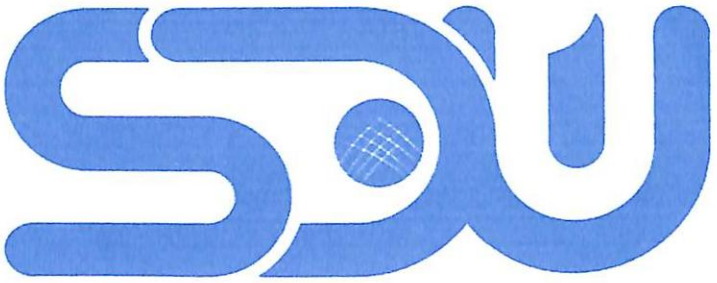
Ministry of Education and Science of the Republic of Kazakhstan
Suleyman Demirel University



Assylay Tolegenova

Automatic error detection and correction of Kazakh text

A thesis submitted for the degree of
Degree of Master of Science in Computer Science
(degree code: 7M06102)

Kaskelen, 2022

Ministry of Education and Science of the Republic of Kazakhstan
Suleyman Demirel University
Faculty of Engineering and Natural Sciences

Automatic error detection and correction of Kazakh text

A thesis submitted for the degree of
Degree of Master of Science in Computer Science
(degree code: 7M06102)

Author: **Assylay Tolegenova**

Supervisor: **PhD Assem Talasbek**

Dean of the faculty:
Associate Professor, PhD Zhamanov Azamat

Kaskelen, 2022

Suleyman Demirel University

Faculty of Engineering and Natural Sciences

Department of Computer Science

Dean of Faculty

Associate Professor, PhD Zhamanov A.

« 30 » May 2022

Topic of the thesis:

Automatic error detection and correction of Kazakh text

Thesis submitted as part of the requirements for the award of the MSc in "7M06102 - Computer Science", SDU, 2020-2022

Head of Department: Associate Professor, PhD Cemil Turan

Academic Supervisor: PhD Assem Talasbek

Master's student: Assylay Tolegenova

Kaskelen, 2022

Abstract

The amount of complicated documents and texts has increased exponentially in recent years, necessitating a deeper understanding of machine learning technologies in order to effectively identify texts in numerous applications. Text normalization is one of the best decision. It is the reduction of all words of the text to the original form. This paper investigates a layered strategy for fixing mistakes in Kazakh language literature downloaded from the Internet. This work is devoted to the study of automatic systems for checking the spelling of the Kazakh language using natural language processing tasks. Currently, most of these type of machines are designed for English, and few are processed for the Kazakh language. The paper discusses the methodology for evaluating automatic spelling checkers and error correction, developed by the author. A description of the selected systems is given. The main goal is to validate the use of n gram for agglutinative languages. On the basis of the study, the best system of use is distinguished, between the n gram and Symspell, and a conclusion is drawn about the further relevance of the development in this area. To complete the auto correction task, we first find an error while typing words, and then choose the most appropriate one.

Keywords: Natural Language Processing(NLP), spell correction, n-gram model, SymSpell algorithm, text normalization, edit distance algorithm, morphology analyze, automatic system.

Аңдатпа

Соңғы жылдары күрделі құжаттар мен мәтіндердің саны экспоненциалды түрде өсті, бұл көптеген қолданбаларда мәтіндерді тиімді анықтау үшін машиналық оқыту технологияларын тереңірек түсінуді талап етеді. Мәтінді қалыпқа келтіру - ең жақсы шешімдердің бірі. Бұл мәтіндегі барлық сөздерді бастапқы түріне келтіру. Бұл мақалада интернеттен жүктеп алынған қазақ тіліндегі әдебиеттердегі қателерді түзетудің көп деңгейлі стратегиясы қарастырылған. Бұл жұмыс табиғи тілді өңдеу тапсырмалары арқылы қазақ тілінің емлесін тексерудің автоматты жүйелерін зерттеуге арналған. Қазіргі уақытта мұндай үлгідегі машиналардың көпшілігі ағылшын тіліне арналған, ал кейбіреулері қазақ тіліне өңделуде. Мақалада автор әзірлеген емлені автоматты түрде тексеру және қателерді түзету құралдарын бағалау әдістемесі талқыланады. Таңдалған жүйелердің сипаттамасы берілген. Негізгі мақсат - агглютинативті тілдер үшін грамматиканы n қолдануды растау. Зерттеу негізінде ngram және Symspell арасында ең жақсы пайдалану жүйесі бөлініп, осы саладағы дамудың одан әрі өзектілігі туралы қорытынды жасалады. Автоматты түзету тапсырмасын орындау үшін алдымен сөздерді теруде қатені тауып, содан кейін ең қолайлысын таңдаймыз.

Түйінді сөздер: Табиғи тілді өңдеу (NLP), орфографияны түзету,н грам моделі, SymSpell алгоритмі, мәтінді нормалау, өңдеу қашықтығы алгоритмі, морфологиялық талдау, автоматты жүйе.

Аннотация

Количество сложных документов и текстов в последние годы увеличилось в геометрической прогрессии, что требует более глубокого понимания технологий машинного обучения для эффективной идентификации текстов в многочисленных приложениях. Нормализация текста — одно из лучших решений. Это приведение всех слов текста к исходному виду. В данной статье исследуется многоуровневая стратегия исправления ошибок в казахскоязычной литературе, скачанной из Интернета. Данная работа посвящена изучению автоматических систем проверки орфографии казахского языка с использованием задач обработки естественного языка. В настоящее время большинство машин этого типа рассчитаны на английский язык, и немногие перерабатываются на казахский язык. В статье рассматривается методика оценки автоматических средств проверки орфографии и исправления ошибок, разработанная автором. Дано описание выбранных систем. Основная цель - подтвердить использование грамматики n для агглютинативных языков. На основе исследования выделяется лучшая система использования, между ngram и Symspell, и делается вывод о дальнейшей актуальности разработки в этой области. Для выполнения задачи автокоррекции мы сначала находим ошибку при наборе слов, а затем выбираем наиболее подходящую.

Ключевые слова: Обработка естественного языка (NLP), коррекция орфографии, модель n-грамм, алгоритм SymSpell, нормализация текста, алгоритм расстояния редактирования, анализ морфологии, автоматическая система.

Contents

Chapter 1

Introduction

1.1 Motivation

At any stage of human development, every society and country is in constant motion with informants, so its natural desire is to regulate and work with automation for processing. In our world, there are and are developing various ways of representing data, but despite this, natural languages remain the most used and most difficult for automatic processing.

Natural Language Processing (NLP) is a general area of artificial intelligence and mathematical linguistics. With the help of this direction, the problems of computer verification and synthesis of natural languages are solved. Analysis with artificial intelligence means understanding machine language, and synthesis means generating literate text. By solving these problems, you can create a sphere where a person and a computer can find interaction. The main parts of natural language processing include data extraction, text and sentiment analysis, language translation, Internet search, generation of words in text, etc.

The most well-known applications of NLP include:

- Question answering (QA);
- systems for extracting information and extracting facts;
- automatic analysis of opinions (Sentiment analysis or Opinion Mining);
- machine translation systems;
- automatic paraphrasing, rewriting, synonymizer;
- Text Summarization;
- man-machine dialogue in system interfaces;
- information retrieval systems;

- automatic classification and clustering of texts;
- automatic spell checker;
- automatic ontology generation;

The system of preventing spelling errors in the text has always been extremely acute not only in documents, but also in public life, since the nation's literacy is its weapon in the struggle for culture. In this regard, the acquisition of the highest level of spelling skills is still the most important thing in natural language learning, as it can be used in search to correctly filter and save data without duplication. The structures that are established by the rules in the spelling system have a certain basis, and most of all associated with the morphological word, as well as with the grammatical form of words and their belonging.

Spelling is, first of all, an established system that society accepts and uses; ensuring uniformity of spellings, even where different spellings are possible; rules for compliance in all languages; written form of speech for further use.

Methods and algorithms for automatic detection and correction of recognition errors are largely based on the application and processing of well-known approaches such as spelling error correction using neural networks, reading the distance of words and characters, hidden Markov models, n-grams of words and one of the main finite automate. Also, systems are used that combine the results of several recognition methods, using an additional database with information and algorithms for their implementation.

In many circumstances, existing methods and systems require the involvement of manual labor, are designed for processing texts with modern contexts and are not suitable in their pure form for processing documents with academic and literary words, characterized by an abundance of archival terms and a significant difference in the quality of detection results.

Modern technology is a powerful tool that facilitates the learning process. Spell checker is a great tool for correcting small mistakes that good spellers make and for common typos. At the moment, there are many types of tools that work for spell checking. Thanks to this introduction, a huge number of users were able to automatically correct texts without wasting their time on lengthy text verification. In modern society, this is one of the most important advantages of text programs. When writing various texts, such as scientific articles, news articles, the theoretical part of projects, legal documents, and other types of

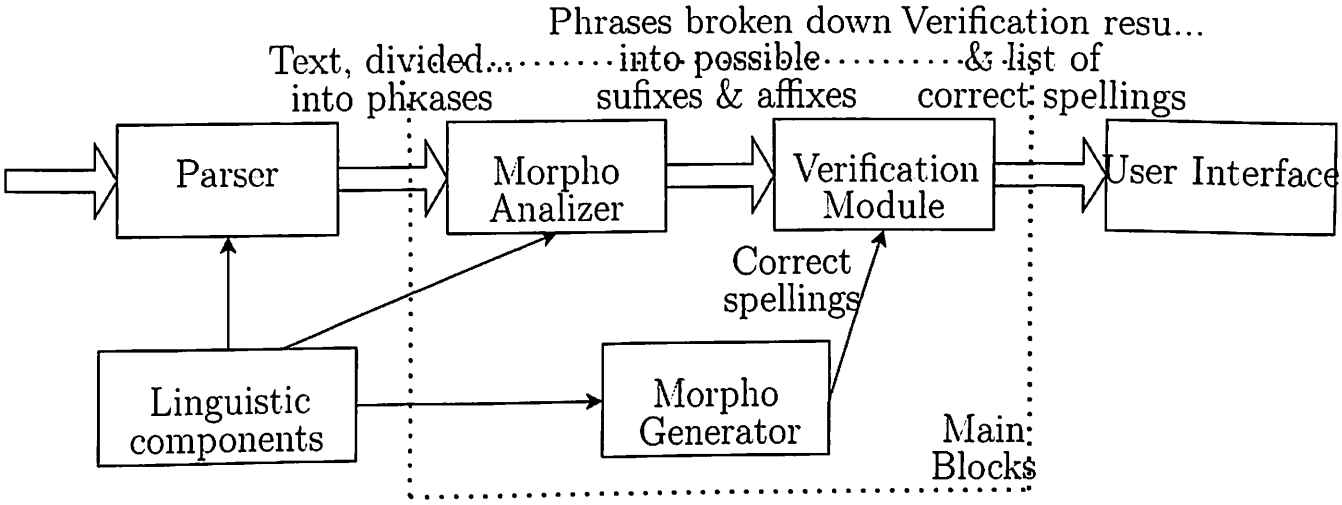


Figure 1.1: Scheme of interaction between the components of the architecture of the spelling verification system

texts, it is often necessary to check it for spelling or punctuation errors. And this work is designed to correct a word unknown to him by clicking on the underlined word or areas of this word and the application will automatically suggest options for correcting it.

