORD CORRECTION USING NOISY CHANNEL

A project/thesis submitted in partial fulfillment for the degree of Master of Engineering of Jadavpur University.

By

Ditu Barai

Registration no: 137108 of 2016-2017

Class roll no: 201610504005

Bi – Annual roll no: M6TCT23040B

Under the guidance of

Dr. Sudip Kumar Naskar

Department of Computer Science & Engineering

Jadavpur University

Kolkata – 700032

2023

Department of Computer Science and Engineering

Jadavpur University

To whom it may concern

This is to certify that **Ditu Barai**, registration number **137108** of **2016-17**, class roll number **201610504005**, Bi – Annual roll no: **M6TCT23040B**, a student of department of computer science and engineering, Jadavpur University has done a thesis under my supervision, titled "**WORD CORRECTION USING NOISY CHANNEL**". The thesis is approved for submission towards partial fulfillment of the requirements for the degree of Master of Technology in Computer Technology, Jadavpur University for the session **2022-2023**.

3rd November, 2023

_____

Dr. Sudip Kumar Naskar

Thesis Supervisor,

Department of Computer Science and Engineering,

Jadavpur University

COUNTERSIGNED BY

_____

Dr. Nandini Mukhopadhyay

Head, Department of Computer Science and Engineering,

Jadavpur University

COUNTERSIGNED BY

_____

Prof. Saswati Mazumder

Dean, Faculty of Engineering and Technology,

Jadavpur University

Certificate of Approval

This is to certify that the thesis entitled "**WORD CORRECTION USING NOISY CHANNEL**" is a bona-fide record of work carried out by **DITU BARAI** in partial fulfilment of the requirements for the award of the degree of Master of Technology in Computer Technology in the Department of Computer Science and Engineering, Jadavpur University. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.

……

Signature of Examiner 1

Date:

……

Signature of Examiner 2

Date:

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

Declaration of Originality and Compliance of Academic Ethics

I hereby declare that this thesis entitled "**WORD CORRECTION USING NOISY CHANNEL**" contains literature survey and original research work by the undersigned candidate, as part of his Master of Computer Technology studies.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: Ditu Barai

Registration No.: 137108 of 2016-17

Class Roll No.: 201610504005

Bi – Annual roll no: M6TCT23040B

Thesis Title: Word correction using noisy channel.

…..…….

Signature

ACKNOWLEDGEMENTS

I express my deep sense of gratitude to my supervisor, Dr. Sudip Kumar Naskar, Assistant Professor Department Of Computer Science and Engineering, Jadavpur University for encouraging me to take Thesis under Natural Language Processing (NLP). I am also very much thankful to him for his valuable guidance, keen interest and encouragement at various stages in the completion of my term paper.

I acknowledge with thanks the kind of patronage, loving inspiration and timely guidance which I have received from my batch-mates. I would like to thank all the faculty members of the Department of Computer Science and Engineering of Jadavpur University for their continuous support.

At last, I would like to express my deep and sincere gratitude to my family members for the constant support that they gave me.

Name: DITU BARAI

Registration No.: 137108 of 2016-17

Class Roll No.: 201610504005

Bi – Annual roll no: M6TCT23040B

Department of Computer Science & Engineering, Jadavpur University

Abstract

In now days we are continuously using social media. Due to the massive uses of social media, we are typed messages different posts in social media. Due to the massive typing sometimes we are write misspelled words in the place of any correct word.

The Aim of this thesis is to detect the misspelled word and the error type. After that the word corrected by noisy channel model and suggest some probable best word against of this misspelled word.

For the correction we uses noisy channel model and one large corpus for the correction. After that we detect the misspelled word between the sentence and correct this word through noisy channel model and suggest the bigram probability of this sentence.

In this project I will describe the process of this correction.

Contents

List Of Figures:

List of Tables:

1.Introduction

1.1 Background

In now days, we are very much active in social media. We are used to send, write about anything and post pictures with caption and also send messages to each other. When we are write about anything in English language sometimes we are type misspelled word and also type wrong spelling of a correct word.

The aim of this thesis is to develop a spell corrector, which is detecting the misspelled word properly and correcting this word by various models and algorithm. After that, it gives the correct spelling and also suggest the similar type of words like spelling suggester in the mobile keyboard.

In this task at first we used to detect the misspelled word and then correct this word by spelling corrector and also suggest a different type of similar word and also detect the misspelled word within a sentence and correct this.

1.2 Project Purpose

In our daily life we are use to send messages to each other and write different things in social media and also write important documents as a official purpose. After writing anything we are searching very much carefully the misspelled word. If the document is very big in size then it is a very much hard work for us. For this reason we want to develop a spell corrector and also a spell suggester.

The work of spell corrector is to correct the misspelled spelling and also give the suggestions of the different words. The spell suggester suggests the different type of spelling of different word. We are choose the correct word, which is appropriate for our writings and also we get the word with correct spelling.

So, the main goal of this project is identify the misspelled words and give the correct word and also suggest any other similar type of words.

2. Literature Survery

2.1 What is NLP?

Natural language processing or NLP, is the area of computer science more precisely, the area of artificial intelligence or AI that focuses on enabling computers to comprehend spoken and written language in a manner that is similar to that of humans.

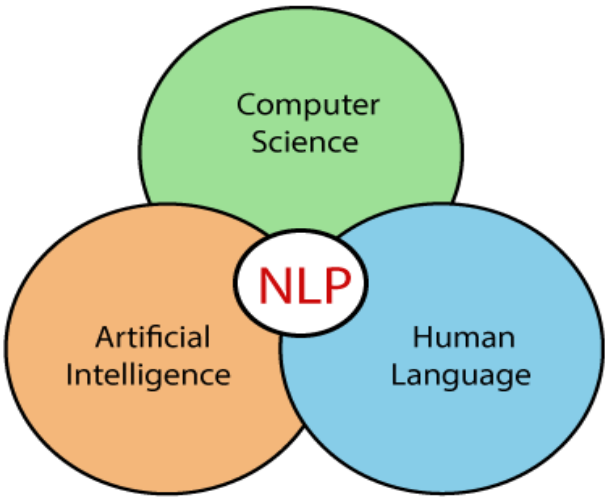

Figure 1. Structure of NLP

Natural Language Processing has a connection with machine learning, and deep learning models with computational linguistics, which is based on the rule of human language. When these technologies are combined, computers can process text or speech data including human language and fully "understand" the meaning, including the speaker's or writer's intent and sentiment.

