Journal of Theoretical and Applied Information Technology
30th November 2021. Vol.99. No 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

CORRECTION OF KAZAKH SYNTHETIC TEXT USING FINITE STATE AUTOMATA

[1]KARTBAYEV A., [2]MAMYRBAYEV O., [3]KHAIROVA N., [4*]YBYTAYEVA G., [5]ABILKAIYR N., [6]MUSSAYEVA D.

[1]Institute of Information and Computer Technologies, Kazakhstan
[2]Department of Computer Sciences, Al-Farabi Kazakh National University, Kazakhstan
[3]National Technical University "Kharkiv Polytechnic Institute",
[4] Department of Cybersecurity, Information Processing and Storage, Satbayev University, Kazakhstan
[5,6]High Schoool of Economics and Business, Al-Farabi Kazakh National University, Kazakhstan
E-mail: [1]a.kartbayev@gmail.com, [2]morkenj@mail.ru, [3]nina_khajrova@yahoo.com,
[4]ybytayeva.galiya@gmail.com, [5]abilkaiyr.nazerke@gmail.com, [6]d_i_n_mus@mail.ru

ABSTRACT

In this paper we investigate the correction of generated synthetic text for resource-poor languages. In most cases, this synthetic text contains many errors that need to be carefully checked and corrected by additional tools. These errors must be corrected automatically to avoid degrading the performance of the system. Our approach to automatic error correction is based on the use of finite automata to suggest candidates for correction of the misspelled word. After selecting correction candidates, a language model is used to assign points to the correction candidates and choose the best correction in a given context. The proposed approach is language-independent and requires only dictionary and text data to construct the language model. The approach was evaluated in Kazakh and achieved an accuracy of 91%.

Keywords: *Synthetic Data, Language Model, Finite State Automata, Hidden Markov Model, Text Generation.*

1. INTRODUCTION

In data processing tasks, natural language text is the main type of data. A large set of training and test data is required to achieve the goal in these tasks. Recently, many countries of the world have tightened the rules of user data collection at the legislative level. Therefore, the task of automatic generation of synthetic texts in natural language, preserving the main characteristics of real texts, is of great importance. Many methods of generating synthetic texts solve the problem of creating such texts. However, at the moment, the available synthetic data generators do not reproduce natural language text data well enough.

In this paper, we present a preliminary study aimed at correcting the results of models for generating texts that have the characteristics of natural language texts. In a series of experiments using different datasets and finite state automata for the Kazakh language, we have shown that the models under study can generate texts with the required quality based on the Hidden Markov Model.

The study has a fundamental novelty both in the formulation of the problem and in the choice of methods for solving the problem. The effectiveness of methods and approaches to solve the problem is based primarily on the comprehensive use of modern advances in the field of artificial intelligence, mathematical linguistics and computer technology related to the development of formal language models, the theory and practice of text generation in natural language.

2. RELATED WORK

For our study we consider the Kazakh language, which is the state language in the Republic of Kazakhstan. Kazakh is part of the Kypchak branch of the Turkic language family and is very rich in morphology compared to languages such as English. Kazakh, in which words are formed by adding affixes to the root form, is called an agglutinative language. We can form a new word by adding an affix to the root form, and then form another word by adding another affix to that new word, and so on. This iterative process can continue on

several levels. Thus, one word in an agglutinative language can correspond to a phrase consisting of several words in a non-agglutinative language. For a review on comprehensive rule-based morphological analysis, we refer the reader to the following studies[1].

The general problem of error correction in texts obtained by different generation methods has attracted considerable attention of researchers in previous years[2][3][4]. Ideas for various error correction algorithms have been proposed by various authors, primarily in the development of natural language processing systems. For example, the application of large language models to error detection with sufficient efficiency is described here[5]. Error correction methods in Mandarin Chinese have also been presented[6]. Almost all the error correction algorithms presented first create lists of candidates, and then select by ranking the candidates with the help of a language model[7]. The Levenshtein distances are used to construct a set of candidates to replace the erroneous one, allowing more accurate correction of cases of erroneous word splitting into several worthwhile words.

The following papers[8] propose improvements to the error correction step errors. Along with the used probabilistic language model of the text, a word reliability measure is introduced, which allows to correct some syntactic and semantic errors at the expense of the information on the neighboring words. The idea of the method is to rank a list of candidates for replacing an erroneous one.

The selection of candidates is rather labor-intensive, due to numerous calculations on the language model. This paper proposes a method based on reducing the computational laboriousness of the error finding procedure. An extended language model is used, which takes into account their mutual information related on parts of speech[9].

The well-known Hidden Markov Model (HMM) is defined as a Markov process with hidden states and an observable variable[10]. These hidden states have a probability distribution over the possible observed outputs. The main task of an HMM is a supervised learning process in which the most likely model that produces the observed sequence is chosen.