Automatic spelling correction is one of the most popular and complex issues in the field of automatic natural language processing. There is still no universal answer for it, however, throughout its history, spelling correction has reflected the actual problems of applied linguistics and absorbed the latest computational methods.

The development of this learning is determined by two main technological needs: the first is the improvement of automatic processing of text that has undergone proofreading (methods for extracting data, optimization of search results) and the second is the convenience of typing for the user.

The relevance of the architecture of the verification system and the correction of spelling is explained by the requirement to use an approach to problem solving such as spell checking with recognition of the work and functioning of the linguistic parts of the language.

The main systems use an information base. A structure is proposed with the components of which influence each other according to the scheme shown in Figure 1.1. Figure 1.1 shows the storage and processing of a database for texts with linguistic components. The advantage of implementing a database over information is that both the user and the developer can fully work with the language components, have full access to information and data components.

The architecture shown above is used in many systems for analyzing and developing an application for text normalization, which is represented by the following steps:

- The first stage is functional parsing, which is responsible for breaking up the text into phrases. At the initial stage, linguistic components are formulated to obtain data on possible suffixes and affixes, which are written separately. Due to the fact that each part of the word is considered by the parser as affixes and the block stores information about separately written affixes, it can catch errors when these particles are misused by the normalization module. As a result, one can see the difference between the proposed parsing for normalization and the classification parsing carried out in existing systems.

- morpho analyzer, which is used to analyze each phrase. The output is a list with a possible phrase scheme divided into stems and affixes. It works according to the developed unified algorithm, in connection with this it is not tied to any particular language. All-linguistic data obtained at this stage from linguistic components is developed for the studied language. At the analysis stage, a stemming algorithm is implemented, which differs from the existing cases of working with words written separately and with long sequences of texts consisting of an unlimited number of terms and elements.

- Generator, the purpose of which is to generate all the correct spellings for given stems based on linguistic spellings.

1.2 Aims and Objectives

Despite the advancement of natural language, many new technologies are being tested and used in languages like English. The Kazakh language is an agglutinative language and therefore has structures that distinguish it from other languages like English, Russian when analyzed by morphology. In a such languages words are formed by combining root words and morphemes. There are roots and several suffixes and affixes, when they are combined, the word modifies or extends its meaning. Kazakh word formation uses a number of phonetic harmony rules. Different languages have different semantic and grammatical features, so often

algorithms that are successfully used to process one language perform very poorly in another language. Because of this, the morphological analysis of the Kazakh language was not considered sufficiently as English and several promising algorithms were not applied. Currently the most used in the industry and showing good results is the work of norvig, as well as the open source JamSpell. These programs are written based on the ngram model and have their own advantages. But the n-gram model has not been used in Kazakh studies, despite the popularity of the n-gram model in languages such as English. However, a full-fledged study that would give a detailed analysis of text correction for the Kazakh language does not exist.

Unfortunately, the problems of text processing and their solution in the Kazakh language are underdeveloped, which brings an obstacle in the development of artificial intelligence, and all this is connected:

1) with the specifics of the Kazakh language as an agglutinative language with a complex and rich morphology;

2) with a lack of resources on the Internet to study Kazakh language in the field of information technology.

With the increase in information in the Kazakh language and the use in electronic portals, surveys, the normalization and classification of texts in practice are very relevant.

The main goal of the research work is to develop an application with a system for automatically detecting and correcting errors received when recognizing documents in electronic form on various topics.

This work will consider the problem "Difficulties that Kazakh speaking learners face in academic writing». And this work will work with data from certain resources with academic lexicons. This study differs from the other studies that have reviewed candidate words for all misspelled words in order to correct misspelled words in Kazakh. The aim of project is built one of the first morphological disambiguate for Kazakh language that can be used to generate and segment word forms. Research is examining whether the n gram model for the Kazakh language will work. And it will also consider the analysis of the difference between the work of the already existing spell correction algorithm like symspell algorithm and the n gram model.

1.3 Thesis Outline

The dissertation work includes an introduction, six chapters including a conclusion, a list of references and an abstract in three languages.

The first chapter is an Introduction. In this chapter, the importance and relevance of the dissertation topic are substantiated, the goals of the dissertation work and the tasks to be solved are formulated, scientific novelty of the work, as well as its practical significance. A summary of the work by chapters is given.

The Chapter 2 provides an analytical review of the subject area and existing systems for spelling correction, determines their advantages and disadvantages for text recognition. It also reveals the need to correct the recognition errors allowed, provides the types of classification and analysis of existing approaches to correction, specifies the requirements for the system being developed.

The chapter 2 contains a description of the methods used and the developed method for automatic correction of recognition errors based on normalization. As well as related work with the n gram model and edit distance algorithm that have been developed in other languages. Related works about spell correction in Kazakh language was also considered.

The chapter 3 describes the methods and architecture. It shows the stages of implementation of the text recognition system, determines the order of the system, describes the algorithm for finding and correcting words for bigrams and trigrams. This chapter provides information about SymSpell, which has been considered for analysis with an ngram model.

The chapter 4 provides the data and datasets that were collected to test the algorithm.

The chapter 5 provides information about the conditions and procedure for testing the developed system, describes the criteria for assessing the quality of recognition. The results of work for each algorithm are shown, and which algorithm is suitable and has advantages over others is analyzed. The results of an experimental evaluation of the proposed correction method are presented and a discussion was shown as an example.

In Conclusion, the results of the work are summed up, the main results of the research, problems with solutions and ways of further development of scientific research are given.

The following tasks were established and solved in the dissertation study to reach this goal:

The quality of the results of existing optical recognition systems is compared, and the main forms of text recognition errors are classified. An examination of current methods for addressing recognition errors. Using a refreshed corpus, develop a mechanism for automatically detecting and correcting document recognition mistakes. Using the algorithms and methodologies provided in the study, designing phases of morphology and constructing a text recognition system. Using Python kivy, a prototype of the normalization system was developed, confirming the method's functionality.

Chapter 2

LITERATURE REVIEW

There have been many efforts for natural language processing. The constant development of word processing has now positively influenced the increase in the use of text recognition systems. This section presents important milestones in the history of the development of the text verification and classification system.

The formulation of the correction and processing of documents consists in the search for the stem of the word among the keywords of the texts, which makes it possible to select the appropriate data as desired. One of the main processes in NLP systems, such as automatic search, machine translation is used to reduce a word to its original stem. Various scientists and scientific groups have analyzed and considered different approaches to language normalization. There are a large number of works in which attempts were made to study all the attributes and in each generation was processed in one form or another.

2.1 Basic representation of the solution to the error correction problem

So, in the 1960s, in the world of information, there was a demand for the processing of texts and language obtained using the optical recognition technique. After discovering the usefulness and benefits of such processing even in the initial stages with easy methods, researchers began to use spelling correction in the wide spectrum area as well. And by the 1970s, researchers began to use modules for automatic correction of machine codes and for training in machine translations. Over the next 10 years, they were added to solve problems such as recogniz-

ing handwritten documents, as well as using these methods in speech synthesis. Processed modules with algorithms for automatically processing steel will be supplemented for emails when finding and correcting spelling. With the emergence of a large number of areas of application, formed for teaching, methodologies for processing text in different languages with morphological and syntactic parsing, semantic homonymy, searches, included a typo correction module. The year 2000 was a year of new information technology needs related to the mobile device and predictive text input when entering texts. In this way, such a flow of information provoked the emergence of independent applications for correcting spelling and grammar. At the moment, the world has changed a lot and the models that are currently being used are undoubtedly several steps higher than in the initial stages, as they are trained on a larger amount of data.

According to Mitton, the words with morphological characteristics and lemmas are divided into two groups. [23] First, dictionary words are normative words of the language included in the dictionary. Are considered as the main contenders for the amendment when correcting errors;

The second is non-dictionary. This word differs from the first one in that this option is not included in the dictionary. Their subparts make up words out of-vocabulary for a normative word, such as neologism, occasionalism and slang. According to Kukich's work, there are two main errata namely errors which are divided into: [14]

1) non-word errors (non-dictionary typos) - are words that cannot be found in the dictionary.

2) real-word errors (dictionary typos) - words that can be found in the dictionary and users misspell them.

In the works of Shavrin, Sorokin, it is precisely shown that, depending on the source of documents, the distribution of types of errors also changes. [31] Errors in optical character recognition and any other technical part of the source may be present even in the most "reliable" publications in terms of proofreading.

2.1.1 Early experiences with automatic correction: 1960-1980

The beginning of the correction began with Johannes Gutenberg in 1460, who made the first documented misprint in history. Finding errors in printed editions and correcting them in subsequent editions has become one way for readers to check texts. With the development of technology has changed the work of publishers. The first attempts to correct errors in documents date back to the early 1960s. It was found that the correction of these errors is closely related to the text recognition feature and their properties. Optical character recognition developed in parallel and did not have the required level of quality - the worse the quality of text recognition turned out to be, the lower the quality of error detection was Hanson in 1976.[11] The scientists had problems although the recognition of errors in characters was 99%,the recognition of errors in words was much less.

During this period, despite the development of text recognition, it continued to be dependent without automatic systems. Verification of texts and their processing has become one of the key points in working with data, as it has begun to be relevant for machine translation and applications with voice input and output. The researchers divided models and algorithms into two types, namely error detection and error correction.

The main method at the initial stage of the 80s was the n-gram method. In a dictionary or database of documents, the probabilities of alphabetic n-grams, for example, two or three alphabetic ones, were considered, and non-dictionary words were reorganized into dictionary ones, according to these probabilities and the smallest distance that was calculated with the Levenshtein algorithm.[27]

Over time, correcting phonetic restrictions on the beginning and end of a word, adding combinations of vowels and consonants, supplementing the gram model with an algorithm that will take into account those specific to each particular language, at the end of this stage, the accuracy of correcting English words was increased to 98%. The researchers used a sample of 8,000 six-letter words, with one error per word.

As a result of these experiments, in the 1970s, the first systems appeared that were able to correct errors. But despite this, one of the main problems was vocabulary errors and errors that were uncorrectable without regard to context.

2.1.2 From rule system to machine learning

Adding a constraint on syllable structure and not working with a dictionary meant that instead of detecting typos, these methods created new errors. And after the 1970s, researchers decided to use a dictionary that was built with a separate language model. The new system also brought with it new challenges, now the researchers had to create a dictionary instead of a model and decide whether a simple check of the dictionary could solve the problems of identifying misspelled words. By the beginning of the 1990s. scientists began to adhere to the opinion that the dictionary should contain only the most frequent lexemes. Three types of misprints were identified: typographical, phonetic and cognitive.

In the 1980s and 1990s, models worked to implement the model in correcting only one type of typographical error, with the exception of Moore's work, in which two candidate search models are formed - for phonetic and typographical typographical errors. [33] The n gram model that showed good results was supplemented with new models to get better results. And also models such as the Levenshtein algorithm began to select candidates using hidden Markov models.

