Satbayev University Institute of Automation and Information Technologies

UDC 004.934

BEKARYSTANKYZY AKBAYAN

Development of end-to-end system for automatic recognition of speech in agglutinative languages

8D06103 – Management information systems

Thesis for the Degree of
Doctor of Philosophy (PhD)

Scientific consultants
doctor PhD,
associated professor
O. Mamyrbayev

doctor PhD,
professor
Mendes Mateus,
(Portugal)

Republic of Kazakhstan
Almaty, 2023

CONTENT

NORMATIVE REFERENCES

In this dissertation work were used references to the next standards:

ГОСО РК 5.04.034-2011. Государственный общеобязательный стандарт образования Республики Казахстан. Послевузовское образование. Докторантура. Основные положения: утвержденный приказом Министра образования и науки Республики Казахстан от 17 июня 2011 года, №261.

Положение о диссертационном совете НАО «КазНИТУ имени К.И.Сатпаева». П 029-04-01.01 – 2021.

Instruction for the preparation of dissertation thesis and author's anstract / MHES Kazakhstan, External attestation committee. Almaty 2004.

GOST 7.1-2003. Bibliographic report.

DEFINITION

In this dissertation the following terms with corresponding definitions are used:

Audio – binary information which stores sound data. In the context of this work recorded human speech.

Automatic speech recognition – ability of computing machines to extract speech data from audio files and to transfer it into text format.

Dataset – set of collected data which could be used for further analysis. In the context of this work, audio files and their transcription in the text format.

Neural networks – system of neurons which can be organic or artificial.

ABBREVIATIONS

AI	– Artificial Intelligence
ANN	– Artificial Neural Network
ASR	– automatic speech recognition
BLSTM	– Bidirectional Long-Short Term Memory
CNN	– Convolutional Neural Networks
CTC	– Connectionist Temporal Classification
DLUNN	– deep locally unified neural network
GMM	– Gaussian Mixture Models
GPU	– Graphical Processing Units
GRU	– Gated Recurrent Units
HMM	– Hidden Markov Models
HMM/DNN	– HMM/Deep Neural Networks
LM	– Language Model
LSTM	– Long-Short Term Memory
MLP	– Multi-Layer Perceptron
MSPC	– Multi-Scale Parallel Convolution
NLP	– Natural Language Processing
TPU	– Tensor Processing Units

INTRODUCTION

Relevance of the research topic. Automatic Speech Recognition (ASR) systems are nowadays widely used in different areas of human life, in order to make it easy for people to interact with computer systems and different applications. For example, smart assistants, smart home systems, commercial and subtitling applications allow to control computer systems without touching, from a distance. Moreover, ASR can make it easy to impaired persons to use electronic devices. For example, Ahmad et al. study the ways of building ASR systems for people with dysarthria. Dysarthria is the type of muscle defects responsible for articulation. ASR development for people with this problem helps them interact not only with digital systems but also with other persons. Next useful example of involving ASR systems is the assessment of hearing loss . This type of system can forecast the level of hearing injury by the quality of answers to questions. But these opportunities are available only for people who knows widely used languages, like English, Chinese and Russian. ASR development for low-resource languages still needs enormous efforts, like data collection and preparation, testing of well-known recognition architectures, as well as studying the ways of adjusting state-of-the art architectures for exact languages or the group of languages. The Turkic group of agglutinative languages, to which the Kazakh language belong to, has many low-resource languages. Besides the problem of shortage of data to train, agglutinative languages have other problems stated out below.

Development of ASR systems for agglutinative languages are complex processes due to the their morphological complexity and richness of grammatical forms in these languages. According to this, development and fine-tuning ASR systems for agglutinative languages require additional studies and specific approaches. Below is a list of several challenges ASR systems for agglutinative languages can face:

1. Morphems' analysis and separation. Morphemes in agglutinative languages can be joined and can be complex, which makes difficult the process of splitting and analysis of morphemes in speech recognition.

2. Grammatical ambiguity. Agglutinative languages can have a variety of grammatical forms which can lead to ambiguity. For example, one expression can have different meanings due to a context. This makes it difficult to exactly recognize and interpret a grammatical unit.

3. Variety of word-formation rules: in agglutinative languages usually there exist various rules of word formation which determine how to concatenate affixes with the root of a word. This also requires complex models and rules for processing those rules in ASR systems.

There is an enormous number of studies dedicated to the development of specific approaches and models to get reliable ASR systems for agglutinative languages. Authors of propose a language model, based on morphemes, where morphemes are understood as any of prefix, root or suffix in a word. As a result, authors got an automatic speech recognition system with large vocabulary. studies

the performance of transformer-based CTC system which depends on context, trained with the word pieces taken as training units. Authors note the effectiveness of their method not only for English and German, but also for one of agglutinative languages - Turkish language. made research on applying transformer architecture for a morphological disambiguator using Turkish language. This disambiguator can be used in any of NLP tasks and speech recognition is not exception here. The transformer architecture performed well also for another agglutinative language - Hindi . Here the transformer architecture along with Connectionist Temporal Classification (CTC), Language Model (LM) showed the lowest error rate for Hindi language: 3.2%. One more example of using Transformer architecture in ASR development for agglutinative language is . Here the author compares the performance of Transformer-XL architecture with LSTM and concluded that perplexity improvement achieved 29% and Word Error Rate (WER) was decreased to 3% for Finnish language. In authors state out that there are the most widely used and effective end-to-end architectures for automatic speech recognition: connectionist temporal classification and attention-based mechanism. Also, in this work mentioned the lack of transcribed audio-text pair resources for agglutinative languages to train in order to develop reliable automatic speech recognition systems.

According to the mentioned researches for agglutinative languages it was noted that dictionary enlarging and transformer architecture are the most effective approaches for developing end-to-end automatic speech recognition systems for agglutinative languages. Moreover, the lack of data to train and common morphological rules and similar soundings of languages from Turkic family of agglutinative languages served as a basis for providing pooling experiments, like transfer learning and multilingual training for these languages.

Purpose of the dissertation. The present dissertation was developed with the aim of studying the ways of improving ASR performance for agglutinative languages on the example languages from Turkic family.

Research objectives.

1. Analysis of existing ASR approaches for general cases and for agglutinative languages.
2. Extension and development of data corpus for agglutinative languages.
3. Development of models and methods for automatic speech recognition of agglutinative languages.
4. System development for automatic recognition of speech in agglutinative languages.

Object of the study. Modern automatic speech recognition methods and approaches, especially pooling metods like multilingual training and transfer learning.

Subject of the study. Agglutinative languages of Turkic family, methods of Machine learning, namely neural networks for Automatic Speech Recognition: attention mechanism, convolutional neural networks, performance improvement methods for critically low-resource languages, a moment from Natural Language

Processing methods: word embeddings, and demonstrative Telebot which uses trained ASR model and web-application, available to translate audio files to a text.

Research methods. Machine learning methods, automatic speech recognition methods and technologies, natural language processing methods, mathematical statistics and probability theory.

Scientific novelty of the research. The thesis proposes scientific and practical novelties, which were applied to practical tasks, especially for improving end-to-end automatic speech recognition systems for agglutinative language - focusing Kazakh language and which can easily be applied to other languages. Moreover, contributions were made to the increase of training data size for Kazakh language. The main positive results, obtained during the research are listed below:

1. Was developed data corpus for agglutinative languages.
2. Were developed effective models for recognition of agglutinative languages from Turkic family: transfer, multilingual, extended language model.
3. System for automatic speech recognition for agglutinative languages.

Theoretical and practical significance of the research. Theoretical importance of the research is that, it proposes the possibility of improving ASR performance improving only the language model with external "Big Text", and shows the possibility of improving performance for all languages included in multilingual training, transfer learning for languages from one family group. The possibility of applying all mentioned theoretical statements to train ASR for agglutinative languages of Turkic family shows the practical significance of the current thesis. Moreover, text processing algorithms can be applied to wide range of text processing tasks. Audio-text pair data, collected during research, can be used in different speech processing tasks.

Statements to be defended. Next statements are proposed to the be defended:

1. Dataset for agglutinative languages was developed.
2. Methods of improving ASR for agglutinative languages were improved.
3. ASR system for agglutinative languages was developed.

Reliability degree and approbation of the results. Researches and their results related to the thesis topic were presented and discussed in different conferences and seminars and some of them were published. Moreover, the author was awarded with certificates as a seminar speaker, for the best presentation (Appendix A):

1. End-to-End Model Based on RNN-T for Kazakh Speech Recognition // 3rd International Conference on Computer Communication and the Internet (ICCCI) (Tokyo, 2021 – 25-27 June).
2. Certificate to the seminar speaker on the topic "Improved Speech Recognition for Agglutinative languages", Coimbra Institute of Engineering (ISEC), (Coimbra, 2023 – 21 April).
3. Certificate for the best presentation speech, "Improve Automatic Speech Recognition for Kazakh Language using Extended Language Model", "ACeSYRI Young Researchers School" (Almaty, 2023 – 5-10 June).

4. Improve Automatic Speech Recognition for Kazakh Language Using extended Language Model // 21 st scientific conference, (Riga, 2023 – 20-21 April).

5. Automatic Speech Recognition Improvement for Kazakh Language with Enhanced Language Model // Recent Challenges in Intelligent Information and Database systems. ACIIDS 2023. Part of The Communications in Computer and Information Science book series. – 2023. - Vol. 1, - P.538-545 (Springer, Cham).

Personal contribution of the researcher. PhD candidate independently performed and solved the tasks of the PhD thesis. The author designed and implemented end-to-end models for Kazakh and Agglutinative languages. Made own contribution in expanding data corpus for Kazakh language. Designed and performed experimental tests and assessments of the models, both existing and improved models.

The connection of the dissertation topic with the plans of research work. Research works under the research topic were conducted within the grant projects: “Development of an end-to-end automatic speech recognition system for agglutinative languages” (2020-2022, governmental registration number: 0120PK00344) in the Institute of information and computational technologies SC MHES RK.

Main results of the dissertation research. There were four papers published under the research topic, one of which is published in a periodical journal with non-zero impact-factor and indexed by databases Scopus and Web of Science, 3 papers published in the journals recommended by the Control Committee in the sphere of education and science of MHES RK:

1. Identifying the influence of transfer learning method in developing an end-to-end automatic speech recognition system with a low data level // Eastern-European Journal of Enterprise Technologies. – 2022. - Vol. 1, №115. - P. 84-92 (Scopus, percentile 34);

2. Integrated Automatic Speech Recognition System for Agglutinative Languages // News of the National academy of sciences of the republic of Kazakhstan. - 2023. - Vol. 1, №345. - P. 37-49.

3. Transfer learning for an integrated low-data automatic speech recognition system // Scientific and technical journal "Bulletin of the Almaty University of Power Engineering and Telecommunications". – 2023. - Vol. 1, №.60. - P. 185-198.

4. End-to-end speech recognition systems for agglutinative languages // Scientific Journal of Astana IT University. - 2023. - Vol. 13. - P. 86-92.

5. Author’s certificate "Software Product UniCodeKaz" №38545 from 21.08.2023 (Appendix B).

6. Author’s certificate "System of transcribing audio files to text" №38833 from 31.08.2023(Appendix B).

Structure and size of the thesis. Dissertation thesis consists of the Introduction, 4 sections, conclusion, bibliography from 163 references, and 5 appendixes. Work is presented in 107 pages and contains 38 figures, 16 tables and 68 equations.

1 STATE OF THE ART

This chapter presents a comprehensive review of related work, both in agglutinative and non-agglutinative languages.

1.1 Methodology

For this section, publications about automatic speech recognition were searched in four topics: first ASR examples, worldwide examples, examples for agglutinative languages and, finally, for Kazakh languages. The keywords used were names of first significant speech recognition systems, names of main methods and architectures for Automatic Speech Recognition (ASR), like Hidden Markov Models (HMM), Recurrent Neural Networks (RNN), attention mechanism, ASR for agglutinative languages and Kazakh languages. It was decided to search for related documents from scientific databases like Scopus, Google Scholar and Semantic Scholar as it is possible to find the results of appropriate scientific researches from these resources. Search with keywords in Scopus database gave 24127 documents, while Google scholar retrieved 2320000 results and Semantic Scholar gave a list of documents with 226000 entries. Due to the fact that the taken lists of documents were very huge, it was decided to narrow down the search process by formulating more detailed keywords: “Multilingual speech recognition”, “Conformer for Agglutinative languages”, “Attention mechanism in speech recognition”, “Speech recognition for Kazakh language”, “Hidden Markov Models in speech recognition”, “Recurrent Neural Networks for speech recognition” and so on. The most appropriate and Open Access documents were chosen from the retrieved list of documents. Detailed results for the basic search keywords are given in Table 1.

Table 1 – Number of results retrieved from databases for different search keywords

Keywords	Databases		
	Scopus	Google Scholar	Semantic Scholar
Multilingual speech recognition	1560	195000	110000
Conformer for Agglutinative languages	1	510	11800
Attention mechanism in speech recognition	1435	2470000	937
Speech recognition for Kazakh language	43	21000	43
Hidden Markov Models in speech recognition	7360	319000	18600
Recurrent Neural Networks for speech recognition	2918	2	20500

1.2 First ASR examples

The first speech recognition system was called “Audrey.” It was developed in Bell laboratory in 1952 and focused not on words, instead it converted speech signals to numbers . In other words, it could recognize numbers from zero to nine spoken by an exact person. Further, there was IBM’s “Shoebox” product introduced in 1962. It was able to understand 16 words of English language, which consisted of digits and names of simple arithmetic operations, like “plus”, “minus”, “total,” and could print

results of basic arithmetic calculations . Shoebox received commands via microphone and the microphone converted speech into impulses of the electricity. Then a measuring scheme classified perceived impulses according to properties of word sounds and it activated an adding machine.

Significant progress was reached in the system "Harpy," developed by Carnegie Mellon University in 1976. This system was based on phoneme recognition and could understand more than 1000 words. The segmenter based on the feature extraction was used to provide initial inputs of symbols. Then segmentation followed by labeling, where the middle points of each segment were compared by templates saved earlier and adjusted.

Application of Hidden Markov Models [13, 14] based on statistical predictions, brings a breakthrough in the 1980's to speech recognition, although researchers studied this algorithm in the field of speech recognition since 1958. Ability of speech recognition applications grew to several thousand words, because HMM allows to predict the most probable sequence of sounds in a speech.

1.3 Introduction of neural models

Next epoch in speech recognition had started with the introduction of Recurrent Neural Networks (RNN), which brought the concept of "deep learning". They are capable of working with sequential and time-based data and can learn features and long-term dependencies. They are also able to map sequences of inputs to output sequences at the current time-slot and predict the following timestamp's sequence. The first significant advances in RNNs were reached in 2006, with solving the issue of optimization: authors of introduced the idea of using fast and greedy algorithms to initialize a slower procedure of learning, which fine-tunes the weights. Next study showed the advantage of gradient clipping, improved momentum techniques and obtained improvements in provided experiments on music and text data. Development of various types of RNNs like Multi-Layer Perceptron (MLP), bidirectional RNN, Convolutional Neural Networks (CNNs), Long-Short Term Memory (LSTM), Bidirectional Long-Short Term Memory (BLSTM), and Gated Recurrent Units (GRU) allowed to build different architectures which receive input data at one end and give output data at another end, which is called End-To-End (E2E) architectures.

E2E brought to speech recognition a simplicity. If traditional ASR systems included many supervised stages, like separate training of acoustic, pronunciation and language models, E2E maps entered acoustic sequences into a sequence of letters, words, because it can play the role of these three models inside a single Neural Network (NN).

Figure 1 [19, p. 052068-2] and Figure 2 [19, p. 052068-2] show the difference between conventional systems and E2E speech recognition systems.

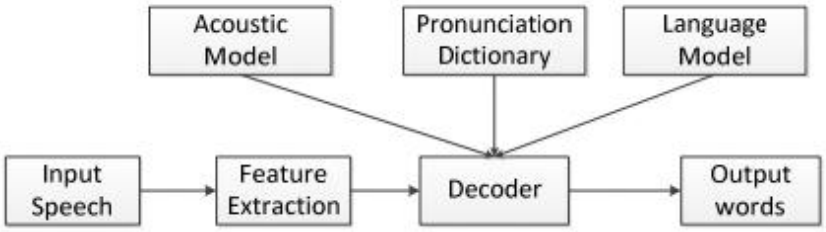

Figure 1 – Example of conventional ASR

Figure 2 – Example of E2E ASR

1.4 Modern models for ASR

Currently, RNN-Transducers (RNNT), Connectionist Temporal Classification (CTC) and attention-based encoder-decoder are the widely used E2E approaches [20, p. 1436] in speech recognition. Attention mechanism was adopted to speech recognition from sequence processing and its introduction started the new area in Natural Language Processing. This also touched the speech recognition as a part of NLP. There is an impressive number of research projects which were done about the application of attention mechanism to speech recognition, such as models with attention mechanism had outperformed its predecessors like RNN and CNN . For example, in authors propose a new type of architecture based on attention mechanism, calling it “Squeezeformer”. This architecture is a modification of so-called conformer architecture, which is mainly based on attention mechanism .

One of the best speech recognition results among end-to-end architectures was obtained by convolution-augmented transformer, which is very popular as Conformer [24, p. 5039]. This architecture takes advantages of both approaches included to its development: transformer and CNN architecture. Because the transformer can manage long-term dependencies, while CNN can capture local relations in a sequence. Trained on the LibriSpeech corpus model, it achieved a Word Error Rate (WER) of 4.3%, without language model, and a WER of 3.9% with language model.

All mentioned E2E architectures work properly only with large amounts of data to train. But annotating of speech data with its text equivalent is not easy and not a cheap process, especially for the tasks of processing the dialects of proper languages or for the speech recognition of specific tasks. Merging different corpora of several languages, providing transfer learning among relevant languages, can give promising results [26-28].

1.5 Agglutinative language examples

There was also research for agglutinative languages from Turkic family, by the representatives of their speakers and Chinese scientists. Researches of some scientists are dedicated to the collection of datasets in order to make them open-source: one of

Uzbek language was developed open-source, with transcribed audio data which totally consists of 105 hours . The quality of the data was tested by using these data to train ASR with DNN-HMM and different end-to-end architectures, like LSTM RNN, Transformer and Conformer. The best result was obtained with E2E-Conformer architecture, WER on test set was 17.4%.

In spite of the lack of data to train, some authors attempted to implement ASR for real applications. In was implemented an ASR using an Azerbaijani speech dataset for emergency call centers. Here authors used 27 hours of dialogue dataset and 53 hours of summary dataset and trained a model on Kaldi. Kaldi is open-source toolkit for speech processing tasks proposed in 2011 . The results of GMM/HMM and DNN/HMM were compared. Authors realized that use of spelling correction in datasets before training and application of DNN/HMM for acoustic modeling and using trigram in language modeling are effective in the recognition of emergency conversations. One more example of using speech recognition for exact task for Azebaijani language is provided in . Here authors test CMUSphinx and Kaldi speech recognition tools on 4 hours of specific data for taxi call applications and realized that Kaldi gives more accurate results over CMUSphinx.

In authors propose the way of improving hybrid CTC-Attention architecture by improving feature extraction where they use different sizes of convolutional kernels. It gives advantage in fusing features of different scales. Moreover, authors improved attention mechanism by using previous attention weight in the calculation of attention weights. Authors used BERT model to initialize language model in decoding. Training the proposed model on Turkish (35 h) and Uzbek (78 h) datasets from Mozilla's Common Voice dataset reduced WER by 7.07% and 7.08%, respectively.

1.6 Kazakh Language examples

Kazakh language is also an agglutinative language from Turkic family. Currently, among all researches for Kazakh language, the best result was obtained in . Transformer architecture in joint use with CTC loss function was trained on the 400 hours speech data. In the results, CER was 6.2% and WER -13.5%. The involvement of language model to joint decoding increased the model size, but reduces CER and WER by 3.7 and 8.3%, respectively. also studies the joint use of CTC objective function and attention mechanism. Here authors highlight the rapidness of training and decoding process with the application of mentioned approach.

Authors of proposed Conformer model for Kazakh language boosted by low-rank approximation for multi-headed self-attention and balanced softmax-function which uses penalty algorithm for the words with high frequency. Here low-rank approximation helped to decrease model size to 20.2 MB by reducing the number of parameters by 5.3 M in comparison with baseline Conformer-CTC architecture. But this algorithm is not stable, because the approximation by decreasing the rank size also affects recognition quality: word error rate goes up, keeping rank size high does not give expected result: number of parameters stays as in baseline model. For example, in the case of training with baseline Conformer-CTC, the number of

parameters is 47.6 and WER is 10.36. Adding low-rank to baseline model with rank size equal to 128 keeps all training results as in the base case. Further decrease for different values of rank size gradually decreases the number of parameters, but increases error rate for word recognition in parallel.

Transfer learning is also applicable for Kazakh language. In was tested transfer learning for Kazakh language over the weights of a model for Russian Language, regarding to the similarity of alphabets and similar sounding of all intersecting letters. Here authors trained 20 hours of Kazakh language dataset on the model trained on 100 hours of Russian dataset. In the end result, authors got letter error rate decreased to 32%. It is observed that transfer learning for Kazakh language with other language from one language family can help improve recognition for all languages. Transfer learning on the example of Kazakh and Azerbaijani languages reduced phoneme error rate to 14.23%.

2 THEORETICAL FOUNDATIONS

This chapter reviews the most important concepts about agglutinative languages, natural language processing, automatic speech recognition, performance metrics and speech recognition models.

2.1 Agglutinative languages (definition and necessity of capturing long term dependencies)

Agglutinative languages are languages in which morphological relations among words are presented by adding suffixes to the root word. In the result, one word can have several morphological morphemes, which give different grammatical meanings. But during adding suffixes the meaning of root word does not change. Next languages are examples of agglutinative and agglutinative like languages: languages from Turkic family [41-45], Finnish, German, Korean languages.

In the range of this thesis Kazakh language will be studied as an example of Agglutinative languages. Agglutinative property of Kazakh language is noticeable in nouns and verbs [43, p. 108]. Grammatical affixes in Kazakh language are added to the root of a word in order to present tense, case and addressee. For example, the noun “адам” can be used in different case and counting forms by the adding various differences of affixes. Different morphological examples of noun “адам” in Kazakh language are given in Table 2. Morphological examples of the verb “оқу” in Kazakh language are given in Table 3.