Computer algorithms that can translate text between languages, react to spoken commands, and quickly summarize enormous amounts of material even in real time are powered by natural language processing (NLP). You most likely have experience with NLP through voice operated.

Because there are so many ambiguities in human language, it is very challenging to create software that reliably interprets text or audio input to represent the intended meaning. Programmers must teach natural language driven applications to recognize and understand irregularities in human language from the beginning if they are to be useful. These irregularities include homophones, sarcasm, idioms, grammar and usage exceptions, and variations in sentence structure.

A number of natural language processing (NLP) activities deconstruct human text and speech input so the machine can understand what it's consuming. Among these assignments are the following:

- **Speech recognition:**The process of accurately translating voice data into written data is known as speech recognition, or speech-to-text. Any program that responds to voice commands or understands spoken inquiries has to have speech recognition. Speech identification is particularly difficult because of the way people speak: they speak

quickly, slur words together, fluctuate in emphasis and tone, have diverse accents, and frequently use improper grammar.

- **Part of speech tagging:** Grammatical tagging, also known as part of speech tagging, is the process of classifying a word or phrase according to its usage and context. "Make" is a verb in "I can make a paper plane," and a noun in "What make of car do you own?" according to part of speech analysis.

- **Word sense disambiguation:** Word sense disambiguation is the process of choosing a word's meaning from among its many definitions by using semantic analysis to identify the word that makes the most sense in the situation in question. Word sense disambiguation, for instance, aids in separating the meaning of the verb "make" in phrases like "make the grade" (achieve) from "make a bet" (place).

- **Named entity recognition:** Named entity recognition is a process, in which we identify a word or different kind of phrases.

- **Co-reference resolution**: When one or two word have same entity then the task of Co-reference resolution is to detect this word.

- **Sentiment analysis:** In this part of NLP we recognize or extract the different kind of emotions Like, happiness, joy, sadness, sarcasm etc, from the text.

- **Natural language generation:** Speech to text is another important approach of NLP. But, in this case the task is put the structured information in to our human language.

2.2 What is Language Model?

The fundamental building block of contemporary Natural Language Processing (NLP) is a language model. It is a statistical model intended to examine the structure of human language and forecast the probability of a word or token sequence.
Language models are used by NLP-based applications to perform a range of functions, including spell checking, summarizing, sentiment analysis, audio to text conversion, and speech recognition.
Let's examine the role language models play in completing various NLP tasks:

- **Speech Recognition:** Speech recognition is the another important part of NLP. In now days, Alexa is very famous. Also we use Speech recognition techniques in mobile phones. It translate the human speech into text. In between the translation ASR mechanism analyze the sentiment of this sentence. It also analyze the homophones.

- **Machine Translation:** In the machine translation we can translate a language from another language. When you translate this language the it can also analyze the text and give different suggestions of the translated languages

Challenges with Language Modelling?

Programming languages are examples of formal languages, which have exact definitions. Every term in the system has a preset meaning associated with it. Without a clear specification, anyone familiar with a particular programming language can understand what's written.

Conversely, natural language is not created; rather, it develops based on each person's convenience and level of knowledge. Natural language has a number of terms that have several meanings. Although there is some uncertainty here, people can still understand it.

Machines can only communicate in numerical terms. All the words must be converted into a series of numbers in order to create language models. This is referred to as encodings by the modelers.

A code can be straightforward or intricate. Generally, each word is given a number; this is known as label-encoding. Every word in the statement "I love to play cricket on weekends" has a number [1, 2, 3, 4, 5, 6]. This is an illustration of one-hot encoding in action.

How does the language model works?

By examining the text in the data, language models calculate the likelihood of the following word. By passing the data via algorithms, these models interpret the information.

In natural language, the algorithms are in charge of formulating rules for the context. By studying the traits and qualities of a language, the models are ready to anticipate words. With this knowledge, the model gets ready to comprehend phrases and anticipate the words that will come after in sentences.

Many probabilistic methods are used to train language models. These methods differ according to the goal that drives the creation of a language model. The method used for creating and analyzing text data varies depending on the volume of text to be evaluated and the arithmetic used for analysis.

There are primarily two types of language models:

2.3 Statistical Language Model

Probabilistic models that can forecast the following word in a sequence based on the words that came before it developed as part of statistical models.

N-Gram: This is among the most basic methods for language modelling. Here, the size of the gram (or sequence of words given a probability) is defined by creating a probability

distribution for a sequence of 'n,' where 'n' can be any integer. A gram would say, "Can you help me?" if n = 4. Primarily, 'n' represents the quantity of context that the model is trained to take into account. Several varieties of N-Gram models exist, including Statistical language models use statistical techniques to determine the likelihood of a word sequence based on probability theory. These models typically display each word in the lexicon as a separate number in order to calculate the likelihood of each word emerging given the words that came before it.

One of the most popular statistical language models is the n-gram model, which splits a string of words into overlapping groups of n consecutive words, or n-grams. A string of three words, such as "I am going," "am going to," "to the grocery," and "the grocery store," would be used to indicate a sentence like "I am going to the grocery store." The likelihood of each 3-gram is then determined by the 3-gram model based on how frequently it occurs in the training data.

2.4 Neural Language Model

These neural network-based language models are frequently regarded as an advanced method for carrying out NLP tasks. Neural language models are employed for complicated tasks like speech recognition and machine translation because they overcome the drawbacks of traditional models like n-gram. Neural language models make use of artificial neural networks to discover the patterns and structures in a language. Each word is often represented by a dense vector or embedding which captures the word's context and meaning. The neural network receives the embeddings and uses them to process and forecast the likelihood of the next word in the sequence.

The ability of neural language models to handle vast vocabularies and intricate syntactic structures is one of its main features. They are particularly helpful for tasks like machine translation and text summarization because they can capture the context and meaning of words in a manner that statistical models cannot.