With the help of a hidden Markov model, it was possible to make a classifier of proper names for five and six languages, also get fruitful results. Spell checking significantly improved the quality with optical pattern recognition,

With this progress in the 1980s, neural networks trained with back propagation began to be applied for the first time. The neural network methodology made it possible to select a set of features that determines the most correct candidate. The results obtained with the use of neural networks have given a boost to their use in other areas of applied linguistics, and they have been used for processing handwritten texts by Burr, for speech synthesis, and for optical character recognition with auto-correction of addresses in mail.[5]

2.1.3 Transition to context models

By the end of the 1980s - the beginning of the 1990s. the programs in which the developers implemented spelling corrections were not as perfect as the researchers wanted them to be. Despite the successful results of algorithms for correcting isolated words, it did not meet the following criteria:

1. a large number of lexicons (more than 150,000);

2. independence from word length (more vocabulary candidates in short words);

3. work speed

4. online bug fixes when pressing the keyboard;

5. Possibility of selection among several candidates.

Methodology .	500 word	1200 word	1900+ word
Minimum Edit Distanse	65%	62.5%	60%
Similarity Key	80%	77.4%	74%
Simple N-gram Vector Distance	77%	75%	75%
SVD N-gram Vector Distance	81.8%	76%	74.3%
Probabilistic	–	78%	–
Neural Net with Back-Propagation Classifier	75%	75%	75%

Table 2.1: Accuracy of some context-independent methods

Table 2.1 shows the accuracy of different algorithms for spelling correction in the early 1990s, although not all of them are directly comparable.

The results show that the accuracy of none of the methods is higher than 80%. For the test, three options were developed with the amount of the database, as you can see the difference with large and small offers. All sentences were generated by optical recognition. With the indicators of these methods, it is possible to find out the need to improve the probabilistic selection model. In this case, you need to take into account the fact that this cannot be done without the use of context models. This is the time to move on to contextual models.

In different contexts, such as morphological, syntactic, semantic, the difference in contextual methods was taken into account. In languages with exact word order and structure, n-grams of word forms were effective, but for languages such as Russian and Kazakh, which have a rich morphological context, this was not the best option.

One of the most successful programs at this stage was CLAWS (Constituent Likelihood Automatic Word-tagging System), which connected three types of analysis: dictionary, morphological and syntactic. The calculation of the probabilities of trigrams in morphology, as well as bigrams in syntactic analysis, included a manual database of 1 million dictionaries.[8] Each dictionary word received two levels of markup - morphological based on the dictionary and syntactic based on

bigrams, and each non-dictionary word - morphological and syntactic based on n-grams.[15] If the gram model was insufficient for a complete adjustment, then the models were added according to the contexts of morphology or syntactic. As a result, the label assignment accuracy reached up to 96%.

2.1.4 Actual development of the industry at the present time

Since the mid 2000s. automatic error correction has risen to a new level. With greater access to corpora containing more than 100 million words, model training has changed. Also, due to the high demand for electronic documents and searches, programs to correct typos have also become much more in demand. Each personal computer has an additional correction module, such as the spell checker in "Microsoft Word", and the popular "aspell" and "hunspell" used in UNIX systems. And also Yandex, which is used by the Russian-speaking community in Kazakhstan.[1] Such programs are designed primarily to detect typos, while automatic correction is carried out mainly in web interfaces: Google by Norvig necessarily includes spelling correction.[21]

Table 2.2: A recall of correction of errors

	yspell	aspell	Google	MS Word	Yandex
Recall	0,862	0,694	0,765	0,709	0,829

The area of programs responsible for predictive text input has been implemented. And also the accelerated process systems are used in devices, where in the process the module itself suggests options for the endings of words and phrases, checking with a dictionary, and entering the correct conclusions of the results.

However, for the Kazakh language, an important and main disadvantage is the inability to implement existing algorithms to cope with the developed morphology. It is worth noting that for languages with agglutinative characteristics and properties, the latest versions of the T9 system provide for the division of the word into a root and quasi-morphemes. The newest programs include working with a large dictionary, and also have the ability to learn for the user: adding new words, taking into account typed text.

2.2 Overview of existing methods:

Consider some existing algorithms for solving problems about the processing and normalization of words in natural languages. Most of the algorithms used implement lemmatization, that is, translate into normal form using stemming and various add-ons. Consider stemmers and the three most popular algorithms for finding and correcting errors in text, based on different principles and allowing word processing.

2.2.1 Stemming algorithm

Stemming is a part of word normalization that provides data pre processing. Each language has multiple variants of a single word, and these variants raise questions when training a module and predicting machine learning. Stemming is engaged in filtering such texts and converting them into serial data. The Stemming algorithm removes word suffix endings so that the part that is chosen is the same for all grammatical types of that word. The NLTK library has several classes for performing stemming on phrases. The most widely known stemming algorithm is Porter stemmer.[16]

Stemming does not use stem bases, but instead removes suffixes, which makes it work quickly. The idea behind Porter's stemmer is that there are a limited number of word-building suffix forms, and word stemming occurs without using any stem bases: only a set of existing suffixes and manually set rules.

At each step in 2.1, the suffix is removed and the part that was left to check will be checked by the rules of the language itself. If the resulting word satisfies all the corresponding rules, then it will execute the next step. If not, then the model looks for another suffix to cut off.

Porter's stemmer does not use any dictionaries and stem bases, which is a plus for speed and a range of applications (it copes well with non-existent words) and at the same time a minus in terms of the accuracy of stem selection.

2.2.2 'Naive Bayes classification

The Bayesian classification approach is based on the theorem that if the distribution densities of each class are known, the desired algorithm may be expressed in

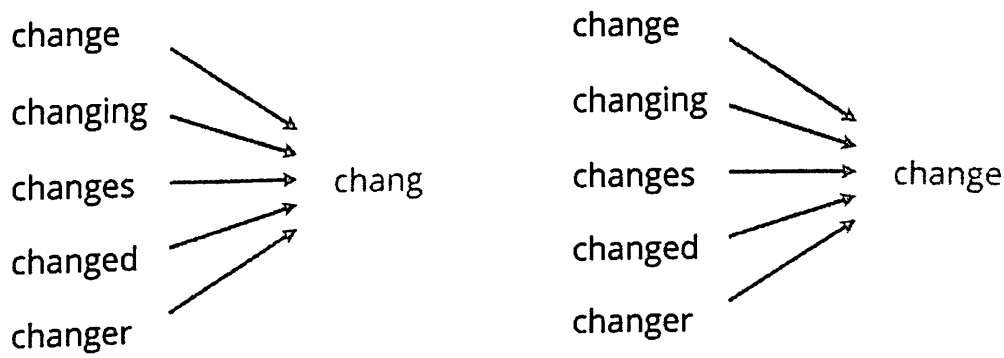


Figure 2.1: Stemming vs Lemmatization

an explicit analytical form. Furthermore, this approach is optimum, meaning it has the lowest possible error rate.

The Bayes formula is used to create Bayesian classifiers:

$$P(y|x) = P(y)\frac{P(x|y)}{P(x)} \tag{2.1}$$

where P(y|x) is the a posteriori probability of a given class, for a given feature value x, P(y) is the a priori probability of a given class, P(y) is the a priori probability of a given class, P(x|y) is the likelihood, i.e. the possibility of a selected feature value for a given class. We can disregard the calculation of the value of P, because the classification problem is being solved rather than the straight calculation of P(x). The value of P(x) has no effect on the value of y, as shown in formula 2.1, hence there is no need to calculate it.[22]

The class distribution densities are generally unknown in practice. The training sample must be used to estimate (recover) them. As a result, the Bayesian approach is no longer ideal, because the density of a sample can only be restored with some inaccuracy. The fewer the sample, the more likely it is to match the distribution to specific data and face the overfitting effect.

Yag shows advantages of the Bayesian classifier:

- works well in practice when the data are of a probabilistic nature;
- simple implementation;

- speed of work;
- separates objects by fairly simple but nontrivial separating surfaces (in the case of normal distributions).

It also has disadvantages: If a categorical property in the testing data set has a value that was not present in the training data set, the model will give it a zero probability and will be unable to generate a prediction.[12] "Zero frequency" is the term for this occurrence. Smoothing can be used to solve this issue. Laplace smoothing is one of the most basic procedures.

The bayes algorithm is a decent classifier, but the predicted odds aren't always correct. As a result, you shouldn't put too much faith in the predict proba method's output. Another bayes algorithm flaw is the assumption of feature independence. Sets of entirely independent traits are relatively rare in life.

Consider the naive Bayes classifier construction procedure. All features are considered to be independent and distributed according to the normal distribution law in this algorithm.

The mathematical expectation and standard deviation of each feature in each class are calculated during the model's training stage. The following steps comprise the stage of obtaining a prediction for object x:

a. for all classes $y \in$Y, steps b. – d. are performed;

b. using the traditional definition of probability, the a priori probability of the existence of a class is calculated;

c. Under the condition of class y, the total a posteriori probability of features x is determined, i.e. the likelihood of acquiring x under a normal distribution with mean and standard deviation calculated during the model training stage;

d. The class that has the highest posterior probability value is returned.

The scikit-learn library can be used. According to J.Chen and Abdalhkim , library offers three different types of Naive Bayes models:[17] [2]

Gaussian distribution (normal distribution). This model is utilized when dealing with continuous attributes and maintains that the feature values follow a normal distribution.

Polynomial (multinomial distribution). When there are separate features, this is used. In a pattern recognition problem, for example, features can display how several times each word appears in a text.

Bernoulli's formula (Bernoulli distribution). It's employed when dealing with binary identified (there is only two values: 0 and 1). A binary feature determines the existence (1) or absence (0) of a certain word in the text when utilizing the "bag of words" approach to classify texts.

2.2.3 K-nearest neighbor method

The compactness hypothesis states that related things are more likely to be in the same class than in distinct ones. This means that the class border is relatively simple, and the categories form compactly defined regions in the object space.In metric algorithms, the item being categorized is described straightly by the vector of distances to other items in the training examples, rather than by a set of features. In such circumstances, featureless recognition is also used.

It is considerably helpful in measuring the similarity of sentences, chemical formulas, and amino acid patterns directly than it is to go to feature descriptions. It is critical in expert systems to not only models can be classified but also to provide the user with an explanation of the recommended classification. Such explanations appear extremely logical in the nearest neighbor method: "The object x is allocated to class C since a close object in the training set belonged to the same class." Experts in a variety of fields are familiar with "precedent" logic (medicine, geology, jurisprudence).

The k Nearest Neighbors (kNN) algorithm is a widely used classification algorithm for a variety of machine learning issues. This, along with the decision tree, is among the most easy - to - understand classification methods. [36] When the procedure is used for classification, the item is assigned to the most common class among the k neighbors of this element, all of whose classes are known.

When the procedure is used for regression, the object is given the average value of the k objects that are closest to it and whose values are already known.

