Design and Implementation of Interactive Web System for the Kazakh Text Recognition and Correction with Using of Parallel Computing

Bakyt M. Kairakbay, Daniyar B. Nurseitov, Yuriy Y. Stolyarov, Ilyas E. Tursunov, Igor A. Ugay, David L. Zaurbekov
K.I.Satpayev Kazakh National Technical University, National Open Research Laboratory of Information and Space Technologies / Satpayev str., 22, Almaty 050000, Republic of Kazakhstan
b.kairakbay@norlist.kz, d.nurseitov@norlist.kz, y.stolyarov@norlist.kz, i.tursunov@norlist.kz, i.ugay@norlist.kz, d.zaurbekov@norlist.kz

ABSTRACT

In given paper we present the integrated interactive computer recognition system for the Kazakh language text with using of parallel computing. The design and integration methodology of the system are based on service-oriented architecture that allows provide an easy, flexible, and extensible integration of any language service into any desktop or mobile client. 4-tier SOA was designed and built based on W3C Web service standard. Using the high-performance cluster demonstrates significant advantages in the Kazakh text processing, especially for large arrays of texts. The main objective of the developing system is to provide any person easy access to text and documentation in Kazakh language with the following possibility of editing and manipulation different docs through respective Web services OCR and morphological analysis/correction. Developed web services cover the Kazakh language text OCR and morphological analysis for subsequent correction of errors after OCR, and show an acceptable quality of the Kazakh texts recognition that is better than existing tools of text recognition.

KEYWORDS

Service-oriented architecture (SOA), Web service, high-performance computing, Kazakh language, natural language processing (NLP), optical character recognition (OCR), Tesseract, Xerox Finite State Tool (XFST), finite state transducer.

1 INTRODUCTION

Permanently growing amount of information and, therefore, of workflow requires more effective and productive tools of a natural language processing. Effective development of the Kazakh language as the state language presupposes the creation of a massive set of various professional and problem areas electronic documents in the Kazakh language, and make available to the public the widest possible access to them. High demand in the system of the Kazakh language text recognition and correction exists from ordinary users, governmental bodies, various office divisions of companies, and educational institutions. The large majority of the documents are, as a rule, in printed or any other material form. The scanning of printed texts and its following processing, recognition, and correction for the transformation to digital text format which allow further manipulation with it as with electronic document, is the usual practice for that. The optical character recognition (OCR) software such as ABBYY FineReader, OmniPage, CuneiForm, and others carry out initial recognition of the original scanned text. OCR is used in document management systems, translators, artificial intelligence systems, digital libraries, and many other applications.
However, the problem is that the accuracy of text recognition, in addition to the quality of its images, depends on a recognizable language. The most known recognition programs work well with, say, English or Russian texts. Moreover, the direct application of these programs to the Kazakh language gives a large number of errors. Therefore, in the first, further improvement of OCR quality is required in

relation of the Kazakh language. Secondly, additional resources are required to correct the errors of OCR in order to achieve an acceptable quality of the processed document output. Such tools are usually depend on the language of recognizable texts and the use and formation of different thesauri and lexicons (including stems and affixes), and can implement their morphological, syntactic and semantic analysis for the final correction of text too. To provide to the public a possibility of processing arbitrary volumes of texts it is required a creation of high-performance and easy of use interactive system with the quick response and free access.

In recent years, a development of Web service technologies and distributed computing has led to the fact that a growing range of tasks, which traditionally use desktop applications, are solved now in a Web environment. These tasks naturally include also text recognition.

The advantages of the Web service architecture in the recognition of texts is not just a possibility of using modern recognition systems from a variety of external sources without installation of these systems, or to automate the process of batch processing large number of pages without load on your computer. Additional advantage is that a placing a process of recognition in the Web environment allows to create flexible, self-learning recognition environment, accumulate experience, and is able to adapt to the peculiarities of the various texts.

In given paper we present a preliminary version of high-performance and Web-oriented system for the Kazakh language text recognition.

2 RELATED WORKS

The Kazakh language belongs to the Ural-Altaic family of agglutinating languages [1]. In such languages, the concept of the word is much broader than simply a set of lexis' items. Word structure can become quite long when you add affixes, and sometimes it contains the amount of the semantic information inherent in the whole sentence in the other languages [2]. Modern Kazakh alphabet based on Cyrillic consists of 33 standard Cyrillic characters of Russian and 9 additional characters that reflect specific sounds of the Kazakh language.

Research and development in different application areas of NLP has made exciting progress over the last few decades, developing effective strategies for automatically analyzing and transforming large amounts of natural language data. However, despite significant progress in this area, most of the developments had implemented in the form of desktop applications that still require from the user a lot of manual and routine work suffering from a lack of relevant services integration. In [3] for a solution of this problem an open service-oriented architecture (SOA) was proposed based on W3C Web services standards [4-6]. Such an approach allows providing an easy, flexible, and extensible integration of any kind of language service into any desktop client, and the user is not worried about the realization of relevant services in this case. Paper [7] is the further development of proposed in [3] solution in application to Web portals. Partial solution of mentioned problem is the development of Web-based interface to application with embedded NLP services that can be obtained through the APIs [8].

There are many OCR systems, both commercial (like ABBYY FineReader desktop package) and different free online OCR realized via Web-services (including ABBYY FineReader too) [9-11]. However, as a whole none of the well-known Web-services (Google, ABBYY FineReader Online, OnlineOCR, FreeOCR, Scanonline, NewOCR, Sciweavers: i2OCR) does not support recognition of the Kazakh language text. Interesting solution is proposed in [12] where there is an opinion that the collaboration of the network and OCR is expected to be useful for making lightweight applications not only for desktop systems but also for small gadgets and autonomous robots.

Finite state methodology is sufficiently mature and well developed for use in a number of computer applications areas of NLP including morphological analysis. The finite state [13] and two-level morphology [14] approach has been used successfully in a broad number of NLP applications both non-concatenated language like Arabic [15] and agglutinative languages. Among the last it can be noted also an automatic morphological analysis of Basque [16], free morphological analyzer for Turkish [17], computer morphological analysis of Turkmen language [18], finite state approach to the Kazakh language [19], and many others.

3 ARCHITECTURE

For the realization of Web-oriented system of the Kazakh text recognition we have chosen the service-oriented architecture (SOA) [20] based on established open standard and protocols SOAP, WSDL, HTTP, XML [4-6].