2.5 Related work

Text Normalization

Normalizing text is one step in the text preparation process, which is a crucial component of natural language processing (NLP). Grouping related tokens—tokens are typically the words in the text—is the aim of standardizing text. Some of the normalizing strategies in this post may not be necessary for you, depending on the text (Pennell &Liuyou 2011) are working with and the kind of analysis you are undertaking.

A hybrid technique for language detection, first using postprocessing heuristics and then machine learning based on CRF. The spelling correction module receives the identified English words and uses them to perform text normalization within a cohesive framework. Based on the noisy channel concept, the spelling corrector takes into account phonetic variances, wordplay, and shortened words. We obtain great accuracy with our hybrid language detection engine. Additionally, experimental data show that our combinative approach to spelling correction outperforms separate, independent models(Dutta et al.,2015). In the text normalization noisy

text is generated very much. When we typed speedily then the noisy text(Jose,2002) is generated. The Correction of noisy text(Schwarm & Ostendorf, 2011) is the another important work was done previously.

A text normalization method based on similarity, which takes into account similarities in context, sound, and spelling. Edit distance are used to quantify how similar two words are: a target (normalized) word and a source (nonstandard) word. The suggested normalization technique is capable of handling the often occurring homophone(Beliga et al.,2011), spelling, and insertion (repeated character) cases in texts(Chotimongkol et al.,2014) on Twitter.

In the text normalization text to speech conversion(Harde&Pooja,2019) is very important. Previously human writing or typing Chinese text in the some part of sentence we are using Arabic n numbers symbol to replace the full Chinese character set. "Non-Standard Words" (NSWs) are strings with Arabic numerals and non-Chinese symbols on the right-hand side; "spoken-form words" (SFWs) are strings with same elements on the left-hand side. NSWs fall into one of two categories: 2) Ambiguous NSWs (ANSWs) and 1) Basic NSWs (BNSWs). BNSWs may typically be recognized and described using rules that are carried out by finite state automata (FSAs). Since a BNSW typically only has one meaning, an FSA can quickly translate a BNSW to its SFW. However, even though FSAs can also identify the structures of ANSWs, an ANSW might have several meanings, which lead to different matching SFWs. An ANSW's precise meaning typically depends on its context.(Liou et al.,2016)

In our daily life we give different opinions(Aliero et al.,2023) and different comments in social media(Maitama et al.2014).The text is reflecting ours sentiment (Hirankan et al.,2013)or mentalities. Previously, Some of researchers work with this. They are extracting different sentiment, emotions, mentality from the text. In the social media we use different type of texts, like :

- The graphic representation of speech sounds is called phonetic typing. Different language dialects might have rather different pronunciations of the same word.
- When written in Roman script, these variances lead to a single word having various spellings. For example, the word afr can be spelled as be, vi, bhi, or bhe.
- Abbreviations are shorter versions(Khan & karim,2012)of words or phrases. It is made up of one letter or several letters that are extracted from the word. For example, Prof. is the abbreviation for Professor.(Jose et al.,2014)
- Word play/intentional misspelling for dramatic effect: These are imaginative spellings that are employed to convey emotions through dramatic effect. In this case, a word's spelling is lengthened by using a letter more than once. As in, yummmmmmy.
- Slang terms (acronyms): Internet and smartphone users utilize the fewest characters possible to communicate intelligibly. They have cleverly begun substituting digits for word segments, which lowers the word count while maintaining the same phonetic effect. For example, 2nite, Fi9, 4ever, 10q, TTLY, and BFNN

In this work they present a model that identify the word level language and the code mix script which is automatic translated and analysis the sentiment.(Sharma et al.,2015)(Branislav Gerazov,2011)

3. Methodology

3.1 Spelling Correction

Spelling correction is a very Important task in Natural Language Processing. It is used in various tasks like search engines, sentiment analysis, text summarization etc. As the name suggests, we try to detect and correct spelling errors in spelling correction. In real worls NLP tasks, we often deal with data having typos and their spelling corrections come to the rescue to improve model performance.
For example, if we want to search university and type “univarsty”, we will wish that the search engine suggests “university” instead of giving no results.
The spell correcting and checking techniques that have been developed and refined over the years of research and development are among the many real-world applications of Natural Language Processing (NLP) that are widely used in our daily lives. Examples of these activities include word prediction while sending texts, spell checker usage while writing documents, query prediction in search engines, etc. Therefore, it makes sense that we discuss how these algorithms have changed over time and how their relative efficacy changes depending on whether they are used alone or in combination with other algorithms.

We now study further how to identify approximate candidate matches of a query string in a big lexicon by utilizing lexicon indexes in conjunction with string matching algorithms.
We first describe a few popular algorithms, or approximation string matching approaches, that are used to identify approximate matches of a query string in a vast vocabulary. We must perform some sort of indexing, or a crude search of the provided vocabulary, in order to extract the likely candidates from the lexicon, as it is computationally impractical to conduct a thorough search of a huge lexicon for every query string. We do fine search to further hone our final findings after extracting the most likely options.

3.2 Approximate String matching techniques

The edit distance between two strings, s and t, of lengths m and n, respectively, is the number of single character insertions and deletions required to convert one string into another. This measure of string similarity was developed by Levenshtein (1965). The most often used operations that are permitted are 1. insert a character into a string
2. Take a character out of a string.
3. substitute another character for a character in a string.
We take a dynamic programming algorithm approach for this method which works in O(mn) time complexity. A matrix of (m+1)(n+1) is maintained and the (*i*,*j*) of the matrix will hold the the edit distance between the strings consisting of the first i characters of s and the first j characters of t. An outline is given:

$$
\begin{aligned}
&edit(0,0) = 0\\
&edit(i,0) = i\\
&edit(0,j) = j\\
&edit(i,j) = min[edit(i-1,j)+1,\\
&\qquad\qquad edit(i,j-1)+1,\\
&\qquad\qquad edit(i-1,j-1)+d(s_i,t_j)]
\end{aligned}
$$

There are more sophisticated versions of edit-distance which consider moves, allocation of different costs to each kind of operation, if the expression:

$$edit(i-2, j-2) + d(s_i, t_{j-1}) + d(s_{i-1}, t_j) + 1$$

1 is added here as argument in the call to minimum, and the resulting modified edit-distance allow the transposition of letters with a cost of 1. For example, for string 'play' and 'plya' the modified edit distance will add 1 for transposition of letters ay and ya in two string.