Table 2 – Examples of different morphological form of the word “Адам”

Word in Kazakh language	English meaning
Адам	Human
Адам-ның	Of human
Адам-ға	To human
Адам-ды	The human
Адам-да	Human has
Адам-нан	From human
Адам-мен	With human
Адам-дар	People
Адам-дар-дың	Of people
Адам-дар-ға	To people
Адам-дар-ды	The people
Адам-дар-да	People has
Адам-дар-дан	From people
Адам-дар-мен	With people
Адам-гер-ші-лік	Morality
Адам-сыз	Without human

Table 3 – Examples of different morphological form of the word “Оқу”

Word in Kazakh language	English meaning
Мен оқи-мын	I study
Сен оқи-сың	You study
Сіз оқи-сыз	You study
Ол оқи-ды	He/she studies
Біз оқи-мыз	We study
Сендер оқи-сың-дар	You study
Сіздер оқи-сыз-дар	You study
Мен оқы-ған-да	When I study

2.2 NLP

Natural Language Processing (NLP) - is the sector of Artificial Intelligence (AI) which studies, develops and applies approaches and models for human-computer interaction using natural human speech. NLP includes wide range of tasks starting from tokenization, recognition and generation. Tokenization includes splitting, syntactic and semantic analyses of a text [50, 51]. Recognition contains tasks of determining some types of entries into a text, answers to questions and text classification [52-54]. Text generation is the process of generating sequence of symbols and words for the tasks of machine translation, dialogue systems and etc. [55-58]. It is very important to know relations of words to some classes in the tasks of recognition and generation. One of the basic techniques in this area is word embedding.

2.2.1 Word Embedding

Featurized representation of word is the probability of the words' relations to different features and these features stored in vector space. Features are learned from contextual information and extracted during the training process by calculating relations among words in sentences and phrases. These representations could be used in sequence generation processes like NLP and speech recognition processes to predict the next element of a sentence.

For example, there is a sentence in a dataset:

I like the process of picking apples.

If the trained model have to predict the next word in new sentence which is not from a known dataset:

I like the process of picking _____?

What word it will choose from the given example table? Most likely, that it will choose the word “cherry”. Because its hot representation is very similar to the hot representation of “apple”. But if your model would not know anything about the word “cherry” it could not predict it correctly. That is why is it very important for your model to know more and more words. Example of featurized representations of words is given in Table 4.

Table 4 – Example of words' featurized representation vector

Words Features	Ер (man)	Әйел (woman)	Әке (father)	Ана (mother)	Алма (apple)	Шие (cherry)
Gender	-1	1	-0.97	0.96	0.00	0.02
Parenthood	-0.25	0.32	-0.99	0.99	-0.03	0.04
Food	0.00	0.00	0.02	0.03	0.92	0.93
Age	0.43	0.38	0.72	0.78	0.04	-0.08
Size	0.05	0.04	0.08	0.09	0.25	0.12
Pet	0.07	0.08	0.01	-0.02	0.00	-0.03
Fruit	0.00	0.00	0.03	-0.01	0.98	0.94

2.2.2 Automatic Speech Recognition

Speech recognition is the complex task which includes the tasks from speech processing area and Natural Language Processing, and its purpose is the transferring human speech into text format. It is based on algorithms, models and approaches which process acoustic speech information and build models of languages to correctly interpret words and phrases in a human speech [49, p. 5].

If divide ASR tasks to exact stages, automatic speech recognition task systems have three main tasks (Figure 3): feature extraction, training and recognition. At the first stage a feature vector is obtained from original speech signal - compressed presentation of speech signals, which contains only the information necessary to recognize.

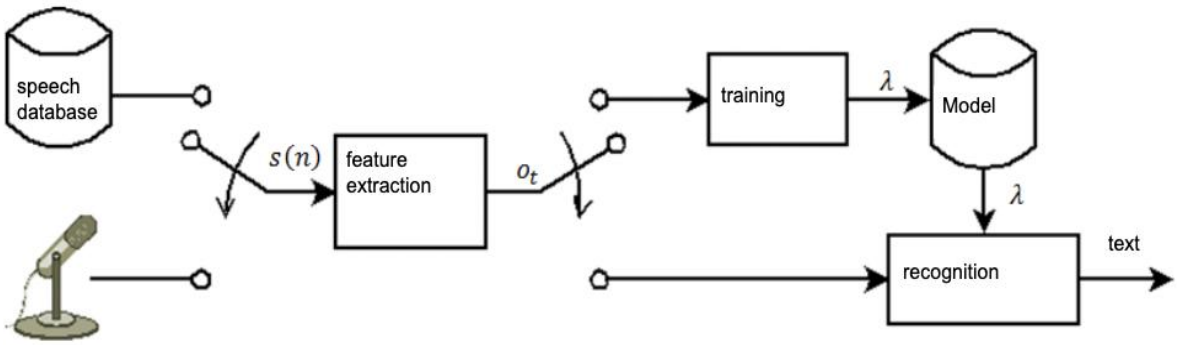

Figure 3 – General scheme for ASR

Human speech in computer systems is presented as audio recordings. Audio recording is a long list of numbers measuring the little changes in air pressure detected by the microphone. The same way our ears perceive the sound and our brain can extract necessary information. In order to make possible to computer to calculate the probability of letter or word correspondence to sound piece it is necessary to form amplitudes of audio record. It is the complicated process where speech signals presented in the form of amplitudes. There are used two types of methods [60, c. 116], first type works with frequencies (Mel-cepstral coefficients, coefficients of linear prediction) and second type is time domain (for example, short-term energy value). But the problem of feature representation of not totally solved problem, that is

why further processing of input speech signals to features is one of the main interests of researches [61-63], because the main purpose is to make neural networks able to extract features from raw input [64-66] and map features to appropriate output of sequences in order to totally exclude human supervision during ASR training process.

Sequence of feature vectors with length equal to T is called acoustic or observed sequence :

$$O = (o_1, o_2, o_3, \dots o_T) \quad (1)$$

Using these features human sends words' sequence, which is equal to :

$$W = (w_1, w_2, w_3, \dots w_T) \quad (2)$$

The main task of speech recognition is to find the sequence of words W, which corresponds to acoustic sequence X [67, 68]. A model Ω will be built in order to solve this task. This model Ω should be able to generate all possible sequences of O, for all word sequences $W \in \bar{W}$. Let function $f(W, \Omega)$ to return all possible O only for given W. Then the recognition is the task of finding the word sequence W, which can generate the closest acoustic sequence according to the model Ω (3):

$$W^* = ArgMin_{W \in \bar{W}} d(f(W, \Omega), O) \quad (3)$$

where $d(O', O)$ - is the distance between O' and O. It means that it is necessary to check all sequences of words W.

According to , the main task of an ASR is to map sequence of speech features to a sequence of notations, like characters or words. If we denote the sequence of speech features as (4):

$$X = \{x_t \in R^D | t = 1, \dots T\} \quad (4)$$

where T is the length of sequence of speech features and x_t is a vector of speech features at the frame of time t, it has dimension D. Here sequence of words can be given as (5):

$$W = \{w_n \in \mathcal{V} | n = 1, \dots N\} \quad (5)$$

where $w_n$is a word in the vocabulary $\mathcal{V}$ which is located at n-th position. Mathematical presentation of ASR can be given as follows based on the theory of Bayes decision (6):

$$\bar{W} = \arg\max p(W|X) \quad (6)$$

where W belongs to all possible sequence of words, $\bar{W}$ is the most probable word.

During training of ASR, after finding the most probable word sequence to the audio from test set performance of ASR is tested by comparing recognized text with original text. There are several main characteristics in speech recognition, which are used to assess the ASR model performance. They are accuracy, WER, CER, loss function value and training time.

ASR systems can be used in wide range of applications, including voice-controlled devices (smart homes, collaborative robots, artificial reality devices), smart voice-assistants, laboratory assistants with the support of interaction via voice, answering machines, dictation systems, transcribing of audio and video records, ASR systems for impaired persons and etc. There are descriptions of some applications of ASR systems:

1. Voice assistants like Siri from Apple, Alexa from Amazon, Google Assistant, Cortana from Microsoft use automatic speech recognition and speech synthesis. These systems allow user to interact with devices using voice-commands.
2. ASR systems also helps get text transcriptions of audio and video files, which is very helpful in big data processing, transcriptions of lecture, medical records.
3. Automatic answering machines are used to automatically recognize and process voice requests from clients.
4. Technologies for impaired persons and persons with hearing loss can be used to make them easy to understand speech and interact with devices and computer applications.
5. Auto translators can help automatically translate a speech from one language to other language without human to interpret.

2.3 Performance metrics

A summary of the results of predictions in the tasks of classification are stored in an error matrix, so called confusion matrix, for being used for statistical calculation. Correct and incorrect results of different observations are classified as TP (True Positive), FN (False Negative), TN (True Negative), and FP (False Positive). A TP is when the model predicts positive and the sample is actually positive. A FP is when the model predicts negative and the appropriate sample is positive. A TN is when the model predicts positive and the given sample is negative. A FN is when the model predicts negative and sample is also negative.

Example of confusion matrix is given in Figure 4.

		Predicted	
		NO	YES
Actual	NO	TN	FP
	YES	FN	TP

Figure 4 – Example of Confusion Matrix

Further will be given definition and mathematical formulations for performance metrics, which are widely used and specific to the research topic, like accuracy, precision, recall, word error rate, character error rate, sentence error rate and training time.

2.3.1 Accuracy

By general definition, accuracy is the sum of correct predictions over the total number of examples (7):

$$Accuracy = \frac{TruePositive+TrueNegative}{Total\ Examples} \quad (7)$$

Accuracy calculation for speech recognition usually includes the comparison of recognized text using ASR with original text [49, p. 10]. Accuracy is determined based on the correctly recognized words or phonemes. A common method for calculating accuracy includes several steps:

1. Storing original text. Use dataset of audio files with correct transcription texts.
2. Generation of audio file transcripts, using ASR system. Align original transcript with recognized transcript to determine matching words and phonemes. Usually, this process is performed by dynamic temporal alignment or alignment of sequences.
3. Calculation of accuracy. Compare aligned original text with recognized text to calculate accuracy metric. The most common metric is Word Error Rate - the metric which determines the percentage of not correct or replaced words in recognized words in comparison with original text. There are other metric types, which could be calculated depending on the type of recognition rate analysis.

2.3.2 Precision and Recall

Precision (8) and Recall (9) are also calculated on the basis of data taken from the confusion matrix [71, p. 1-16]:

$$Precision = \frac{TruePositive}{TruePositive+FalsePositive} \quad (8)$$

$$Recall = \frac{TruePositive}{TruePositive+FalseNegative} \quad (9)$$

Precision is therefore a ratio that shows how much of the positive predictions are correct, while Recall is a ratio that shows how many of the positives are correctly determined by the model.

2.3.3 Word Error Rate (WER)

WER (10) is the ratio of errors in a recognized text to the total text in the initial utterance. It is assessed in percentages and it is one of the key metrics of model performance for ASRs.

$$WER = \frac{\mathrm{S+I+D}}{WN} \quad (10)$$

where S is substituted misspellings, I is insertions of words absent in the initial transcript, D is deleted (missed) words, WN is the number of words in the initial text. In alternative, WER can be calculated using accuracy (11):

$$WER = 100 - \mathrm{Accuracy} \quad (11)$$

2.3.4 Character Error Rate (CER)

CER is the ratio (12) of errors in a recognized text to the total number of characters in the initial utterance. It is calculated based on the Levenshtein distance concept, measured in percentage. It is one of the key metrics of model performance for ASR:

$$WER = \frac{\mathrm{S+I+D}}{CN} \quad (12)$$

where S is substituted character misspellings, I is insertions of characters absent in the initial transcript, D is deleted (missed) characters, CN is the number of characters in the original text.

2.3.5 Entropy and perplexity

Entropy measures information amount for a random variable. By definition: entropy, so called self-information is the average uncertainty value (13) of a random variable [49, p. 32]:

$$H(p) = H(x) = -\sum_{x \in X} p(x) log_2 p(x) \quad (13)$$

here $p(x)$ is the probability mass function of some random variable X, in terms of language modelling over the alphabet. It means entropy can be used to evaluate the

performance of a language model as good as perplexity. Perplexity is calculated on the basis of cross entropy in order to perform calculation results in a non-logarithmic way [49, p. 35]. The less the value of perplexity, the better the performance of a language model. The relationship between perplexity and entropy is shown here (14):

$$Entropy = log_2 Perp \tag{14}$$

There are different types of entropies, like joint entropy, conditional entropy, Kullback-Leibler divergence, and maximum and minimum entropy that can be applied to different tasks. For example, authors of [74, p. 242] give the following way of entropy calculation: if given corpus to test is $s = \{d_1, \dots, d_L\}$ and language model denoted ad $p(s)$, the mathematical formulation of them will be as follows (15):

$$Perp = \sqrt[|s|]{\frac{1}{p(s)}} = \sqrt[|s|]{\prod_{i=1}^{L} \frac{1}{p(w_1 \dots w_{|d_i|}, d_i)}} = \sqrt[|s|]{\prod_{i=1}^{L} \prod_{l=1}^{|d_i|} \frac{1}{p(w_l, d_i | w_i \dots w_{l-1})}} \tag{15}$$

where d is the length of the test document, $|s| = \sum_{i=1}^{L} |d_i|$ denotes the test corpus length, w_i means the word in a sequence into the test document. Anyway, the main purpose of language modeling is to predict the word sequences which are more natural.

2.3.6 Training time

Training time is the time spent for full training of a model. In general training time can vary from several hours, to weeks or even months, depending on several factors. Some of the factors are as follows:

1. Size and complexity of the system.

Large and complex neural networks with big size of layers and many parameters need more time to train, in comparison to simple networks. Training of deep CNNs, like ResNet or Inception, can require days or weeks on the most powerful Graphical Processing Units (GPU) or specialized hardware systems, like Tensor Processing Units (TPU).

2. Size and quality of the data to be trained.

Bigger size of data needs more time to train. If you have a dataset which needs preprocessing and augmentation, it can require long time to process and increase the time for training.

3. Computing resources.

Use of the most powerful resources, such as GPUs or specialized hardware platforms, can significantly increase the training time. Parallelization of computing in different units also can shorten the time of training.

4. Choice of optimization algorithms.

Many optimization algorithms do exist, like stochastic gradient descent (SGD), Adam, RMSProp and others. Choice of appropriate algorithms and their correct tuning can have impact on training time.

5. Purpose of training.

Training time also can depend on the exact task which is to solve. Some tasks like image classification, usually require less time in comparison with the tasks of Natural Language Processing (NLP) or text generation.

2.4 Speech recognition models

2.4.1 Mathematical model

The process of automatic translating a speech to text can be represented as a search of the most probable word (16) sequences according to two estimates: acoustic and linguistic:

$$W = ArgMax_w P(X|A) = ArgMax_w P(A|W) \cdot P(W) \qquad (16)$$

where $P(A|W)$ is the probability of hypothesis occurrence by acoustic model, $P(W)$ is the probability of hypothesis W by language model.

2.4.2 Acoustic models

For acoustic modeling of speech Hidden Markov Models (HMMs), are usually used. Here each allophone (speech sound) is represented by one continuous HMM of the first order. The phoneme model most often has three states: the first describes the beginning of the phoneme, the second represents the central part, and the third the ending. The HMM of a word is obtained by connecting phoneme models from the corresponding phonemic alphabet into a chain. In a similar way, word models are connected to each other, forming phrase patterns. HMM states are described by means of mixtures

Gaussian probability density distributions (Gaussian mixture model - GMM), which provide a fairly complete coverage of possible variants of pronunciation of phonemes, taking into account phonetic contexts and speaker differences. The goal of training acoustic models based on the HMM is to determine, from the training sequence of observations, such model parameters with which the probability of this sequence would be maximum. Context-independent phonemes or context-dependent phonemic implementations can be used as acoustic units in speech recognition systems. The advantage of using context-dependent units is their ability to model the effects of coarticulation between adjacent sounds. Therefore, in modern speech recognition systems, context-independent models (monophones), which correspond to phonological units of a phonemic set, are often replaced by context-dependent models (triphones). HMM is the most widely used method for modeling acoustic units, but HMM is not without drawbacks. In particular, they have weak discriminative abilities, that is, the ability to separate classes of words.

The most common language models are statistical models based on n-grams of words, which estimate the probability of the occurrence of a sequence of words in some text. n-grams are a sequences of n elements (for example, words), and the n-gram language model is used to predict an element in a sequence containing n-1 predecessors . The disadvantage of n-gram models is that they predict a word based on a pre-existing context of a certain length. Usually a context of three words

(trigrams) is taken, less often - of four or five words. The use of a longer context is problematic, since, firstly, it requires a very large amount of training data, and secondly, it significantly increases the size of the language model and, as a result, the speed of speech recognition falls down.

Use of artificial neural networks in speech recognition systems allowed to increase recognition accuracy in comparison with basic models (HMM for acoustic modelling, n-grams for language modelling). Basic types of neural networks used in speech recognition are given in Figure 5.

Neural networks can be used for both acoustic and language modeling, improving recognition accuracy. NNs can be divided into feed-forward networks and backward networks. There exist different varieties of NNs, among which the main types can be distinguished: perceptrons, autoencoders, convolutional neural network (CNN), NN with time delays (time delay neural network; TDNN), deep belief networks (deep belief networks; DBN), NN with long short-term memory (Long Short-Term Memory; LSTM), Bi-LSTM(Bidirectional LSTM).

In some studies , it was shown that the use of NNs together with HMMs allows to improve the accuracy of speech recognition, while HMMs provide the ability to model long-term dependencies, and NNs provide the possibility of discriminant learning [7].

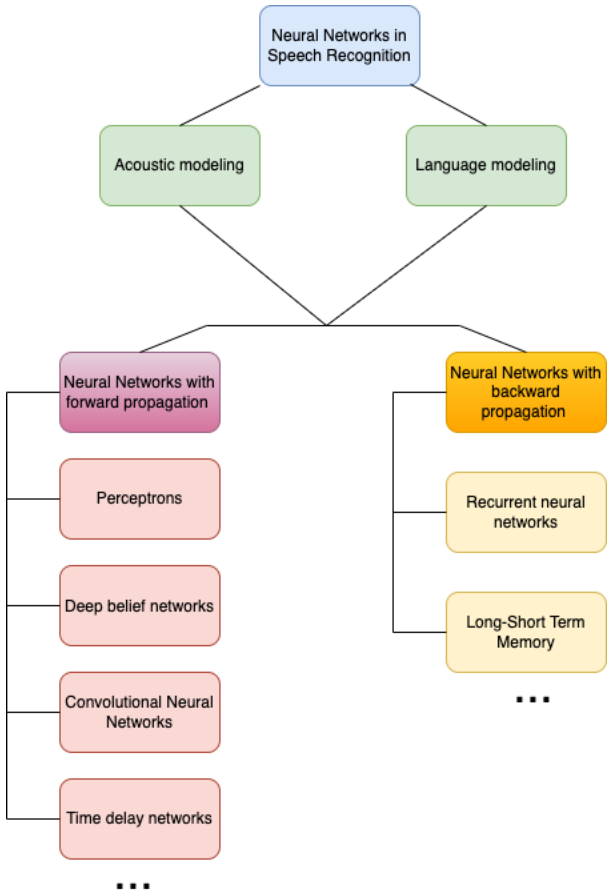

Figure 5 – Classifications of NNs in speech recognition

Acoustic models are usually built on the basis of deep neural networks (DNNs), which are forward-propagation ANNs containing more than one hidden layer between the input and output layers. There are exist many methods for combining neural networks and HMM. Two main methods among existing methods are: 1) building hybrid HMM/DNN models; 2) construction of tandem models. In hybrid systems, neural networks are used to obtain HMM posterior probabilities.

In the tandem method, the output of the neural network is used as an additional feature stream for HMM training.

To increase the accuracy of recognition, the bottleneck method was also used. In a bottleneck neural network, the middle layer has fewer elements. The input data for the neural network are features such as mel-frequency cepstral coefficients (MFCC) or perceptual linear prediction (PLP) coefficients. After training, the layers behind the bottleneck layer are removed. The output of neurons in the bottleneck layer serves as acoustic features for standard speech recognition systems using HMM.

Studies on combining ANN and HMM for acoustic modeling were started in the late 1980s . However, such studies were not popular at that time, due to the fact that NN is a resource-intensive task and requires high-performance computers. In recent years, due to the increase in the computing power of computers, the use of ANNs in speech recognition systems, including for acoustic modeling, is becoming increasingly popular. The development of a parallel computing platform using the NVidia CUDA graphics processor made it possible to significantly reduce the training time of deep ANNs on large data volumes, which contributed to an even greater spread of neural network models in speech recognition systems .

Last studies in the area of acoustic dedicated to the ways of excluding processing raw features by HMM. For example, in , the possibility of obtaining features directly from a neural network without converting output probabilities to features suitable for HMM was studied. Experiments were carried out using a five-layer bottleneck perceptron in the middle layer. After training the network, the output from the bottleneck layer was used as features for the speech recognition system. At the same time, an increase in recognition accuracy was obtained when these features were used instead of probabilistic features; in addition, the size of the model was reduced, since only part of the neural network was used.

Parameters are one of important factors with neural networks. The paper describes a study of which parameters of neural networks are most important for the operation of a speech recognition system. It has been shown that with the increase in the size and depth of the model, the efficiency grows only up to certain limits. In addition, a comparison was made of standard deep neural networks, convolutional neural networks, and deep locally unified neural networks(DLUNNs), which showed that DLUNNs can significantly improve recognition accuracy.

In , neural network for acoustic models were trained using Kaldi and Python deep learning toolkit (PDNN) software. Acoustic models were trained as follows: first, acoustic models using Gaussian Mixture Models(GMMs) were created using Kaldi, then a deep neural network was trained using PDNN, and finally the trained

neural network models were loaded into Kaldi for speech recognition. The article describes four implementations: hybrid model; a tandem model using features obtained from the bottleneck layer; joint use of previously mentioned two methods a hybrid model based on a convolutional neural network.

A convolutional neural network consists of one or more pairs of convolutional and pooling layers. The architecture of a convolutional neural network is shown in Picture 2.4. In a convolutional neural network, the activation signal of each neuron is calculated by multiplying a small part of the input data(several vectors of features from Figure 6) by the weight matrix W. Then the weight matrix is shifted for the next part of the input data, thus the weight matrix is shifted over the entire input feature space. At the output of the layer, a feature map is formed. The pooling layer performs downsizing of the input feature map by selecting the maximum element. The merging layer allows you to reduce the influence of speaker variance on the model parameters. A convolutional neural network for acoustic modeling was used in , where neural network adaptation to the context of convolutional neural networks was studied, which made it possible to reduce the relative recognition error by 6%.