3.1 General Description

Multitier architecture of the system is shown in fig.1 and contains 4 tiers: *Clients*, *Presentation Logic*, *OCR and Morphological Analysis*, and *Data Layer*.

Tier Clients. This layer provides an access to the system. It can be as a desktop application, web application, and application of mobile devices (such as Android or IOS), or any other new application (created either by us or by the third-party developers) which are connecting with our system through the Web interface. This layer provides the functionality for services OCR and morphological analysis.

Tier Presentation Logic. This layer consists of WebServer, WebPortal, и WebServices. WebServer is responsible for the handling clients' requests. WebServices provides access to Web services OCR and MorphologicalAnalysis. They are responsible for communications between clients and applications OCR, morphological analysis and correction of text, preparation input/output data flows, and transform for a transfer the data into proper kind. WebPortal provides Web interface for the access to the services.

Tier OCR and Morphological Analysis. This layer contains the functional core of the system that is closed for users, and interaction with it is carried out via Web services. It consists of *Applications*, which directly realize the functional of optical recognition and morphological analysis. A high-performance computing cluster that implements a parallel processing of data is also located in this layer.

Tier Data Layer. This layer contains documents processed by the web service, morphological

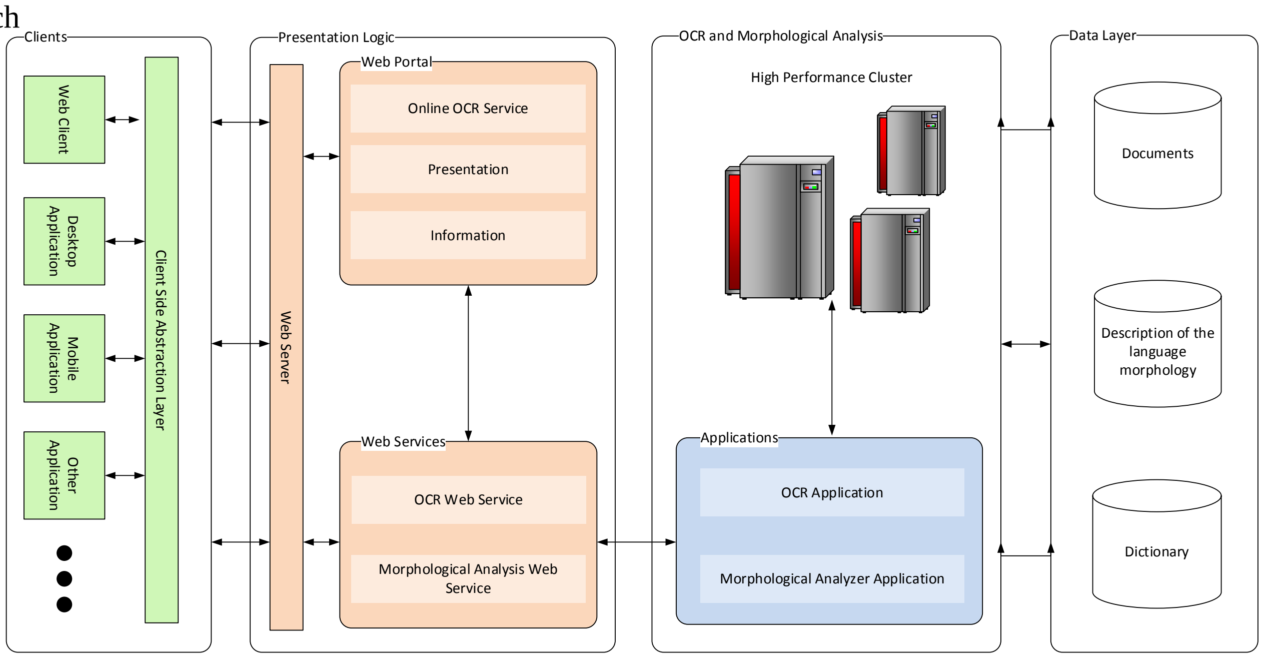

Figure 1. Architecture of Web

description of the Kazakh language, and dictionary databases.

3.2 Implementation

Let's consider some details of our architecture realization.

Web services. For the implementation of a simple and remote access to our services of recognition and text analysis, we used the technology of Web services based on the standard defined by the consortium W3C: "A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."[4]. To call the methods of a Web service client applications must obtain a description of the Web service on how to interact with it. This description is in the WSDL files. After receiving descriptions, client applications can exchange SOAP messages. Our developments are implemented with Microsoft Visual Studio and C# tools.

Client part. We published the Web service on server, and now it is accessible for the calling from clients' applications. A creation proxy-client object carries out the calling methods of the Web services. Now we can call the methods of the Web service through it, as shown lower:

```
SinglepageServiceClientproxySinglePage
= new SinglepageServiceClient();
intJobIdentifier                    =
proxySinglePage.NewOCR(0);
```

The use of Web services allows a developer of the client application very easy to integrate his/her application into our architecture, and it does not need to worry about implementing remote procedure calling or writing different code for network interaction.

4 OCR COMPONENT

4.1 Scheme of Work and Adaptation of the Tesseract to the Kazakh Language

To create the Kazakh language OCR component open source core libraries of the Tesseract were used [21]. Now, Tesseract is considered as one of the most accurate and high-quality open source programs in existence. It is a console application on input of which the input image in format TIFF is fed, and the result of the program code execution is "pure text" in txt format.
Tesseract works on the following scheme (fig.2). On this scheme, one can see that after you download the image with the text Tesseract produces division of the available text on the fixed fields. Then it divides these fields into the words that are identified with the help blanks and unclear places. Further, the text goes through two processes of recognition. It is done in order to get a more accurate result.

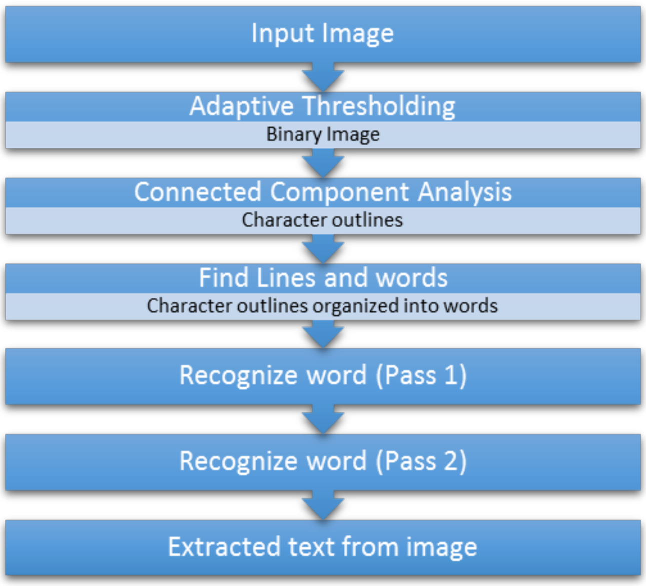

Figure 2. Scheme of work of the Tesseract

Initially, work was carried out to add a support of the Kazakh language to the Tesseract. To do this some Kazakh text was selected. Then the text was edited and converted to a TIFF image. Further, image files were processed with a bias for better recognition of the Kazakh text. There were created several versions for recognition of the Kazakh text. The versions were tested. After that the most accurate and stable version of the Kazakh text recognition was revealed. Further improvement of the component was continued based on this version.