3.3 N gram

Spell corrector and spell suggester is depending on the probability theory. Before explaining the probability theory we need to discuss about the corpus. Corpus is a big size text file , which includes millions of words. In a corpus different types of texts are collected from different background.

After looking out the corpus, we need to count the number of wordform types in the corpus.

Now we start with N gram. We all have a basic knowledge about probability theory. Our main motivation is to count the probability of a word (w) from given some history (h). Corpus is a main things in the calculations of N gram.

In the N gram we are count the relative frequency of the word w from the corpus. After that we count the probability that how many times the word w is follow the history h in the corpus.

To count the probability of this, we use P(w|h).

In this way we are estimating the probability of the new word after the history and we could suggest the appropriate word by calculating this in the help of the corpus.

In the N gram we calculate the probability of the single word. If we want to calculate the joint probability of the word in the whole sentence, then what we can do?

In this case we found a cleverer way to calculate this. In this case we taking a particular variable Xi. Which is taking the value of the word. To calculate the joint probability of each word in a sentence have a particular value of the every word, like P(X=w1,Y=w2,Z=w3…..W=wn).

To calculate this we are using the chain rule of probability.

The rule is,

$$\begin{aligned} P(X_1 \ldots X_n) &= P(X_1)P(X_2|X_1)P(X_3|X_{1:2}) \ldots P(X_n|X_{1:n-1}) \\ &= \prod_{k=1}^{n} P(X_k|X_{1:k-1}) \end{aligned}$$

Applying the chain rule to words, we get

$$P(w_{1:n}) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2})\dots P(w_n|w_{1:n-1})$$
$$= \prod_{k=1}^{n} P(w_k|w_{1:k-1})$$

3.4 Bigram

The bigram model also counting the probability of the words . In the bigram model we calculate the approximate of the probability of a word also calculate the probability of all the previous words . We calculate the probability of the all previous word by using the conditional probability of the preceding word P(Wn|W1^n-1) .

We are using the bigram model to predict the conditional probability of the next word . So we use , P(Wn|W1^n-1) ~ P(Wn|Wn-1) . Here (W1^n-1) indicating the full string and (Wn-1) is the previous word .

A Markov assumption is the belief that a word's probability solely depends on the word that came before it. The class of probabilistic models known as Markov models makes the assumption that we can estimate the likelihood of a future unit without having to look too a long time ago. The bigram, which looks one word into the past, can be generalized.

Now let's look at a general formula for this n-gram estimate of the conditional likelihood of the subsequent word in a series. Here, N will refer to the n-gram. Hence bigrams are represented by N = 2 and trigrams by N = 3. Next, we estimate the likelihood of a word in light of its complete context as follows:

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1})$$

Given the bigram assumption for the probability of an individual word, we can compute the probability of a complete word sequence by substituting Equation,

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

3.5 Spelling Correction

In natural language processing, spelling correction is a crucial task. Text summarization, sentiment analysis, search engines, and other applications all use it. We attempt, as the name implies, to identify and fix spelling mistakes in spelling correction. Spelling correction is useful for dealing with typographical data, which is common in real-world natural language processing jobs, as it enhances the performance of the model. If we were to type "aple" into a search for "apple," for instance, we would want that the search engine would suggest "apple" rather than return no results.

Two importent stages of spelling correction task is:

1. Spelling error detection
2. Spelling error correction

Spelling Error Correction

In natural language processing, Spelling error correction identifying the misspelled word and rectifying this word. This is a very important task in NLP. It uses only search engines, chatbots, word processors and different communication approach.

In the real word spelling correction the important task is detecting and correcting the spelling errors.The errors can happens from the typographical errors like, insertion, deletion, substitution, transposition. We use the edit distance algorithm to detect the errors between the correct and the misspelled word and also use the noisy channel model to correct the spelling.

3.6 Edit Distance Algorithm

In the edit distance algorithm we check the similarity between two strings. By this Edit Distance Algorithm we can calculate the minimum number of single character edits.

Different types of edit distance allow different sets of string operations. For instance:

- The Levenshtein distance allows the deletion, insertion and substitution.
- The longest common subsequence distance does not allow transposition.
- The Hamming distance allows the substitution.
- The Damerau–Levenshtein distance allow the four type of edits, Like-Insertionm,Deletion,Substitution,Transpotition.
- The Jaro distance work only with the transposition.

It is possible to calculate the Damerau–Levenshtein distance by the use of dynamic programming. Building a matrix with each cell representing the edit distance between the original strings' substrings is the fundamental principle. The final cell has the minimum edit distance and is filled in by going left to right and top to bottom in this matrix. A edit distance matrix is created using dynamic programming algorithms for sequence comparison, with one row representing each symbol in the source sequence and one column representing each symbol in the target sequence. This matrix is the edit distance matrix for the minimal edit distance.

Every cell edit distance[i,j] contains the distance between the first i character of the target and the and the first j character of the source.

The value in each cell computed by taking the minimum of the three possible paths through the matrix.

To express Damerau Levenshtein distance between two string a and b, a function $d_{ab}(i,j)$ is defined, whose value is a distance between an I symbol prefix(initial substring) of string a and a j symbol prefix of b.

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

Each recursive call matches one of the cases covered by the Damerau Lavenshtein distance.

$d_{a,b}(i-1,j)+1$, denotes the deletion (from a to b)

$d_{a,b}(I,j-1)+1$, denotes the insertion(from a to b)

$d_{a,b}(i-1,j-1)+1$, denotes a match or mis match, depending on weather the respective symbols are the same.

$d_{a,b}(i-2,j-2)+1$, denotes the transposition between two successive symbols.

The Damerau Levenshtein distance between a and b is the given by the function value for full string:

$D_{a,d}$ (|a| and |b|), where i=|a| marked the length of the string a, and j=|b| is the length of b.