To generate synthetic text, HMM is applied as follows. In the first phase, the source text is tagged with a partial speech tagger, and then the process of computing the most likely model capable of producing the desired text is done[11][12]. All transitions from the hidden state and variable are counted and used to estimate transition probabilities.

The correction method we use in this paper assumes that the dictionary is represented as a finite state machine (FSM). Our solution selects a number of dictionary words with corrections, and then measures the distance between the incorrect word and all selected words. Another method that works on the same principle is the similarity key method[13]. In this method, words are divided into classes according to their characteristics, where the comparison is made with the class of words. In addition to the considered method, there are other methods based on finite state automata, for example, in which words are considered a separate language over an alphabet[14].

Our proposed method imposes no restrictions on the edit distance between the input word and the candidates. But it can accept several constraints on symbols that can replace certain other symbols. For context-dependent error correction, we mainly apply a candidate set ranking with respect to the context of the corrections.

3. TEXT CORRECTION USING FSM

3.1 Description of the method

Our approach to correcting synthetic text consists of three main steps: detecting incorrect parts of a sentence, generating possible candidates for correction, and choosing the most appropriate corrections. The most obvious way to detect incorrect sentences is to search for each occurrence of a phrase in the dictionary and to look for words not found in the dictionary. However, we can represent the dictionary as a finite state automaton to make this process more efficient. In the proposed approach, we build an FSM that represents the path for each word in the input string. We then combine this with the dictionary FSM. The result of the operation is the intersection of these words, which are present in both the processed string and the dictionary. If we find the difference between the FSM containing all the words present and this FSM, we get the FSM for each incorrectly written text. Figure 1 illustrates the FSM containing the input string.

Further the problem of generating candidates for incorrect phrases can be divided into the following subtasks: generating a list of words close to the input word by distance, and selecting a subset of words from the dictionary. To perform these tasks, we generate one transducer from FSM representing the word, which generates all the words within a certain distance from the input word. After finding the wrong words represented by the FSM, we can filter out the words that are not in the dictionary.

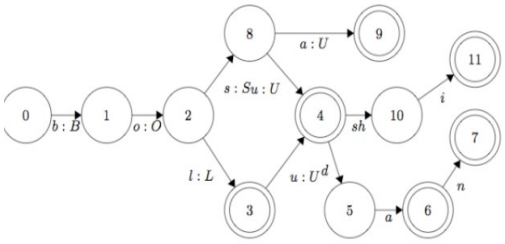

Figure 1: the FSM containing the input string

To select the best choices from the set of candidates, we use a language model to assign probability to the sequence of words. To obtain the desired word sequence, we consider the context in which the incorrect word appeared, replace the incorrect word with a fixed candidate, and retrieve n-grams containing candidates at probable positions in the n-gram. We then find a score for each n-gram using the language model and assign the corresponding score to the candidate as the average score of all n-grams. Before choosing the best candidate, we downgrade the fixes that require more editing in order to give preference to candidates with minimal editing.

3.2 Generating candidate corrections

Here we give a description of the formal apparatus of two-level rules, as well as a full description of the two-level model of Kazakh language morphology and morphological analyzer built on its basis using our modified source formats of the PC-K1MMO toolkit[15] and belonging to the class of pragmatic conceptual-formal models. PC-K1MMO is a computer program which uses a linguistic description of the phonology and morphology of the native language to recognize and generate words in this language.

Models implemented using the PC-KIMMO can be used as stand-alone modules in other language processors. In particular, the Kazakh morphological analyzer based on PC-KIMMO is used as part of a rule based machine translation system from Kazakh to English[16]. The morphological analyzer can be effectively used also as a software tool for studying, researching and developing natural language morphology.

PC-KIMMO, from the morphological analyzer's developer point of view, consists of two user created files. The first file is the rules file, which describes the alphabet and phonological rules. The second file is the Lexicon, which contains the vocabulary of the lexical units (root and affix morphemes) and their interpretations, as well as the description of the morphotactical rules. A lexicon consists of sub lexicons (sublexicons) divided by selective features and paradigmatic classes. The structure of the sublexicons forms a connected graph, with the root lexicon at its apex, which starts the analysis of the input word[17]. All the rules of the second morphology component are written in the language of regular expressions[18]. The analysis technology is based on a kind of finite-state transducer (FST). An FST is an automaton in which each transition between states in a network has an output label in addition to the input label. The original morphological lexicon is compiled into a lexicon transducer, and the rule component is compiled into a two-level rule transducer. The resulting lexical finite state machine, i.e. a complete morphological representation of a language, is a lexical transducer obtained by a composition of a lexicon transducer and a rule transducer. The character alphabet of a finite automaton is called Sigma. Sigma of lexical TCS consists of the alphabet of the analyzed natural language and special grammatical tags that express the meaning of the selective features and grammars (e.g., +Verb - verb, +Active - active voice, +P1 - 1st person, +P1 - plural, etc.).