As a result, the Kazakh text recognition component on Tesseract basis was created that has shown more exact recognition comparing with other OCR products. It has been successfully learned and improved OCR component for the Kazakh font.

4.2 Functions

Functions of component are in recognition of the Kazakh font and transformation of the text from the image into electronic form. The use of this component is to work on the Kazakh text optical recognition and provision as a cloud service.

In perspective, this recognition service can be applied to mobile gadgets like Android and iPhone for recognition of printed or typed Kazakh texts. At the subsequent learning and improving component its functions can be adapted for recognition of handwritten Kazakh texts.

Tesseract was originally designed to detect only the English text. Further efforts were made to change the core and learning system for use with other languages and characters from UTF-8. Tesseract can handle any character Unicode (UTF-8 encoded) but there are limitations with respect to certain languages.

4.3 Technology and Features of Realization

For the learning OCR component to recognition of the Kazakh language it is necessary to create multiple data files in a subdirectory *tessdata*, and then merge them into one file using the utility *combinetessdata*.

Format of file denotation – [lang].[fontname].exp[num]

For the learning, the following files are necessary:

1) [lang].freq-dawg
2) [lang].word-dawg
3) [lang].user-words
4) [lang].inttemp
5) [lang].normproto
6) [lang].pffmtable
7) [lang].unicharset
8) [lang].DangAmbigs (unicharambigs)

In result of learning from the received files the following file will be obtained:

9) [lang].traineddata

File *traineddata* is the union of the input files with the directory that contains the offset of known files types (a list of adopted names has been defined in the file *ccutil/tessdatamanager.h*).

For every image used in learning one needs to create a *box-file* (*box-file* is the file in which every character corresponds to the coordinates of the rectangle around it in the string of the image) which will contain the coordinates of every character in the image (fig. 3).

Some problems and their solutions:

– «Gluing» of characters when consecutive characters are defined as a single character for which just one coordinate is recorded in the string of image. Solution is to divide the characters and indication for each of them their position.
– Tesseract recognizes the Cyrillic character ‘Ы’ as two separate characters ‘Ь’ and ‘І’. In this situation, the characters coordinates should join in one coordinate.
– Tesseract also recognizes the quotation marks as two separate characters. In this case, the quotation marks coordinates should join too.
– If in the box-file character coordinates are specified inexactly or wrong then it is necessary to set them manually. The coordinate (0, 0) is in the bottom left.

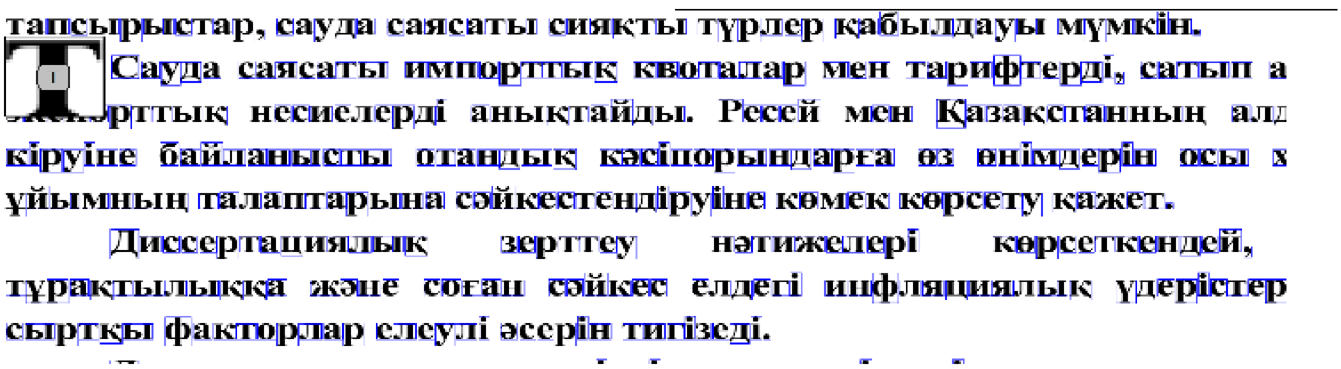
тапсырыстар, сауда саясаты сияқты түрлер қабылдауы мүмкін.
Сауда саясаты импорттық квоталар мен тарифтерді, сатып а
рттық несиелерді анықтайды. Ресей мен Қазақстанның ал
кіруіне байланысты отандық кәсіпорындарға өз өнімдерін осы х
ұйымның талаптарына сәйкестендіруіне көмек көрсету қажет.
Диссертациялық зерттеу нәтижелері көрсеткендей,
тұрақтылыққа және соған сәйкес елдегі инфляциялық үдерістер
сыртқы факторлар елеулі әсерін тигізеді.

Figure 3. Box-file (extraction)

In the process of learning Tesseract uses the dictionaries. For each language one can be used up to 5 dictionaries, as well as in the process of learning there is the ability to create your own dictionaries. Dictionaries are encoded with the help of extension *Directed Acyclic Word Graph (DAWG).* The list of words in dictionaries is formatted as ordinary text file in UTF-8 format with one word per string.

After all the files for learning has been generated, they should be collected into a single file, which will be responsible for the recognition of a new font in the Tesseract. To do this it is necessary to add the prefix of the Kazakh font (*kaz*) to all above-created files. The resulting file *[lang].traineddata* is the component that is able to perform OCR of the Kazakh language texts.

4.4 A Comparison of Recognition Quality

A comparative testing of OCR-tools Tesseract, Cuneiform, and ABBYY FineReader showed the following results (table 1). For the testing, we use texts on the Kazakh language (Times New Roman 14 points font) scanned with high resolution 600 by 600 dpi.