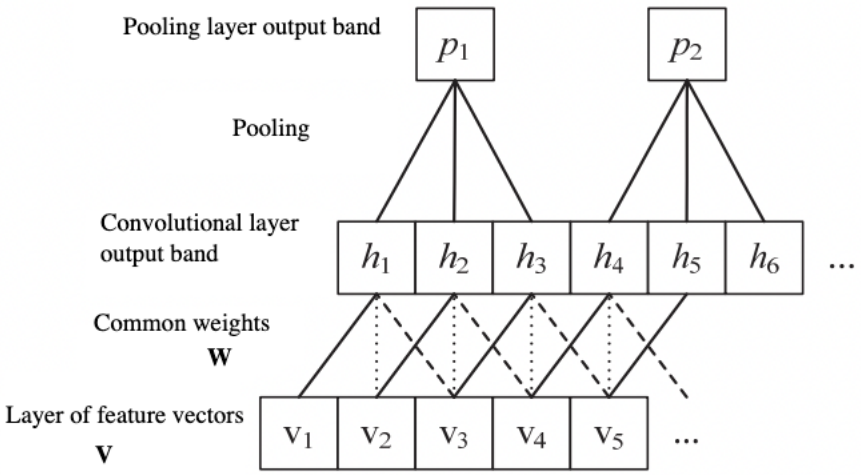

Figure 6 – CNN architecture

Neural networks with time delays are also used for acoustic modeling. They are a multilayer neural network of direct propagation, the nodes of which are modified by introducing time delays . An example of a node with N delays is shown in Figure 7. In the figure, $U_1 \dots U_j$are the inputs of nodes; each of the J inputs is multiplied by the corresponding factor of weighting w; $D_1 - D_j$are time delays and F is an activation function . Here short-term memory is built into the artificial neural network. The introduction of a time delay makes it possible to make the ANN invariant to time shifts. In , the use of neural network with time delays made it possible to obtain a relative decrease in the error rate for word recognition by 2.6%.

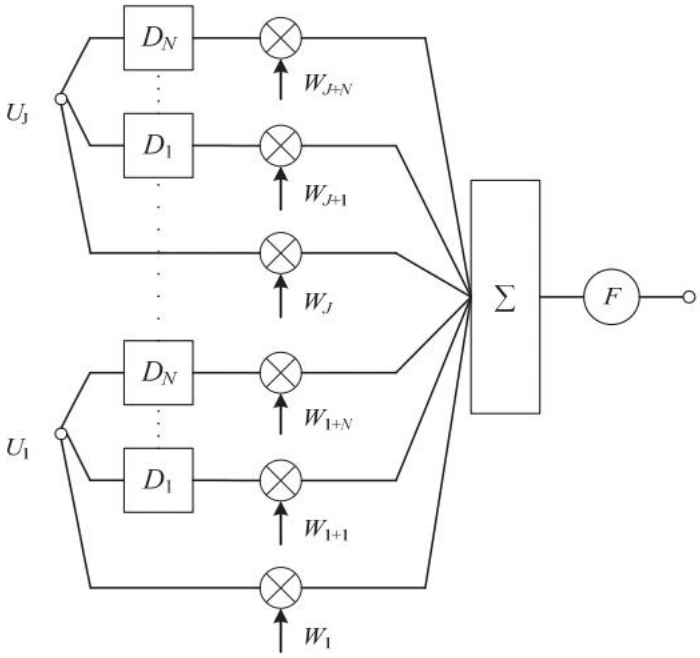

Figure 7 – Example of neural network with time delays

Another type of neural networks are recurrent neural networks. The presence of feedback provides the neural network with memory, which makes it possible to simulate dynamic processes. One of the types of RNNs used for acoustic modeling is the LSTM network containing special elements called memory blocks. Memory blocks contain cells that store the temporary state of the network, as well as multiplicative elements called gates (gates), which control the flow of information. Each block of memory contains an input gate, an output gate, and a forget gate. An example of an LSTM network memory block is shown in Figure 8. In the figure, x_t is the input vector at time t, h_t is the output vector. An LSTM network cell can be considered as a complex network element capable of storing information for a long time. Gates determine when the input is relevant and needs to be remembered, when the information should continue to be remembered or forgotten, and when the information should be output. It was shown in [91, p. 631-634] that the use of LSTM in a hybrid RNN/HMM model makes it possible to reduce the word recognition error compared to the use of DNNs.

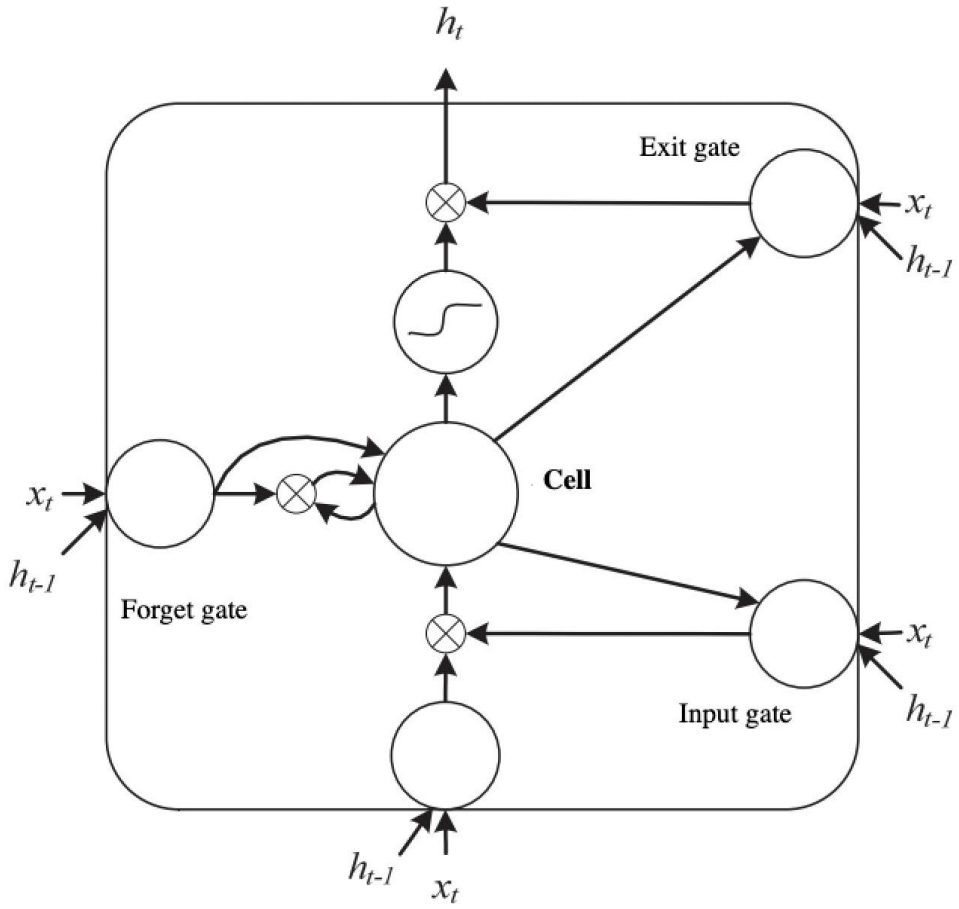

Figure 8 – Example of LSTM unit

Recently, end-to-end speech-to-text systems using only neural networks are popular. They do not train HMM models. End-to-end systems consist of two submodules: an encoder and a decoder. The encoder reads the input signal, calculates the features of the signal and converts it into an intermediate parametric representation. The decoder converts the parametric representation of the signal into a sequence of symbols. In , an end-to-end system was built on the basis of a convolutional neural network and the neural network temporal classification method (CTC) . The developed approach was tested for the problem of phoneme recognition, and the phoneme recognition error was 18.2%. The network with LSTM units was used to build the end-to-end system described in . Without the use of linguistic information, the word recognition error was 27.3%, the use of a dictionary made it possible to reduce the error to 21.9%, with the trigram model of the language, the word recognition error decreased to 8.2%.

Despite of breakthrough achievements in E2E STT systems, some of actual ASR systems use architectures similar to old constructions and consequently use methods derived from them: discriminative training methods, HMM, GMM and HMM/DNN. This approach needs a hand-made phoneme dictionary built on developers' suggestions which is then used to build context dependent phonetic model. During the building of phonetic model text tokenized not into words, text splitted to phoneme groups . According to this, given detailed descriptions of these

methods concurrently with modern STT architectures and approaches are given below.

2.4.2.1 HMM

Hidden Markov Models are static models, which are used for modelling sequential data. They are widely used in the tasks where data has temporal or sequential structure, like speech recognition, NLP, bioinformatics and other applications. HMM consists of two main components: hidden sequence of states and observable sequence of characters. Inside of HMM exists the set of states which form the hidden sequence. Each state can generate certain characters, forming observable sequences. Transfer probabilities of between states and probabilities of character generation are determined by model parameters.

The main assumptions of HMM:

1. Markov Assumption: a current state depends only on previous state. It means that current state depends on the state that exactly prior to it and does not depends on other previous states.

2. Assumption about observability: Observed variable (character) depends only on the current state, and does not depends on other states or other characters.

The main tasks related to HMM are probability definition of observed sequence, decoding of state sequences, model training. The main task of probability definition of the observed sequence is the calculation of character sequence under the given model of HMM. The main task of state sequences decoding is the finding of the most probable sequence of hidden states which is appropriate to the given sequence of characters. The main task of model training is the assessment of parameters on the basis of selected data from observed sequences. HMM training is usually performed by using Baum-Welch algorithm which is the variation of expectation-maximization (EM) algorithm. Additionally, forward-backward algorithms and Viterbi algorithm are widely used for decoding.

Application of HMM in speech recognition

HMM is defined as set of triple parameters :

$$\Omega = (A, B, \pi) \tag{3}$$

where A is the matrix of transition probabilities, B is the probability matrix of output observations, and π is the probability vector of initial states. Matrix A consists of elements a_{ij}, where these elements are probabilities of transitions from state i to state j. Matrix B contains elements $b_i(o_k)$, where $b_i(o_l)$ is the observation probability of feature vectors in the state i. Finally, π consists of components, called π_i - probabilities of being in the state i at the beginning time.

Statistical models of phonemes, words and whole phrases are created with the help of HMM. The choice of a specific language object depends on the tasks to be solved by the speech recognition system being developed. Usually, it is possible to distinguish the following approaches to the construction of the HMM (they can be both mutually exclusive and complementary) [97, p. 275]:

1. HMM is used to model phonemes - sound letters of the language, which can be combined into words.

2. Phonemes are modeled using three states - initial, middle and final (Figure 9). This is due to the fact that speech trajectory cannot change its characteristics instantly, and while moving from a phoneme to another phoneme it goes “through” intermediate states.

3. Phonemes can sound in different ways depending on other sounds around. This process is called coarticulation Two types of phoneme models exist depending on whether this phenomenon will be ignored or not: monophones and triphones.

4. A separate HMM is composed for each word from the dictionary and the most appropriate is chosen during recognition. This approach is very good for recognition of distinct words.

5. One HMM can be composed by joining HMMs for through internal states(for example, silence), according to the language grammar. It is necessary for the recognition of continuous speech.

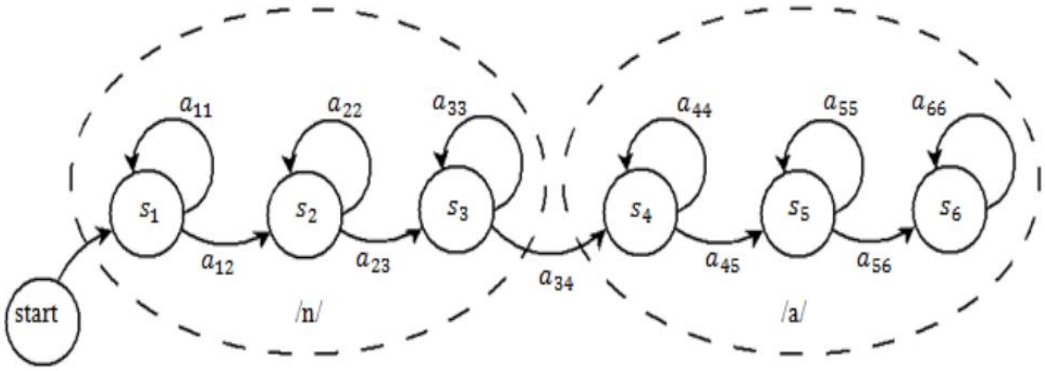

Figure 9 – HMM fragment for "n” and “a” phonemes which include tree states: initial, middle and final

Use of HMM for the recognition of isolated words is based on the calculation of probabilities with forward propagation, which is defined as the probability of observed sequence $O = (o_1, o_2, o_3, \dots o_T)$, which is in the state j at the time moment t for the model $\Omega = (\mathrm{A}, \mathrm{B}, \pi)$ (18):

$$\alpha_1(j) = \pi_j \cdot b_j(o_1)$$
$$\alpha_t(j) = \left[\sum_{i=1}^{N_s} \alpha_{t-1}(i) \cdot a_{ij}\right] \cdot b_j(o_t) \quad (18)$$

Calculation of $\alpha_t(j)$ goes recursively. Achieving the end of observed sequence, exactly to $t = T$, it is necessary to sum up $\alpha_T(j)$ for all states, after taking the probability of observing sequence $O = (o_1, o_2, o_3, \dots o_T)$ for the given HMM Ω (19):

$$P(O|\Omega) = \sum_{j=1}^{N_s} \alpha_T(j) \quad (19)$$

This probability can be used to recognize isolated words: each word is modeled by HMM Ω_k. At recognition it is important to choose the HMM which available to generate observed O with the highest probability (20):

$$w^* = ArgMax\ P(O|\Omega) \quad (20)$$

The method described for calculating $P(O|\Omega)$ is called the algorithm of forward pass and it is also the base for reassessment procedures of HMM (Baum-Velshch algorithm). Another famous algorithm is the algorithm of Viterbi, which is used to find the optimal sequence of HMM states $Q = (q_1, q_2, q_3, \dots q_T)$, which corresponds to the given sequence of observations. These algorithms also works recursively, but instead of increasing sum at each step, movement goes by maximum.

2.4.2.2 Gaussian Mixture Model – GMM

There are two main approaches to determining the probabilities of observations: mixtures of Gaussian probability density distributions (Gaussian Mixture Model; GMM) and artificial neural networks.

Until about 2010, the Gaussian mixture model was used in practice to specify the distribution of the observed signal depending on the phoneme. To do this, the audio signal is divided into small sections (10-50 ms), to apply traditional signal processing in the frequency domain, a fast Fourier transform is performed for each section of the signal. Further, the logarithm of the resulting spectrum was used in connection with the well-known logarithmic perception of the sound scale by the human ear. Finally, using the discrete cosine transform of the logarithm of the spectrum, practically independent features were obtained – cepstral coefficients, the distribution of which was written as a mixture of Gaussian random vectors with diagonal covariance matrices.

In a system using mixtures of Gaussian distributions probability densities, the probabilities of observations are defined as (21):

$$P_e(x_t|q_t = i) = \sum_{j=1}^{J} c_{ij} N(x_t, \mu_{ij}, \Sigma_{ij}) \quad (21)$$

where J is the number of components in mixture, c_{ij} is the weight of Gaussian distribution $N(x_t, \mu_{ij}, \Sigma_{ij})$, μ_{ij} and Σ_{ij} are elements of mathematical expectations vector and covariance matrix.

2.4.3 Hidden Markov Models and Artificial Neural Networks - HMM/ANN

In the hybrid HMM/ANN model, the probabilities of observations are calculated using a neural network. The neural network calculates the probabilities depending on the class $P(x_t|q_t = i)$. It means, we can calculate the probabilities of observations using Bayes' theorem (22):

$$P_e(x_t|q_t = i) = \frac{P(x_t|q_t=i)}{P(x_t)} = \frac{P(q_t=i|x_t)}{P(q_t=i)} \quad (22)$$

Various neural network architectures, like Multilayer Perceptron, Recurrent Neural Networks, Long-Short Term Memory, Gated Recurrent Units and Convolutional Neural Networks are used to build hybrid models [101, с. 81]. The architecture of the hybrid model is shown in Figure 10.

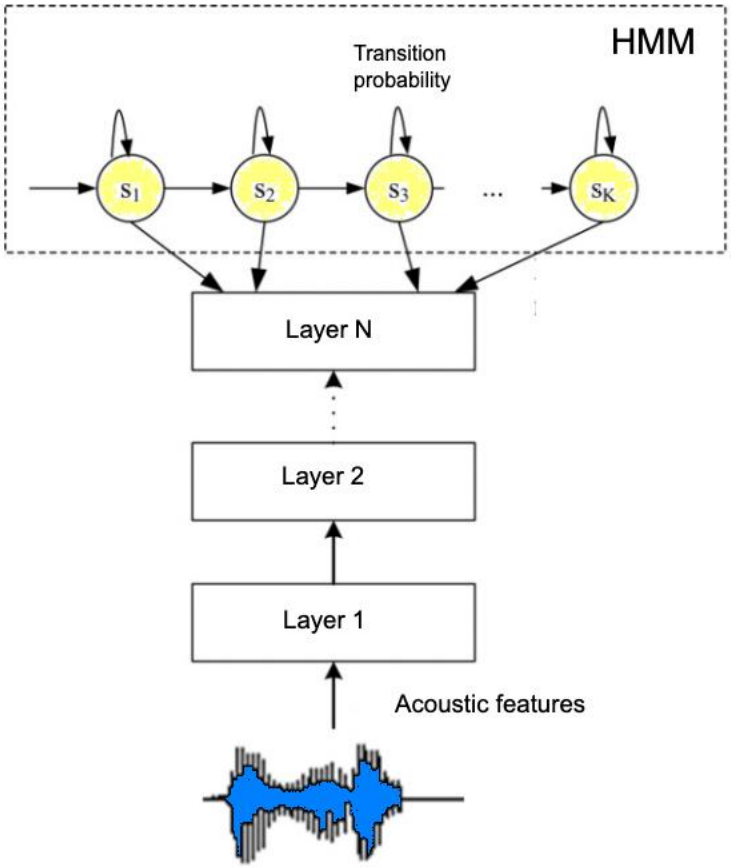

Figure 10 – Hybrid HMM/DNN model architecture

2.4.4 End-to-end models

It has been shown in many works that the use of neural networks at each step of the scenario of a standard speech recognition system improves the quality of its work. So, for example, in language models were trained using RNN, in a dictionary was obtained using LSTM networks, in deep neural networks showed good results for building acoustic models, in a method was presented feature extraction using limited Boltzmann machines . Consequently, the idea arose to use artificial neural networks at all stages of speech recognition.

In the case of speech recognition, the integral approach tries to calculate $P(W|X)$ "globally". Let the input be a sequence of sound features X, and the corresponding sequence of words is W. Thus, the neural network calculates the probabilities P$P(\cdot|x_1), P(\cdot|x_2), \ldots P(\cdot|x_T)$, where the probability arguments are not

the sequences of words themselves, some of their representations (hereinafter referred to as labels).

Figure 11 [101, c. 83] shows a general diagram of the integrated system. At the moment, there is a huge amount of architectures and neural network types for implementing integral models. Further will be given detailed and short descriptions of working principle of the most popular approaches for speech recognition.

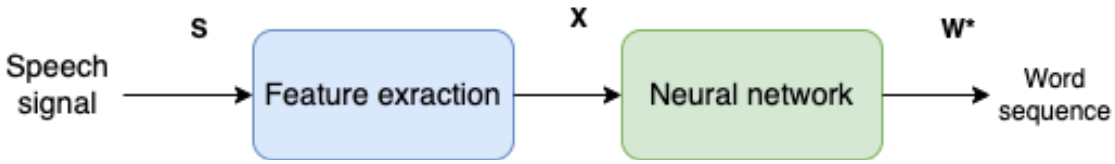

Figure 11 – General scheme for E2E ASR system

2.4.5 Sequence models

Sequence models belong to the class of supervised learning and consist of artificial neurons, which have feedback loops and can be used to solve various problems such as speech recognition, speech synthesis, music generation, different type of classification tasks, machine translation, and video activity recognition. But the limitation of these models is only input or the output can be a sequence. In other words, sequence models can be used to solve any type of supervised learning problem that contains time series in either the input or output layers.

Traditional neural networks assume that all inputs (and outputs) are independent of each other and therefore will not work in sequence prediction because previous inputs are inherently important in predicting the next output. For example, when predicting the next word in a text sequence, we need to know at least several words before the word to be predicted. Traditional neural networks require that the lengths of input and output sequences be constant across all predictions. Sequence model networks can solve this problem directly.

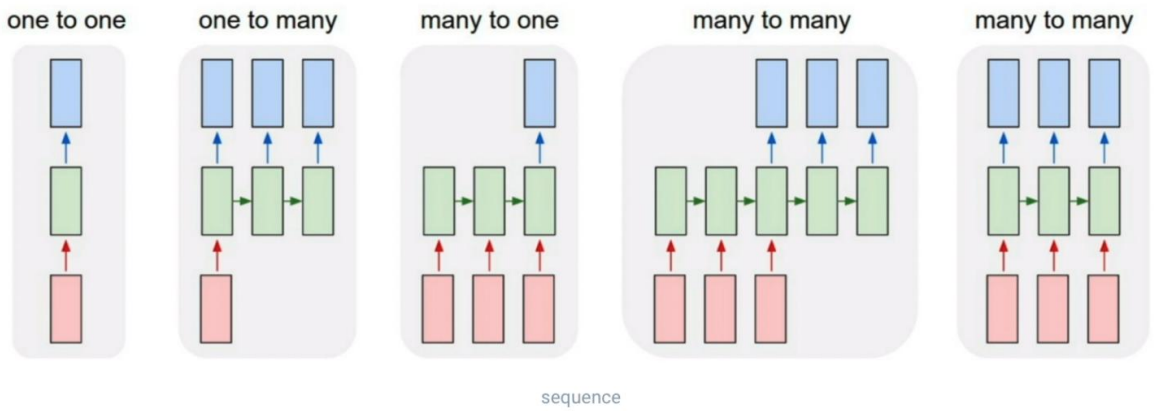

Figure 12 – Types of sequence models

Sequence modeling has many types of networks: including one-to-one, one-to-many, many-to-one, and many-to-many, as shown in Figure 12. When generating

music, the input can be an empty set, and the output can be a song (one-to-many), while in speech recognition, only one word can be obtained from a long set of sound characteristics(many-to-one). Many-to-many architecture where the length of the input and output sequences are different can be implemented using the encoder and decoder approach. Many-to-many models are known as "sequence-to-sequence" models.