A green circle test pattern is shown in Figure 2.2. The blue squares will be classified as class 1 and the red triangles as class 2. Class 1 or 2 should be assigned to the green circle. If the region under consideration is a small circle, the object is categorised as class 2 since there are two triangles and only one square inside the circle. Because there are three squares inside the circle instead of two triangles, a huge circle (with a dotted line) will be classified as first class.

The parameter k must be chosen carefully in order to produce accurate classi-

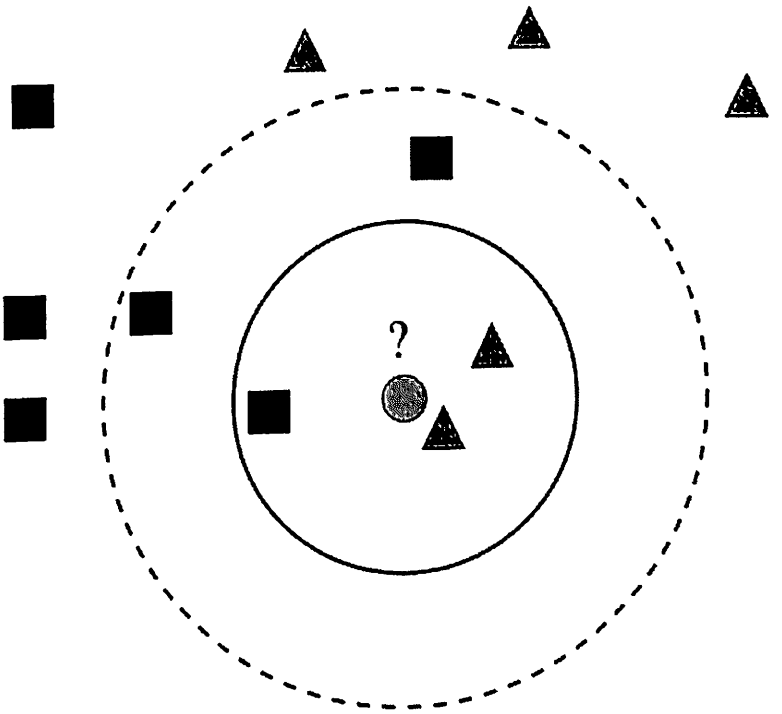


Figure 2.2: An example of k-nearest neighbor classification

fication results. When a classification judgment is made based on a small number of examples and has low relevance, an overfitting effect develops if the parameter value is tiny. Overfitting in decision trees, where there are numerous rules for a small number of samples, is analogous to this. If you set k=1, the algorithm will simply assign the class label of the nearest item to any new observation.

Furthermore, it should be noted that using lower k values increases the effect of noise on classification results, since minor changes in the data result in big changes in the classification results. At the same time, though, class divisions become increasingly apparent (the class take advantages with a large score through voting).

On the other hand, if the parameter's value is too large, the categorization process includes many objects from various classes. This classification turns out to be overly broad and inaccurately reflects the data set's local characteristics. As a result, the choice of parameter k is a trade-off between model accuracy and generalization ability.

2.2.4 Edit Distance algorithm

The concept of edit distance is used in information theory and computational linguistics (a metric that measures the difference between two character sequences).

The Levenshtein technique is commonly used to compute such edit distances. It is the smallest number of character deletions, insertions, and replacements required to convert one string to another. [26]

The Levenshtein distance is a number that tells you how different two strings are. The higher the number, the more different the two strings are.

According to work Iskander we can take one of examples: the Levenshtein distance between "kitten" and "sitting" is 3 since, at a minimum, 3 edits are required to change one into the other.[35]

kitten → sitten (substitution of "s" for "k")

sitten → sittin (substitution of "i" for "e")

sittin → sitting (insertion of "g" at the end).

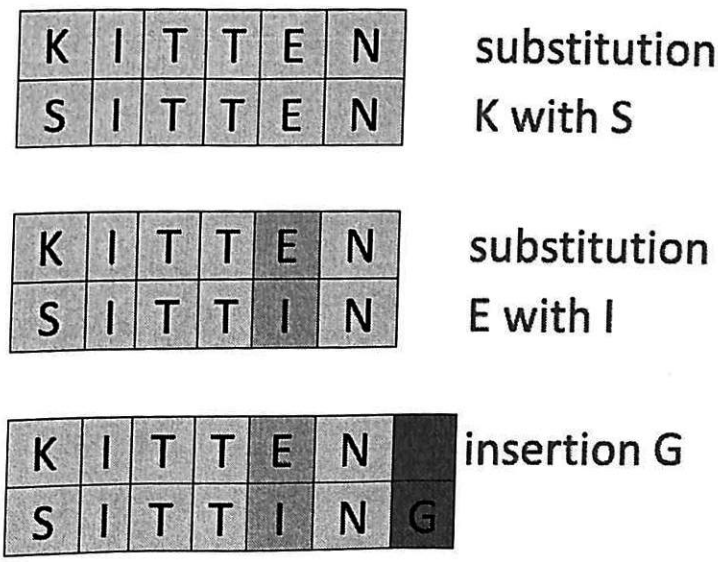


Figure 2.3: Levenshtein distance

The weight of the operation (insert, remove, replace) varies according to the type of operation and the amount of content to be processed, representing the varied chances of a text recognition error:

- cost(α, β)
- cost(E, β)
- cost(α, $\mathcal{E}$)

The complexity of the operation (insert, remove, replace) varies according to the type of operation and the amount of content to be processed, representing the varied chances of a text recognition error:

Algorithm 1 Computing Levenshtein edit distance

$D(i,j) = \min\{D(i-1,j-1) + \gamma(A\langle i\rangle \to B\langle j\rangle), \quad D(i-1,j) + \gamma(A\langle i\rangle \to \Lambda), \quad D(i,j-1) + \gamma(\Lambda \to B\langle j\rangle)\}$

The goal of approximation string matching is to locate matches for short strings in a large number of lengthier texts in scenarios where only minor changes are expected. The brief strings could, for example, come from a dictionary. One of the strings is usually short, and the other can be any length. Spell checkers, correction systems for optical character recognition, and software to help natural language translation based on translation memory are just a few examples.[19]

Although the Levenshtein distance can be computed between two lengthier strings, the cost of doing so is roughly proportional to the product of the two string lengths, making it impractical. As a result, when utilized to aid in fuzzy string searching in applications like record linkage, the compared strings are typically short to aid in comparison speed.

Two ideas underpin the suggested algorithm was showed by Yi Mar Myint. [18]

1. The range of filled cells in a row is dynamically decided based on the previous row's values, resulting in a more thorough cutoff of optional calculations.

2. To the initial matrix being used to estimate the Levenshtein distance, an additional matrix is added. The necessary features are kept in this scenario, as demonstrated below, for recurring filling of this matrix and breaking out unnecessary calculations in it. This results in a more comprehensive filtering of calculations that have no bearing on the final outcome.

2.3 Related Works

2.3.1 Text processing for Kazakh language

Consider the work that examines the morpheme architecture in the Kazakh language corpus, as well as the extraction of stems and affix segmentation by Altenbek. [7] Inflectional affixes are first segmented using a finite-state machine. Four sorts of endings are given as inflectional affixes in the work of author Kessikbayeva: plural, possessive, case, and personal.[10] Works can be credited in the field of morphological analysis of the Kazakh language. A morphological analyzer

is developed with end-state tools using a two-level morphological approach, and a rule-based realization of the morphological tester is described in the same work. The work "Rule-Based Machine Translation From Kazakh To Turkish" compares and contrasts the morphological characteristics of the two languages.[25] Ontology was compared, a uniform system of morphological feature symbols was constructed, and the morphological principles of the Kazakh and Turkish languages were represented using a new set of symbols.

The universal morphological analysis algorithm is used to create the unified morphological analyzer. The paper of Mukhanova, [9] proposes a method for standardizing the main forms of Kazakh endings. The aforementioned works primarily explored a certain (limited) class of Kazakh language endings, although complicated forms of language change were not provided, relying on the analysis and generation of Kazakh language endings (suffixes and affixes), which does not cover the language's fullness. The authors of the study, D.R. Rakhimova and A.O. Turganbaev, provide a new method to the classification of Kazakh language endings, completeness of coverage, and practical implementation, which will be employed in this study.[24]

There are many works performed on the general spelling correction problem, but we don't have many of works related to text normalization and processing. One of the first works with the result was "Spelling Correction for Kazakh" by Nazarbayev University students.[**aibek**] In this work, a spelling correction was developed with the help of the FSA and which had an accuracy of 83% in generating correct proposals, but also had a poor candidate correction rating and a shortened candidate correction lists. This work can be seen in two ways: first, one of the first morphological disambiguators for the Kazakh language was developed, which can be used to generate and segment word forms; and second, the text summarization tool was utilized to develop a spelling correction tool. The created FSA generates morpheme chains, which can then be combined with lexical roots to construct inflectional word forms. The Bayesian approach, which mixes error and source models, was employed to achieve this goal. The noisy channel techniques proposed by Church and Gale have been shown for these error models.The job of employing FSA has several flaws, most notably the relatively poor ranking of viable repairs and the reduction in catalogues of potential solutions.

And recent work by new NU students who have also worked with spell cor-

rection in text has shown the disadvantages of method FSA.To compare the outcomes, the author used both statistical machine translation (SMT) and neural machine translation (NMT) techniques. [37] The fundamental experiment was chosen to be SMT. A pretty conventional set of tools were utilized in this workflow. The goal was to create scalable NLP tools in Python, hence a phrase-based statistical machine translation system was created because phrase-based methods outperformed other methods.Also demonstrated was the use of a neural network model, namely the sequence-sequence model (Seq2Seq), to solve the problem of secondary normalization. In the test set, the statistical technique yielded 21.67 BLEU, while the sequence-sequence model yielded around 30 BLEU points. Both outcomes can be regarded averages on the whole. It's possible that sparse datasets are to blame for this behavior.

The algorithms shown above, such as the Naive Bayes theorem and the Levenshtein algorithm, have been developed and used in other languages. These algorithms were developed with students from Suleiman Demirel University.[4] They have created a database with matching words used in each region of Kazakhstan. The dictionary includes words, dialects and words that are often used in social networks, which often reduces the number of letters in a word. Here, two algorithms have been implemented and the best option, namely Bayes' theorem, has been shown with an accuracy of 99.19%. The advantage of the method proposed by the authors is that it can be iteratively improved by adding new rules/transitions to the normalization and new entries to the root lexicon.

2.3.2 Other language

1. Turkish language: Works for the processing and correction of texts in Turkish were considered, due to the similar structure and origin of agglutinative languages. Zemberek is among the most popular tools for Turkish developers. The work of Ebru Yılmaz Ince, show how work Zemberek, it defined Turkish vocabulary, including word roots and suffixes, according to Turkish language standards. [13] For Turkish, an agglutinative language, n-gram algorithms and edit distance were used to produce research, spell checking, and correction software. When using the morphological structure of Turkish words, nZemberek is utilized to check and rectify spelling problems.