Table 1. Comparative testing

#	Program name	Percentage of correct recognition of the Kazakh text (%)	Percentage of correct recognition of the Kazakh characters (%)	The rate of recognition (full page per second)
1	Tesseract (without dictionary)	77	94	4
2	Tesseract (with dictionary)	88	99,52	4
3	Cuneiform	60	0	5
4	ABBYY FineReader	86	85	5

The main emphasis was made in the comparison on the recognition of the text and recognition of Kazakh characters.

At initial learning, to test the selected software the learning was held without creating dictionaries (see *Tesseract (without dictionary)*). While testing there was created two test packages, with the processing with dictionary and without it. Processing with dictionary increased the recognition of text approximately by 10%. The main advantages of the Tesseract in recognition of the Kazakh text with comparable with other OCR-tools processing rate are the correct recognition of Kazakh characters and the correct recognition of the text itself. Its additional merits include also the possibility of learning and modification based on the openness of the code. One of the main advantages of the Tesseract is that recognition is carried out without any additional external processing that allows in future creating them and significantly improving the recognition quality of text and Kazakh characters.

5 MORPHOLOGICAL MODULE

5.1 General Description

Morphological analysis and generation of the Kazakh word forms are carried out with help of finite state transducer (FST) for the Kazakh language constructed with using of Xerox Finite State Tool (XFST) technology [13]. Morphological analysis module is responsible for a correction of errors after optical recognition of text for the improvement of recognition quality.

XFST forms the representation of the word based on so-called two-level morphology [14]. Surface representation or lower side reflects how the word is usually written. Lexical representation or upper side expresses relevant lexical categories of word morphemes. Morphological analyzer is built on the basis of the finite state machine that allows achieving high performance.

In the general morphological analyzer consists of *Finite State Transducer, Finite State Machine u Controller* (fig.4)

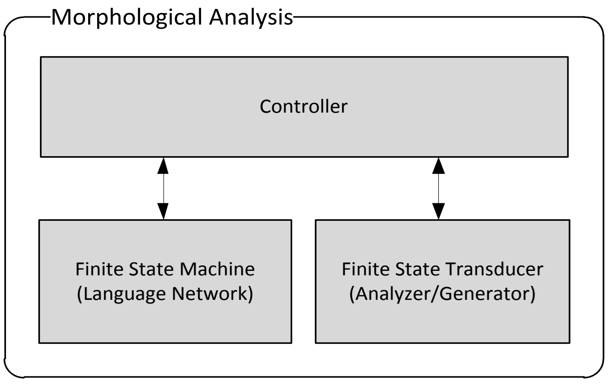
Morphological Analysis
Controller
Finite State Machine (Language Network)
Finite State Transducer (Analyzer/Generator)

Figure 4. Components of morphological analysis module

Finite State Transducer (fig.5). FST is two-level finite state automaton that carries out the morphological analysis and generation of word forms.

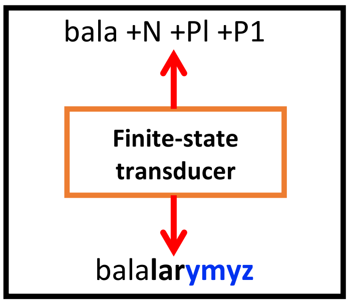

Figure 5. Finite state transducer

Finite State Machine (fig.6). This is finite state automaton describing complete network for the Kazakh language.

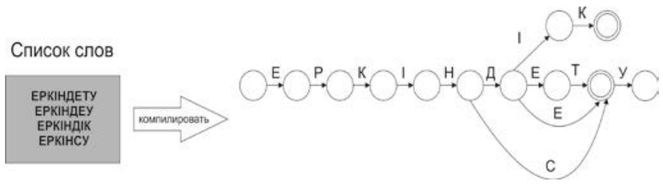

Figure 6. Finite State Machine

Controller. This application is responsible for the text processing such as a parsing of sentence, a verification of word correctness and a searching of possible corrections.

5.2 Analysis and Generation

Analysis (lookup):

– It begins from the StartState;
– Comparison of the string input characters with the characters lower-side on arcs, taking of input characters, and finding a path to a final state;
– If successful then return the string of upper-side characters on the path as the result;
– If not successful nothing to return.
– Generation (lookdown):
– It begins from the StartState;
– Comparison of the string input characters with the characters upper-side on arcs, taking of input characters, and finding a path to a final state;
– If successful then return the string of lower-side characters on the path as the result;
– If not successful nothing to return.

With using of finite state transducers, the following can be realized:

1. Combination of morphemes can be encoded as a network of finite states.
2. Rules defining form of every morpheme can be realized as FST.
3. Lexical network and rules transformations can be merged into single network (lexical transducer). It contains all morphological information on the language (fig.7).

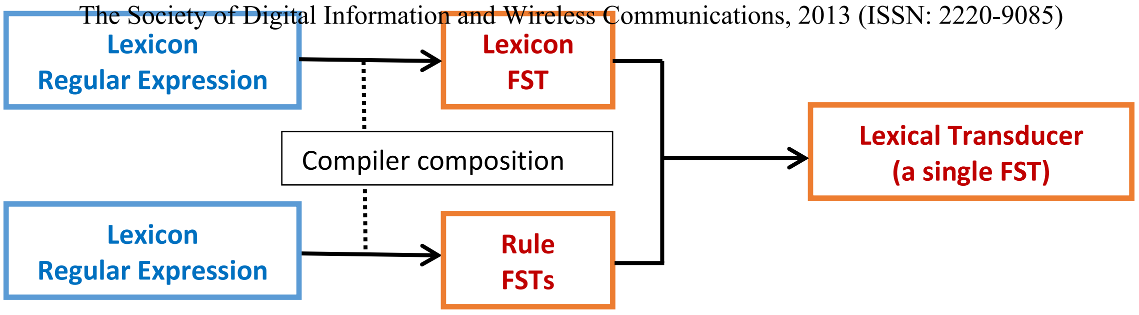

Figure 7. Lexical network

5.3 Methodology of Finite State Transducer Construction with Using of XFST

5.3.1 The Using of Finite Automata Methods for the Kazakh language

A creation of morphological analyzer and generator is based on the nominal paradigm of the Kazakh language. For the construction of finite automaton (transducer) we use XFST (Xerox Finite State Tool).
The build of morphological analyzer consists od the following stages:

1) The noun morphotactics description;
2) Description of morphophonemics rules;
3) Finite automaton network construction.