The root lexicon calls the sub-lexicons. Expressions in lexicons are a pair of forms: lexical and surface forms, separated by a colon. The TCS builder interprets such a pair as a regular relation. The '#' grid marks the end state. The uniqueness of the path of transitions in the finite automata network gives uniqueness to the morphological interpretation. The path variants leading to a finite state in the TCS network specify a plurality of interpretations for the surface form, which corresponds to morphological multivaluedness.

In the two-level approach, phonology is defined as the relationship between the lexical level of deep representation of words and their realization at the surface level, by virtue of which the theoretical model of PC-KIMMO phonology is called a two-level phonology. PC-KIMMO includes two functional components - a generator and a recognizer.

The generator inputs the lexical form, applies the rules of phonology, and returns the corresponding surface form. The lexicon is not used. The recognizer receives on the input the surface form, applies the rules of phonology, refers to the lexicon and returns the corresponding lexical forms with their comments (interpretations). Figure 2 shows the structural and functional diagram of the two-level morphological analyzer.

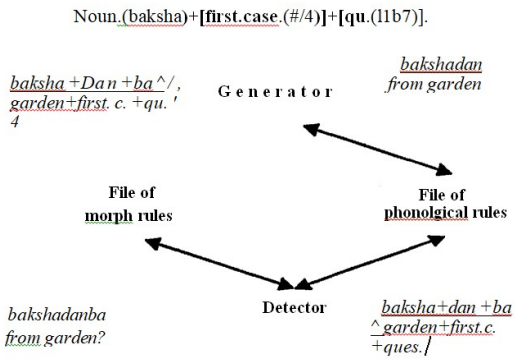

Figure 1: decompoings the surface form "bakshadan" into its components

The generator, using the file of phonological rules, translates the lexical notation "baksha+da" into the surface form "bakshadanmyn" ("from the garden"). The recognizer, using both the file of phonological rules and the file of morpho-tactical rules, decomposes the word form (surface form) "bakshadanmyn" into its components and their corresponding substantive descriptions.

Two-level rules are similar to the rules of classical generative phonology, but differ in several important points. Here is an example of a generative rule notation: (R1) RULE x -> y/z. The two-level rule has the following form: (R2) RULE x:y =>z. The difference between the formalisms of the two rules is not only in their writing, their meanings are also different.

Generative rules have three main characteristics: 1) transformation rules - they transform or rewrite one character into another. Rule (R1) states that x becomes (changes into) y when it precedes it. After (R1) x is rewritten as y and x does not exist further; 2) consistently applied generation rules transform deep forms into surface forms through any number of intermediate levels of representation; 3) generative rules are unidirectional - they can only convert deep forms into surface forms, but not vice versa.

In contrast to (R1), two-level rules are declarative. They establish certain relationships (connections) between lexical (i.e., deep) forms and their surface forms. (R2) establishes that the lexical x corresponds to the surface y before g; it does not change into y and only takes place after this rule is applied. Since the two-level rules express the connection of characters rather than their overwriting, they are applied in parallel rather than sequentially, not forming intermediate representations as with (R1). Only lexical and surface levels are allowed, no other intermediate levels. This is their property, which is why they are called two-levels. Moreover, since a two-level model is defined as a set of links between lexical and surface representations, two-level rules are bidirectional. A given lexical form PC-K1MMO translates into a surface form and a surface form into a lexical form.

An important characteristic of two-level rules is that they require a one-to-one correspondence between lexical and surface letters, i.e., there must be an equal number of lexical and surface letters and each lexical letter must cover exactly one surface character and vice versa. A phonological process that removes or inserts characters corresponding to the NULL symbol into the two-level model is written as 0 (zero). Another special character is the BOUNDARY (boundary) character, written as #. It is a boundary character that represents either the beginning or the end of a word. It can only be used in the context of a rule and can only correspond to another boundary character, i.e. #.

In PC-KIMMO, character classes are listed with one name (one or more characters, without a space). These character classes are defined in SUBSET statements in the rules file. For example, the following declarations define CS as the set of consonants, VOWEL as the set of vowels, S as the set of mute consonants, and NASAL as the set of nasals.

The main mechanism for representing two-level rules as a two-level computer model

is the finite-state transducer technology. It consists of finite states and directed transient arcs. As a minimum, it must contain an initial state, a final state, and an arc between them. A successful transition from one state to another is possible when the next character of the input line matches the character on the arc connecting the states.

Transducers differ from automata in that they operate on two input sequences. Transducers are automata in which each transition between states in the network has an output label in addition to the input label.

For example, it recognizes whether two chains are valid correspondences (or translations) to each other. Suppose that the first input chain for a transducer is a language chain containing elements x and y and defined as b1={hunx\n> 0}. Correctly constructed chains for this language are: xx, xuh, xuhuh, xuhuh, etc. As the second input we define chains of the language b2 corresponding to chains of the language b1 where every second occurrence of the element y corresponds to an element. Fig. 3. shows a diagram of correspondence between languages b1 and b2.