According to the study, software that uses an n-gram based on word length

and an edit distance algorithm produces a spell checker with a 95% rate of success and suggests an appropriate spelling repair option with an 86% success rate. This study varies from others in that it considers the length (n) of the studied terms n, (n-1), and (n-2) when correcting Turkish misspelled words.[20] The word was corrected using grams. This optimizes the application for agglutinative languages as if it were a library with a direct cyclic word graph tree with sources such as a root, suffixes, and related special instances.

2. Russian language:

A language-independent typo correction model was created in research on the Russian language for social media texts by Alexey and Tatiana.[30] This was one of the final works to fix Russian literature. Another benefit is its adaptability, which allows you to insert arbitrary functions at the word and phrase level. They discovered that the quality of the language model is the most important aspect in the spelling corrector's efficacy after experimenting with several types of features.

According to the scheme of Alexey, another work for the Russian language gathered scores from many levels, including a vocabulary prototype, an n-gram language model, a weighted error method, and a morphological error model, and combined them into a single linear classifier.[29] The author of the paper utilized a reranking method commonly used in machine translation: the program first generated the n best lists of candidate sentences based on the simplest of the models, and then used a logistic regression classifier to rerank these hypotheses. As a result, after implementing the error model, morphological and semantic aspects did not improve any more.

Chapter 3

Research method

This section of the report will introduce methods for performing the spelling correction process. As noted in the literature review, there are several different approaches that can be used to automatically recognize and correct errors in text. As shown, the n-gram model has been and remains one of the best methods for working with text correction in every stage of the development of natural language processing, since it has the properties to work with and without a dictionary. In this case, the n-gram method is used for the Kazakh language, despite the statement about the non-working of this model for the Kazakh language in previous works [37]. This work will provide an analysis of the operation of the SymSpell algorithm and n-gram model.

3.1 Morphology Analyze

3.1.1 Classification of the endings of the Kazakh language

Affixes in the Kazakh language include suffixes and ends. The researcher U. Tukeyev guided the authors in developing a thorough system of Kazakh language endings. [34]

The structure of endings for the nominal roots of phrase and the verbal roots of the Kazakh language are two types of endings.
There are four different sorts of endings for the nominal bases of Kazakh words:

- Plural endings (K)
- case endings (C)

- possessive endings (T)
- personal endings (P) (J).

The study examines many alternatives for inserting different types of endings, as indicated by the formula 3.1:

$$^{n}P_{k} = \frac{n!}{(n-k)!} \tag{3.1}$$

There are a total of n objects. We'll choose k items from among them and rearrange them in any way imaginable (both the structure of the selected endings and their order change). Placements of n objects by k are the resultant combinations. The number of alternative locations as a consequence of the calculation is 64, although some of them are semantically wrong.

The following are semantically valid: KT, TC, CJ, KC, TJ, KJ, KTC, KTJ, TCJ, KCJ, KTCJ. In all, 4 places from one kind, 6 from two main types, 4 from three different types, and 1 from four categories are permissible. In words having nominal stems, there are a total of 15 types of acceptable placements.

The forms and amount of endings are chosen based on the sorts of endings received. So, there are 1213 ends for parts of speech with nominal stems (all plural versions are taken into consideration), and 432 endings for parts of speech with verbal stems: verbs - 432 , moods - 240, participles - 1582, pledges - 80, gerunds - 48. (They are not ends, but suffixes that construct forms of the verb, thus they will be treated as such.). There are 3565 endings in total.

3.1.2 Classification suffixes of word

In Kazakh, there are two sorts of suffixes:

- word-forming (TdJr)
- shaping (TrJr)

Suffixes that produce new words are known as word-forming suffixes (The word's definition shifts). For example, we can take word "write" - "жазу", "writer" - "жазу+шы", or "knowledge" - "ақыл", smart - "ақыл+ды".

Word forms are formed using formative suffixes. For instance: "бір" - "бір+інші" ("one" - "first"), "оқы" - "оқы+лды", ("read" - "was read").

Customized and grammatical suffixes are separated from formative suffixes. Suffixes of the degree of evaluation of adjectives; suffixes of ordinal labels of numbers; suffixes of verbs that constitute the voice, amplification, and negative types of the verb are examples of modified suffixes.

The affix database will not include negative verb suffixes or derived noun suffixes. For example: "бармаңыздар" - "бар + ма (verb) + ңыз (JJ) + дар (KJ)". However, instead of the word "бар", the basis of "бар + ма" should be displayed, because they have different meanings.

Grammatical suffixes are indicative of the verb's change. There are time suffixes, gerund suffixes, mood suffixes, participle suffixes.

There are a total of 26,526 potential suffixes.

Number	Suffixes and ends used in Kazakh nouns with verb stems	Examples
1	V + Ks1 + JJ	бар+а+мын, кел+іп+ті, сөйле+й+міз, көр+е+сіңдер, +ып+пыз
2	V + KS2 + V + JJ	бар+ғалы отыр+мын, жатыр+сыздар
3	V + E1 + JJ	жүр+ген+сың, жүр+етін+біз
4	V + E2 + TJ + CJ	қара+ры+ң+ды, қара+мағ+ым+ның
5	V + E1 + KJ + TJ + CJ	қара-ған-дар-ым-ның, қара-йтын-дар-ы-на
6	V + R2 + TJ + V + Ks1 + Sh3	жүр+гі+міз ке+ле+ді

Table 3.1: Table with examples of the formation of affixes in the words

Some of the categories listed in Table 3.1 are lexically and semantically correct, but others do not. Only the most commonly used affixes have been included to the database.

The order where such affixes are added to the root is shown below in order to highlight the right stem; however, derivational suffixes were not taken into account because they modify the roots of the word and the context of the definition.

Suffixes that form words are applied to nouns.

$$\begin{aligned} W &= N_i + E_i \\ &= A_i + S1_i \\ &= R_i + S2_i \\ &= V_i + S3_i + s4_i + E_i \\ S3 &= E_t, K_p \\ S4 &= K_{s1}, K_{S2}, Sh3, Sh4, E1, R2, R1, R2, R3 \end{aligned} \tag{3.2}$$

In 3.2, N is the set of root for nouns and pronouns; V is a set of verb root; R is a set of numerals; A is a set of adjectives; W is a set of words; S, S1, S2, S3, S4 is a set of formative suffixes; Kp - amplifying type of suffixes; Ks - gerund suffixes; E - set of endings; Et - voice suffixes.

3.2 Modelling

3.2.1 N-gram model algorithm

For text recognition systems with a big vocabulary, a language model is required. The n-gram statistical model is one of the most advantageous language models. People may usually use N-Gram in NLP to anticipate or evaluate if a sentence is plausible based on a corpus. N-Gram, on the other hand, can be used to calculate the difference between two strings. This is a frequent fuzzy matching technique. This essay will begin here and then demonstrate readers how N-Gram can be used in natural language processing in a variety of ways. [3] N-grams are a positive point of texts and language since language is not just a random set of words.

N-grams are the foundation of several natural language processing algorithms:

1. Language detection: techniques that use N-gram letters are more accurate (google langdetect);
2. Text generation: a pattern of N-grams in which the end of the i-th N-gram corresponds to the start of the i+1th N-gram. A syntactically connected text is known as an N-gram.

3. Search for semantic errors: when a term isn't used in a context, the likelihood of seeing an N-gram containing that word in that context is low (or 0), indicating a semantic error.

Every text element is divided into a group of n-grams. Developing a system based on n-grams necessitates first calculating the probability of error for each n-gram separately. The number of times it appears on incorrect words out of the amount of times it occurs on words in the text is the probability. It's also feasible that an n-gram (a sequence of n letters) is valid in one context but not in another.

As a result, this strategy necessitates a learning corpus. The invalid n-grams are only those created from the erroneous words.[32] An n-gram that appears on both the proper and incorrect forms of a word is not considered to be related to the error.

$$p = E/(E + V) \tag{3.3}$$

The probability of error (P) of an n-gram is given by the following formula 3.3 where: E is the number of times it has been classified as an invalid n-gram; V is the number of times it has been classified as a valid n-gram.

This is how the N-gram model works:

We go through a frame of N words in length and count the how often each combination (n-gram) occurs, according to the text which used train and validate the model. Similarly, while querying the model, we have to go through sentence frame and calculate the combination of all n-gram probabilities. The number of such n-grams in the experimental text is used to estimate the likelihood of seeing one.

The total of all n-grams of size n that compensate this sentence is nearly equal to the chance P(w1,..., wm) of approaching a sentence (w1,..., wm) of m words:

$$P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, \ldots, w_{i-1}) \approx \prod_{i=1}^{m} P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1})$$

The number of times this n-gram appears in comparison to the number of times the identical n-gram occurs without the last word determines the likelihood of each of the n-grams:

Firstly, i created dictionary of ngrams and the words that contain them. After that we should check word from dictionary. if the word exists in the dictionary,

$$P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1}) = \frac{\text{count}(w_{i-(n-1)}, \ldots, w_{i-1}, w_i)}{\text{count}(w_{i-(n-1)}, \ldots, w_{i-1})}$$

function lookup return True. And also with function ngrams given a word, return the set of unique ngrams in that word.

Listing 3.1: ngram

```
def lookup(self, word):
        return word in self.words
def ngrams(self, word):
        all_ngrams = set()
        for i in range(0, len(word) - self.ngram_size + 1):
            all_ngrams.add(word[i:i + self.ngram_size])
        return all_ngrams
```

Function "suggested_words" take a word, then return a list of possible corrections. After that sort by descending frequency. In this code ngram_size=2, len_variance=1.

Listing 3.2: ngram

```
def suggested_words(self, target_word, results=5):
    word_ranking = collections.defaultdict(int)
    possible_words = set()
    for ngram in self.ngrams(target_word):
        words = self.ngram_words[ngram]
        for word in words:
            if len(word) >= len(target_word) -
            self.len_variance and \ len(word) <=
            len(target_word) + self.len_variance:
                word_ranking[word] += 1
    ranked_word_pairs = sorted(
      word_ranking.items(),key=itemgetter(1),reverse=True)
```

3.2.2 Smoothing

In training, such a model is rarely employed in its basic state because of the following issue. If no n-grams were detected in the learning text, the entire phrase would be given a zero probability. Get one of the smoothing options to remedy this issue (smoothing).

This is the addition to choose one over the frequency of occurrence of all n-grams, or the usage of smaller n-grams in the exclusion of a higher-order n-gram in a more complex form.

Kneser–Ney smoothing is the most widely used smoothing technique.[6] However, it necessitates keeping additional information for each n-gram, and the benefit over simpler smoothing was not significant (at least in experiments with small models, up to 50 million n-grams). For the sake of simplicity, we'll treat the probability of any n-gram as the total of n-grams of all items, such as trigrams.

Now that we have a language model, we can choose among the options for repairing errors for which the language model will provide the best estimate, taking into account the context. In order to avoid a high number of false positives, we will also impose a minor penalty for modifying the original term in the estimate.

Changing this penalty allows you to vary the amount of false positives: in a text editor, for example, we can ignore the percentage of mistakes higher, and for automatic text correction.

The n-gram model is used in the proposed method:

Based on n-gram statistics and lexical resources, it can discover the correction ideas by assigning weights to a list of scope correction candidates.