The most popular and widely used algorithms for sequential models are: RNN, LSTM, GRU.

2.4.6 RNN

The whole point of processing a connected sequence of data is to be able to take into account the connection of elements in addition to extracting a response for each element. For example, it is possible to represent an image as a set of vectors, each containing the pixels of one column. If we want to teach the network to classify natural language sentences (for example, determine the emotional coloring of a sentence), and feed one word after another to the network, we want the network to "remember" the words already transmitted. If we want the network to translate a sentence from one language to another, then it would also be good to take into account the beginning of the sentence when translating the middle and end.

It is the task of "remembering" the elements of the sequence that have already been looked at and is supposed to be solved using a recurrent network. To do this, in addition to the output vector, the network must also have some vector that describes the current internal state of the network, i.e. it contains memories of all the elements already viewed by the network. More formally, it looks like this.

Consider, we have set of input vectors $(X^{(1)}, X^{(2)}, \dots, X^{(n)}$ and they will sequentially transformed (23):

$$H^{(t)} = \sigma(W^{hx} \cdot X^{(t)} + W^{hh} \cdot H^{(t-1)} + b_n)$$
$$Y^{(t)} = W^{yh} \cdot H^{(t)} + b_y \quad (23)$$

Moreover, in addition to the output $Y^{(t)}$ (which we may not need at each step), we also have a vector $H^{(t)}$ describing the current state. Thus, the network consists of cells of the form as shown in Figure 13.

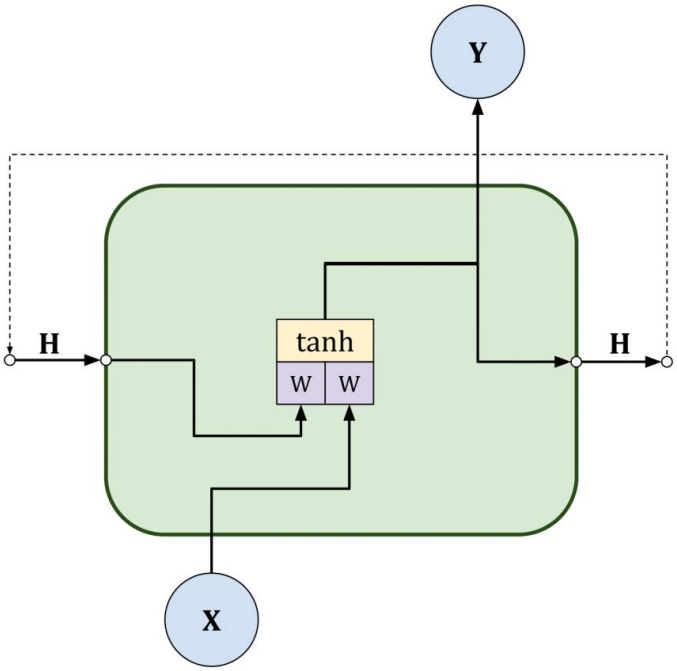

Figure 13 – RNN cell

These cells are assembled into a sequence, passing the internal state from the cell to the one following it in time. Note that the weights in this case for all cells are the same (Figure 14).

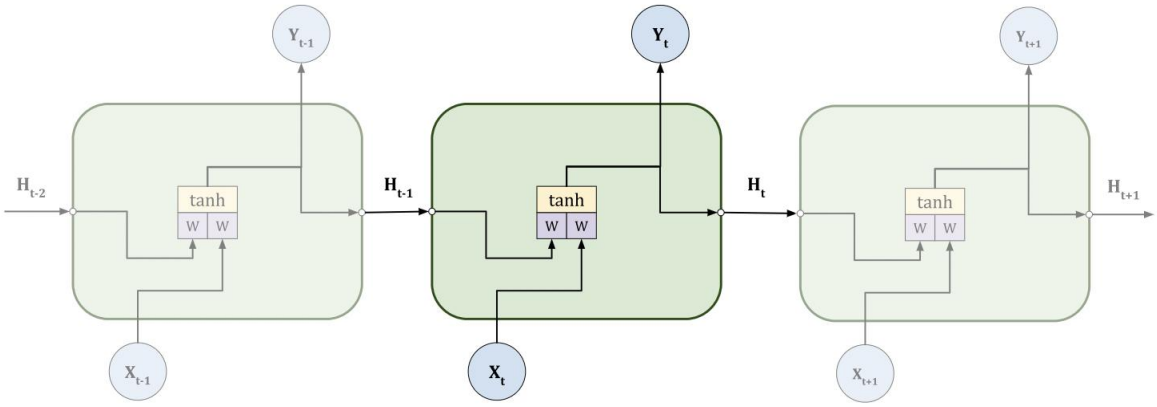

Figure 14 – Sequence of RNN cells

2.4.7 LSTM

The problem with basic RNN cells is that they cannot "keep in memory" very long sequences. This is due to the fact that when we pass gradients through a sufficiently long sequence, we encounter one of two problems: either the gradients decrease so much that errors at the end of the sequence no longer affect its beginning, or the gradients increase, and the process diverges. The same problem exists in conventional networks: by adding layers, appear difficulties with training (hence architectures of ResNet and other approaches). To overcome this problem, it was

proposed to replace conventional RNN cells with a more advanced version: the LSTM. In the basic version of LSTM, was added one more internal state of the cell S_t which helped to extinguish the problem of vanishing or exploding gradients. Also, an input and an output gate were added to the LSTM cell. This approach proposed the solution that input data will have effect on internal state and internal state will affect the output. Its working principle is given in Figure 15.

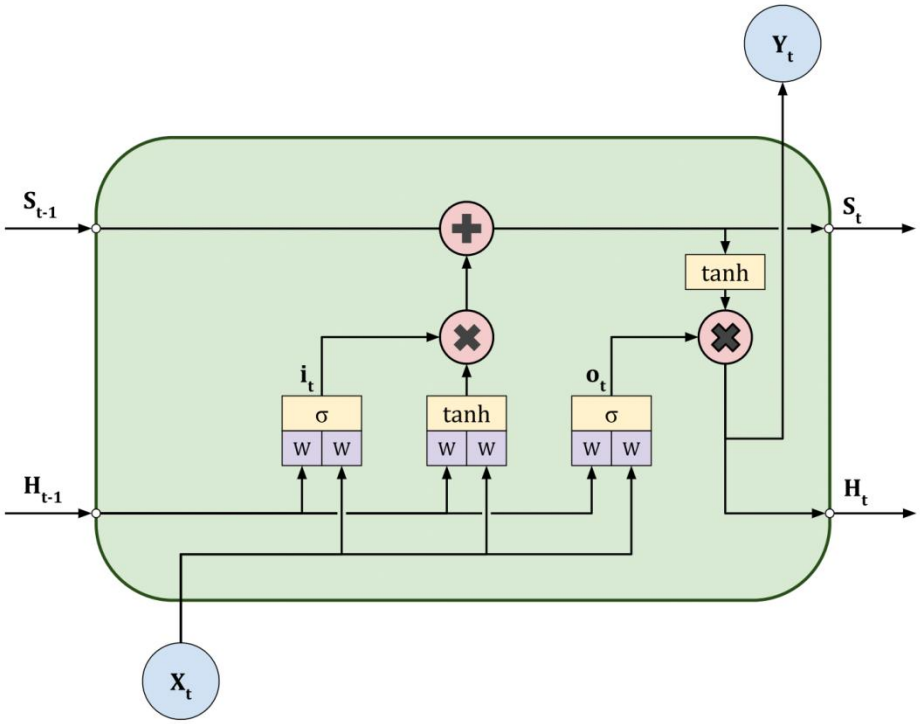

Figure 15 – LSTM cell structure

If we present the working principle of LSTM formally, it looks like as follows (24):

$$\begin{aligned} g^{(t)} &= \phi(W^{gx} \cdot X^{(t)} + W^{gh} \cdot H^{(t-1)} + b_g) \\ i^{(t)} &= \sigma(W^{ix} \cdot X^{(t)} + W^{ih} \cdot H^{(t-1)} + b_i) \\ o^{(t)} &= \sigma(W^{ox} \cdot X^{(t)} + W^{oh} \cdot H^{(t-1)} + b_o) \\ S^{(t)} &= g^{(t)} \odot i^{(t)} + S^{(t-1)} \\ H^{(t)} &= \phi(S^{(t)}) \odot o^{(t)} \end{aligned} \tag{24}$$

Here $i^{(t)}$ and $o^{(t)}$ are the input and output gates, respectively, and $\odot$ denotes the operation of elementwise multiplication. Those Here $i^{(t)}$ and $o^{(t)}$are assumed to be vectors of 1-s and 0-s that allow to pass or do not pass some components of the vector through themselves.

The concept of the gate was so effective that later they were added another one: forget gate (Figure 16). This gate allowed additional zeroing of some components of the internal state $S^{(t-1)}$ before passing them on.

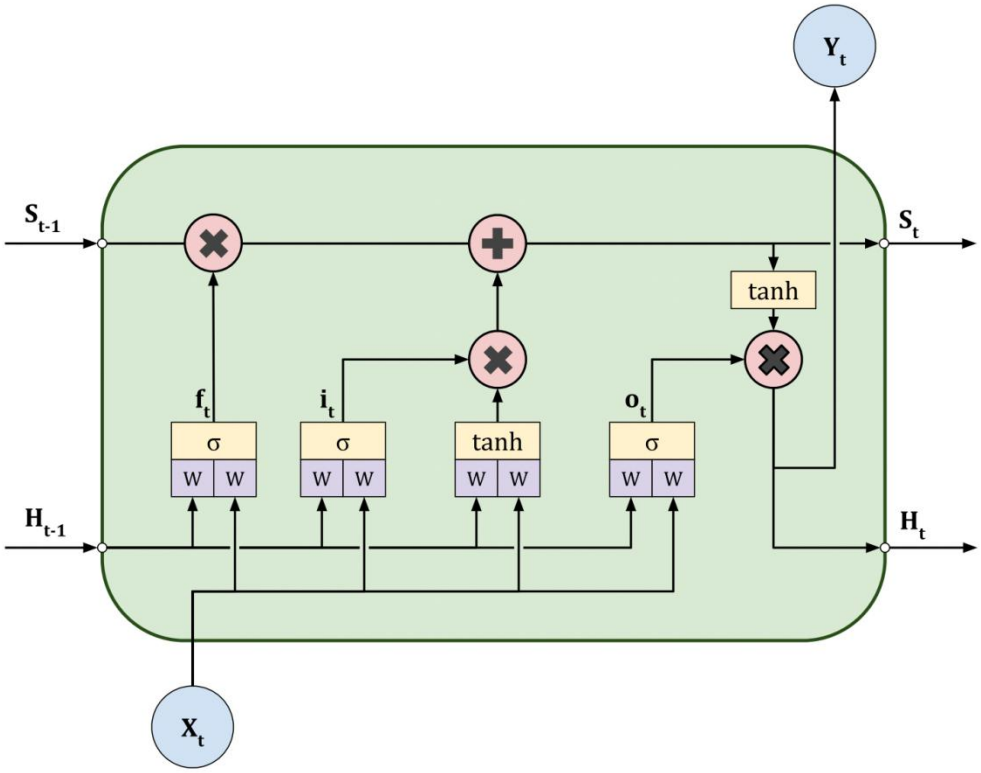

Figure 16 – LSTM with forget gate

Formally it looks this way (25):

$$\begin{aligned} g^{(t)} &= \phi(W^{gx} \cdot X^{(t)} + W^{gh} \cdot H^{(t-1)} + b_g) \\ i^{(t)} &= \sigma(W^{ix} \cdot X^{(t)} + W^{ih} \cdot H^{(t-1)} + b_i) \\ o^{(t)} &= \sigma(W^{ox} \cdot X^{(t)} + W^{oh} \cdot H^{(t-1)} + b_o) \\ f^{(t)} &= \sigma(W^{fx} \cdot X^{(t)} + W^{fh} \cdot H^{(t-1)} + b_f) \\ S^{(t)} &= g^{(t)} \odot i^{(t)} + S^{(t-1)} \odot f^{(t)} \\ H^{(t)} &= \phi(S^{(t)}) \odot o^{(t)} \end{aligned} \tag{25}$$

Here $f^{(t)}$ is a forget gate.

2.4.8 GRU

Gated Recurrent Unit (GRU) is a modified, simplified version of LSTM, in which long-term and short-term memory are combined into a so-called Hidden State. It only has a latent state that can combine both long-term and short-term memory. The GRU was introduced in 2014 to solve a common vanishing gradient problem faced by programmers.

Vanishing gradient problems occur when the gradient tends to decrease after backpropagation over time and ceases to be of benefit in the learning process. Therefore, in registered neural networks, if the first levels gain the least amount of gradient, their learning process stops. Since these layers are not trained, the RNN

does not remember anything from the experience of longer runs of data and runs into short-term memory problems.

GRUs are a variation of the RNN design. They use the gating process to manage and control the flow of information between the cells of a neural network. GRUs can make it easier to catch dependencies without ignoring past information from massive chunks of serial data. GRU does all this using its gates, which help solve vanishing gradient problems often encountered in traditional registered neural networks. These gates help control the information that should be kept or discarded at each step. It is also worth remembering that controlled recurrent blocks use reset and update gates. GRU structure is given in Figure 17.

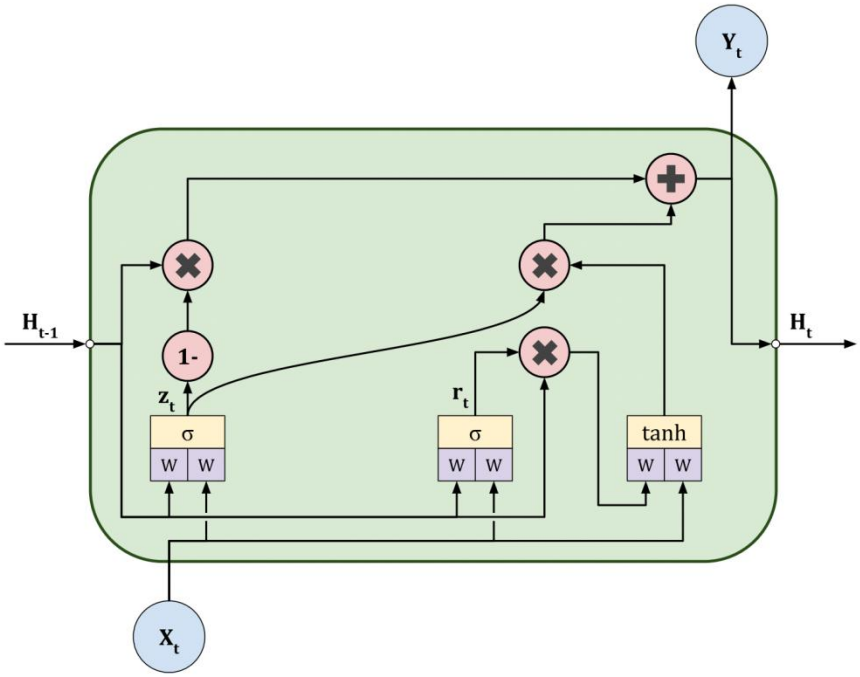

Figure 17 – GRU structure

GRUs have an update gate function. The main function of an update gate is to determine the ideal amount of previous information that is important for the future. One of the main reasons why this feature is so important is that the model can replicate every detail of the past to eliminate the vanishing gradient problem.

GRUs also have a reset gate function. The main reason why reset gates are vital is because they determine how much information should be ignored. It would be fair to compare the reset gate to the forget gate in LSTM as it tends to classify unrelated data and then force the model to ignore it and act without it.

Formally it looks this way (26):

$$\begin{aligned} z^{(t)} &= \sigma(W^{zx} \cdot X^{(t)} + W^{zh} \cdot H^{(t-1)} + b_z) \\ r^{(t)} &= \sigma(W^{rx} \cdot X^{(t)} + W^{rh} \cdot H^{(t-1)} + b_r) \\ \widetilde{H}^{(t)} &= \phi(W^{\widetilde{hx}} \cdot X^{(t)} + W^{\widetilde{hh}} \cdot \left(r^{(t)} \odot H^{(t-1)}\right) + b_{\widetilde{h}} \end{aligned} \quad (26)$$

$$H^{(t)} = \left(1 - z^{(t)}\right) \odot H^{(t-1)} + z^{(t)} \odot \widetilde{H}^{(t)}$$

Here $z^{(t)}$ is the output of update gate, $r^{(t)}$ is output of reset gate, $\widetilde{H}^{(t)}$ is a candidate state, $H^{(t)}$ is a hidden state, $\odot$ the operation of elementwise multiplication.

2.5 Attention mechanism and Connectionist temporal classification

There are two major architectures for ASR: Connectionist-Temporal Classification (CTC) and attention-based methods, both of them based on RNN. In CTC, the acoustic model emits not only output characters, but also emits blank neutral symbols which serve as connectors for the final output sequence of symbols. Neutral symbols allow the network to implicitly align longer acoustic features to output sequences of characters which is very short in length [96, p. 287].

CTC is a widely used deep learning algorithm of ASR systems. CTC was proposed by Alex Graves in 2006 [93, p. 1-8] and is widely used in modern ASR systems because it is able to handle variable-length sequences of input. CTC operates on the output of a neural network. The main purpose of CTC is to map the output E2E network to a sequence of characters which is the text of the spoken words. CTC considers all possible versions of alignments between the output of an E2E network and the target output sequence and computes the probability for alignment. The output of the CTC algorithm is the alignment with the highest probability. Advantages of CTC over traditional ASR algorithms are as follows: CTC does not require explicit segmentation of the input signals, that is why this algorithm is more robust to different variations in the speed of speaking and pronunciation.

Attention-based methods use the attention-based neurons to explicitly align acoustic frames to output characters. Attention-based methods can model statistical dependency between input and output sequences [96, p. 287].

2.5.1 Connectionist temporal classification

Neural networks in speech recognition are usually trained using individual fragments of audio recordings of speech. To do this, it is necessary to allocate separate marks corresponding to each frame, which entails the need to align the audio track and transcription. However, alignment is only reliable after training the neural network, which leads to a circular relationship between segmentation and recognition (known as Sayre's paradox). Moreover, in speech recognition tasks based only on word transcription, alignment is not useful.

Connectionist Temporal Classification [93, p. 1-8]. is a function that allows recurrent neural networks to be trained to recognize a sequence of words without initial alignment of input and output sequences.

To describe how the CTC works, let's start with an approach in which the CTC function is used as a loss function to train a neural network. The output layer of the neural network contains one block for each character of the output sequence (letters, phonemes, punctuation marks) and one more for an additional "blank" character, which corresponds to an empty output character. The output vector w_m is normalized

using the softmax function, which is interpreted as the probability of occurrence of a blank character with index k at time m (27):

$$P(k,m|x) = \frac{\exp(w_m^k)}{\sum_{k'=0}^{|w_m|} w_m^{k'}} \quad (27)$$

where w_m^k is the k-th element, w_m is the length of a word. Let α be a sequence of "blanks" and characters to align. The probability $P(\alpha|x)$ can be represented as a product probabilities of occurrence of symbols at each moment of time (28):

$$P(\alpha|x) = \prod_t^T P(\alpha_t, t|x) \quad (28)$$

For a given output sequence $|w_m|$, there are as many possible alignments as possible ways of placing "blanks" between characters. Let "-" mean "blank". For example, the alignments (x,-, y, z, -, -) and (-, -, x, -, y, z) correspond to the sequence (a, b, c). When identical characters appear consecutively, these repetitions are removed: (x, y, y, y, z, z) and (x, -, y, -, z, z) correspond to (x, y, z). Let's denote that B is an operator that first removes all repetitions, and then removes "blanks". Thus, the total probability of the output sequence w is equal to the sum of the probabilities of all possible corresponding alignments (29):

$$P(w|x) = \sum_{\alpha \epsilon B^{-1}(w)} P(\alpha|x) \quad (29)$$

where B^{-1} is the operator reverse to B.

This sum over all possible alignments allows the neural network to train on non-segmented data. It means, without knowing the exact location of the labels, we summarize over all the locations where they can be. This sum can be calculated using dynamic programming [93, p. 1-8]. Let w^* be the target sequence of words, then the neural network can be trained to minimize the CTC function (30):

$$CTC(x) = -logP(w^*|x) \quad (30)$$

A neural network can be trained with any optimization algorithm that uses a gradient.

Figure 18 shows a diagram of the CTC model, where the encoder can be a DNN, LSTM, BLSTM, CNN or any other kind of neural networks. In [93, p. 1-8], a CTC forward-backward algorithm is proposed, which uses a dynamic programming algorithm similar to the forward-backward algorithm for HMM [97, p. 281]. The main idea of this algorithm is that the sum over all alignments is split into the sum over the alignments corresponding to the prefixes of their output sequences. This sum can be efficiently computed using recursive direct and inverse variables.

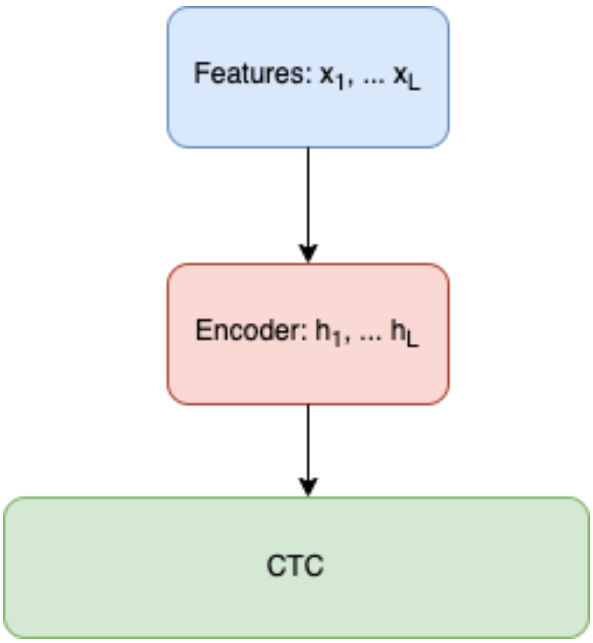

Figure 18 – ASR system with CTC

As for CTC decoding, two options for decoding integral CTC models were presented in [93, c. 1-8]. The first method is based on the assumption that the most likely alignment matches the most likely output sequence (31):

$$\arg max\ P(w|x) \approx B(\alpha^*) \tag{31}$$

where $\alpha^* = \arg\max P(\alpha|x)$. Calculating the best alignment is a simple task, because α^* is the concatenation of the most "active" outputs at each time step. However, this does not guarantee finding the most probable sequence of words.