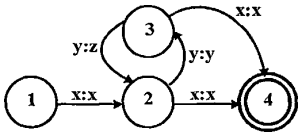

Figure 3: a diagram of correspondence between languages b1 and b2.

Transducers can also be represented in the form of finite state tables, with the only difference being that the column headings will indicate pairs of correspondences, such as: x:x, y:y, and y:z. For example, the diagram shown in Fig. 3, can be represented as the following table of finite states:

Table 1: finite state tables indicating pairs of correspondences.

	x x	y y	y z
1.	2	0	0
2.	4	3	0
3.	4	0	2
4:	0	0	0

For example, let's take the execution of a two-level rule as an example: (R1) RULE t:d =>_y:

The operator => in this rule means that the lexical symbol t is realized as a surface symbol d only when (but not always) it precedes the environment (context) y: y.

The correspondence t:e declared in rule (R1) is special. The two-level description contained in rule (R1) must also contain a set of default correspondences, such as k:k, a:a, t:t, y:y, etc. The set of all special and default correspondences forms the set of probable pairs.

Let the description contain (R1) and the set of all default matches. Suppose a lexical form(LF) “katyk” is fed to the input of the generator. The generator starts browsing from the first character of the input sequence and looks to see what correspondences are set for it. At a certain point in time the generator has a symbol t as its input, and for a successful t:d match by rule (R1) the next input symbol in the chain for it must be a y:y match. Having found that this condition is satisfied, the generator sets t:d.

Table 2: the execution of a two-level rule.

LF:	K	a	t	y	k
DC	3	2	1	2	1
SF:	K	a	t	y	k

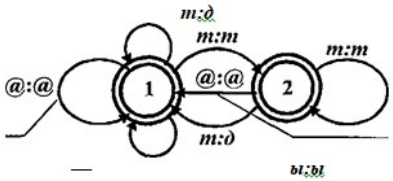

Figure 4: the execution of a two-level rule.

Since there are no more input characters in the input lexical chain, the generator will produce a surface form of the “kadyk”. However, the generator does not complete its work. It continues to return to the previous characters and tries to find alternative implementations of the lexical form.

First it makes a return to the last match of the input character y:y, then it recycles the third lexical character t again. A t:d correspondence has already been set for it, so the generator will set the next possible t:t correspondence, defined by default. Then the

generator moves on to the last character k, for which the default correspondence (DC) k:k is set. All other rollbacks are unsuccessful. Therefore, the generator completes its work and will produce a second surface form (SF) of "katyk".

3.3 The phonological rules file description

The rules file consists of a list of keyword declarations and their corresponding content. The rules file uses the following set of keywords: ALPHABET, NULL, ANY, BOUNDARY, SUBSET, RULE and END.

1) ALPHABET.

This is a list of 42 characters required for a complete representation of the Kazakh alphabet.

In the base shell PC-KIMMO the Latin alphabet is used for symbols, so that complicates the realization of phonological rules file and Lexicon for languages based on the Cyrillic alphabet. In this connection we have carried out modification of program toolkit with use of system Visual Studio and programming language C#.

The modified system was supplemented with additional capabilities to work with characters of the Unicode code table, respectively, providing an opportunity to use languages based on the Cyrillic alphabet. In addition we have developed plug-in .dll and .net-modules for morphological analysis and text synthesis. These libraries were developed in the Microsoft Visual Studio .NET application development environment, allowing the two-level model to be used on any alphabetic basis, including Cyrillic, in cross-platform systems.

There is also a lexical form of writing, which at the surface level is implemented according to phonological rules. % - is applied to words that do not obey the law of vowel harmony. For example, the word bale (trouble) attaches allomorphs with "soft" vowels, rather than "hard" vowels, as the rules of vowel harmony suggest (ends in a "hard" syllable). The lexical form of the word form construction of the word form bale+LY is formed as follows: bale%+LY, where the lexical symbol y in this case corresponds to the surface "soft" symbol and not to the "hard" symbol y according to the law of vowel harmony.

2) NULL O

3) ANY @

4) BOUNDARY #

The components of the rules file indicate the purpose of the corresponding characters to be used in writing the rules. The SUBSET section is used to make the rule file more compact.

5) SUBSET CS is the designation of the set of all (ConSonants) letters appearing as consonants (25 letters).

The rules file is then followed by the Rules themselves, which establish character matching depending on the context, i.e. the character environment in the word form. The phonological rules indicate in what environment the appropriate lexical character is to be changed when the word-form is generated.