We'll have to take a bit of probability mass from the more common occurrences and give it to the events we've never seen to keep a language model from assigning 0 probability to unseen events. Smoothing or discounting is the term for this change.

Smoothing is as simple as adding one to all the bigram counts before normalizing them into probabilities. All counts that were previously 0 will now be 1, counts of one will be 2, and so on.

Laplace smoothing is the name of the algorithm. One option to add-one smoothing is to shift a smaller portion of the probability quantity from observed to unknown events.

```
#If mispelled word is last word of a sencence take
probaility of previous word
    if cnt == len(sentence)-1:
        try:
        p2 = cf_biag[corrected[len
        (corrected)-1].lower()].
        prob(sug.lower())
        prob.append(p2)
        except:
        prob.append(0)
#If mispelled word is first word of a sencence take
probaility of next word
    elif cnt == 0:
        try:
            p1 = cf_biag[sug.lower()].prob
            (sentence[cnt+1].lower())
            prob.append(p1)
        except:
            prob.append(0)
```

3.2.3 SymSpell algorithm

SymsSpell is an algorithm for quickly locating all strings within a maximum edit distance out of a large array of strings. It could be used to correct spelling and search for fuzzy strings. SymSpell gets its speed from the Symmetric Delete spelling correction method and uses prefix indexing to keep its memory usage low. For a given Damerau-Levenshtein distance, the Symmetric Delete spelling correction technique decreases the difficulty of edit itemsets and vocabulary lookup.

The Damerau-Levenshtein distance is an additional operation to the Levenshtein distance method.[28] Transposition is the process of rearranging similar characters in different locations. The approach works well together with single-byte encodings, however in some languages, such as Unicode, text can only be described using variable-length encodings. He discovered that transpositions account for 4/5 of all typing errors.

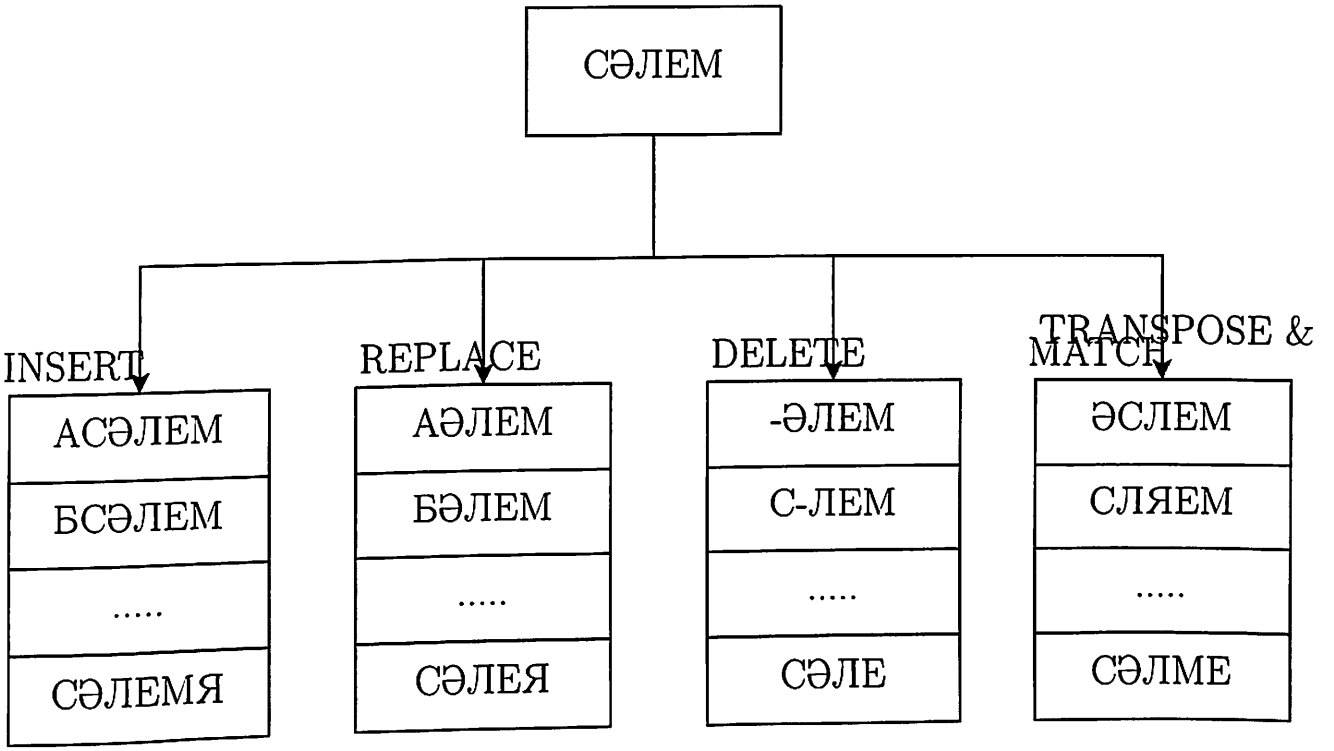


Figure 3.1: Scheme of interaction between the components of the architecture of the spelling verification system

An edit distance algorithm is a technique for quickly producing a second line from a single line. Most of the time, actions are indicated in the table. Letters from the alphabet of the language in which the word is to be located are used in insert operations in Table 3.2.

Table 3.2: Edit distance algorithm's operations

Operation	Description
D	Delete
I	Insert
R	Replace
T	Transpose
M	Match

For example, for the strings "СӘЛЕМ" and "САЕМ" you can build a transformation as shown in the Figure 3.1.

A technique for quickly creating a second phrase from a single line is called an editorial prescription. In most cases, actions are listed in the table. Insert procedures utilise letters from of the alphabet of the language under which the

term is to be found. The speed is due to the use of low-cost delete-only edit candidates and pre-calculation.

Within a maximum edit distance of 3, an average 5 letter word contains about 3 million possible spelling errors, however SymSpell only needs to generate 25 deletes to handle it all, including both pre-calculation and at lookup time.

The number of returned results can be controlled using the Verbosity parameter:

1. Top: The recommendation with the highest phrase frequency among the smallest edit distance suggestions discovered.
2. Closest: All recommendations with the shortest edit distance, ordered by term frequency.
3. All: All ideas within maxEditDistance, sorted by edit distance first, then term frequency.
4. The Maximum edit distance parameter specifies the maximum distance at which dictionary terms should be treated as suggestions.
5. The needed Word frequency dictionary could be produced from a large text corpus or directly received from data files (LoadDictionary) (CreateDictionary).

Tokens are created from the input string. The Symmetric Delete spelling correction technique is then used to generate individual token suggestions. Ideas are also reviewed for every bigram (a concatenated pair of consecutive tokens), but only if one of two consecutive tokens produces no options or the perfect solution has an edit distance > 1.

Listing 3.3: Edit Distance

```
def lookup_word(word, verbosity, max_edit_distance):
    suggestions = symspell.lookup(word, verbosity,
    max_edit_distance)
    words_list = [item.term for item in suggestions]
    if verbosity == Verbosity.CLOSEST:
        return words_list
```

```
    if len(words_list) > 0:
        return words_list[0]
    return None
```

Firstly, we input word to lookup. Then check the Verbosity mode: Verbosity.TOP, Verbosity.ALL or Verbosity.CLOSEST. After that take max edit distance with Leveshtein, If verbosity is defined as Verbosity.TOP and the input word has the most repeated word within max edit distance, the output will be verbose. If there are no terms in the dictionary matching input word within max edit distance. This can happen, for example, when you pass a word in a different language. It returns the most common word.

Listing 3.4: SymSpell

```
suggestions = symspell.lookup(word, verbosity,
max_edit_distance)
words_list = [item.term for item in suggestions]

if verbosity == Verbosity.CLOSEST:
    return words_list

if len(words_list) > 0:
    return words_list[0]
```

There are four different comparison pair types:

- dictionary entry==input entry,
- delete(dictionary entry,p1)==input entry
- dictionary entry==delete(input entry,p2)
- delete(dictionary entry,p1)==delete(input entry,p2)

Chapter 4

EXPERIMENT SETUP

Like most NLP models, machine language methods require data to produce results. To form an algorithm for this research, I first collected articles from news web pages like tengrinews.kz and texts on forums, to strictly controlled documents such as web pages, wiki materials.

The latter form the backbone of our corps. All these resources allow us to observe a wide variety of language works. The corpus consists of 50 to 100 pages of various articles and reports written by different authors.

Table 4.1: Number of inputs

Source	Number of posts	Number of pages	Number of Words
tengrinews.kz	18	84	184 555
kk.wikipedia.org	29	100	147 880
Total	47	184	332 435

The problem of preliminary text processing (text correction) using the Kazakh language is tackled utilizing a collection of algorithms and techniques (Table ??). This method accounts for changes in keyboard layout, different types of mistakes, and so on.

A news text corpus was created and processed to construct a language processing model, and it was derived from the websites of several electronic newspapers. As a result, the corpus is comprised of texts that have a high number of speeches and direct text, representing the characteristics of current language. The assembled body was processed automatically. The text was broken down into sentences, with sentences of five words or less eliminated, as well as text in

brackets. Punctuation marks have been eliminated and common abbreviations have been deciphered.

To test the created language model, a text corpus containing the material of the electronic newspaper site as "tengrinews.kz" was assembled.

Chapter 5

RESULTS AND DISCUSSION

5.1 Results

In this section, the results of the work performed will be presented and analyzed.

In this study, software was developed to check and correct Kazakh spelling. The software was built on the Python platform, and kivy was applied to create a mobile application. The accuracy level of a word is checked as the user types it in. If the word is spelt correctly, the user will be prompted "The spelling is right" and the word will be displayed. The level of correctness of a word is tested when it is entered by the user. If the spelling of the term is correct, the user is informed that "The spelling is correct."

Otherwise, after selecting the "CHECK" button, the user automatically corrects the word's problem and displays it on the screen. In 5.1 we can see one of example of errors and their corrections.

For spell checking and correction, N-gram and edit distance algorithms are utilized, using 3 and 2 as thresholds, respectively. The n-gram is calculated using the misspelled word length as n, (n-1) and (n-2). The nearest two distance distance editing method selected candidate words, which were added to the suggestion list.