3.7 Noisy Channel Model

The misspelled word is treated as though it had been "distorted" by passing via a noisy communication channel in the noisy channel model. This channel adds "noise" by changing or adding other elements to the letters, making it challenging to identify the "real" word. So, our objective is to construct the channel model. Then, using this approach, we identify the correct word by passing through each word of the language using our noisy channel model and determining which one comes closest to the word that was misspelled.

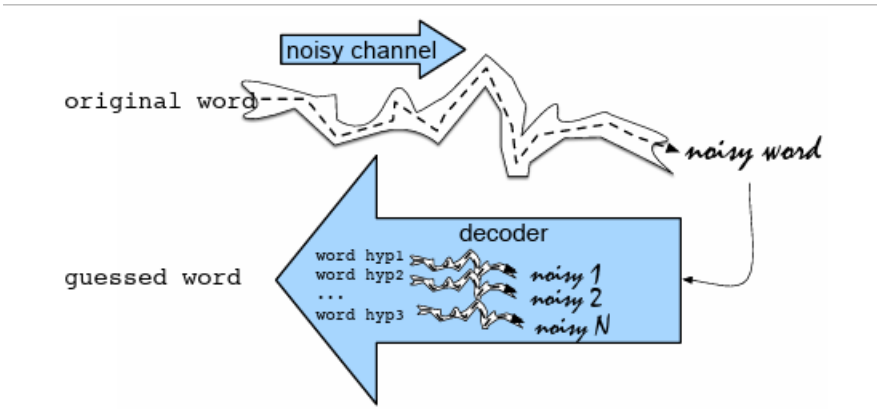

Figure 2. Noisy Channel Model

According to the noisy channel hypothesis, the surface form that is visible to us is actually a "distorted" version of the original word that has been transmitted through a noisy channel. Each hypothesis is run through a model of this channel by the decoder, which then selects the word that most closely resembles the noisy word on the surface. This model of noisy channels is an example of Bayesian inference.

Our task is to identify the word w that gave rise to the misspelled word x that we observe in the observation. We are looking for a word w such that P(wx) is highest among all the words that are feasible in the vocabulary V. "Our estimate of the correct word" is denoted by the cap ^ notation .

$$\hat{w} = \underset{w \in V}{\operatorname{argmax}} P(w|x)$$

The function $\operatorname{argmax}_x f(x)$ means “the x such that f(x) is maximized”. In the previous Equation means, that out of all words in the vocabulary, we want the particular word that maximizes the right-hand side P(w|x). The intuition of Bayesian classification is to use Bayes’ rule to transform previous equation into a set of other probabilities. Bayes’ rule is presented in the next equation; it gives us a way to break down any conditional probability P(a|b) into three other probabilities:

$$P(a|b) = \frac{P(b|a)P(a)}{P(b)}$$

We can substitute the current equation into previous equation to get the new modified equation.

$$\hat{w} = \underset{w \in V}{\operatorname{argmax}} \frac{P(x|w)P(w)}{P(x)}$$

We can conveniently simplify the current equation by dropping the denominator P(x). Why is that? Since we are choosing a potential correction word out of all words, we will be computing P(x|w)P(w) /P(x) for each word. But P(x) doesn’t change for each word; we are always asking

about the most likely word for the same observed error x, which must have the same probability P(x). Thus, we can choose the word that maximizes this simpler formula:

$$\hat{w} = \underset{w \in V}{\operatorname{argmax}} P(x|w)\, P(w)$$

To summarize, the noisy channel model says that we have some true underlying word w, and we have a noisy channel that modifies the word into some possible misspelled observed surface form. The likelihood or channel model of the noisy channel producing any particular observation sequence x is modeled by P(x|w).

P(w) models the prior probability of a hidden word. Given several reported misspellings x, we may calculate the most likely word ˆ w by multiplying the prior P(w) and the likelihood P(x|w) and selecting the word for which this product is largest. By taking any term that is not in our spelling dictionary, we can use the noisy channel approach to fix non-word spelling problems. We do this by creating a list of candidate words, ranking them based on the previous equation, and selecting the word with the highest ranking. The above equation might be changed as follows to use this list of potential words rather than the entire vocabulary V:

$$\hat{w} = \underset{w \in C}{\operatorname{argmax}} \overbrace{P(x|w)}^{\text{channel model}} \overbrace{P(w)}^{\text{prior}}$$

In our project how the prior and channel model is calculated?

3.8 Channel Model

In this case the important factor is the predict an insertion, deletion, substitution and transposition the the correct letter.

In the project we put a misspelled word for the correction. At first we check the type of error between the misspelled word and the suggested word and then check the edit distance between this. Through the edit distance algorithm we detect the type of error in the word. In the misspelled word the letters are how many times substituted by another letter or inserted or deleted by mistake in the large corpus, it is represented by the confusion matrix. This is a 26X26 matrix. This matrix is represent one letter how many times incorrectly used instead of another.

A confusion matrix was computed by coding of a collection of spelling errors with the correct spelling and then count the number of times different errors occurred. Kernighan et al(1990) corpus used for create the confusion matrix. It creates four confusion matrix, one of each type of single error.

- del[x,y] contains the number of times in the training set that the characters xy in the correct word were types as x.
- ins[x,y] contains the number of times in the training set that the character x in the correct word was typed as xy.
- Sub[x,y] the number of times that x was typed as y.
- Trans[x,y] the number of times that xy was typed as yx.