3.4 Description of the lexical components file

The Lexicon contains a list of lexical entries found in the description. Lexical input can be a single morpheme (such as root, prefix and suffix) or a morphological complex of words (prefix plus root and suffix; for agglutinative languages this order would be: root plus affix morpheme). In word recognition, lexical components work together with rule components. The general structure of the lexicon is a list of keyword declarations. The set of valid keywords includes ALTERNATION, LEXICON, INCLUDE and END. The declarations can occur in any order except that LEXICON must be declared after ALTERNATION. The obligatory single declaration is LEXICON INITIAL; that is, a lexical file must at least contain a sub lexicon called INITIAL (beginning).

The skeleton of a LEXICON file looks like this: ALTERNATION End End LEXICON INITIAL 0 End "[" LEXICON End 0 # "]" END.

Lexical components also use automata. Morphtactic constraints are represented in the lexicon by structuring it as automata. Since two-level phonological rules use transducers that can operate on two strings simultaneously, the process of recognizing morpheme sequences in the lexical form of a word deals with only one level. Thus, it uses a less complex automata formalism that operates on only one line. The PC-KIMMO lexicon is an automata in which (1) each changing name is a state; (2)-joining classes are arcs that point to the next state; (3)-the sublexicon of lexical occurrences are labels on the arcs.

The morphotactic rules file is designed on the basis of morphotactic schemes and

defines the relationships between the base and affixal groups. The description of morphotactic rules for the verb in the file <kazakhV.lex> and for nouns in the file <kazakhN.lex> is demonstrated here.

The lexicon of root lexemes is built on the basis of the modern Kazakh language and consists of a number of lexicons filled in according to the relevant PC-KIMMO requirements. The sub-lexicons contain rows of lexical entries consisting of the following three parts: the first part is a lexical atom (a Kazakh root word); the second part is an accession class (or continuation); that is, something that may follow immediately after this atom - a sub-lexicon that may have other lexical units. Accession classes can follow many other morphemic units. The lexicon ALTERNATION in PC-KIMMO is a list of names of sublexicons, the order of which determines which class can be followed by which, while only one definition is possible, i.e. it is a restriction inherent to the sublexicon; the third part is its interpretation (description of grammatical features). As a rule, any morphological, grammatical, lexical, or semantic properties of a lexical unit are recorded here. When the word recognizer processes a word, the interpretation of each selected morpheme is added to the result line.

(1) Nouns. The lexicon includes about 20 thousand Root nouns.

(2) Verbs. The lexicon contains about 8 thousand verb roots.

(3) Adjectives.

As it is known, Kazakh is an agglutinative regular language subjected to strict rules. At the same time, as in any natural language, there are exceptions, most often also subject to certain rules. For example, superlative adjectives have prefixes written with a hyphen '-'. For example: the root word 'red' in the superlative degree is written as kyp-kyzyl ('very red'). The Lexicon of Adjectives contains over 3,000 basic roots and additionally includes a lexicon of 140 superlative adjectives with prefixes. The following Lexicons, which constitute a small fraction of the total vocabulary of about 30 thousand root words with specific morphotactic rules inherent to the selected word groups, are also defined: (4) Adverbs. (5) Pronouns. (6) Numerals. (7) Postpositions. (8) Conjunctions. (9) Interjections (Exclamations).

The ALTERNATION parameter has 8 inputs for word forms (So, in this description, it is defined that there are 8 different possibilities for a kazakh word's beginning): VERB (verb), a sublexicon for verbs; NOUN (noun), a sublexicon for nouns; ADJECTIVE (adjective), a sublexicon for adjectives; ADJECTIVE2 (adjective2), a sublexicon for adjectives; NUMERAL, a sublexicon for numbers; PRONOUN, a sublexicon for pronouns, postpositions; ADVERB, a sublexicon for adverbs; SPECIAL, a sublexicon for conjunctions, interjections.

3.5 Description of the base of morphotactic rules

The list of verb forms for recognition is written to the special file <kazakh.rg>, which is fed to the input of the two-level morphological analyzer.

Suppose the file <kazakh.rg> contains the following words: baru bargandar barma barmasa bardy. Then the recognition result recorded in the file <kazakh.rec> will be:

bar+U[V(bar)+NOMINATIVE(y/Y/B)]bar+GAN+DAR[V(6ap)+PAST_UNDEF(GAN)+PLURAL(DAr)]bar+mA[V(6ap)+NEGATTVE(MA)]bar+mA+sA[Y(6ap)+NEGATIVE(MA)+CONDITIONAL(cA)] bar+Y [V(bar)+COUSATIVE(DY)].

Next, here is a description of the morphotactic rules file for the Kazakh verb with examples and comments.