The comparison of the method's results with expert estimations is the key criterion for evaluating the quality of automatic text classification algorithms. In this situation, the "ideal" algorithm is one in which the system's results are compatible with professional appraisers' opinions.Recall and accuracy are the most

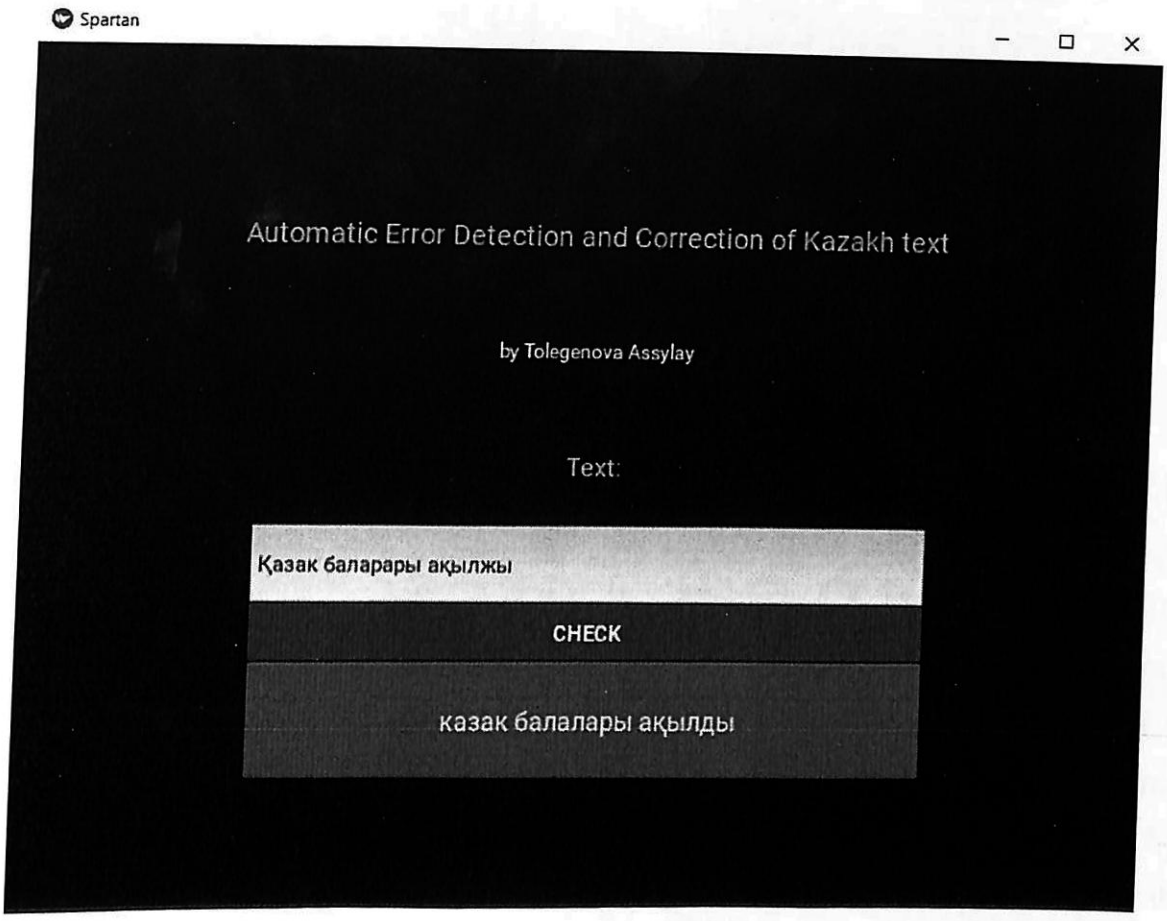


Figure 5.1: Automatic error correction application

commonly used quality measures.

Definition (completeness of document classification): The ratio of the number of documents correctly assigned (automatically) to the total number of documents associated to this item is used to calculate recall of document classification:

$$Recall = TruePositives/(TruePositives + FalseNegatives) \tag{5.1}$$

Definition (document categorization precision): Precision: The ratio of the number of documents accurately assigned (automatically) to the total number of documents assigned to a specific item is determined as follows:

$$Precision = TruePositives/(TruePositives + FalsePositives) \tag{5.2}$$

Methods	Error Rate	Fix Rate	Precision	Recall
N gram model	3%	89%	91%	82%
SymSpell	3,7%	71%	88%	70%
Bayes Theorem	3.80%	74.50%	98%	81.33%

Table 5.1: Results of algorithms

According to the results of the 5.1 we can see that the ngram model is leading in the secondary corrected version and also has fewer errors that were found when found using the algorithm. A fixed rate per gram model with 89% beats the symspell model by 18%. And also the errors that were after checking and correcting in the n gram model showed 3%, which is 1% less than it was in the Bayes theorem. But despite good results in secondary correction and fewer errors, Bayes' Theorem leads in algorithm accuracy with 98%, which is 7% better than the ngram model and 10% better than the symspell. If we look at and compare the completeness of the algorithm, then in this case the gram model and the Bayes theorem are not much different, while the symspell algorithm showed the worst result among the three algorithms. Since it only uses deletion, the percentage of completeness was also low.

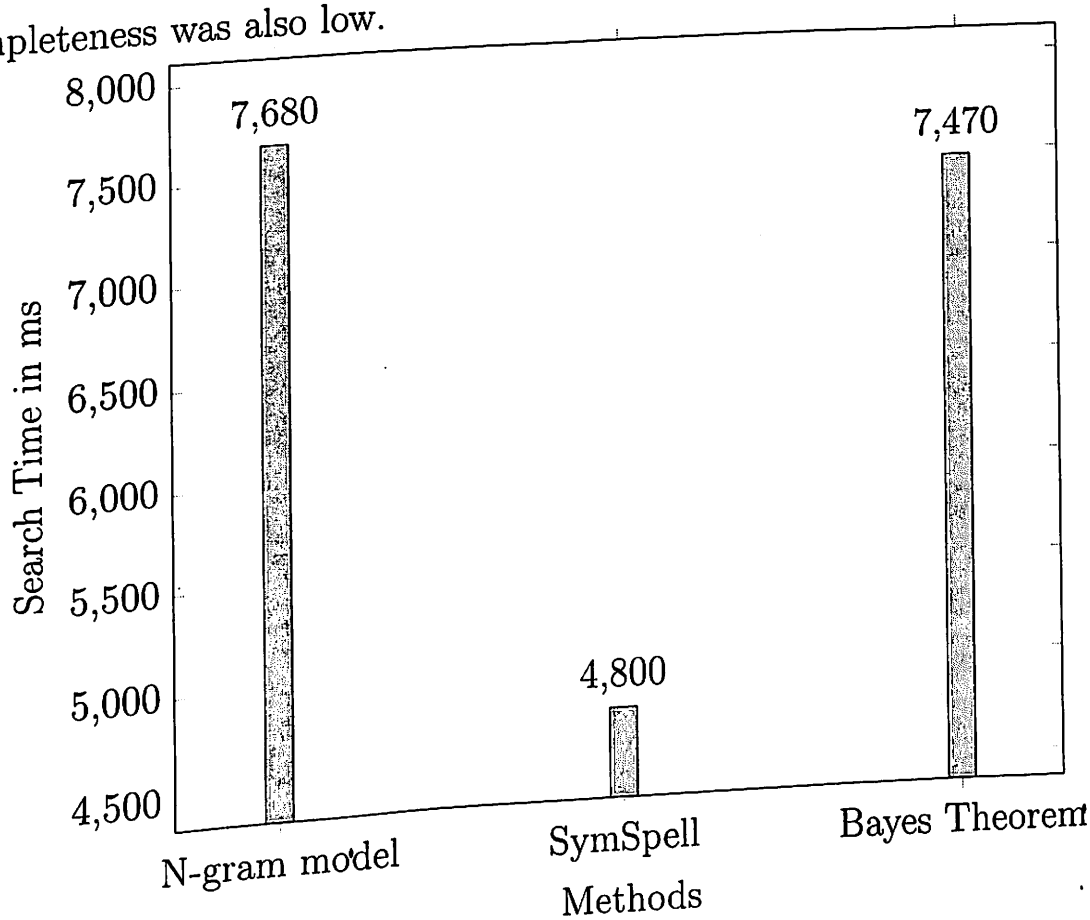


Figure 5.2: Tested search time for 1019180 words.

SymSpell tool is very fast. We can see from the chart that speed is vital when utilizing SymSpell. It is several orders of magnitude faster than several distance-testing tools, such as the BK-Tree, and several orders of magnitude faster than the ngram model approach. The time it takes SymSpell to look up a word grows slightly as the dictionary size and maximum edit distance grow.
In terms of search speed, it outperforms the other three algorithms. This comes at the expense of increased memory usage and precalculation time. If the precomputed data is serialized, the precalculation time occurs only once when program or server launch, or even only during program development.

5.2 Discussion

In this section of the report, the analysis and investigation of the model's performance will be shown.
Some tests are being carried out to assess automatic spelling in the following way: we are concerned in assessing the quality of the proposed sentence. In order to accomplish so, the whole list of Kazakh texts with frequent errors and news offered in Chapter 4 was assessed. To begin, use the bigram case to spell out all of the misspelled words from the list, then simply correct the first sentence. The proposed technique successfully corrected 354 misspelled words (98%) while failing to correct 6 misspelled terms (2%).
The system was then put through its paces again, this time with the addition of purposeful errors, which were split into numerous categories. It's a system for putting letters in words, deleting them, reordering them, and replacing them. It can also be identified by repeated characters that have been removed or added, as well as differences of one character or a difference test that involves substituting two consecutive letters.
A test set of 500 spelling errors was developed, with spell check correcting 435 words with errors (87%) and 65 words with errors (13%). This revealed low efficiency and performance when utilizing more than the standard, such as inserting or removing, replacing, or moving more than two letters. As stated in the 5.2, the spell check passed on 390 words (78%) and failed on 110 words (22%).

	Bigram	more than two letter(trigram)
Correct	435 (87%)	390 (78%)
Wrong	65 (13%)	110 (22%)

Table 5.2: Results of bigram and trigram

An analysis of the works on automatic detection of spelling errors leads to the conclusion that the following approaches exist today: N-gram, based on the use of valid letter combinations, and dictionary, based on the use of reference and frequency dictionaries. As shown above in this work, the n gram model was considered with symspell. In the experiments, the statistics of symbolic n-grams of length 3 was used, since it was verified that this length provides the maximum reliability of the classifier.

Statistical language models are inherently a type of model that assigns probabilities to sequences of words. The N-gram model, like many statistical models, depends heavily on the training corpus. As a result, probabilities often encode specific facts about a given academic building. In addition, the performance of the N-gram model depends on the change in the value of N. For this, smoothing was used. This means that any N-gram occurring enough times can have a reasonable estimate of its probability. But since any corpus is limited, some perfectly acceptable sequences of English words are bound to be missing from it.

Once tested, the idea of using an n-gram can see distinct benefits that can be applied to many problems such as speech recognition, translated word, word correction, prediction, and spelling correction. This statistical method does not require much knowledge of the language of the document. One of the main advantages of the n-gram is that it automatically captures the most common roots. N-gram can be used in two ways: without a dictionary or together with a dictionary, but for this, lemmitization must be used to determine the properties of the language. The N-gram model uses a dictionary that has been collected from different sources, this method is used to find exactly at which position in the wrong word the error occurs. If there is a way to change the wrong word so that it contains only the correct n-gram, that is, as a correction.

This approach has a low performance; yet, it is simple and does not require a dictionary; however, we cannot declare that it is an appropriate solution for the Kazakh language. The n-gram, on the other hand, is utilized in conjunction with the dictionary to establish the distance between words, but the words are always

double-checked. There are a variety of approaches to this, including a weighted examination of how many n-grams the misspelled word and the dictionary term share.

While the findings are impressive, it's worth noting that the two implementations use totally opposite prioritization models. ...