The second method (the method of finding prefixes) is based on the fact that by modifying the forward-backward algorithm described above, one can efficiently calculate the probabilities of successive extensions of output sequence prefixes.

2.5.2 Attention Mechanism

Encoder-decoder models are often used for problems where the lengths of the input and output sequences are variable [55; 116]. The Encoder is a neural network that transforms the input $x = (x_1, x_2 \dots, x_L)$ into some intermediate representation $h = (h_1, h_2 \dots h_{L'})$. The decoder is usually an RNN that uses this intermediate representation to generate output sequences. The encoder can be any neural network, for example: DNN, LSTM, BLSTM, CNN. Figure 19 shows a diagram of the model.

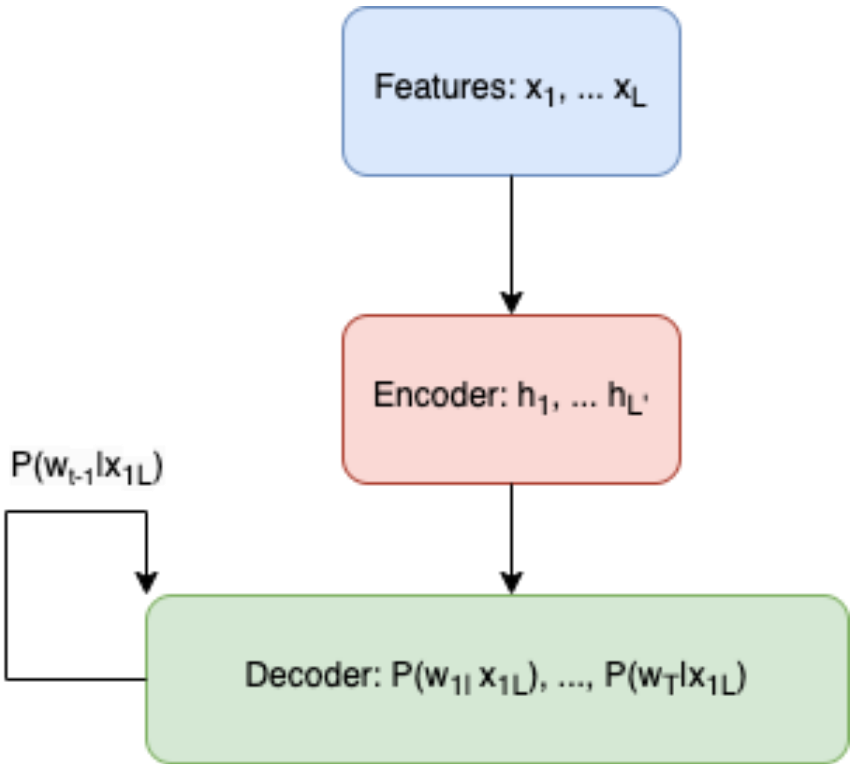

Figure 19 – Encoder-decoder ASR system

In , it was proposed to use an Attention-based Recurrent Sequence Generator (ARSG) as a decoder. It is a recurrent neural network that stochastically generates an output sequence $y_1, \ldots, y_t$ on the base of input h. The ARSG consists of the RNN and a subnet called the attention-mechanism. The attention mechanism selects a subsequence of the input sequence, which is then used to update the hidden states of the RNN and to predict the next output value. In the i-th step, recurrent sequence generator generates output y_i, focusing on certain elements of h (32)-(34):

$$\alpha_i = Attend(s_{i-1}, \alpha_{i-1}, h) \tag{32}$$

$$g_i = \sum_{j=1}^{j=L} \alpha_{i,j}\, h_j \tag{33}$$

$$y_i = Generate\ (s_{i-1}, g_i) \tag{34}$$

where s_{i-1} is the i-1-th state of the RNN, which is called Generator, $\alpha_i \epsilon \mathbb{R}$ is attention weights (another name is alignment). g_i sometimes called “glimpse”. Step is finished by recurrently (usually recurrent function if LSTM or GRU) calculating the new state for the generator (35):

$$s_i = Recurrency\ (s_{i-1}, g_i, y_i) \tag{35}$$

Scheme of attention mechanism is given in Figure 20.

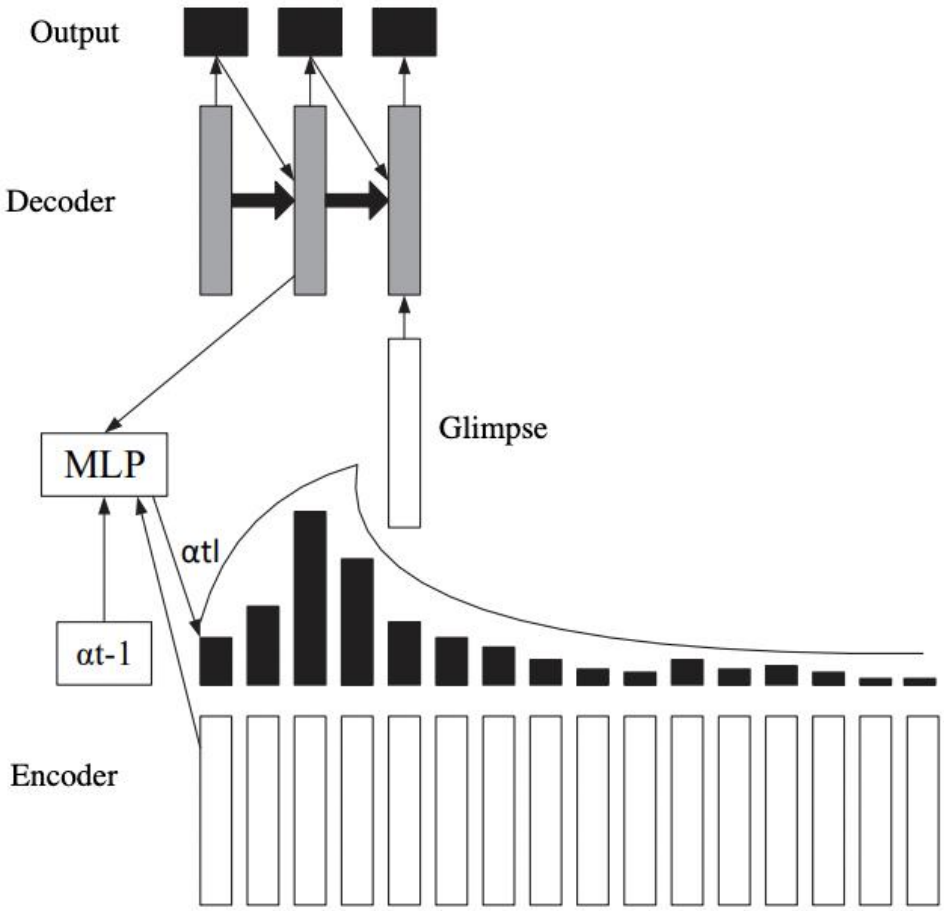

Figure 20 – Integral model with attention-based mechanism

In [117, p. 579], it was proposed to divide attention mechanisms into three types: location-based, content-based and a hybrid. Hybrid is the most general type. If Attend does not depend on α_{i-1}, then this is an content-based attention mechanism . Attention can be thought of as a normalized sum of the metrics of each h element (36), (37):

$$e_{i,j} = Score(s_{i-1}, h_j) \tag{36}$$

$$\alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{j=1}^{j=L} \exp(e_{j,j})} \tag{37}$$

The main limitation of such a scheme is that the same or very similar elements of h are considered the as the same, regardless of their positions in the sequence, which in speech recognition has significant importance. This problem is called "the problem of similar fragments of speech". Often this problem is partially solved by an encoder, such as Bi-LSTM or deep CNNs, which encrypt the context information into h elements. However, the sizes of h and their elements are always limited, which does not fully solve this problem.

For example, the location-based attention mechanism calculates alignment using the state of the generator and the previous alignment: $\alpha_i = Attend(s_{i-1}, \alpha_{i-1})$.

This type of attention mechanism predicts the distance between successive phonemes or symbols only by s_{i-1}, which can be difficult due to the large variance of this distance.

The hybrid attention mechanism uses the previous α_{i-1} alignment to select the short subsequence h, according to which the content attention mechanism will select the most relevant elements without the problem of similar fragments of speech.

2.5.3 Self-attention

$A(q, K, V)$ is attention-based vector representation of a word. Self-attention mechanism means the calculation of this vector for each word in a sentence($A^{<1>}$ ... $A^{<5>}$). Here, representation of each word is computed in parallel depending on the relations of a word to another word in a sentence (Figure 21). For example vector A will be calculated for the word "Дубайға". Of course, there could be used word embedding, but depending on the context the word "Дубайға" can be considered as historical place or the destination for holiday. In order to understand the context and choose appropriate representation for the considered word, the surrounding words would be looked up.

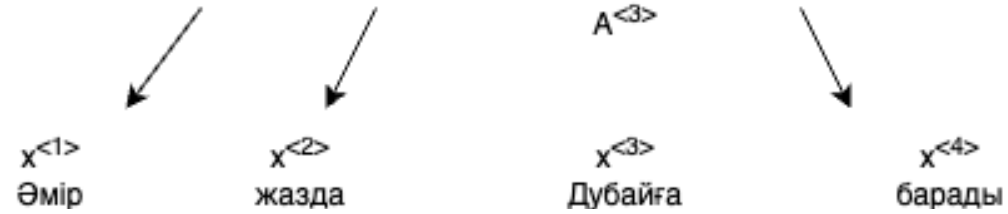

Figure 21 – Attention calculation for each word related to other words in a sentence

Vector A of the transformer attention is described in (38). This attention type includes softmax in its divisor as well as RNN attention (39):

$$A(q, K, V) = \sum_i \frac{\exp(q \cdot k^{<i>})}{\sum_j \exp(q \cdot k^{<j>})} v^i \tag{38}$$

$$a(t, t') = \frac{exp\left(e^{t,t'}\right)}{\sum_{t'}^{T_x} exp\left(e^{t,t'}\right)} \tag{39}$$

Before calculating attention, we firstly associate each word with three vectors: $q^{<i>}, k^{<i>}, v^{<i>,}$. Here $q^{<i>}$ is query, $k^{<i>}$ is key and $v^{<i>}$ is value. These vectors are found using the following matrices: W^Q, W^K, W^v. These matrices are learned parameters of the algorithm and they allow to pull-up query, key and value vectors for each word (40):

$$\begin{aligned} q^{<i>} &= W^Q \cdot x^{<i>} \\ k^{<i>} &= W^K \cdot x^{<i>} \end{aligned} \tag{40}$$

$$v^{<i>} = W^V \cdot x^{<i>}$$

where x^i is word embedding for the word i. After getting: $q^{<i>}, k^{<i>}, v^{<i>}$ vector it is necessary to take four values marked in white blue figures in Figure 22 in order to figure out more relevant representation and compute the softmax over them. For example the value $q^{<3>} \cdot k^{<2>}$ corresponding to word “жазда” can have the largest value. Further taken softmax values will be multiplied with $v^{<i>}$ values. Finally these multiplications will be summed up and will give $A^{<3>} = A(q^{<3>}, K, V)$.

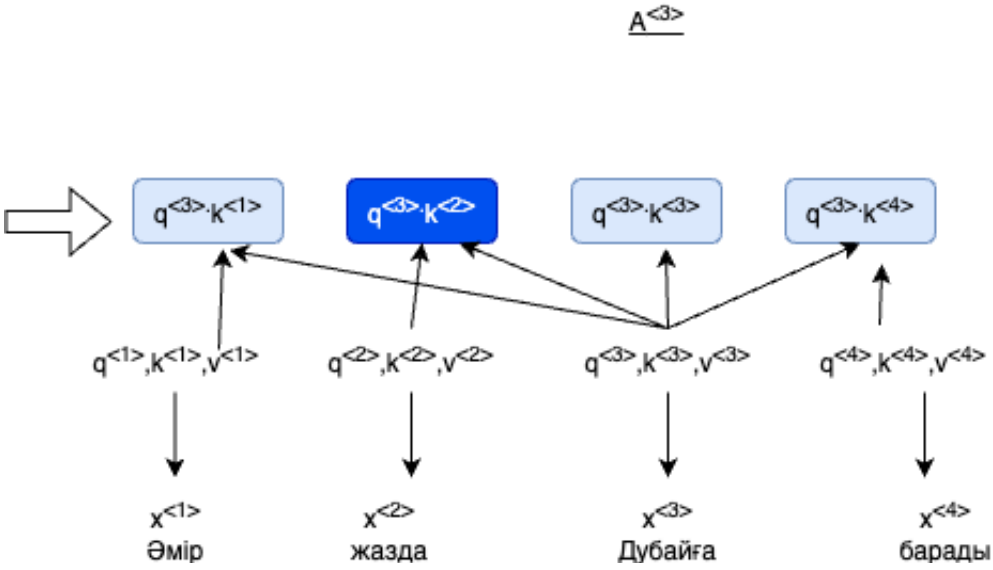

Figure 22 – Softmax calculation for inner products of query and key values

2.5.4 Multi-head attention

It's not easy to get dot product attention to work: bad initialization with random values can destabilize the learning process. This is handled by simultaneously calculating attention with several attention heads at once and concatenating the results, here each head has its own learning weights (41), (42):

$$h_i^{l+1} = Concat(head_1, head_2, \dots, head_K)O^l \quad (41)$$

$$head_k = Attention(Q^{k,l}h_i^l, K^{k,l}h_j^l, V^{k,l}h_j^l) \quad (42)$$

where $Q^{k,l}, K^{k,l}, V^{k,l}$ are trainable weights of k-th attention head, and O^l-is projection which helps dimensions h_i^{l+1}, h_i^l to match all layers. In essence, multiple "heads" allow the attention mechanism to "make multiple bids at once", allowing you to look at different transformations or aspects of features from the previous layer.

2.5.5 ASR architectures based on attention mechanism

2.5.5.1 Transformer

Content based global relations in sequences can be captured by transformer models [24, p. 5036]. Transformers are mainly based on the attention mechanism which is firstly mentioned in . Transformers were originally developed for machine translation and have gradually replaced RNNs in mainstream NLP tasks. The architecture has a fresh approach to learning representations: completely getting rid of recurrence, transformers for each word build features using the attention mechanism to determine the importance of all other words in the sentence for this word. Thus, the constructed features for a given word are simply the sum of linear transformations of the features of all words, weighted by this "importance".The described architecture in formal language looks like this way: so, h is the hidden representation of the i-th word in the sentence S - from layer l to layer $l+1$ is updated like this (43):

$$h_i^{l+1} = Attention\left(Q^l h_i^l, K^l h_j^l, V^l h_j^l\right)$$

$$h_i^{l+1} = \sum_{j \in S} w_{ij} (V^l h_j^l) \tag{43}$$

where (444):

$$w_{ij} = softmax_j (Q^l h_i^l, K^l h_j^l) \tag{4}$$

where $j \in S$ is a set of words in a sentence, Q^l, K^l, V^l trainable linear weights(abbreviated from Query, Key, Value). The attention mechanism is computed in parallel for each word in a sentence to get their updated features at once. It is the advantage of the transformer architecture over RNN which updates features word by word. The mechanism of attention is easier to understand from of steps from
Figure 23:

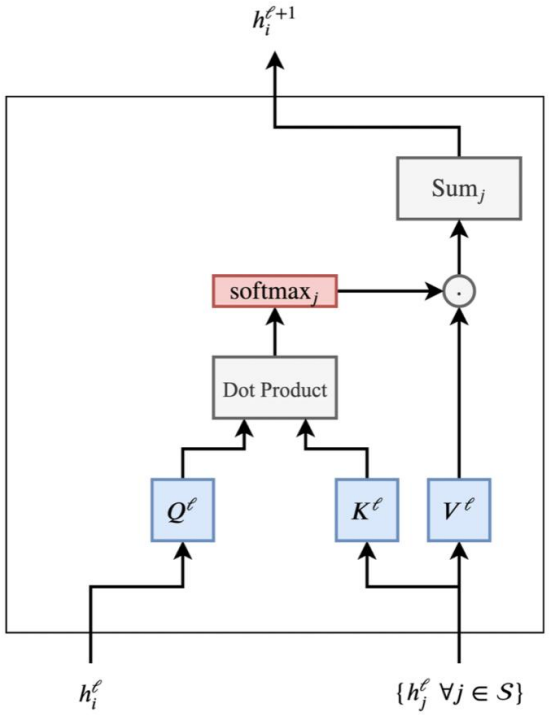

Figure 23 – Attention mechanism in Transformers

In Figure 23, features for a given word h_i^l and all other words from a sentence $h_i^l,\ \forall j \in \mathcal{S}$ will be calculated attention weights w_{ij} for each pair (i, j) as a scalar product. Afterwards this will be applied softmax-function through all j. At exit updated features h_i^{l+1} for the word i will be taken summing up by all h_j^l weighted by corresponding w_{ij}. Each word of the sentence concurrently passes the same chain of calculations.

The key problem, due to which the architecture of transformers looks like as it is: the values of the features for each word after applying the attention mechanism can be very different in magnitude. First problem may appear due to too "sharp peaks" or, on the contrary, uniformity in the attention distribution w_{ij}. Second problem is that when we concatenate the outputs of several heads for each word, and they can also be very different in "scale". Therefore, in the final vector h_i^{l+1} the value spread can be large. According to the practices accepted in machine learning, here it makes sense to add a normalization layer to the calculation chain.

Second problem is solved by transformers using LayerNorm, which normalizes and learns the affine transformation at the feature level. In addition, dividing the attention dot product by the square root of the dimension helps to hold the first problem.

Final trick to deal with the problem of scaling: the values at each position are transformed by a two-layer perceptron with a special structure. After applying multi-head attention, they project h_i^{l+1} into an absurdly high dimensionality using trainable weights, afterwards they are transformed with a non-linear ReLU activation function, and then values are projected into the original dimension, to further pass an another normalization (45):

$$h_i^{l+1} = LN(MLP(LN(h_i^{l+1}))) \quad (45)$$

here LN means Layer Normalization, MLP stands for Multi-Layer Perceptron. Final structure of Transformer architecture is given in Figure 24. These layers made "Transformer" architecture deeper, and this allows the NLP community to increase both the number of parameters and the size of the datasets. Residual connections between the inputs and outputs of each "sublayer" of multi-headed attention and fully connected "sublayer" are the key detail that allows you to stack the layers of the transformer on top of each other.

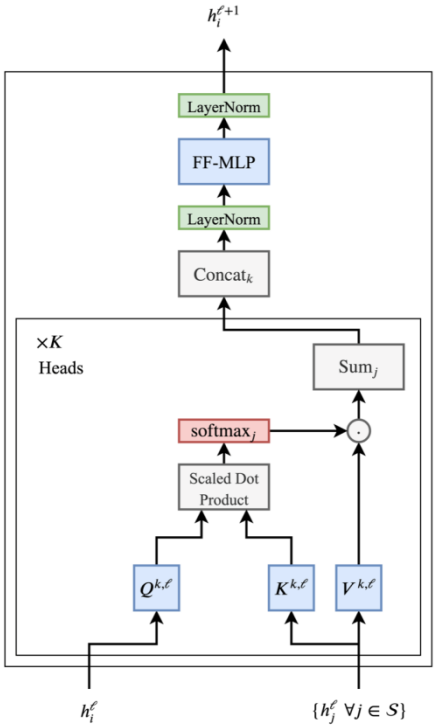

Figure 24 – Transformer with Layer Normalization and Multilayer Perceptron

2.5.5.2 Conformer

The conformer architecture is effective in many speech processing tasks. It has advantages of convolutions and attention mechanisms. Convolution is good for short-term local dependencies, while self-attention is good for capturing long-term dependencies [24, p. 5036; 120]. The reason of these specific abilities can be described next way: the transformer using the self-attention mechanism captures the global context well, but does not extract local features very well. Convolutional neural networks, on the other hand, use local features efficiently, but require a large number of layers to capture the global context. A conformer combines convolutional layers with a self-attention mechanism.

First, the data supplied to the input of the Conformer is augmented. The SpecAugment method is used for speech recognition. SpecAugment applies three types of deformations to the Mel-spectrogram: time distortion (lengthening or compression of a certain interval of the record), removal of a certain time interval

from the record, and removal of a certain frequency interval. Thus, when training on noisy data using SpecAugment, the network is trained on features that are resistant to time deformation, partial loss of frequency information, and loss of small segments of speech. The conformer processes the final augmented inputs with a convolutional neural network consisting of a pooling layer, a fully connected layer, and a dropout, and then with a sequence of Conformer blocks.

Conformer blocks are a sequence of two macaron-like feed forward modules [120, p. 17629], between which there is a Multi-Head Self Attention module and a convolutional module, followed by normalization layer (Figure 25).