Kazakh.lex {File containing sublexicons of all lexeme classes} ALTERNATION BEGIN VERBS {VERBS is a list of verb bases that are the initial input for the analyzer} Example: LEXICON VERB bar verb "V(6ap)" kel verb "U(kel)" kara verb "U(kara)"

ALTERNATION verb { here the affix classes that can follow the verb are specified} REFLEX MODAL NOMINATIVE INFINITIVE PARTICIPAL CONTRARY IMPERATIVE REQUEST CONDITIONAL TENSES CONDJFUTURE1 End { in our case the specified affix classes, each of which is further predefined up to the corresponding affix group}

ALTERNATION End End {Signify the end of affix accession or zero affix accession} LEXICON INITIAL O BEGIN "[" INCLUDE verb.lex; {connect file containing verb bases}

What follows is a description of the affix base of Kazakh verb word forms. Here is a description of the fragment from this file. LEXICON REFLEX {group of reflexive affixes denoting the form of pledge}

The first part of the lexicon gives the affix morpheme, then the name of the class of morphemes that may follow this affix. The third component reflects an interpretation, a commentary regarding a given lexical input.

Morphotactic rules specify affix groups and their ordering. The recognition function accesses both the phonological and morphotactic rules file.

The scheme of morphotactic transitions for verbs is constructed taking into account the grammatical categories of inflection, negation tense, voice, number and person of the verb. The verb stem is presented in the dictionary in the form of the 2nd person of the Imperative: e.g. bar - 'go', kel - 'come'. All affixes in the scheme are presented in the lexical form (LF), that is, depending on the environment; they acquire different surface forms (SF). For example, LF: bar (go) +Gan kel (come) + Gan SF: bargan (went) kelgen (came). As can be seen from the example, here the affix -GAn appears in two surface forms: -gan and -gen.

3.6 Description of the lexical semantics

When describing the semantics of affixal morphemes, we proceed from the statement that each morpheme is used to encode a meaning in some context, reflecting some local "picture of the world. The use of affixal morphemes allows us to significantly reduce the number of root morphemes for the transmission (coding) of some meaning, i.e. serves as an element reducing the lexical space needed to form the context.

A local "world picture" is a formalized description of some context reflecting objects and their relations. The division of lexemes or groups of lexemes into objects and relations is a rather conventional procedure and depends on semantic roles performed by lexemes or groups of lexemes reflecting certain meanings in a certain context. It is known that the meanings of morphemes form a certain context, which is most fully revealed in the semantic situation formed by the word-form or their combination, and each affix can be used in the formation of different contexts.

Affixal morphemes as minimal meaningful units of the language, by definition, have at least one meaning, manifested when it is used in the word-form. In the Kazakh language, often, depending on the environment, affixal morphemes have different interpretations, i.e. depending on the context have different meanings, and the same situation is not always conveyed by the same class of morphemes. Formal semantic models allow us to most fully reflect the meanings of affixal morphemes in some fragment of the real world and build morpheme correspondence tables for the pair of translated languages and mathematical linguistic models of translation using these tables. The methodology of comparing the meanings of affixal morphemes based on the object-predicate relation system allows us to effectively identify the elements of similarity and difference between languages at the deep semantic level, and to build mathematical linguistic models to use them in the tasks of machine translation and multilingual search.

4. EXPERIMENTS

We are developing our model in a data-poor environment and mostly on synthetic Kazakh texts generated from very different data sources. Unlike the machine translation data we have previously collected, we do not yet have public texts to train our correction model, so we collect both training and evaluation data almost from scratch[19]. As training data, we mainly use generated data from synthetic text.

Our hypothesis about this kind of data is that possible users who will actually try to generate the data follow our representation of it; accordingly the result will correspond to their possible intention, and from this sequence of possible text actions we can potentially extract samples of incorrectly and correctly written texts.

Obviously, this in reality involves a much greater variety of actions on the data, and so additional filtering is required to obtain representative data to train the error correction model. By filtering the data from our dataset, we get about 2 million pairs of misspelled and corrected training data. For testing, we use the same dataset as in our previous work on machine translation.

Table 3: the training dataset.

Text data, thousands	training	testing
words	2584	500
errors	713	181

The software modules of the system are implemented on the basis of morphological models described in the second chapter of the paper. The modular structure of the system contains user and algorithmic parts, and the algorithmic part is language-independent, which, if necessary, allows you to build correction models for different languages.

Table 4: the performance of correction model on the test data sets.

Metrics	Recall	Precis	F-score
baseline	99.87	94.36	97.0368
pure random	71.49	71.59	71.54
refined	99.91	89.91	94.65
Test set1	71.52	75.86	73.63
Test set2	86.23	86.57	86.40
Test set3	88.69	92.15	90.39
Test set4	60.55	69.17	64.57
Test set5	62.92	66.18	64.51
Test set6	76.05	79.21	77.60

Let's consider the stages of execution of modules in the order of phrase processing on the example of Kazakh synthetic text. Let the following sequence of word forms, forming a sentence in the Kazakh language "Men kuzgi zhol bardym", come to the input of the system. Examples of processing of this phrase are given below as a result of execution of the module for the Kazakh language.