Frequency of the addition of letter

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	8	4	39	28	10	0	8	3	74	1	8	28	11	64	1	5	0	82	26	27	32	5	2	5	12	2
b	31	9	0	0	34	0	0	0	1	0	3	0	0	7	0	0	6	6	3	8	0	0	0	0	3	0
c	27	0	75	0	51	1	1	31	0	52	3	0	2	70	1	4	6	10	29	23	1	0	2	2	3	0
d	19	0	0	32	101	1	2	0	47	1	0	4	0	0	18	0	0	6	4	1	5	0	1	0	1	0
e	131	0	53	77	78	0	14	0	51	1	4	14	18	60	19	11	9	94	78	12	7	9	1	19	13	2
f	33	0	1	0	52	53	0	1	18	0	0	4	0	0	20	0	0	16	0	3	14	0	0	0	0	0
g	13	1	0	2	50	0	10	6	15	0	0	3	1	0	8	0	1	7	4	4	19	0	1	0	0	0
h	18	0	0	0	61	0	2	4	41	0	0	3	1	0	21	0	0	12	2	8	12	0	0	0	5	0
i	43	1	44	26	82	5	10	2	6	0	5	7	9	36	32	3	2	15	67	25	7	9	0	1	6	17
j	0	0	0	0	1	0	0	0	1	0	0	0	0	0	2	0	0	0	0	0	1	0	0	0	0	0
k	1	0	1	0	33	0	1	0	4	0	1	0	0	1	1	0	0	0	3	1	0	0	0	0	1	0
l	35	0	1	1	102	1	1	0	73	0	0	223	1	1	26	2	0	1	4	4	7	1	0	0	6	0
m	34	5	0	0	86	0	2	0	39	0	0	1	79	9	20	0	0	2	3	2	8	0	0	0	2	0
n	37	2	50	23	112	5	33	1	75	0	5	2	6	106	18	1	1	3	49	65	16	2	2	0	4	2
o	33	1	20	3	18	2	6	3	15	1	2	10	32	44	43	9	0	40	27	7	111	4	39	0	5	0
p	17	0	1	0	101	0	0	38	33	0	0	15	0	1	12	48	0	25	2	5	5	0	0	0	1	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
r	33	2	10	10	169	1	0	3	82	0	0	7	2	10	31	0	0	100	20	6	17	1	0	1	24	0
s	39	0	48	3	139	2	3	44	70	0	2	3	2	1	20	3	0	0	119	28	22	0	1	0	3	1
t	63	0	8	2	185	0	1	30	93	1	0	2	2	0	27	0	0	18	20	79	28	0	1	0	3	0
u	23	1	11	4	37	0	6	2	20	0	1	9	6	15	1	6	0	34	4	14	1	1	0	0	0	0
v	15	0	1	0	79	0	0	0	32	0	0	0	0	2	14	0	0	1	0	0	5	1	0	0	2	0
w	5	0	0	1	33	0	0	32	7	0	0	2	0	1	8	0	0	0	1	0	0	0	0	0	0	0
x	2	0	20	0	7	0	1	6	3	0	0	0	0	0	0	1	0	0	34	1	0	0	0	1	0	3
y	0	0	0	0	16	0	0	4	0	0	0	0	0	0	1	1	0	5	0	1	0	0	0	0	0	1
z	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 1

Frequency of the deletion of letter

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	24	165	55	11	18	68	0	115	0	3	94	35	182	5	26	0	96	37	74	97	13	3	0	19	0
b	16	14	0	0	117	0	0	0	35	0	0	9	0	0	23	0	0	35	4	10	19	0	0	0	1	0
c	85	0	167	0	98	0	0	##	125	0	36	6	0	0	198	0	6	43	5	59	66	0	0	0	4	0
d	43	0	0	24	146	0	11	0	107	2	1	9	1	1	25	0	0	6	10	0	10	0	0	0	9	0
e	280	0	199	80	97	36	21	6	52	38	0	52	66	126	16	10	77	106	136	28	11	11	0	27	10	3
f	37	0	0	0	41	107	0	0	148	0	0	29	0	0	31	0	0	106	0	6	6	0	0	0	1	0
g	48	0	0	0	76	0	51	23	32	0	0	11	0	27	11	0	0	50	8	5	151	0	0	0	1	0
h	45	0	0	0	129	0	0	1	61	0	0	6	1	7	44	0	0	5	4	9	18	0	0	0	22	0
i	53	2	121	24	74	19	119	0	0	0	2	28	23	99	39	16	0	16	138	82	5	9	0	0	0	3
j	0	0	0	0	2	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	11	0	0	0	0	0
k	1	0	0	0	48	0	0	0	10	0	0	0	0	9	0	1	0	0	13	0	0	0	0	0	0	0
l	55	0	2	7	130	5	0	0	102	0	0	295	0	0	52	3	0	2	6	9	5	11	0	0	9	0
m	114	22	0	0	108	9	0	0	85	0	0	0	223	8	40	57	0	0	8	0	14	0	0	0	14	0
n	113	0	93	62	163	7	65	0	156	5	2	1	1	189	36	2	8	0	155	104	20	23	0	31	4	0
o	35	10	33	17	31	4	12	1	23	1	3	25	53	93	56	30	0	111	11	29	251	11	61	13	16	1
p	101	0	0	0	188	0	0	37	61	0	0	39	0	20	65	295	0	133	13	17	9	0	0	0	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0	0	0	0	0	0
r	203	0	24	8	276	1	14	28	192	0	1	4	16	28	74	1	0	162	23	72	14	1	1	0	13	0
s	64	0	357	126	5	0	73	97	0	30	1	0	2	87	65	10	0	163	156	152	0	15	0	30	0	0
t	120	1	3	0	247	0	1	48	162	0	0	8	4	4	121	0	0	139	32	131	81	0	2	0	8	0
u	42	31	33	19	9	13	0	77	0	0	17	7	75	8	8	0	64	30	14	0	0	0	0	0	0	0
v	50	0	0	0	112	0	0	0	45	0	0	0	0	15	0	0	0	0	0	1	0	0	0	0	0	0
w	30	0	0	3	31	0	0	66	14	0	0	9	0	1	32	0	0	5	2	0	0	0	0	1	0	0
x	20	0	45	0	8	0	0	52	18	0	0	0	0	0	17	14	0	0	8	0	0	0	0	1	0	0
y	6	0	8	0	16	0	0	0	6	0	0	0	9	0	5	2	0	1	17	1	0	0	0	0	0	0
z	0	0	0	0	3	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2