5.3.2 The noun morphotactics description

Morphotactics description is performed using special language lexc, which is high-level and declarative programming language. The finite automaton build is done using compiler lexc (LEXICON compiler) from the XFST tools.
Affix morphemes indicated by the so-called Multichar Symbols. They need to describe in the beginning of the file after the announcement Multichar_Symbols (Multichar_Symbols statement).

```
        Multichar_Symbols
        +N             ! Noun
        +Pl            ! Plural
        +Sngl          ! Singular
        +Poss12Sngl    ! Possession 1-2:Singular
        +Poss12Pl      ! Possession 1-2:Plural
```

After announcement of Multichar Symbols we describe lexc program body. It consists of LEXICONs. LEXICON is one of the components of the word morphemes. In the beginning, LEXICON Root must be declared. It corresponds to the Start State of resulting Network. There may be declared roots of words, if the file is constructed for one part of speech, or you can declare parts of speech, if the network is built for the whole language.
Further, after LEXICON Root all remaining morphemes are described according morphotactics rules.

```
LEXICON Root
        Noun;
LEXICON Noun
        ă̋ke            NounTag;
        tôlem          NounTag;
        ùj             NounTag;
LEXICON NounTag
%+N:0                  SingularPlural;
LEXICON SingularPlural
%+Pl:lAr/DAr                   #;
%+Sngl:0                       #;
```

Example above describes part of the noun morphotactics. LEXICON Root declares part of speech as Noun. Then it needs to describe LEXICON Noun, which contains roots of words: ă̋ke, tôlem, ùj; and specifies the transition to the next morpheme NounTag. LEXICON NounTag does not contain any morpheme that indicated zero in expression %+N:0, but it attaches Multichar Symbol +N, which will allow us to identify the word as noun in morphological analysis. Symbol % escapes symbol + in order to

use it as a symbol, because it is the operator of lexc and will be treated as an operator by the compiler, which in this case will cause an error. In LEXICON NounTag the next morpheme SingularPlural is indicated. Here we join plural morpheme lAr/DAr (lAr/DAr is an adopted designation of plural affix, which after application of morphophonemics rules is transformed into one of final endings: lar/ler, dar/der, tar/ter) and correspondent Multichar Symbol: +Pl for the morphoanalysis. Symbol # indicates the end of a word, i.e. this morpheme is the last one for the current paradigm.
In the same way a description of natural language all morphotactics is carried out.

5.3.2 Morphophonemics Rules Description for the Noun

Morphophonemics rules are described in XFST using Regular Expressions and Replace Rules.
General scheme of substitution rules is expressed by the following:

upper -> lower || left _ right

where upper, lower, left и right are regular expressions designating regular languages.
Example:

a -> e || [g | k | ņ] _ [g | k | ņ]

In natural language given rule can be read as "character 'a' is replaced to 'e' if one of the characters [g | k | ņ] is placed before it and one of the characters [g | k | ņ] locates after it.
Examples:

```
define plural1f   [ {lAr/DAr} -> {der} || FrontStem [LMNN | JZ] _ ];
define plural1b  [ {lAr/DAr} -> {dar} || BackStem  [LMNN | JZ] _ ];
```

where plural1f and plural1b are declared names of the replace rules; FrontStem, BackStem, LMNN, JZ are the regular expressions.

```
define Consonants [ b | v | g | ġ | d | ž | z | j | k | ķ | l | m | n | ņ | p | r | s | t | f | h | ḥ | c | č | š | ŝ | " | ‘ ];
# This expresses the consonants
define BackVowels [ a | o | u̇ | y | ë | û | â | u | i ];
# This expresses the back vowels
define FrontVowels [ấ | ô | ù | ì | e | u | i ]; # This expresses the front vowels
define BackStem [ Consonants* BackVowels+ Consonants* ]; # This expresses back syllable
define FrontStem [ Consonants* FrontVowels+ Consonants* ]; # This expresses front syllable
define JZ [ ž | z ]; # It expresses ž or z character
define LMNN [ l | m | n | ņ ]; # It expresses the characters l | m | n | ņ
```

Let us consider one of the rules:
[{lAr/DAr} -> {der} || FrontStem [LMNN | JZ] _];
This rule means that "Replace {lAr/DAr} to {der} if front syllable ending by one of the characters [l m n ņ ž z] is placed before it"
In the same manner all morphophonemics rules are described.

5.3.4 Finite Automaton (Transducer) Network Construction

After we described the morphotactics and morphophonemics we need to merge them in finite state transducer network for the analysis and generation. This is done using XFST. So, we get the following:

Upper side:
ấke+N+Pl ấke+N+Sngl ùj+N+Pl ùj+N+Sngl tôlem+N+Sngl tôlem+N+Pl
Lower side:
ấkeler ấke ùjler ùj tôlem tôlemder

When we apply to the input of transducer Upper side-forms we can generate the required word form, and applying to the input the Lower side we can implement word form analysis (fig.8).

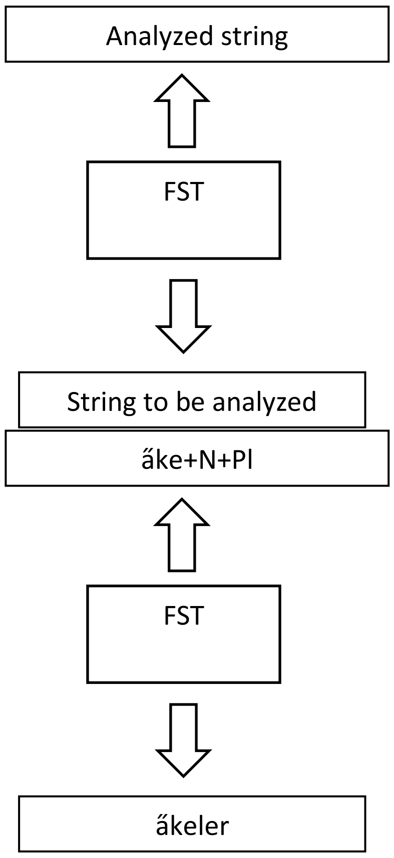

Figure 8. Generation/analysis by FST

5.4 Error Correction

A correctness of word is checked by the searching it in finite network. If the word is found, it will be correct. If the word is not in the network then or this word is mistakenly, or it is absent in our dictionary database.