1) The module Two-level morphological analyzer, described in chapter 2, using morphotactic files and two-level rules compiled into finite state automata, gives the analyzed word forms with assigned morphological features:

1. men [Pro1_Sing(Men)]
2. kuzgi[N(K63)+CASE_POINT(TBI)]
3. zhol [(zhol)]
4. bardym [V(6ap)+POST_DAF()+1 PSJSing()]

The morphological analyzer in the form of a plug-in dll-module is implemented in .NET application development environment, which provides compatibility of services in different application systems and its functioning in cross-platform systems. The word processing speed of the dll-module is about 100 word forms per second.

2) The module sentence variant builder is used to build variants of sentences obtained as a result of multi-word morphological analysis of word forms related to lexical uncertainty. For our example, it generates all possible variant sentences:

3) All variant sentences arrive at the input of the correct sentence construction module, where the correct sentence selection algorithm is executed to select one based on the input sentences, after which the module searches for the most relevant words from the input variants.

4) Next, the found correct sentence is fed to the input of the verification module, where the affix and root morpheme database, based on a formal semantic model of affix values, is used to verify the elements of the sentence.

5) As a result of all these actions, on the basis of data from the module of two-level morphological analyzer, using morphotactic files and two-level rules of the Kazakh language, compiled into finite state automata, the system will generate the output sequence "Men kuzgi zholmen bardym".

6) The output data preparation module allows outputting the data with appropriate formatting of the input data.

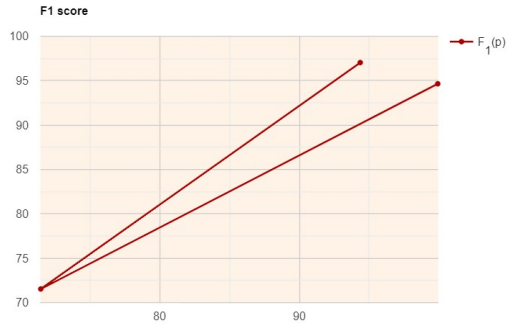

Figure 4: Distribution of F1-score values after correction of distorted texts.

The proposed method is based on a multistage application of the approach described above; at each stage the text fragments that remained distorted after the previous stage are corrected.

Non-word forms and word forms, the probability of occurrence of which in the text according to the chosen probabilistic model is

less than a given threshold, are considered distorted. Word forms are defined as continuous sequences of alphabetic characters separated from each other by spaces or punctuation marks. Fig. 4 shows the distribution graphs of F1-score values during the correction of distorted texts. For the method, the F1-score distribution graph is calculated in cases where the list of candidate words was composed of words within a Levenshtein distance of up to 4 from the word being corrected.

5 DISCUSSION

A two-level model of Kazakh morphology has been developed, which belongs to the class of formal models and is a complete computer model of Kazakh morphology. On its basis the two-level morphological analyzer, which generates word forms on the basis of phonological rules and decomposition of arbitrary word forms into morphemes on the basis of phonological and morphotactic rules, has been constructed.

Mathematical linguistic model of morphology is described on the basis of two-level formalisms of PC-K1MMO software toolkit modified by us for the tasks solved in the framework of this thesis. The two-level rules are implemented using finite state automata. The mathematical model of morphotactic rules is bi-directional and is implemented based on finite state transducers.

In contrast to generative rules, two-level rules are declarative and establish certain relations between lexical forms and their surface forms. Since two-level rules express the connection of characters rather than their overwriting, they are applied in parallel rather than sequentially, without forming intermediate representations, as with generative rules. Since the two-level model is defined as a set of relations between lexical and surface representations, the two-level rules are bidirectional; respectively the same mathematical linguistic model is used for both text generation and recognition.

To describe the phonological rule file of the Kazakh language, 42 rule records are used. The rules implement the law of vowel harmony as well as a number of exceptions and irregularities caused mainly by the lack of phonetic assimilation of borrowed lexemes.

The file of morphotactic rules is developed on the basis of morphotactic schemes and defines relations between the base and affixal groups. The lexicon of root lexemes is built on the base of Contemporary Kazakh language and consists of 9 lexicons filled according to PC-KIMMO requirements: Nouns, Verbs, Adjectives, Adverbs, Pronouns, Numerals, Postpositions, Conjunctions and Exclamations. The total volume of the dictionary is 30,000 root words. Complete morphotactic base for all selected classes of root lexemes was developed.

In the basic PC-KIMMO the Latin alphabet is used for symbols, so that there are difficulties in the implementation of the file of phonological rules and Lexicon for languages based on the Cyrillic alphabet. In this connection we have carried out modification of PC-KIMMO software toolkit using Visual Studio system and C# programming language. The modified system has been supplemented with the additional possibilities to work with symbols of Unicode code table, accordingly giving the opportunity to use languages on the Cyrillic alphabet. Besides, plug-in .dll and .net-modules for morphological analysis and text synthesis have been created in Microsoft Visual Studio application development environment, which allows using two-level model on any alphabet basis, including Cyrillic, in cross-platform systems.