Table 2

Frequency of the substitution of letter

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	5	19	12	579	3	2	37	222	0	8	41	7	22	306	1	2	74	25	12	102	5	3	1	7	2
b	1	0	0	32	8	2	0	0	1	0	0	7	10	3	0	36	0	3	0	3	0	7	1	0	2	0
c	36	0	0	8	36	3	20	9	21	0	77	5	3	11	12	4	27	20	401	79	8	2	0	41	3	5
d	18	29	5	0	12	3	31	0	8	4	3	12	2	13	2	1	0	9	14	53	3	1	0	0	2	0
e	625	3	10	20	0	13	6	23	596	0	0	23	14	11	134	9	3	46	19	33	130	3	9	1	22	2
f	11	0	4	4	4	0	1	2	10	0	0	4	2	4	0	27	0	19	9	14	3	41	2	0	0	0
g	5	2	46	36	14	4	0	3	2	12	9	3	5	20	3	20	8	4	6	7	2	2	0	0	5	2
h	30	4	5	0	42	2	1	0	35	2	15	3	0	7	18	5	0	5	9	15	14	0	4	0	6	0
i	317	2	15	8	715	12	7	19	0	1	0	49	19	41	86	4	1	42	28	40	92	6	1	0	115	1
j	0	0	3	38	0	0	23	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0
k	4	0	43	2	2	1	8	5	2	0	0	3	0	9	8	0	1	2	0	9	0	0	1	0	0	0
l	50	1	8	1	30	7	0	2	24	0	8	0	0	17	19	2	0	33	2	25	17	2	3	0	1	0
m	13	7	2	4	13	2	4	0	5	0	0	3	0	171	6	7	0	7	4	1	7	2	4	1	3	0
n	26	2	9	17	30	4	7	23	32	0	4	8	134	0	28	1	0	30	13	24	29	5	9	3	4	0
o	298	4	21	5	199	2	3	17	68	0	9	16	10	10	0	6	1	17	5	5	142	3	9	0	8	0
p	15	32	14	3	18	25	1	6	4	0	4	5	4	4	5	0	0	34	28	17	1	1	0	1	1	0
q	3	0	38	0	2	0	10	0	0	0	7	0	0	1	0	3	0	0	2	0	0	0	0	1	0	0
r	65	2	4	8	49	2	5	9	37	0	4	22	7	28	28	7	0	0	10	13	37	3	10	0	8	0
s	16	1	213	4	21	5	3	13	11	0	1	4	5	14	6	15	2	19	0	36	12	1	4	9	3	27
t	21	2	70	65	27	20	13	13	30	0	17	29	3	24	12	5	2	22	88	0	6	3	2	1	5	0
u	124	2	4	5	161	0	8	19	82	1	2	19	3	26	185	2	1	45	7	20	0	2	30	0	12	0
v	2	8	2	5	1	39	0	0	1	1	0	2	1	1	2	0	0	5	1	4	5	0	7	0	2	0
w	3	1	1	3	7	1	1	10	2	0	0	8	8	10	8	0	0	13	2	1	39	1	0	0	4	0
x	1	0	22	1	2	1	10	0	5	0	1	0	3	1	0	1	0	2	22	0	0	2	0	0	1	8
y	16	0	3	2	61	0	0	1	109	0	1	3	0	2	3	1	0	6	4	3	9	1	2	1	0	1
z	0	0	1	1	1	0	1	0	0	0	0	0	0		0	0	0	27	0	0	0	0	0	2	0	0

Table 3

Frequency of the transposition of letter

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	2	3	0	5	0	1	0	29	0	7	12	2	16	3	1	0	17	2	4	8	2	0	0	4	0
b	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	4	0	0	0	0	1	0	0	3	13	0	1	4	0	0	2	0	0	0	0	4	3	0	0	2	0
d	3	0	0	0	13	0	7	0	3	0	0	1	0	0	1	0	0	0	0	0	4	0	0	0	1	0
e	20	0	0	14	0	3	0	0	46	0	0	37	13	11	2	1	0	37	13	8	14	0	1	0	1	2
f	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
g	4	0	0	0	4	0	0	0	5	0	0	0	0	7	1	0	0	1	0	0	3	0		0	0	0
h	8	0	0	0	7	0	0	0	3	0	0	0	0	4	0	0	0	0	0	13	1	0	0	0	0	0
i	17	0	10	2	76	0	4	0	0	0	0	8	4	11	10	0	0	21	17	9	2	0	0	0	0	1
j	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
k	0	0	0	0	5	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
l	7	0	0	4	59	0	0	0	10	0	1	0	0	0	9	0	0	0	3	4	5	0	0	0	3	0
m	9	1	0	0	7	0	0	0	7	0	0	0	0	1	3	0	0	0	1	0	1	0	0	0	0	0
n	12	0	1	5	12	0	5	0	18	0	3	1	1	0	7	0	0	0	2	3	1	0	0	0	0	0
o	5	0	1	0	5	1	1	0	10	0	0	11	3	1	0	5	0	33	2	2	8	1	6	0	0	0
p	0	0	0	0	5	0	0	2	1	0	0	2	0	0	1	0	0	3	0	0	1	0	0	0	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
r	10	0	0	0	48	0	1	1	20	0	2	0	1	1	19	0	0	0	3	0	9	0	0	0	3	0
s	1	0	5	0	16	0	0	0	9	0	0	0	0	0	1	1	0	0	0	8	0	0	1	0	2	0
t	3	0	8	0	12	0	0	10	5	0	0	3	0	0	4	0	0	1	4	0	1	0	0	0	0	0
u	22	0	2	2	2	0	0	0	9	0	0	3	1	4	5	0	0	11	2	0	0	0	0	0	0	0
v	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
w	0	0	0	0	0	0	0	6	2	0	0	1	0	3	5	0	0	1	0	0	0	0	0	0	0	0
x	0	0	0	0	0	0	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
y	0	0	0	0	4	0	0	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4

Using the value of four confusion matrix, we calculate the channel model p(x|w). In this case we have for formula for each type of edit. w_i is the i_{th} character of the correct word w and x_i is the i_{th} character of the typo x:

$$P(x|w) = \begin{cases} \dfrac{\text{del}[x_{i-1}, w_i]}{\text{count}[x_{i-1} w_i]}, \text{ if deletion} \\ \dfrac{\text{ins}[x_{i-1}, w_i]}{\text{count}[w_{i-1}]}, \text{ if insertion} \\ \dfrac{\text{sub}[x_i, w_i]}{\text{count}[w_i]}, \text{ if substitution} \\ \dfrac{\text{trans}[w_i, w_{i+1}]}{\text{count}[w_i w_{i+1}]}, \text{ if transposition} \end{cases}$$

3.9 Prior model

The prior probability of each correction p(w) is the language model probability of the word w in context. This is the unigram probability p(w). In the spelling correction process this is extend up to trigram or 4 gram probabilities. In the prior model first we take a large corpus to calculate the total no of words of the corpus. When we input a misspelled word, after this and after that calculate the occurrence of the suggested word. After that calculate the probability of this.