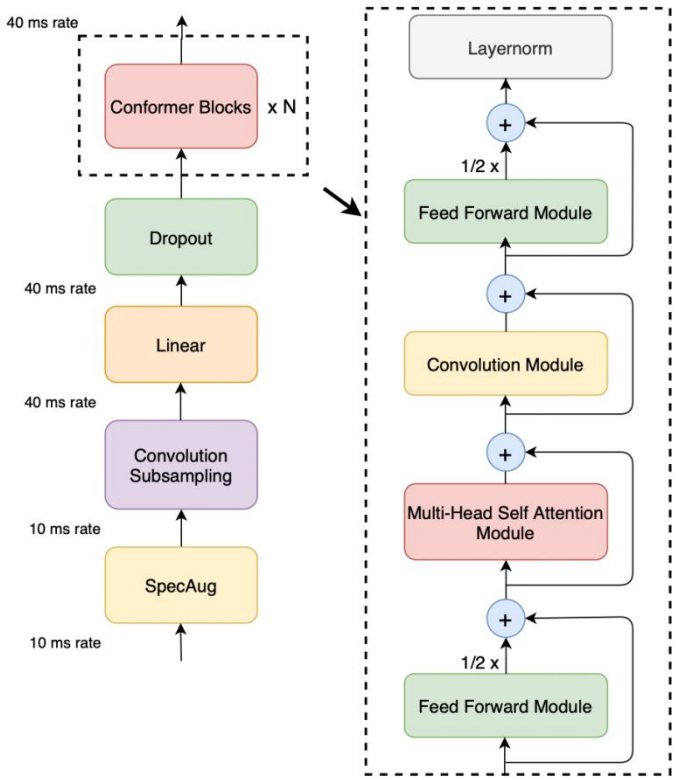

Figure 25 – Architecture of Conformer encoder

Mathematical description of Figure 25 looks like as follows (46):

$$\begin{gathered} \widetilde{x_i} = x_i + \frac{1}{2} FeedForward(x_i) \\ x_i' = \widetilde{x_i} + MultiHeadSelfAttention(x_i) \\ x_i'' = x_i' + ConvolutionModule(x_i') \\ y_i = NormalizationLayer(x_i'' + \frac{1}{2} FeedForward(x_i'')) \end{gathered} \tag{46}$$

here x_i is the input, y_i is the output of i-th Conformer block.

2.5.5.3 Branchformer

This architecture was proposed in 2022 in the study [120, p. 17629]. Here authors state out limitations of Conformer encoder, like combining self-attention and convolution sequentially. After looking for questions dedicated to relationships in different layers of Conformer encoder, and importance of initial layer operations authors proposed Branchformer architecture calling it flexible and customizable. This type of encoder architecture has two branches one of which uses self-attention to capture long-term dependencies and another one that uses gated Multi-layer Perceptron (gMLP) for processing local relations (Figure 26). Two branches run in parallel.

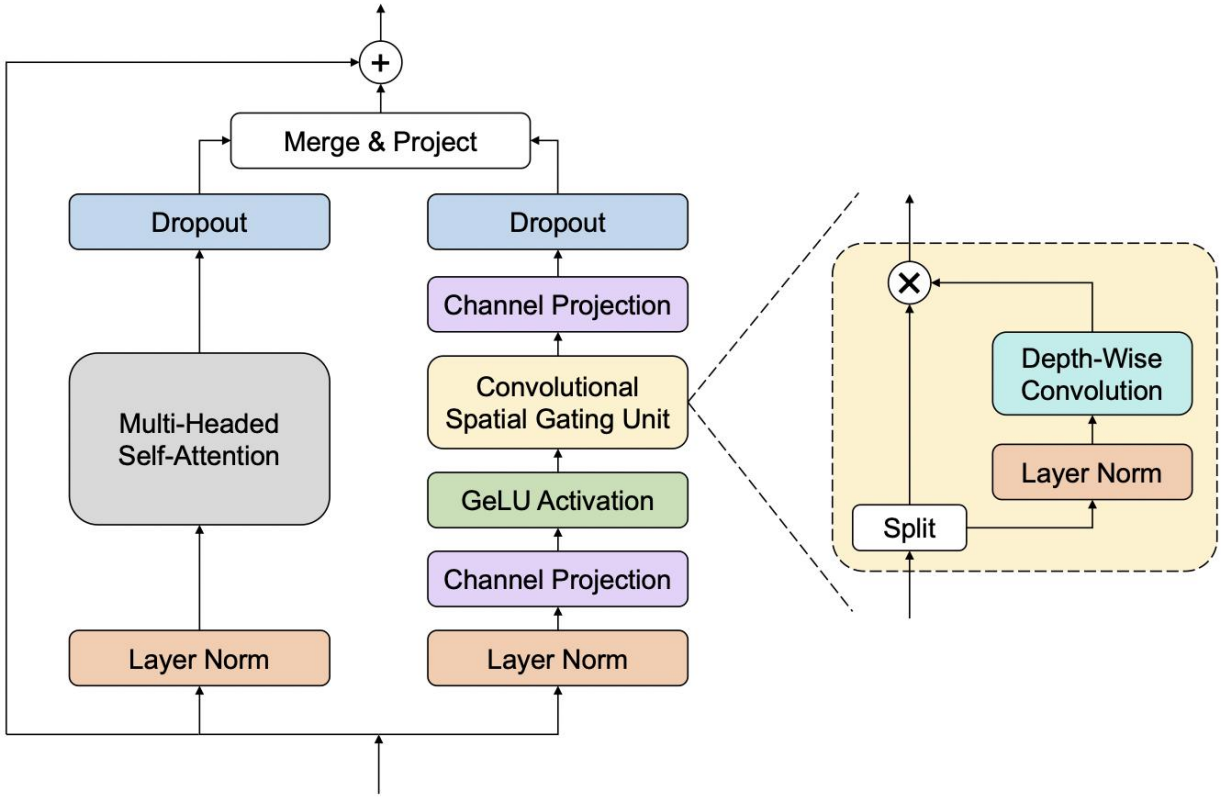

Figure 26 – Branchformer architecture

Outputs of two branches are concatenated. This concatenation also can be replaced by weighted average in order to make it interpretable where local and global relationships of context were used in layers. Concatenation can be performed along the dimension of features and the result can be projected to original dimension again (47):

$$Y_{merged} = concat(Y_{att}, Y_{gmlp})W_{merge} \quad (47)$$

where W_{merge} is trainable matrix, Y_{att}, Y_{gmlp} output sequences of attention branch and gMLP branch respectively. Despite the effectiveness of merging by concatenation, it is not easy to modify. As a result Branchformer authors proposed weighted average method. This approach firstly summarizes the output sequence from each branch with attention-based pooling method (48):

$$y_{att} = AttentionPooling(Y_{att})$$
$$y_{gmlp} = AttentionPooling(Y_{gmlp}) \quad (48)$$

Further in order to get weights of branches, these vectors are projected to some scalars and modified to normal case by softmax function (49):

$$w_{att}, w_{gmlp} = softmax(W_{att}y_{att}, W_{gmlp}y_{gmlp}) \quad (49)$$

here W_{att}, W_{gmlp} are linear transforms. Finally weighted average captures all of local and global connections and looks like this (50):

$$Y'_{merged} = w_{att}Y_{att} + w_{gmlp}Y_{gmlp} \quad (50)$$

To Branchformer can be applied branch dropout, which means the dropping of an entire branch for attention mechanism.

3 EXPERIMENTS AND RESULTS

3.1 Data collection

3.1.1 Introduction

Improving automatic speech recognition for low-resource languages is one of the most urgent problems of today. The Kazakh language belongs to the group of agglutinative Turkic languages with few resources. In general, all the languages which belong to the Turkic language family are representatives of the group of agglutinative languages, and almost all of them are representatives of languages with few resources [8, p. 84]. Due to the lack of information in the form of audio-text pairs for these languages, they are called low-resource languages. For example, one of the largest publicly available corpora is the Open-Source Uzbek Speech Corpus of the Uzbek language, which is only 105 hours long and has 258 hours of Uzbek language data collected as part of Mozilla's Common Voice project. Common Voice is Mozilla's project, where collected and made available to the public audio-text information of the world's languages. Any volunteer can participate in the development of this project. For the Kazakh language, ISAAI has a corpus of 1,200 hours consisting of 600,000 sentences . Information about most of other Turkic languages can be found only in Common Voice.

There are several research works on speech recognition of the Kazakh language. For example, some works have improved speech recognition by using well-known recurrent neural networks and some by using the architecture with single hybrid neural systems [38, p. 263]. If one paper considered the improvement the automatic speech recognition for the Kazakh language by conducting transfer learning over the model of the Russian language [40, p. 5884], another paper developed mutual transfer teaching of the Kazakh and Azerbaijani languages, which belong to the group of related languages [8, p. 84]. In another work, the Kazakh language was trained together with other Turkic languages. All these works have achieved better results than the previous ones. And it can be observed that the more information that is taught to everyone, the more accurate the speech recognition will be.

Due to the fact that the tuning the hyperpameters for transfer learning method is complicated, and the fact that combined language training basically improves an acoustic model, the probability of error in identifying speech in a specific language for short contexts is high, the relevance of collecting "Clean" information for specific languages has not disappeared. Therefore, within the framework of this research work, the work of enlarging the corpus of the Kazakh language was also carried out.

3.1.2 Methodology: raw data collection, combining the collected data into single corpus and data normalization

For the development of reliable automatic speech recognition system, it is very important to have a sufficient amount of transcribed data, as in all other areas of machine learning. At this stage of the research, it was very important to combine the collected data for all periods into one large corpus for further ASR, which could be

used in practical applications. Therefore, it was decided to combine the following data sets:

1. 283 hours of data, previously collected in the laboratory of the Institute of Information and Computing Technologies.
2. Data collected for the 2022 year and marked in the laboratory of the Institute of Information and Computing Technologies. This audio data contains phone conversations, audio recordings of zoom meetings, news channels: 195 hr 11 min 25 sec (It was expected that in the end we would have 478 hours of data, but after removing duplicates, only 407 hours remained. It is very important to exclude duplicates in order to preserve the quality of the resulting ASR model).
3. Writing a script for collecting data with different encodings into one file with UTF-8 encoding(UTF-8, UTF-16, rk1048).

195 hours of prerecorded phone dialogues and audio files of zoom meetings specially collected for researches, were collected and further processed for comprehensive improvement of speech recognition in the Kazakh language. During the processing of telephone dialogues, audio files were separated from two-channel audio into two separate channels using ffmpeg software, and the audio information of each channel was recorded in a separate file (Figure 27). Audio files were further cut into short audios of 8 seconds duration. These audio files were further used for text annotation.

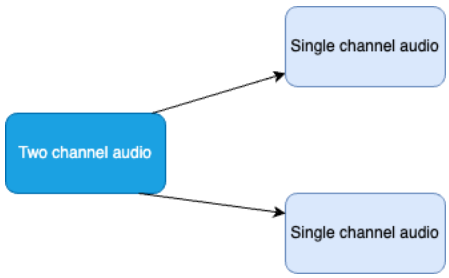

Figure 27 – Split two-channel audio-recordings into single channels

The script for editing audio files was written in C++. However, these script lines can also be used in other programming languages.

1. Script line for dividing files into channels:

str += string("ffmpeg -i ") + filename + " -map_channel 0.0.0 " + left + " -map_channel 0.0.1 " + right +" -report";

2. Script line for cutting files into 8 second files:

lstr="ffmpeg -i "+left+ " -f segment -segment_time 8 "+trunk_path+ltmp+"%03d.wav";

3. Script line for changing the frequency of files:

str += string("ffmpeg -i ") + filename + " -ar 16000 " + out;

The most effort and time in data collection was needed to process the data, collected in 2022: as the markup was done on different operating systems the human factor was unavoidable. The data contained encoding UTF-8, UTF-16, rk1048), and this required the development of an algorithm which is able to collect the data into

one file using only one script. Python was choosen as the language for writing the script, as this language throws an exception by default, in the process of all other encodings, except UTF-8. Next, all files that were thrown out to the exception processing block, depending on the presence of bytes b’ \xef\xbb’ , b’ \xff\xfe’ , were decoded into strings with UTF-8, UTF-16 or rk1048 encoding. The encoding type selection algorithm depending on the availability of one or another byte is shown in Figure 28. The complete program code is given in (APPENDIX C).

```
try:
    if bytes.startswith(b'\xef\xbb'):
        bytes = bytes.replace(b'\xef\xbb', b'')
        print(bytes)
        sentence = str(bytes, 'UTF-8')
    elif bytes.startswith(b'\xff\xfe'):
        bytes = bytes.replace(b'\xff\xfe', b'')
        print(bytes)
        sentence = str(bytes, 'UTF-16')
    else:
        sentence = str(bytes, 'rk1048')

    if sentence == "":...
    else:...
except:
    txt_file_path = str(subdir + "/" + main_title + ".txt")
    os.system("cp " + full + " " + after_broken_dir)
    os.system("cp " + txt_file_path + " " + after_broken_dir)
```

Figure 28 – Encoding selection for files that were thrown into an exception after attempting direct decoding with UTF-8

Next step was dedicated to the normalization of data, collected to one large text file: removal of numbers, punctuation marks, extra, special and invisible symbols. As the comparison of one symbol to a long speech leads to force alignment all numbers were rewritten to text format. Also all letters were converted to lowercase, because of the reason that, ASCII code of uppercase and lowercase letters and having different cases will have impact on the performance of ASR. The same reason served as a base for clearing the text from punctuation marks.

ASR training tools like Kaldi and ASR throw exceptions for special and invisible symbols. In order to make the collected corpus trainable on different tools, the symbols, like '\u200c', '\u0x00' and etc. were removed from transcripts. The script for removing these symbols is given in Figure 2929.

```
import unicodedata, re, itertools, sys

all_chars = (chr(i) for i in range(sys.maxunicode))
categories = {'Cc'}
control_chars = ''.join(c for c in all_chars if unicodedata.category(c) in categories)
# or equivalently and much more efficiently
control_chars = ''.join(map(chr, itertools.chain(range(0x00,0x20), range(0x7f,0xa0))))

control_char_re = re.compile('[%s]' % re.escape(control_chars))



def remove_control_chars(s):
    return control_char_re.sub('', s)
```

Figure 29 – Script for removing invisible symbols from a text

After combining and processing all the necessary data, the validity of the data was checked by training the ASR on ESPNet. ESPNet was selected as a tool for obtaining an integrated model. An architecture using a conformer encoder and a transformer decoder was chosen. The input audio information is processed by CNNs, Bi-LSTM neural system is used to obtain hidden layers in the encoder. The decoder improves the attention mechanism by fixing connection class coefficients, and additionally calculates the weight of the language model with a coefficient of 0.3 during decoding. As experiment held only with the purpose to check trainability of the data, other parameters of neural network weren't analyzed.

The obtained model showed that the quality of the collected data is suitable for ML training dedicated to automatic speech recognition. Even if the model parameters were not carefully chosen, the accuracy of the model was obtained within reasonable values (Table 5).

Table 5 – Performance of test ASR trained on collected audio-text data

Dataset types	WER (%)	CER (%)
Train	20.4	8.2
Test	22.4	9.3

3.1.3 Use of trained ASR model

The ability of the trained ASR model was presented by a Telegram bot. User sends audio messages to the bot and received the answer as a transcribed text form of sent audio file. Source code of Telebot is given in (APPENDIX D) and its interface in Figure 30.

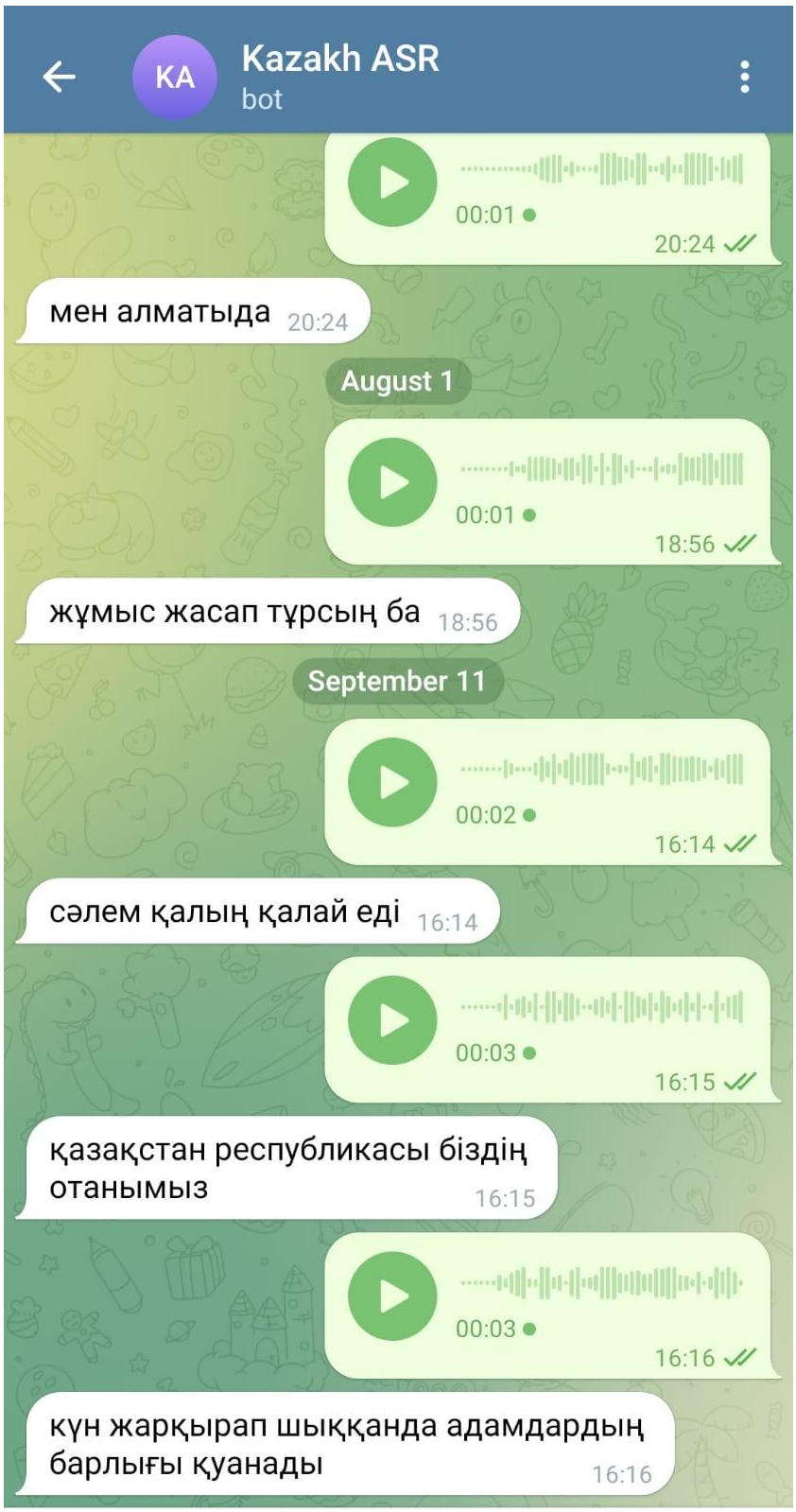

Figure 30 – Interface of Telebot, implemented using trained ASR

3.1.4 Conclusion

Constructing the model needed to build a speech recognition system, like any other type of machine learning, requires an adequate dataset. But processing the information collected in the real environment in the form of audio-text and making it usable is one of the most complex types of information collection. The 195-hour

marked database collected and edited by the authors in more than 1 year is suitable material for use in experiments for the purpose of creating speech recognition systems in the Kazakh language, and currently for expanding other databases. After removing duplicates and merging the collected database with 283 hours of previously collected data the total volume of data reached 396 hours. This is evidenced by the results of experiments conducted for the purpose of testing the database.

3.2 Multilingual training experiments

3.2.1 Introduction

Kazakh language is a low-resource agglutinative language from the Turkic family languages which belong to agglutinative languages. Moreover, almost all these languages suffer from data shortages [8, p. 84]. Data scarcity for these languages is found in the presence of less-transcribed audio. For example, the largest open-source corpus, which is an Open-Source Uzbek Speech Corpus, has only 105 hours. There were 258 recorded hours and 97 validated hours in Uzbek in Common Voice. Common Voice is a Mozilla's project for collecting open-source datasets of transcribed audio data for all possible languages in the world. Anyone can participate in improving this resource. ISAAI's Kazakh language corpus contains 335 hours of transcribed audio (two hours in Common Voice). Some agglutinative languages only have corpuses in the Common Voice.

Inspired by the results presented in some papers [125, 126] we decided to use a multilingual model using datasets of several agglutinative languages because multilingual models demonstrate stable gains over monolingual models [8, p. 86]. Therefore, it is supposed that the multilingual model taken for the group of agglutinative languages can decrease Character Error Rate (CER) and Word Error Rate (WER) for distinct languages included in the experiments. Moreover, it is assumed that the model can be used as a base for providing transfer learning for distinct groups of languages.

Experiments provided on 11 languages from the Roman family give better results on Multilingual Deep Neural Networks (DNN) compared to monolinguals . Transfer learning conducted using the English language model significantly improved the CER for 12 languages (some agglutinative languages) [126, p. 4221]. The reason for this hypothesis is that agglutinative languages have common characteristics including linguistic structure, agglutinative morphology, and vowel harmony . Furthermore, attempts to use a language model of the Russian language to Kazakh [40, p. 5884] resulted in unsatisfactory Error Rates because word formations of these languages are different [8, p. 86].

According to previous research, it was decided to train a multilingual model for languages from the Turkic family using the Cyrillic alphabet. The reason for choosing such a method is that the similarity of the alphabet can lead to better compatibility among language words that have common roots.

Most of existing combining methods of datasets of different languages does not take into account relations of languages to each other. The basic idea of this research is to study the impact of combining languages from Turkic language family, with

similar scripts. Common word and sentence formation rules of the selected languages with similar scripts allow to get a working model for each of languages included in the experiments. The contributions of this research are:

1. ASR development method for critically low-resource languages.
2. Improving ASR performance for languages from one language family.

Further subsections present a review of relevant work found in the literature, describe the datasets and the methods used in this study, present the main results of the study, contain a discussion of the results and a comparison with state-of-the-art methods. The last subsection draws conclusions and proposes directions for future work.

3.2.2 Related Work

Pooling resources from different languages is helpful for low-resource languages [127, p. 8622; 129-132].The same idea can also be applied to distinct language families. Investigations on combining experiments, such as Transfer and Multilingual training on agglutinative languages, and improvements in WER and CER were mentioned in [8, p. 86; 133]. Applications of ESPNet and its benefits for languages of this family can be found in [30; 133, p. 454]. ESPNet is a deep neural network-based automatic speech recognition toolkit that was proposed in 2018 . ESPNet's conformer encoder and transformer decoder, which are used for low-resource languages, obtained an improvement of more than 15% [135, 136]. The application of the conformer encoder in the multilingual end-to-end (E2E) model yielded better results than the others [131, p. 1-4].