6 CONCLUSION

Despite the adequacy of the described approach to error correction, it has disadvantages: the language models and correction models used in practice are rather crude. A possible compromise is multistage methods that provide successive approximations to the optimal solution.

In addition to the Levenshtein distance [20], its modifications can also be used to construct a set of candidate words to replace an erroneous one, allowing for more precise correction of cases of erroneous word splitting or erroneous joining of adjacent words.

To study the impact of text noisiness on the performance of machine translation systems, we propose to develop parallel corpora with artificially introduced distortions [21]. The following types of distortions are considered: changing the case of individual letters, replacing words with phonetically similar ones, dropping one or more letters in a word to reduce it, dropping words from a sentence to

reduce it, merging neighboring words in a sentence.

The paper studies distorted texts of natural way. Distortions in such texts are sometimes made by the users themselves when writing for speed of writing or due to illiteracy. The method allows to noticeably increasing the accuracy of the correction. In the experiments conducted the quality of correction in F1-measure values for medium distorted texts increased by 6%.

ACKNOWLEDGMENTS

This research has been funded by the Science Committee of the Ministry of Education and Science of the Republic Kazakhstan (Grant No. AP09259309).

REFRENCES

[1] Bekbulatov E. and Kartbayev A., "A study of certain morphological structures of Kazakh and their impact on the machine translation quality", In Proceedings of the IEEE 8th International Conference on Application of Information and Communication Technologies, 2014, p.495-501.

[2] Joseph J Pollock. "Automatic Spelling Correction in Scientific and Scholarly Text", Communication of the ACM, 1984, No.4, pp.358-368.

[3] Golding, A.R., Schabes, Y., "Combining trigram-based and feature-based methods for context-sensitive spelling correction", In Proceedings of the 34th annual meeting on Association for Computational Linguistics, Morristown, 1996, p.71–78.

[4] Sjobergh J., Chunking: an unsupervised method to find errors in text. In Proceedings of the 15th NoDaLiDa conference, 2005, p.180–185.

[5] Andersen O.E., "Grammatical error detection using corpora and supervised learning", In Proceedings of the 12th Student Session of the European Summer School for Logic, Language and Information, 2007, p.269-275.

[6] Lei Zhang, Ming Zhou, "Changning Huang. Multifeature-based Approach to Automatic Error Detection and Correction of Chinese Text", Microsoft Research China Paper Collection, 2000, p. 193-197.

[7] Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto and Christopher Bryant. The CoNLL-2014 Shared Task on grammatical error correction. Proceedings of the 18th Conference on Computational Natural Language Learning: Shared Task, 2014, p.1–14.

[8] Yin, F., Long, Q., Meng, T., Chang, K.-W., "On the Robustness of Language Encoders against Grammatical Errors", In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Stroudsburg, 2020, p. 3386–3403.

[9] Samanta, P., Chaudhuri B.B., "A simple real-word error detection and correction using local word bigram and trigram", In Proceedings of the 25th Conference on Computational Linguistics and Speech Processing, ROCLING 2015, Kaohsiung, Taiwan, 2013, p.124-138.

[10] Rabiner, L.R., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", Proceedings of the IEEE, 1989, p.257-286.

[11] Y. Li, H. Duan, and C. Zhai, "A generalized hidden markov model with discriminative training for query spelling correction," in Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2012, p. 611–620.

[12] E. Brill and R. Moore. An improved error model for noisy channel spelling correction. In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics, Hong Kong, 2000, p.378-395.

[13] Kukich K., "Techniques for Automatically Correcting Words in Text". ACM Computing Surveys, 1992, Vol.24, No.4, p. 377-439.

[14] Bojanowski, P. Grave, E. Joulin, A., Mikolov, T., Enriching Word Vectors with Subword Information. Trans.Assoc. Comput. Linguist., 2017, p.135–146.

[15] Antworth E.L., "PC-KIMMO: a two-level processor for morphological analysis". Technical Report Occasional Publications in Academic Computing, Summer Institute of Linguistics, Dallas, Texas, 1994.

[16] Kartbayev A., SMT: A Case Study of Kazakh-English Word Alignment. ICWE Workshops, 2015, p.40-49.

[17] Kartunen L., Constructing Lexical Transducers, 15th International Conference on Computational Linguistics, 1994, p.406-411.

[18] Xerox, MLTT-95/Application of Finite-State Networks. Report, 1995, p154.

[19] Kartbayev A., Using Kazakh Morphology Information to Improve Word Alignment for SMT, AECIA 2015, Paris, 2015, p.351-359.

[20] Levenshtein V., Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10(8), 1966, p.707–710.

[21] Khairova, N.; Kolesnyk, A.; Mamyrbayev, O.; Ybytayeva, G.; Lytvynenko, Y. Automatic multilingual ontology generation based on texts focused on criminal topic. In CEUR Workshop Proceedings, 2021, 2870, pp. 108–117.