After the calculation of the channel model and the prior model for this misspelled word, we multiplied the value of the channel model and the prior model with 10^9 for readability, and get the result.

3.10 Bigram probability of a word

In the bi gram probability of a word we compute the probability of a word depends on the previous word and also the next word.

This is called Markov assumption. In the Markov assumption we can use the bi gram, trigram and also n gram. In this case the general equation of n gram approximation to the conditional probability for the next word is:

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)}$$

We are also simplify this equation. We take the sum of all bigram counts, which is start with a word W_{n-1}, must be equal to the unigram count. The simplified equation is:

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

By this formula we can compute the probability of a sentence by the bigram probability of the each word of the sentence. We calculate the bigram probability of the each word with the previous and also the next word of the sentence. If we give a misspelled word in the sentence, then this misspelled word corrected by the noisy channel model and create a suggestion list of words. After that calculate the bigram probability of the all suggested word given by the noisy channel model. After that multiply all the bigram probabilities of the sentence with the suggested word one by one and get the probabilities value of the sentence.

The highest probability value is taken by us as a correct suggestion of the bigram probability of the sentence.

This is the simplest and the best way to estimate the probabilities. It is very important to calculate the bi gram probability of word a, given by the previous word b. We compute the bigram count and normalize the sum of all the bigrams.

3.11 Results

For the spelling correction task the noisy channel model is used for the correction. At first detect the error and then computed the edit distance from the minimum edit distance algorithm. After that the prior is calculated and the confusion matrix is created by the data of the corpus. The Channel model is calculated by the edit type of this word. If we calculated the bigram or trigram probability of a word after the correction, then we use the Maximum likelihood estimation formula to calculate the bigram or trigram probability of a word in a sentence.

5. Conclusion

In this project we discuss about the very important part of Natural Language Processing. Previously different work was done in this area. In our project mainly we discuss about the spelling correction. Here we work with noisy channel model, n gram and bi gram probabilities. Here we see the misspelled word has mainly four types error. Our aim is, firstly detect the error type and correct this word. After that we correct misspelled word in the middle of the sentence. Also give the correct probable word after this word.

To work with this project, I read about the noisy channel model and also the different part of NLP. The project is dependent on the corpus. This is the one of the limitations of this project.

In this project we work with the English language. If I get the chance in future, I want to work with the same model to correct the Bengali word.

Bibliography

- Dutta, Sukanya, et al. "Text normalization in code-mixed social media text." *2015 IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS)*.
- Liou, Guan-Ting, Yih-Ru Wang, and Chen-Yu Chiang. "Text normalization for Mandarin TTS by using keyword information."*Coordination and Standardization of Speech Databases and Assessment Techniques (O-COCOSDA), 2016 Conference of The Oriental Chapter of International Committee for*. IEEE, 2016.
- Sharma, Shashank, P. Y. K. L. Srinivas, and Rakesh Chandra Balabantaray. "Text normalization of code mix and sentiment analysis." *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on*. IEEE, 2015.
- Jose, Greety, and Nisha S. Raj. "Noisy SMS text normalization model." *Convergence of Technology (I2CT), 2014 International Conference for*. IEEE, 2014.
- Schwarm, Sarah, and Mari Ostendorf. "Text normalization with varied data sources for conversational speech language modeling." *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*. Vol. 1. IEEE, 2002.
- Chotimongkol, Ananlada, Kwanchiva Thangthai, and Chai Wutiwiwatchai. "Utilizing social media data through similarity-based text normalization for LVCSR language modeling." *Co-ordination and Standardization of Speech Databases and Assessment Techniques (COCOSDA), 2014 17th Oriental Chapter of the International Committee for the*. IEEE, 2014.
- Geazov, Branislav, and Zoran Ivanovski. "Text Normalization and Phonetic Analysis Modules for Macedonian TTS synthesis." 2011 19thTelecommunications Forum (TELFOR) Proceedings of Papers. 2011.
- Beliga, Slobodan, and Sanda Martinčić-Ipšić. "Text normalization for croatian speech synthesis." *MIPRO, 2011 Proceedings of the 34th International Convention*. IEEE, 2011.
- Hirankan, Pawanrat, Atiwong Suchato, and Proadpran Punyabukkana. "Detection of wordplay generated by reproduction of letters in social media texts." *Computer Science and Software Engineering (JCSSE), 2013 10th International Joint Conference on*. IEEE, 2013.
- Khan, Osama A., and Asim Karim. "A rule-based model for normalization of sms text." *Tools with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference on*. Vol. 1. IEEE, 2012.
- Jose, Greety, and Nisha S. Raj. "Lexical normalization model for noisy SMS text." *Computational Systems and Communications (ICCSC), 2014 First International Conference on*. IEEE, 2014.
- Maitama, Jaafar Zubairu, et al. "Text normalization algorithm for facebook chats in hausa language." *Information and Communication Technology for The Muslim World (ICT4M), 2014 The 5th International Conference on*. IEEE, 2014.

- Pennell, Deana, and Yang Liu. "Toward text message normalization: Modeling abbreviation generation." *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011.
- Aliero, Abubakar & Bashir, Sulaimon & Aliyu, Hamzat & Tafida, Amina & Kangiwa, Bashar & Dankolo, Nasiru. (2023). Systematic Review on Text Normalization Techniques and its Approach to Non-Standard Words. International Journal of Computer Applications. 185. 975-8887.
- Harde, Pooja. (2019). An Experimental Technique on Text Normalization and its Role in Speech Synthesis.