At detection of the incorrect word, a search in the network is starting for possible corrections a given word. To search probable corrections it is used Levenshtein distance (or edit distance) [22]. In our case maximum distance equal to 1 is established. This means that it is allowable the correction of words with one error. The distance is established in the connection with the specifics of our task because prevalent number of errors after OCR is equal to 1.

There are 3 types of possible errors (operations):

1) An incorrect character (operation *replace*);
2) An extra character (operation *deletion*);
3) A missing character (operation *insertion*).

For each correction of possible errors the following algorithm of search in the network of a finite state automaton is implemented:

Step 1. Since the beginning of the string take one character from the string and memorize its index.

Step 2. Look for this character in the network at a depth equal to the index of the current character.

Step 3. If this character is found then increase the index, and continue search at the daughter nodes of found node from the *Step 1* for remaining characters of the string.

Step 4. If this character is not found then take all nodes on this level, and continue search at its daughter nodes from the next character.

Step 5. If the index has got the end of the string then return found current state, and finish the work.

For the preliminary testing we chose 5 pages of standard Kazakh text (1630 words of economic and business lexis) which previously were processed by our OCR-module for the Kazakh language text recognition. Then we used the generated from the dictionary word forms by morphological module for the comparing with words from OCR-processed text with the aim of search matching. Results are given in the table 2.

Table 2. Preliminary Testing of Error Correction in Selected Text (Economic and Business Lexis)

File name	The number of words	The number of incorrect words after OCR-processing (% from the number of words)	The number of corrected words (% from number of errors)	The number of introduced errors (% from the number of words)

scan1.tif	315	31(10%)	29(94%)	23(7%)
scan2.tif	295	14(5%)	11(79%)	19(6%)
scan3.tif	293	21(7%)	17(81%)	17(6%)
scan4.tif	352	41(12%)	32(78%)	21(6%)
scan5.tif	375	50(13%)	33(66%)	18(5%)
In total	1630	157(10%)	122(78%)	98(6%)

As seen our algorithm allows improve correction ratio of OCR-module in 1,67 times on average. Introduced errors (in last column of table 2) are connected with incompleteness of the Kazakh language paradigms formulation for another (than a noun) parts of speech. If we take into account only number of corrected which is caused namely formulated nominal paradigm then we get average 78%-level of correction. Further improvement can be achieved by the completeness of formulation of the Kazakh language paradigms, addition of specific professional lexicons, editing and cleaning up of dictionary base, and using error-tolerant algorithms of error correction [22].

6 WEB SERVICE DESIGN AND IMPLEMENTATION

6.1 General Description

Windows Communication Foundation (WCF) technology is the program framework that is used for exchange of data between applications. Interoperability principles laid down in this technology allow to organize work with other platforms for what technologies of platforms interaction are used.

Web services developed on the basis of WCF always are represented as an interface and class, which implements this interface. But a simple declaration of the interface does not make it a web service as a simple method declaration does not make it a method of the web service, etc. To do this, architecture WCF provides contracts:

– service contracts that define the signature of the methods supported by the web service;
– data contracts, which define the types of data received and sent to the web service;
– message contracts define the parts using SOAP messages, and allow you to manage these parts;
– error contracts determine what errors are triggered by the web service, and also as a web service will process them and distribute to your users.

Based on the capabilities of WCF a comparative analysis of the Windows Communication Foundation and web services ASP.NET was done (tabl.3).

Table 3
Comparative characteristics of WCF and web services ASP.NET

WCF	Usual web services ASP.NET
Publication	
IIS, WAS	IIS
Access protocols	
HTTP, TCP, MSMQ, P2P, protocol of named channels	HTTP
Functional capabilities	
Secure data transmission, guaranteed delivery, transaction support, sending long-term messages	Secure data transmission
Multithreading	
Supported	Not supported
Communication channel provision	
BasicHttpBinding, WSHttpBinding, WSDualHttpBinding, etc	SOAP, XML

Web service was designed, starting with the definition of interfaces and connections between them (fig.9).

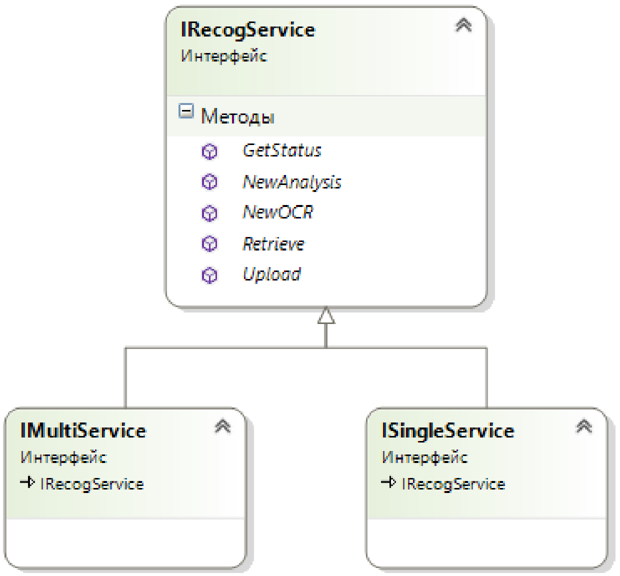

Figure 9. Diagram of web service interfaces

According designed diagram of classes the following interfaces are identified:
– IRecogService contains a declaration of all methods, supporting by the web service;
– IMultiService is an interface inherited from interface IRecogService, representing itself the parallel processing of multiple-page documents;
– ISingleService, interface inherited from interface IRecogService, representing itself the parallel processing of single-page documents.
To provide the web service by the mechanisms of text recognition and morphological analysis the following interfaces were created:
– IOCRService that contains the signature of the method responsible for the process of text recognition;
– IAnalysisService contains the declaration of the method responsible for the process of morphological analysis.
Then implementing classes were designed, the classes diagram of which is shown on fig. 10.