Suggestions for further use of multilingual models as a basis for transfer learning could be justified by the results of . A multilingual deep neural network (DNN) and a matrix factorization algorithm were used to extract high-level features. In the second stage, the authors applied a joint CTC-attention mechanism with shallow Recurrent Neural Networks (RNNs) for high-level features extracted at previous levels. The authors state that the proposed architecture is the best among all the existing end-to-end transfer models. The use of a multilingual model in , built on a collection of adversarial languages for further providing transfer learning for absolutely different languages, shows improvement by decreasing WER up to 10.1%. This study uses the IARPA Babel. IARPA Babel is a program aimed at improving automatic speech recognition in a large number of different languages 1. Using subword units in a CTC-attention-based system on the LibriSpeech 1000h dataset obtained an improvement over a character-based hybrid system, reducing the WER to 12.8% without a language model . Moreover, the authors stated that using subwords helps to avoid out-of-vocabulary problems. This can be assessed as a reason for choosing languages from one language family because a large part of the words from these languages have common roots.

Another successful example of using end-to-end ASR for agglutinative languages can be found in . In this study, experiments were conducted using a conformer model and CTC-Attention on data augmentation, noise injection, and exponential moving average. For the Corpus of Spontaneous Japanese, the authors

achieved a state-of-the-art CER of 3.2%. The paragraphs above summarize information about previous works on Turkic languages and the reasonability of the models and network chosen for the experiments in this study.

Experiments in [126, p. 4218] were conducted using Mozilla's Deep Speech v0.3.0 . A total of 26 h of data for Tatar and 10 h of data for Turkish were used to train the ASR model for these languages over the English language model. The CER for Tatar was 26.42%, whereas that for Turkish was 27.52%.

A Deep Neural Network with Hidden Markov Model (DNN-HMM) system was applied to Turkish, one of the most popular and widely used Turkic languages. The dataset consisted of 6.1 hours of data, collected from mobile devices. In comparison with GMM-HMM systems, the authors obtained a WER of approximately 2.5 in comparison with the GMM-HMM systems.

The authors of [31, p. 634-1] investigated questions of speech recognition in emergency call centers for the Azerbaijan language. In the experiments, two types of datasets were used: dialogue dataset (27 h) and summary dataset (57 h). The GMM-HMM and DNN-HMM were applied to train the acoustic model. The authors found that the DNN-HMM showed better results in the experiments, and the trigram language model gave no risk of overfitting.

In [37, p. 8337-1], the authors considered a transformer architecture with self-attention components that can shorten the training process by parallelizing the processes for the recognition of speech in the Kazakh language. Application of the Transformer + CTC LM model decreased the CER and WER to 3.7% and 8.3%, respectively, for 200 h of read speech.

The hybrid model used in [63, p. 48720] comprised a CTC with an attention mechanism for 400 h of data. The results were as follows: CER = 9.8% and WER = 15.3%. Including Language Model (LM) in this composition led to a significant decrease in the CER and WER (5.8%, 12.2%).

The end-to-end conformer model in for the Uzbek language for 105 h of data volume gave more effective results over E2E+LSTM and E2E+Transformer for Uzbek language from the Turkic family. The end-to-end conformer model showed lower error rates (CER=5.8%, WER=17.4%) when the Language Model was included in the decoder.

In [8, p. 84], an attempt was made to fit a model trained on the Kazakh language dataset to the Azaerbaijani dataset. In this experiment, the (NMF) algorithm was used to extract features from the audio data. NMF is necessary to reduce hidden level outputs and decrease redundant values from high-level vectors [137, p. 18-1]. Furthermore, these characteristics were trained on the attention mechanism of the joint CTC. This approach gave Phoneme Error Rate (PER) equal to 14.23%.

Authors of [133, p. 448-158] study single E2E Automatic Speech Recognition (ASR) using the ESPNet toolkit for the commonly used languages in Kazakhstan: Kazakh (KZ), English (EN), and Russian (RU). The combined dataset of the three languages has a total volume of 975.6 hours. To solve the issue of grapheme compatibility, the authors also combined the grapheme sets of all languages. The training results showed an average WER of 20.5%.

Table 6 – Summary of different models and approaches applied for agglutinative languages

Model	Language	CER (%)	WER (%)	Volume of data (hours)
CTC and attention	Kazakh	9.8	15.3	400
CTC and attention + LM	Kazakh	5.8	12.2	400
Transformer and CTC + LM	Kazakh (Read speech)	3.7	8.3	200
Transformer and CTC + LM	Kazakh (Conversational telephone speech)	9.6	15.8	200
E2E-Conformer	Uzbek	7.5	21.2	105
E2E-Conformer+LM	Uzbek	5.8	17.4	105
Transformer architecture	Combined data of Kazakh, English and Russian languages	n.a.	20.5	975.6
Transfer learning over English ASR model on DeepSpeech (A six-layer unidirectional CTC model, with one LSTM layer)	Tatar	26.42	n.a.	26
Transfer learning over English ASR model on DeepSpeech (A six-layer unidirectional CTC model, with one LSTM layer)	Turkish	27.55	n.a.	10

Table 6 shows a summary of the comparative results of different models for different volumes of data for the Kazakh and all mentioned Turkic languages. The Table 6 compares the CER and WER achieved using the different models and dataset sizes.

3.2.3 Materials and Methods

3.2.3.1 Datasets

Almost all Turkic languages have common rules of word formation, and words can have the same meaning in these languages. Table 7 shows some examples of phrases formed using words with similar soundings and meanings for the Azeri and Kazakh languages, which belong to the Turkic language family.

Table 7 – Comparative table of expressions with the same meaning in Kazakh, Kyrgyz and Azerbaijan languages

Kazakh	Kyrgyz	Azerbaijani	English meaning
бір алма [biyr alma]	бир алма [bir alma]	bir alma [bir alma]	one apple
терең көл [tereng kyol]	терең көл [tereng kyol]	dərin göl [daerin qyol]	deep lake
ақ доп [aaq dop]	ак топ [aaq top]	ağ top [aack top]	white ball
қара қой [qara qoy]	кара кой [kara koy]	qara qoyun [qara qoyun]	black sheep

To make the experiment reproducible, data from the open-source dataset Common Voice was used in the present work. Another reason for choosing this open-source resource is that the dataset for the Kazakh language is only one hour long. This allows us to observe the effect of a multilingual approach on critically low-resource languages. When the experiments of the current investigation began, Mozilla's Common Voice Corpus 8.0 [126, p. 4219] was the latest available.

Languages with Cyrillic scripts were chosen from the dataset: Kazakh (1 h), Bashkir (265 h), Kyrgyz (44 h), Tatar (29 h), and Saha (6 h). The datasets mentioned contain a wide range of speaker ages for both males and females, as shown in

Table 8 and Table 9. For some languages, there are no samples of older speakers, and in the case of Tatar, there are no samples of speakers less than 19 years of age. Male speakers were predominant in terms of gender, with the exception of Bashkir.

Table 8 – Distribution of the data used – number of speakers, by age

Languages	Total hours (validated)	Ages (%)					
		<19	19-29	30-39	40-49	50-59	60-69
Tatar	29		5	73		1	
Kazakh	1	6	26	3		11	
Sakha (Yakut)	6	11	2	44	7		
Bashkir	265 (255)	4	17	17	6	5	20
Kyrgyz	6.5	19	67	8	1		

Table 9 – Distribution of the data used – number of speakers, by gender

Languages	Gender	
	Male	Female
Tatar	79	2
Kazakh	42	3
Sakha (Yakut)	54	10
Bashkir	30	40
Kyrgyz	54	36

3.2.3.2 Speech Recognition Models

Our experiments included five agglutinative languages with Cyrillic scripts: Bashkir (ba), Kazakh (kk), Kyrgyz (ky), Sakha (sah), and Tatar (tt), with the next grapheme set (Gr_{ba}, Gr_{kk}, $Gr_{ky,}$ Gr_{sah}, Gr_{tt}). The grapheme set is understood using letters of the languages. Examples of grapheme letters in the five languages are listed in Table 10. Bold and red letters indicate specific letters for distinct languages.

Table 10 – Graphemes for Turkic languages with Cyrillic alphabet, included in the experiments

ba	ky	sah	kk	tt	All
1	2	3	4	5	6
а	а	а	а	а	а
ә			**ә**	**ә**	**ә**
б	б	б	б	б	б
в	в	в		в	в
г	г	г	г	г	г
ғ			**ғ**		**ғ**
		Ҕ			**Ҕ**
д	д	д	д	д	д
е	е	е	е	е	е
ё	ё			ё	ё
ж	ж	ж	ж	ж	ж
				җ	**җ**
ҙ					**ҙ**
з	з	з	з	з	з
и	и	и	и	и	и
			і		**і**
й	й	й	й	й	й
к	к	к	к	к	к
қ					**қ**
			ҟ		**ҟ**
л	л	л	л	л	л
м	м	м	м	м	м
н	н	н	н	н	н
	ң				**ң**
		ҥ			**ҥ**
ң	**ң**		**ң**	**ң**	**ң**
о	о	о	о	о	о
ө	**ө**	**ө**	**ө**	**ө**	**ө**
п	п	п	п	п	п
р	р	р	р	р	р
с	с	с	с	с	с
ҫ					**ҫ**

Table continuation 10

1	2	3	4	5	6
т	т	т	т	т	т
у	у	у	у	у	у
ү	ү	ү	ү	ү	ү
			ұ		ұ
ф	ф	ф		ф	ф
х	х	х	х	х	х
h		h		h	h
ц	ц	ц		ц	ц
ч	ч	ч		ч	ч
ш	ш	ш	ш	ш	ш
щ	щ	щ	щ	щ	щ
	ъ			ъ	ъ
	ы	ы	ы	ы	ы
	ь	ь		ь	ь
	э	э		э	э
	ю	ю	ю	ю	ю
		я	я	я	я

The training sets for each language are defined as a pair $\{X_i, Y_i\}$ for language i. Therefore, the datasets are defined as follows (51):

$$\{X_i, Y_i : i \in (ba, kk, ky, tt, sah)\} \quad (51)$$

where X_i is an input given as acoustic features, Y_i is the corresponding output or target sequence of characters. The training and grapheme datasets for the multilingual language model were combined from the data for distinct languages according to [133, p. 451].

The multilingual dataset is defined as the union of five datasets of languages (52):

$$\{X_{all}, Y_{all}\} = \{X_{ba}, Y_{ba}\} \cup \{X_{kk}, Y_{kk}\} \cup \{X_{ky}, Y_{ky}\} \cup \{X_{tt}, Y_{tt}\} \cup \{X_{sah}, Y_{sah}\} \quad (52)$$

The multilingual grapheme set is the union of five grapheme sets from different languages (53):

$$Gr_{all} = Gr_{ba} \cup Gr_{kk} \cup Gr_{ky} \cup Gr_{tt} \cup Gr_{sah} \quad (53)$$

For languages with critically low-transcribed data, it is possible to miss some letters and sounds in the dataset. The chosen approach of combining the data of languages with common scripts and language families can close the gap of absence. For example, in Table 10, formed on the basis of letters from datasets, letters ‘в [vae],

‘ё [io], and ‘ф’ [fae] are absent for the Kazakh language. However, these letters exist in the Kazakh alphabet and are often used in words from other languages. In addition, some examples of common letters for all languages included in the experiment (Bashkir, Kazakh, Kyrgyz, Saha, and Tatar) and common letters for only some of them (e.g., between Tatar and Kazakh) are presented in Figure 31. It is important to note that the soundings of these letters are similar in all these languages. This implies that the proposed multilingual approach can help improve the ASR model for critically low-resource languages.

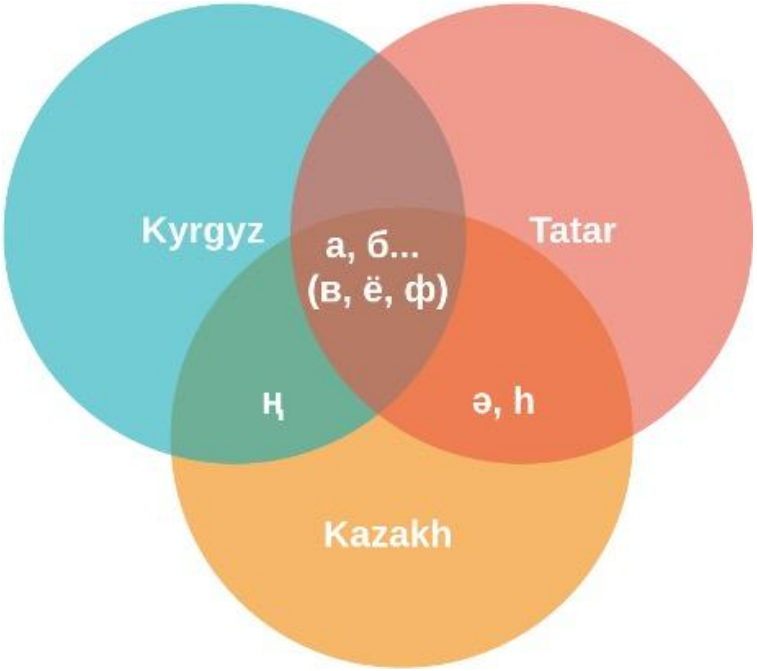

Figure 31 – Example of common letters for some of the languages included in the experiments

For training all E2E ASR models were used conformer encoder and transformer decoders. Connectionist Temporal Classification (CTC) and attention mechanisms were used in both stages of training: encoding and decoding. The weights of CTC and Attention in the hybrid model were given by the hyperparameter ctc weight. This parameter was left in its default value: $\gamma_{CTC} = ctc\_weight = 0.3$, because in , it was proved that this proportion is the best among other values. The weight of the attention mechanism is $\gamma_{att} = 0.7$ according to (54).

$$\gamma_{CTC} = 1 - \gamma_{att} \tag{54}$$

where γ is the coefficient that controls the weights of the CTC and attention mechanism [125, p. 521]. This coefficient is also used in decoding, considering the weights of model (55):

$$log\ p_{hyp} = \gamma \log p_{CTC} + (1-\gamma) lop\ p_{att} \tag{55}$$

In (55), p_{hyp} is a score used in the beam search [125, p. 522]. Probabilities are applicable to each output character.

Figure 32 shows the architecture of the system used for training the multilingual model. The procedure for training ASR was the same as for Common Voice datasets, except for the feature type: feats type parameter. This parameter was set to fbank_pitch because in [127, p. 8620], it was found that features extracted by applying filter bank and pitch methods in training CTC gave better results, decreasing the CER value. In this study, a deep Convolutional Neural Network (CNN) was chosen as an encoder function. Here, the main difference from the initial architecture is the method of combining the input features and the output sequence of characteristics.

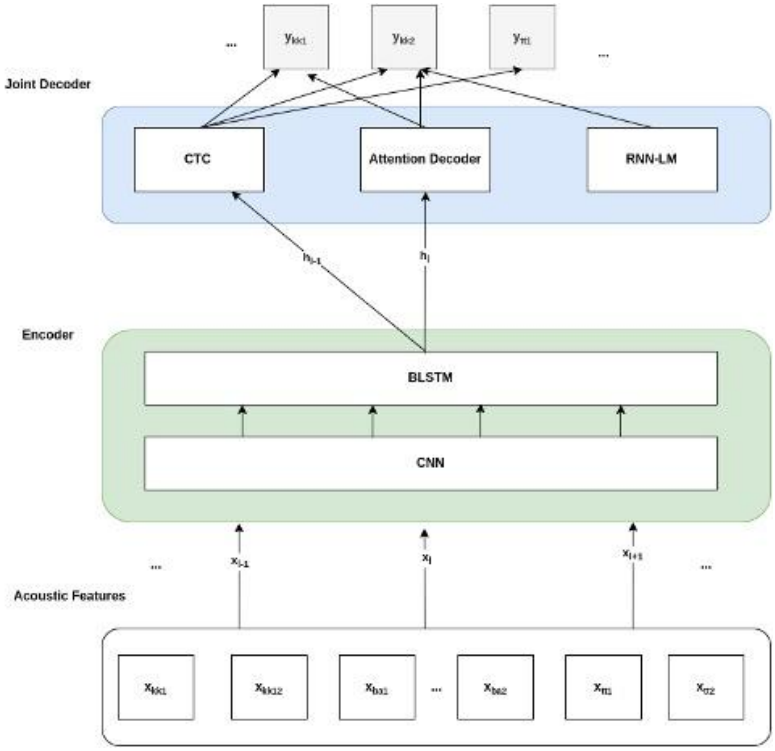

Figure 32 – Network architecture for multilingual model of Turkic languages with Cyrillic alphabet

3.2.4 Results

3.2.4.1 Monolingual ASR Models

The first monolingual ASR models were trained using the training datasets $\{X_i, Y_i\}$ for each distinct language, where $i \in \{kk, ba, ky, sah, tt\}$. Speed perturbation was applied to languages with critically low-resource data. Only the Bashkir language was trained without speed perturbation. The CER and WER results obtained for the training and test sets are listed in Table 11. In the initial training, the results for the Kyrgyz language showed that the CER and WER were lower in the test set

than in the training set. Considering this error, duplicates from the Kyrgyz corpus were removed, and only 6,5 hours of data were kept from 44 h.
Table 11 – Monolingual ASR model results

Languages	Details	Total hours	# Uttr-s	Validation set		Test set	
				CER	WER	CER	WER
Tatar	s.p.: 0.9,1.0,1.1	29	train: 20204, val: 2812	4.5	17.0	7.0	22.5
Kazakh	s.p.: 0.9,1.0,1.1	1	train: 406, val: 316	66.3	123.6	67.7	124.2
Sakha (Yakut)	s.p.: 0.9,1.0,1.1	6	train: 1633, val: 1083	29.2	79.7	32.8	85.5
Bashkir	no s.p.	265 (255)	train: 178522, val: 14577	1.8	6.4	1.7	6.1
Kyrgyz	s.p.: 0.9,1.0,1.1	44(6.5 kept)	train: 4010, val: 502	17.7	54.3	17.9	55.3

3.2.4.2 Multilingual ASR Models

The multilingual ASR model was trained on the basis of the multilingual dataset $\{X_{all}, Y_{all}\}$ with the following overall utterances: train, 227031; test, 20401. Speed perturbation was not applied to the multilingual ASR model. Test folders of the distinct languages were used for decoding. Comparing data from Table 1111 and Table 1212, especially test/WER and test/CER, it is possible to conclude that multilingual ASR gives very promising results for languages with critically low-resource data: test/WER for Kazakh language decreases from 124.2 to 64.3, test/CER decreases from 67.7 to 19.3.

Table 12 – Multilingual ASR model results

Languages	Validation set		Test set	
	CER	WER	CER	WER
ALL: Cyrillic	3.8	11.8		
Tatar	n.a.	n.a.	5.3	19.7
Kazakh	n.a.	n.a.	19.3	64.3
Sakha (Yakut)	n.a.	n.a.	18	58
Bashkir	n.a.	n.a.	1.7	6.4
Kyrgyz	n.a.	n.a.	3.9	11.4

A comparison graph formed based on Table 1111 and Table 1212 for test/WER is presented in Figure 33. Here, it is possible to determine that for the language with maximum hours, Bashkir's language results of multilingual and monolingual models are nearly the same. Kyrgyz, Saha and Tatar have also made improvements.

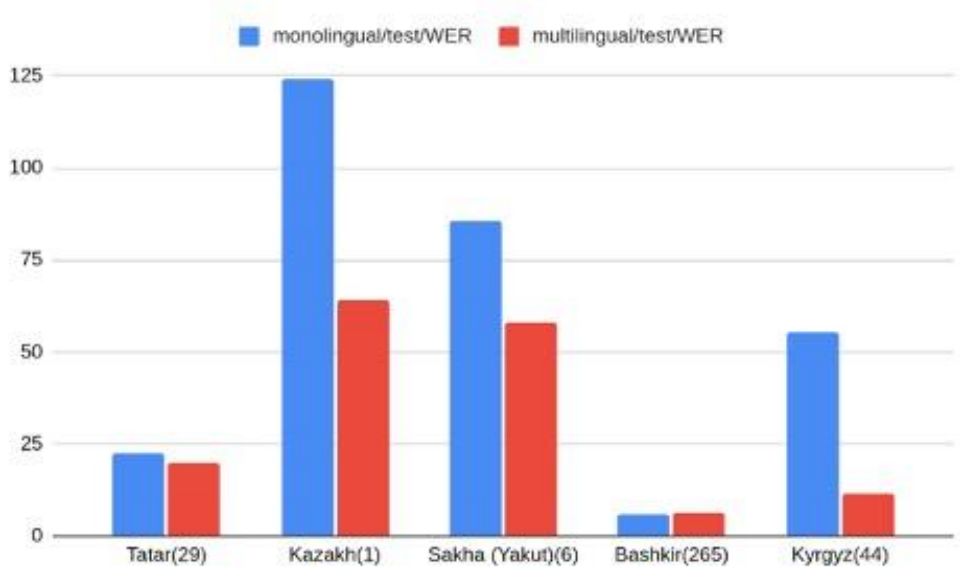

Figure 33 – Monolingual ASR model WER comparison with multilingual

The results obtained in this study show significant differences in comparison with the results of [126, p. 4221]. Transfer learning results for languages from the Turkic family over the English language model, especially for the Tatar language, showed a higher CER than in our investigation (26.42%). In our experiments, the CER was higher in the multilingual model (5.3) than in the monolingual model (4.5) for Tatar. However, the use of languages of one family and one type of scripting, moreover, according to the advantages of the chosen training system, resulted in one-fifth less CER than in the result reported in the state of the art.

The results of [125, p. 524], obtained by training languages from different language groups, provided improvements for all languages included in the experiment. But there is no dramatic improvement as in the current study: in our investigation ASR gives very promising results for languages with critically low-resource data: test set WER for Kazakh language decreases from 124.2 to 64.3, test set CER decreases from 67.7 to 19.3.

The results of this work prove that multilingual training with CTC + Attention mechanisms, including language models, can help obtain meaningful results for languages with critically low-level data if we train languages of one family and have similar alphabets.

3.2.5 Conclusion

Almost all languages in the Turkic family are low-resource languages. As these languages have words with similar roots and word formation rules, the proposed approach can help improve ASR models by providing multilingual training by combining datasets to train on the ESPNet system with CTC + Attention mechanism + LM. To maintain robustness, languages with similar letters (Cyrillic) were chosen for the study and experiment. The results showed that the ASR system chosen and the data combining approach can provide better results than the state-of-the-art system. This approach can also help solve the challenge of letter absence in critically low-

resource languages. The proposed method can be applied to languages of other families, but different letters can lead to different results.