As a result of this, the N-gram matrix for any training corpora should have a significant number of cases of assumed "probability zero grams". Accordingly, using n gram, we proved that the proposed combined algorithm is efficient and in many cases outperforms other modern classification methods in practicality, even for agglutinative languages.

Rechecking the n-gram model was very helpful because all text paths that were similar to the original text were found. The main conclusion is that when using variations close to the lower limit, the graph will strictly display the relationship between words.

Also in the course of the study, it was found that nodes and edges, which are considered noise and which, as it was believed, introduce an incorrect classification of texts, on the contrary, reduce the size of graphics, which reduces the requirements for calculations and memory. Anti-aliasing was used to eliminate the limitations. For example, as a result of the analysis, it turned out that "yes or de" is a more common case, and the program checked the suffixes at the end of the letter. For example, in the Kazakh language, suffixes like ta-ta or those follow a root that ends in the letters "r, t, s, k" and changes at the end to "da or de, pa or pe". Such techniques and distinctions of the Kazakh language were not fully developed with the n gram model, since such features are not available for the English language that was previously used. And these suffix checks were checked through a dataset that consists of parsed dates.

When it comes to the time difference between ngram modeling and symspell, it's due to the edit distance algorithm's employment. To determine the distance limit, this algorithm first loaded a collection of misspelled words and then added a dictionary to it, which comprises the various Levenshtein distances as keys and the number of words with this distance from their proper spelling as values. When we compared the numbers, we discovered that the distance between most misspelled words is less than or equal to 2.

To calculate the DL distance between two strings and to establish the ideal trac-

ing, the SymSpell engineers created efficient caching and multi-core linear space algorithms (edit sequence). These algorithms reduce the amount of space required to solve issues, allowing them to solve far larger problems than previously conceivable.

Chapter 6

Conclusion

This article described the main approaches to solving the problem of correcting the text in the Kazakh language. In particular, the expansion of the editorial prescription with Kazakh letters and its application, the expansion of the database with Kazakh words, the analysis and comparison of gram and SymSpell, as well as with probability theory, changing the layout of the text, demonstrating an example on a flowchart and the results of experiments with Kazakh words and sentences written in python language.

In this work, a program for finding and correcting errors in the Kazakh language was implemented. An n-gram model was employed for text processing, which used n, (n-1), and (n-2) grams to correct the word, taking into account the length (n) of the studied words. From the revised training sentences, the computer determines the frequency (number of occurrences) of all words and bigrams (sequences of two words) case-insensitively. The system is built around a function that takes an incorrect sentence and returns a corrected version of it.

The system should scan the phrase for words that are not in the dictionary (the collection of unique words in the training set), and select the word in the dictionary with the shortest edit distance and highest bigram probability for each word that is not in the dictionary. Only the existing conditional probabilities were applied for the first and last words of the sentence. To avoid zero probability, smoothing techniques have been used.

Like many statistical models, the N-gram model is strongly reliant on the training corpus. As a result, probabilities are frequently used to encode unique information about a certain academic structure. Furthermore, the N-gram model's performance is affected by changes in N's value. The size of the dictionary has a

direct impact on the classifier's efficiency. The greater the vocabulary, the more time and computational resources are required.

Furthermore, having too many terms in a document collection might dramatically degrade the categorization quality. There may be a huge number of so-called noise features with poor classification performance among the terms.

To minimize the vocabulary and improve the effectiveness of the classifier, each phrase is first lemmatized, which involves converting the terms into dictionary form or stemming and cutting off the endings. However, even after getting all of the terms to the normalized form, the document's feature space may still be too vast.

I'd like to include edit distance in future work, which is a way for determining how similar two strings are by counting the least number of operations required to turn one string into another. A function that calculates all words with the smallest edit distance to the misspelled word and then adds it to the bigram probability reading process. As a result, the number of words found for further correction may be reduced. You can also create substitution rules, which are made up of ends that are added to the word's root. For instance, ('м', ''), ('ым', ''), ('им', ''). Because the correction time will not be dependent on the dataset, this will also help to shorten the correction time.

References

[1] Baytin A. "Correction of search requests in Yandex". In: *Rossiyskie internet-tekhnolog* (2008).

[2] Abdalhkim A.M. "Implementation of Naive Bayes algorithm for spell Correction in Web based learning for written English". In: *Seminar on electrical, informtics, and its education* (2011).

[3] Jean Hugues Chauchat Artur Silic. "N-Grams and Morphological Normalization in Text Classification: A Comparison on a Croatian-English Parallel Corpus". In: *Proceedings of the aritficial intelligence Portuguese conference on Progress in artificial intelligence* (2007). DOI: 10.1007/978-3-540-77002-2_56.

[4] Alina Amangeldiyeva Assina Abdussaitova. "Normalization of Kazakh Texts". In: *Student Research Workshop Associated with RANLP* (2019). DOI: 10.26615/issn.2603-2821.2019_001.

[5] Burr D.J. "Experiments with a connectionist text reader". In: *Proceedings of the First international conference on neural networks* 4 (1987), pp. 717–724.

[6] Hui Zhang David Chiang. "Kneser Ney Smoothing on Expected Counts". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* 1 (2014). DOI: 10.3115/v1/P14-1072.

[7] Xiaolong Wang G. Altenbek. "Kazakh Segmentation System of Inflectional Affixes". In: *CIPS-SIGHAN* (2010).

[8] Sampson G. Garside R. Leach G. "The computational analysis of English: A corpus-based approach." In: (1987).

[9] M. Mukhanova Gaukhar Slamova. "Text Normalization and Spelling Correction In Kazakh Language". In: *International scientific journal "Bulletin of Science* 39.6 (2021).

[10] Ilyas Çiçekli Gulshat Kessikbayeva. "A Rule Based Morphological Analyzer and A Morphological Disambiguator for Kazakh Language". In: *Linguistics and Literature Studies* 4 (2016), pp. 96–104. DOI: 10.13189/lls.2016.040111.

[11] Fisher Hanson A.R. Riseman E.M. "Context in word recognition". In: *Pattern Recognition* 8 (1976), pp. 35–45.

[12] Daocai Fu Huaixin Chen. "An Improved Naïve Bayes Classifier for Large Scale Text". In: *2nd International Conference on Artificial Intelligence: Technologies and Applications* 146 (2018).

[13] Ebru Yılmaz Ince. "Spell Checking and Error Correcting Application for Turkish". In: *International Journal of Information and Electronics Engineering* 7.2 (2017), pp. 68–71. DOI: 10.18178/IJIEE.2017.7.2.663.

[14] Kukich K. "A comparison of some novel and traditional lexical distance metrics for spelling correction". In: *Proceedings of the International neural network conference.* 2 (1990), pp. 309–313.

[15] Hugo Van Hamme Loris Pelemans Lyan Verwimp. "Expanding n gram training data for language models based on morpho-syntactic transformations". In: *IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)* (2015).

[16] Thekra Abbas Manhal Elias Polus. "Development for Performance of Porter Stemmer Algorithm". In: *Eastern-European Journal of Enterprise Technologies* (2021), p. 8. DOI: 10.15587/1729-4061.2021.225362.

[17] Muhammad Noman Muhammad Iqbal Malik Muneeb Abid. "Review of feature selection methods for text classification". In: *International Journal of Advanced Computer Research* 10 (2020).

[18] Yi Mar Myint. "Edit distance computation with minimum number of edit operations in database management system and information retrieval". In: *International Journal of Scientific and Research Publications* 9 (2019). DOI: 10.29322/9.09.2019.9385.

[19] Artur Niewiarowski. "Similarity detection based on document matrix model and edit distance algorithm". In: *Computer Assisted Methods in Engineering and Science* 26.3-4 (2019). DOI: 10.24423/277.

[20] Oflazer. "Error-tolerant Finite-state Recognition with Applications to Morphological Analysis and Spelling Correction". In: *Computational Linguistics* 22.1 (1996).

[21] Norvig P. "How to write a spelling corrector". In: (2010).

[22] M. S. Dhotre Pouria Kaviani. "Short Survey on Naive Bayes Algorithm". In: *International Journal of Advance Research in Computer Science and Management* 4 (2017).

[23] Mitton R. "Spelling checkers, spelling correctors, and the misspellings of poor spell". In: *Information Processing Management* 23.5 (1987), pp. 495–50.

[24] Turganbaeva A.O. Rakhimova D.R. "Normalization of kazakh language words". In: *Scientific and Technical Journal of Information Technologies, Mechanics and Optics* 20.4 (2020), pp. 545–551. DOI: 10.17586/2226-1494-2020-20-4-545-551.

[25] I. Salimzianov S. Bayatli S. Kurnaz. "Rule-Based Machine Translation From Kazakh To Turkish". In: *Proceedings Of The 21st Annual Conference Of The European Association For Machine Translation* (2018), pp. 49–58.

[26] Jaiteg Singh Shama Rani. "Enhancing Levenshtein's Edit Distance Algorithm for Evaluating Document Similarity". In: *Communications in Computer and Information Science* 805 (2018).

[27] Guangrong Bian Shengnan Zhang Yan Hu. "Research on string similarity algorithm based on Levenshtein Distance". In: *IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)* (2017). DOI: 10.1109/IAEAC.2017.8054419.

[28] Kileev V.V. Sidorkina I.G. "Kodirovka simvolov peremennoi dliny v algoritme Damerau Levenshteina". In: *Vestnik Chuvashskogo.universiteta* 3 (2013).

[29] Alexey Andreevich Sorokin. "Spelling Correction for Morphologically Rich Language: a Case Study of Russian". In: *Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing* (2017). DOI: 10.18653/v1/W17-1408.

[30] Shavrina T. O. Sorokin A. A. "Automatic spelling correction for Russian social media texts". In: *Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference* (2016). DOI: 10.18653/v1/W17-1408.

[31] Shavrina T. O Sorokin A.A. "Automatic spelling correction for Russian social media texts". In: *Moscow: Russian State Univ. for the Humaniti* (2016), pp. 688–702.

[32] Tobias Scheffer Steffen Bickel Peter Haider. "Predicting Sentences using N-Gram Language Models." In: *Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference* (2015). DOI: 10.3115/1220575.1220600.

[33] Moore R. Toutanova K. "Pronunciation modeling for improved spelling correction". In: *Proceedings of the 40th Annual meeting on Association for Computational Linguistics. Pierre I.(ed.). Philadelphia: Association for Computational Linguistic* (2002), pp. 144–151.

[34] Abduali Tukeyev Turgunbayeva. "Lexicon free stemming for Kazakh language information retrieval". In: *International Conference on Application of Information and Communication Technologies* (2018).

[35] Dali S.Naga Viny Christanti M. "Fast and Accurate Spelling Correction Using Trie and Damerau-levenshtein Distance Bigram". In: *TELKOMNIKA* 16.2 (2018).

[36] Dongfeng Cai Y.Cai D.Ji. "A KNN Research Paper Classification Method Based on Shared Nearest Neighbor". In: *Proceedings of NTCIR-8 Workshop Meeting* (2010).

[37] Zhandos Yessenbayev Zhanibek Kozhirbayev. "Kazakh Text Normalization using Machine Translation Approaches". In: *CEUR Workshop Proceedings* 2780 (2020).