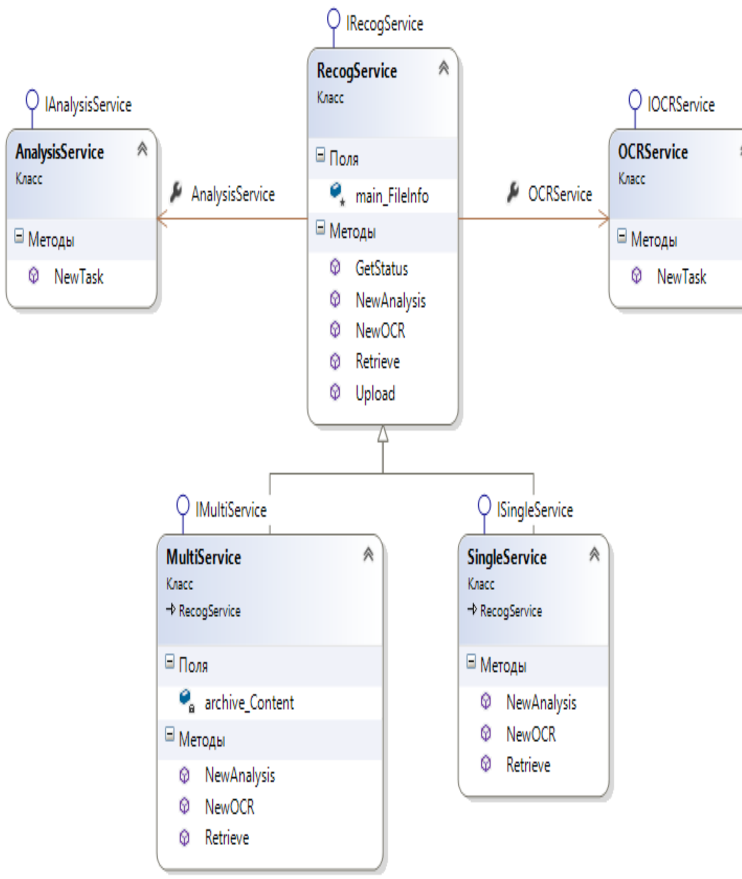

Figure 10. Diagram of implementation classes

The following classes were developed according designed diagram of classes:
1. RecogService implements the interface IRecogService. It contains a field *main_FileInfo*, which is a reference to the class of information about the file. GetStatus method performs an HTTP request in the result of which a web service client receives the data, such as current status of the task and the percentage completed. The Upload method is responsible for the transmission of files on the server;
2. SingleService class is inherited from the class RecogService. Because of this, class SingleService inherits the implementation of the methods defined in the base class. Business logic of method NewOCR is responsible for the development work, adding a new task, and the approval on the run (all these actions implement the mechanism of performing the recognition of one-page documents). Method *NewAnalysis* implements the logic of the single-page document analysis. Primarily this method

performs the job creation process, and then you add new tasks in the job and at the end you can approve of the job. *Retrieve* method generates a URL on the result of the processing;
3. MultiService class inherits RecogService. Method NewOCR is responsible for creating jobs, add new tasks, and approval at the start, i.e. it is the mechanism of recognition of multi-page documents. Method NewAnalysis contains the business logic that performs unpacking of the archive, creation of jobs, add new tasks, the analysis of the text, and approval of the job. Method *Retrieve* is responsible for the archiving of results of processing and generation of the URL to the archive.
Next step was to create a task scheduler for parallel processing. To do this, a static class Scheduler was realized.
The following methods were realized in class Scheduler:
1. GetNodesList method is responsible for returning a list of the nodes names of a cluster;
2. NewJob method performs the creation of tasks on parallel processing on the computing cluster;
3. SubmitJob method performs the approval of the job that runs parallel processing.
The logic of adding new tasks to the job was realized in the following implementing classes:
1. AnalysisService class is responsible for the implementation of the method NewTask, which in turn performs the task creation of text morphological analysis;
2. OCRService class is responsible for the implementation of the method NewTask, which performs the task creation for recognition of the text.
In the result, parallel processing mechanism was realized in the hardware and software of the Kazakh National Technical University computing cluster Fujitsu-Siemens.
All the methods interact with Web service HPC Pack [23]. As a task for each project string was supplied that launches the Tesseract application with input parameters: original image and name of resulting file. The first action of a user is a transfer of file. This action is carried out by the proxy-client call to Web service that calls Web service method which is responsible for the file loading. After download proxy-client returns the user a message about a successful upload or, in case of any errors by the user, it returns information about the cause of the error. Further the user starts the recognition process. Proxy-client calls the method of adding the project to the queue of HPC cluster, and also it generates the task for the recognition of one page of scanned document. The next step is a direct run of created project. During whole computation proxy-client returns information on current state of the project and percentage of the processing implementation. After finishing of recognition process the user is returned the result of recognition as URI-reference for download of the text file.

6.2 Preliminary Testing

After realization of a beta-version of Web service it was carried out a preliminary testing of correctness of recognition the text on the Kazakh language. As input data we use 5 pages A4 format with the text on the Kazakh language (fig.11). The result of recognition was checked manually.
In fig.11 the words with errors were marked by yellow color. This was done for all 5 pages. Results are shown in table 4.

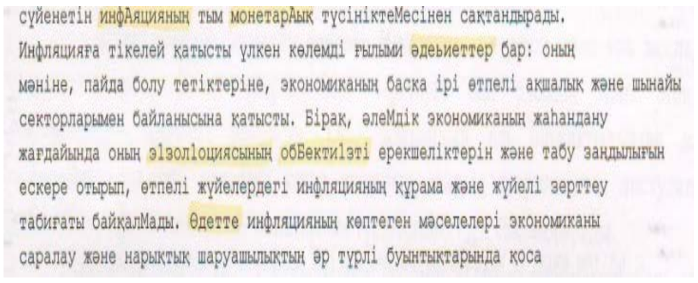
суйенетін инфАяцияның тым монетарАык түсініктеМесінен сақтандырады.
Инфляцияға тікелей қатысты улкен көлемді ғылыми әдеьиеттер бар: оның
мәніне, пайда болу тетіктеріне, экономиканың баска ірі өтпелі ақшалық және шынайы
секторларымен байланысына қатысты. Бірақ, әлеМдік экономиканың жаһандану
жағдайында оның әlзолlоциясының обБектиlзті ерекшеліктерін және табу заңдылығын
ескере отырып, өтпелі жүйелердегі инфляцияның құрама және жүйелі зерттеу
табиғаты байқалМады. Өдетте инфляцияның көптеген мәселелері экономиканы
саралау және нарықтық шаруашылықтың әр түрлі буынтықтарында қоса

Figure 11. Scanned text (extraction)

Table 4. The results of the text recognition

#	The total number of words on the page	The number of incorrectly recognized words

1	315	31 (10%)
2	295	14 (5%)
3	293	21 (7%)
4	352	41 (12%)
5	375	50 (13%)
Total:	1630	157 (10%)

Thus the preliminary testing of the beta-version of the Web service has demonstrated the successful recognition with a 10% error in the average, or 90% successfully recognized text. This result demonstrates a more efficient process for recognition than paid application ABBYY giving on average of 85% successfully recognized text.

Additionally the beta version of the Web service was testing on performance. The main objective of this testing was to determine how effective is the recognition with multiple computing nodes of HPC cluster compared with one computing node. Table 5 presents results showing decreasing of the time processing especially considerable for the large arrays of texts. As initial data for the testing there were used scanned pages A4 format of the Kazakh texts.

Table 5. Comparative data on the processing time

Number of pages	**One computing node**	**4 computing nodes**
1	3 sec	3 sec
10	126 sec	56 sec
100	6170 sec	528 sec

7 CONCLUSIONS

We presented in the paper the integrated high-performance and Web-oriented computer recognition system for the Kazakh language text. The design and integration methodology of the system are based on service-oriented architecture that allows provide an easy, flexible, and extensible integration of any kind of language service into any desktop client, and the user is not worried about the realization of relevant services. We have designed and built 4-tier SOA on the basis of W3C Web service standard. We published the Web service on server, and now it is accessible for the calling from clients' applications. The use of Web service allows a developer of the client application very easy to integrate his/her application into our architecture, and it does not need to worry about implementing remote procedure calling or writing different code for network interaction. Use the high-performance cluster shows significant advantages in the Kazakh text processing, especially for large data arrays.

The main objective of the developing system is to provide any person easy access to text and documentation in Kazakh language with the following possibility of editing and manipulation different docs through respective Web services OCR and morphological analysis/correction. The Kazakh OCR module currently has some advantages in accuracy of the Kazakh text recognition comparing with other known OCR systems. Further the quality of OCR would be improved via augmentation of Kazakh dictionary's database, and more precise determination and extension of the Kazakh fonts. The morphological module implements subsequent analysis and correction of recognizable Kazakh text and shows acceptable correction after OCR processing. In the future quality of the morphological processing would be improved via the completeness of formulation of the Kazakh grammar paradigms, addition of specific professional lexicons, addition, editing and cleaning up of dictionary's database, and using error-tolerant algorithm of error correction. In prospect, our system would be developed in direction of interactive Web portal with complete functionality providing a broad spectrum of NLP Web services to all interested parties. Basing on developed Web portal in the future there can be solved a wide range of problems associated with the recognition of Kazakh texts, namely the

Kazakh text mining, the Kazakh language corpora, Kazakh segment of Wordnet, machine translation, questioning-answering systems, information retrieval systems, etc.

8 ACKNOWLEDGEMENTS

We would like to express a gratitude to reviewers for valuable comments.
The work supported by the grant № 0914GF-OT of Ministry of Education and Science of the Republic of Kazakhstan.

9 REFERENCES

1. Baskakov N.A.: Altaic Family of Languages and its Study. Nauka publishers, Moscow, Russia (1981). (In Russian).
2. Baskakov N.A., Khasenova A.K., Issengaliyeva V.A., and Kordabayev T.R. (eds.): A comparative grammar of the Russian and Kazakh languages. Morphology. Nauka publishers, Alma-Ata, Kazakhstan (1966). (In Russian).
3. Witte R., Gitzinger T.: A General Architecture for Connecting NLP Frameworks and Desktop Clients using Web Services. In: Proc. 13th International Conference on Applications of Natural Language to Information Systems (NLDB 2008), pp. 317–322, Springer LNCS 5039, London, UK (2008).
4. Web Services Architecture, http://www.w3.org/TR/ws-arch/
5. Web Services Description Language (WSDL), http://www.w3.org/TR/wsdl
6. Simple Object Access Protocol (SOAP), http://www.w3.org/TR/soap/
7. Bakalov F., Sateli B., Witte R., Meurs M.-J., König-Ries B.: Natural Language Processing for Semantic Assistance in Web Portals. In: IEEE International Conference on Semantic Computing (ICSC 2012), pp. 67-75, IEEE, Palermo, Italy (2012).
8. Cunningham H., Maynard D., Bontcheva K., Tablan V., Aswani N., Roberts I., Gorrell G., Funk A., Roberts A., Damljanovic D., Heitz T., Greenwood M.A., Saggion H., Petrak J., Li Y., Peters W.: Text Processing with GATE (Version 6). University of Sheffield, Department of Computer Science (2011).
9. Google OCR web-service, http://drive.google.com
10. ABBYY OCR web service, http://finereader.abbyyonline.com
11. Online OCR web service, http://onlineocr.net
12. Goto H.: An Overview of the WeOCR System and a Survey of its Use. In: Proc. Image and Vision Computing, pp. 121–125, Hamilton, New Zealand (2007).
13. Beesley K.R., Karttunen L.: Finite State Morphology. CSLI Publications, Stanford, CA (2003).
14. Koskenniemi K.: Two-level morphology: A general computational model for word-form recognition and production. Publication 11, University of Helsinki, Department of General Linguistics, Helsinki, Finland (1983).
15. Attia M., Pecina P., Toral A., Tounsi L., van Genabith J.: An Open-Source Finite State Morphological Transducer for Modern Standard Arabic. In; Proc. 9th International Workshop on Finite State Methods and Natural Language Processing, pp. 125–133, Association for Computational Linguistics, Blois, France, (2011).
16. Alegria I., Artola X., Sarasola K., Urkia M.:. Automatic morphological analysis of Basque. Literary and Linguistic Computing, 11, pp. 193-203 (1996).
17. Çöltekin C.: A Freely Available Morphological Analyzer for Turkish. In: Proc. 7th International Conference on Language Resources and Evaluation (LREC'10), pp. 820-827, Valletta, Malta (2010).
18. Tantuğ C., Adalı E., Oflazer K.: Computer Analysis of the Turkmen Language Morphology. Lecture Notes in Computer Science, 4139, pp. 186-193 (2006).
19. Kairakbay B., Zaurbekov D.: Finite State Approach to the Kazakh Nominal Paradigm. In: Proc. 11th International Conference on Finite State Methods and Natural Language Processing, pp.108-112, Association for Computational Linguistics, St Andrews, Scotland (2013).
20. Erl T.: SOA Principles of Service Design. Prentice Hall/PearsonPTR, New Jersey (2007).
21. Tesseract, http://code.google.com/p/tesseract-ocr/
22. Oflazer K.: Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. Journal of Computational Linguistics, 22, pp. 73-89 (1996).
23. Methods and classes description library accessible through HPC Pack web service, http://msdn.microsoft.com/en-us/library/hh560258(v=vs.85).aspx