3.3 Enhanced LM with enlarged raw text data

3.3.1 Introduction

The rich set of parameters which describes words' relationships and can help to improve the recognition performance of Automatic Speech Recognition Systems (ASR). There is a number of researches where distinct language models trained on enlarged dataset of texts were used in order to improve the performances of ASR systems [143-146]. One more advantage of this method is the possibility of decreasing the cost of ASR, due to the fact that collection of raw text is much cheaper in comparison with text-audio data collection.

The present section is dedicated to the investigation of ASR by using language model, trained on enlarged text data. There was proposed the method of applying external language model to the workflow of end-to-end ASR, by using it in decoding stage. Moreover, the ASR model of big dataset was used as a basic model for transfer learning, which also had impact on the performance of ASR.

Subsections of this part of the thesis describe related works, methodology used and provided experiments, discussions and conclusions.

3.3.2 Related works

3.3.2.1 Researches, dedicated to improve ASR for Kazakh Language

Multi-Scale Parallel Convolution (MSPC) was proposed by some authors [35, p. 7319-2] as the method of ASR improvement in the architecture CTC-attention. Here authors use this method with bidirectional long short-term memory (Bi-LSTM) and achieved the performance improvement for end-to-end model. Authors of this research used data for Turkish and Uzbek languages from Common Voice and augmented them by adding noises, this allowed them to check the performance and increase the size of data, to be trained. The result of this study showed that WER and CER were decreased by using proposed approach in joint use with language model. Here the increase of beam width to 16 also was one of factors which helped to improve results.

In authors trained twenty two different languages together in multilingual training in order to observe the impact of multilingual approach for representators from one language family. Results of this approach showed that this multilingual approach decreases error rates for each language. In was proposed the idea of using external LM, trained on the data collected from different books in Kazakh language, but authors proposed the idea of only parts of words.

3.3.2.2 Text corpus enhancement for ASR: general case

The use of improved word embeddings was proposed in [143, p. 96] and authors noticed its usability in speech translation and ASR. The importance of word embedding vectors in decreasing error rates was studied and proved in [144, p. 8515].

Improved vectors of embedded words can decrease WER being used in decoding process, because they can obtain rare words as improvement of these vectors are usually provided using large-scale data [145, p. 690]. Authors of [146, p. 366] studied approaches of translating written text form of human speech into the format which can serve as a basis for building LM which also can improve ASR performance.

3.3.3 Methodology

Current research studies the impact of distinct language model to the improvement of ASR characteristics. An enlarged language model was integrated into decoding stage. This language model was trained on the data which consists of text data from different resources and the text of paired entire dataset for training the ASR. General structure of proposed decoder is depicted in Figure 34.

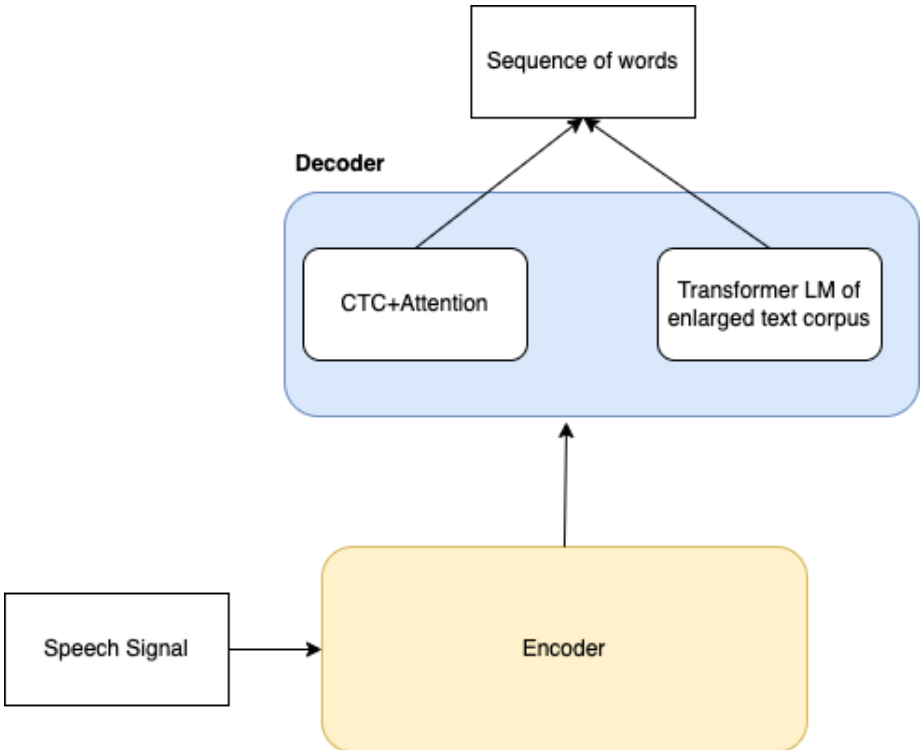

Figure 34 – Improved decoder with the LM of enlarged raw text

3.3.3.1 LM enhancing

As ESPnet supports the state-of-the-art architectures for ASR, like transformer and conformer, it was choosen as a tool for providing the experiments for this task. Convolutional Neural Networks (CNNs) were used for processing input signals and the method of joint decoding (CTC-attention) was applied for output. A Language Model (LM) was trained [134, p. 3; 135, p. 2] and was integrated in decoding (56):

$$\log p\ (y_n|y_{1:n-1}, h_{1:T}) = \log p^{hyp}\ (y_n|y_{1:n-1}, h_{1:T}) + \beta \log p^{lm}\ (y_n|y_{1:n-1}, h_{1:T}) \quad (56)$$

here p is for the next word's probability, $p^{hyp}$is the word's probability given as suggestion found by ASR model, p^{lm} is the probability of next word calculated by

extended language model, $\beta$is the coefficient of language model ($0 \leq \beta \leq 1$, float value).

Use of additional raw text information can improve hot vectors which are called E-vectors, because it can obtain different cases of usages of words and their combinations.

3.3.3.2 Featurized representation

The probability of word's relation to different classes of features is called featurized representation. These features can be retrieved extracted by calculating relations of words into utterances, sentences or phrases. This type of information is very useful in the tasks of generation sequence, for example in NLP and ASR, because both of these tasks need the prediction of next token.

Let's consider, we have this sentence in our data collection:

I like to drink milk.

If the newly trained model is expected to predict the word in a sentence, in the expression which is not in the trained dataset:

I like to drink _____?

Which token will be chosen from the table of word representations? It is expected, that it will pick up word "juice" due to the fact that hot representation of this word is nearly the same as "milk"s (Table 13). It means, if the model will not have any information about the "juice", the probability of predicting it for the sentence in near to zero. This is the reason for increasing the "knowledge" of models about sentences, expressions and words.

Table 13 – Featurized representations of some words in Kazakh Language

Features of words	Ана (mother)	Әке (father)	Ұл (boy)	Қыз (girl)	Сүт (milk)	Шырын (juice)
Parenthood	-1	1	-0.27	0.26	0.00	0.02
School	-0.25	0.32	-0.99	0.99	-0.03	0.04
Drink	0.00	0.00	0.02	0.03	0.92	0.93
Wet	0.02	0.03	0.04	0.01	0.94	-0.98
Size	0.05	0.04	0.08	0.09	0.55	0.62
Fruit	0.07	0.08	0.01	-0.02	0.09	0.53
Flower	0.15	0.20	0.09	-0.21	0.08	0.34

3.3.3.3 LM architectures

Enhanced text corpus was trained with the use of two different LM architectures: sequential RNN model and Transformer language models. Training of "Big text" by transformer resulted in the lowest perplexity value: 2.99. "Big text" also gave improvement, when in was trained with sequential RNN in comparison with training entire text of the used dataset: 3.99 agaist 9.09. (Table 14). This table also depicts, that the biggest number of trainable parameters was extracted with transformer LM (50.54 M).

Table 14 – Impact of different language models on perplexity

Type of language model	Perplexity	Trainable parameters	Number of sentences
RNN language model (transcript of basic dataset)	9.09	6.83 M	5774
RNN language model (enhanced text data)	3.99	6.84 M	139810
Transformer(enhanced text data)	2.99	50.54 M	139810

3.3.3.4 RNN language model

Architectures of language models, tested during this experiment are based on statistical probability. Application of probability to the sequences of letters and words was proposed in the 80's of twentieth century . In this approach the probability of whole sentence or expression is calculated as the product of each word's probability which depends on the previous part of expression.

$$P(s) = P(w_i w_{i+1} \dots w_n) = P(w_i|w_{i+1}) \dots P(w_n|w_i w_{i+1} \dots w_{n-1}) \quad (57)$$

In (57) s stands for sentence, w_i stands for the i-th word. Sentences in Kazakh language can be very long, that is why RNN LM with LSTM cells was used . This language model was trained in the architecture with two layers and 650 LSTM units in each layer. Linear decoder was used with the number of features 650 for input and 48 for output. The batch size was set to 48. Model was trained in 20 epochs.

3.3.3.5 Transformer language model

Transformer is one of the generally used and efficient architectures [136, p. 5874]. The key mechanism in this architecture is attention weights, which firstly successfully solved the task of text translation . In this research this type of language model has embedded type of sequential layers. Dropout for this model is 0.1 and activation function is rectified linear unit. Each layer of encoder has eight heads and 512 units in each head. After position-wise feed-forward, there was placed two normalization layers. Dropout is 0.1. Learning rate was chosen as 0.001 and model was trained in 25.

3.3.4 Description of ASR architecture and the results of training with LM on "Big text"

ASR uses conformer as an encoder and transformer as a decoder. The layer used as input layer is convolution layer which is two dimensional. Activation function for input layers is ReLU. Multi sequential encoder with 12 layers is used. Relational positioning attention heads are in each layer. Each layer has four attention heads. This layer is is followed by two layers of position-wise feed-forward and convolution layers. Activation function of these two different types of layers is swish. Normalization layers are used after each layer of encoder.

Decoder, which uses transformer architecture has positional encoding in the embedded layers of input. After these layers there are placed six attention layers with

multi heads. Joint decoding uses the loss function CTC with the weight equal to 0.3. There was chosen sufficiently high value of learning rate which is equal to two, due to the fact that 15 hours for ASR training is very low.

Language models, tested during this experiment was used jointly with the decoder of conformer architecture. Results for 15 hours of audio-text Kazakh speech are given in Table 15. Weight of language model for experiments, listed in Table 15 is 0.3.

Table 15 – Comparison of word error rate and character error rate values for ASR system trained in different cases for 15 hours of Kazakh language

LM type	WER/val (%)	WER/test (%)	CER/val (%)	CER/test (%)
Sequential RNN language model with basic dataset text	53.1	54.0	19.1	20.1
Sequential RNN language model with enlarged text data	48.7	49.1	18.3	18.9
Transformer language model with enlarged text data	46.2	46.8	17.7	18.2
TransformerLM with enlarged text data and cross lingual transfer learning from English language (encoder)	45.5	46.3	17.3	18.3

After clarifying the fact that language model trained on enlarged text with transformer model can decrease error rates for ASR, appeared the suggestion that increase in the value of language model weight can make further improvements in the performance of ASR. This suggestion is based on the fact that this type of language model has better representation for words. Empirical experiments on choosing proved our hypothesis. The lowest error rate achived with the language model weight equal to 0.45. Values of error for experiments with different values of lm_weight are given in Table 16.

Table 16 – Effect of different values of lm_weight of ASR performance

Lm_weight	WER/train (%)	WER/test (%)	CER/train (%)	CER/test (%)
0.35	43.5	45.6	17.0	17.9
0.4	44	45.1	17.3	18.2
0.45	43.2	43.2	17.2	17.4
0.5	42.7	43.5	16.9	17.8

3.3.5 Discussions and Conclusion

Use of large size of raw text, so called “Big text”, for training external language model have effect on all types of ASR performance metrics, especially on word error rate and character error rate. Significant impact of this approach was on word error rate, but different language models gave different improvements: training “Big text” with RNN language model decreased word error rate by 5%, transformer language model by 7.2%. It means that the ability of transformer model to increase

the number of trainable parameters makes it more effective and this model can decrease the perplexity.

The transfer learning for Kazakh language over the model for English language can improve the results taken by using in the decoding language model by training "Big text" by transformer architecture. This study allows to conclude that the use of Enhanced language model in decoding is suitable for Kazakh language. Taken results also help to conclude, that Transformer model in comparison with RNN model is more effective, and it decreases word error rate by 10% in the case of choosing optimal value for language model weight against ASR model, which used in decoding language model, trained only with the text of entire dataset.

3.4 Transfer learning experiments

3.4.1 Introduction

The performance of ASR systems built by end-to-end methods depends on the size of data to be used in training process. The study, were the author of this thesis participated [8, p. 90] proved this idea. Also here stated out that the most agglutinative languages from Turkic family are low-resource. They are: Azerbaijani, Kazakh, Kyrgyz, Tatar, Turkish and etc. The larger the data, better the accuracy in final model. In and were obtained better results for large data by training with CTC and attention-based end-to-end models. However, introducing the complex of computational layers can result to huge number of parameters which cannot be reached by training low-resource languages. In order to avoid parameter leakage problem for each language the transfer learning method was applied for languages which belong to one language family: Kazakh and Azerbaijani languages. Here we also use the advantage of end-to-end models for agglutinative languages which was proved in , which states the fact that end-to-end models for these languages does not need the integration of language models.

Transfer learning is the approach which adapts the models which was trained on one data to another collection of data for training. This research showed next three improvements:

1. The representations taken in the result of training for one language (Kazakh) reduces training time for other language against the training from scratch.
2. Transfer learning allow to use less data for another language for evaluation
3. GPU memory usage decreases because transfer learning does not need the support for gradients of all layers.

3.4.2 Related works

As high-quality transcribed data is not available for languages from Turkic family, it is necessary to study the ways of improving ASR result parameters for these languages. There was performed review for transfer learning and combining of end-to-end methods for speech recognition which served as the bases for current research.

DNN based end-to-end ASR systems can use the advantages of transfer learning. In acoustic model was built on the data taken from the phone call records of

call center. The model trained with 20 hours of target data over the acoustic model trained with large corpus from call centers improved the accuracy by 7.8% in comparison with the model trained only with target data for Turkish language.

Other research proposes another type of transfer learning, called language-adversarial. This learning type could enable SHL model's common layers which allowed to learn the features, invariant for languages. The dataset IARPA BABEL was used for experiments. The result of this study showed that this approach could reduce word error rate by 10.1%. In spite the model good performance of the model the systems got heavier.

The next study [40, p. 5884] used the pretrained model on the 100 hours of Russian speech taken from VoxForge as a basis for transfer learning. The knowledge, extracted from the mentioned dataset was used as a basis for 20 hours speech of Kazakh language. This model used LSTM and BiLSTM NNs. In the case of training by Bi-LSTM Lemma Error Rate (LER) was reduced to 32%.

Two major types of end-to-end architectures were compared in the next study for Mandarin speech. Here was chosen the best unit type for recognition and the best architecture between CTC and attention-based mechanism. As a result, characters of Chinese language were found as the best unit for recognition and encoder-decoder mechanism on the basis of attention mechanism (35.2%) showed better performance over CTC model (35.7%). In the experiments the stage of feature extraction from input signals was skipped. Attention mechanism also was used for 60 hours of Russian language in . Here authors tests joint use of attention encoder-decoder with CTC model on Russian continuous speech and took comparatively better results in comparison with other approaches.

Effectiveness of hybrid approach also was shown in another study . Jointly use of LSTM and transformer gave faster output and outperformed the ASR trained by transformer by 11.9%. The made review shows that combination of end-to-end methods improves performance of ASR systems, while transfer learning can improve speech recognition for low-resource language. Both of mentioned approaches were used in the current study.

3.4.3 Methodology

3.4.3.1 Training by transfer

To avoid the problem of resource leakage the transfer learning method was applied in end-to-end architecture. The acoustic similarities of Kazakh and Azerbaijani languages were used from common layers. Feature extraction was performed from multilingual dataset where Kazakh and Azerbaijani languages were combined, for embedding acoustic knowledge which are general for two languages. Firstly, were trained independent RNN layers using hidden layers which are common for languages included in the experiment. Figure 35 demonstrates the input for hidden shared layers for transfer learning.

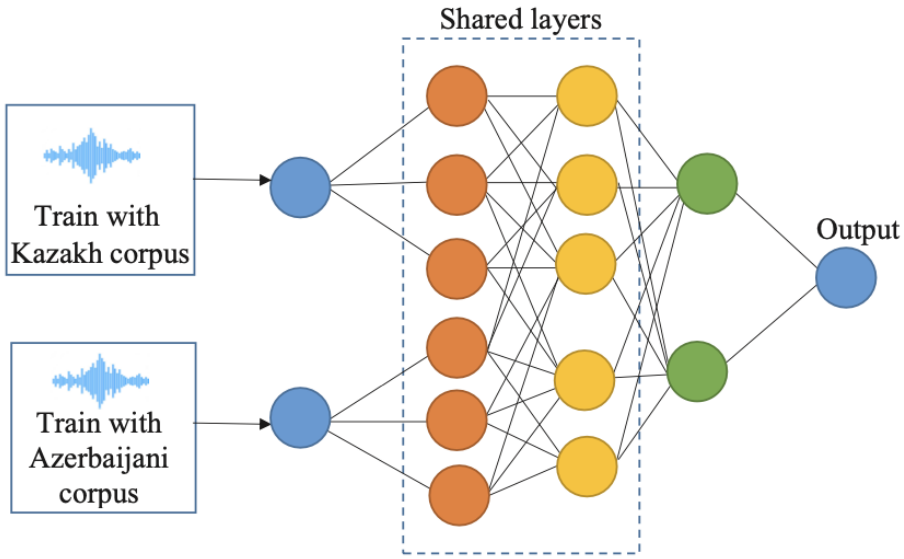

Figure 35 – RNN with common hidden layers

Two languages were trained in parallel. The output of each hidden layer for RNN is presented as dot product:

$$x_n = y_n \times M_y, n \in [1, N] \tag{58}$$

where layer l-th output for n-th frame is y_n, M_y presents the vector of binary elements. Vector's each element stores information about whether corresponding item was changed or remained. The activation function is maxout function. Additionally, dropout is applied in training process in order to extract the best features in common layers and minimize the risk of overfitting. In order to find the maximum for each layer was performed the max pooling. The maximum was found next way:

$$y_n(i) = \max\left(s_n(k * i - 2), s_n(k * i - 1), s_n(k * i)\right), n \in [1, I] \tag{59}$$

where I is the number of each hidden layer's single output, s_n is the vector which contains these outputs, is the size of pooling is k. RNN processes phonemes extracted by Gaussian Mixture Models (GMM) from low-level acoustic data. All parameters were moved below the last hidden layer and an additional layer with SoftMax was added in order to get the features of low-dimension from recurrent neural network. This adaptation which does not destroy entire structure of NN provides maximum level of possible nonlinearity for the further calculations.

3.4.3.2 CTC and attention mechanism in joint use

This section contains discussion about end-to-end approach which provides transfer learning on the previously obtained high-level features. Previous studies of some authors [38, p. 265] tested the architecture which has one encoder and joint

CTC-attention decoder. The already extracted high-level features were trained by shallow Bi-LSTM encoder and joint decoder. The described architecture is given in Figure 36 [8, p. 88].

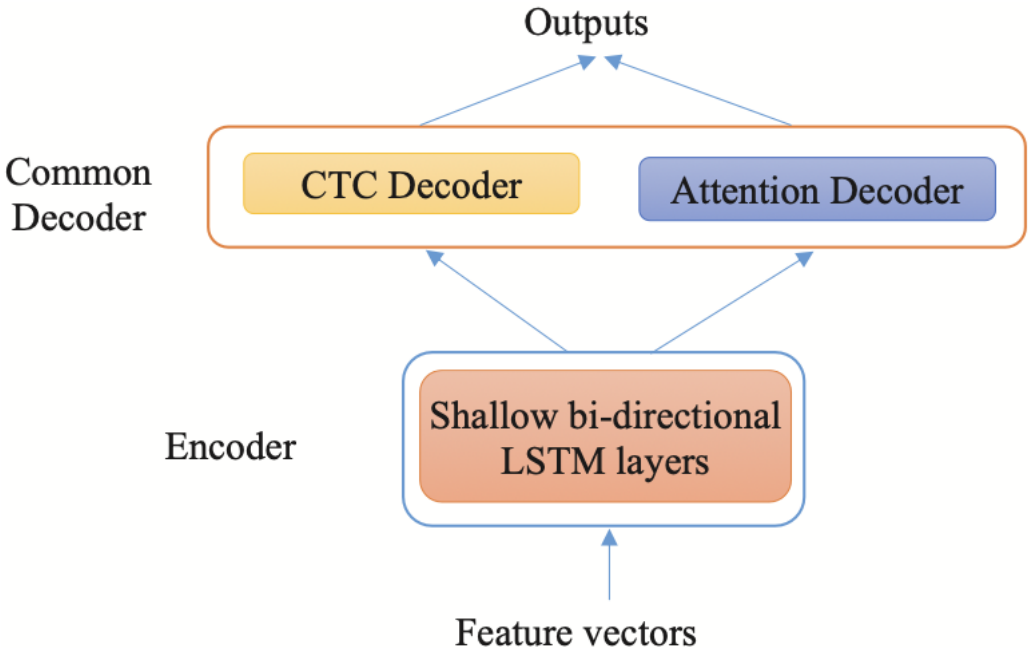

Figure 36 – The architecture with shallow Bi-LSTM encoder and joint CTC-attention decoder

The distribution of probability $P(S|X)$ over the input, which is given as audio signals S in independent conditions:

$$P_{CTC}(S|X) = \sum_{d\in N(S')} P(d|X) \approx \sum_{d\in N(S')} \prod_{t=1}^{T} P(d_t|X) \quad (60)$$

where $P(d|X)$ is the initial data at time t for the symbol which is given as d_t. And this is calculated for all X which mean the input.

The encoder uses Bi-LSTM and location aware attention. Weights of attention $g_{k,t}$ contains outputs of k-th encoder together with the t-th decoder. Weights of previous layer g_{k-1} and decoder's hidden outputs l_{k-1}, encoder's output h_t are used to find $g_{k,t}$:

$$f_n = F * g_{k-1} \quad (61)$$

$$e_{k,l} = \omega^T \tanh\left(V^S S_n + V^H H_n + V^F f_{k,1} + b\right. \quad (62)$$

$$r_k = \sum_{t=1}^{T} g_{k,t} h_t \quad (63)$$

$$P(s_k, |s_1, \dots s_{k-1}, X) = Decoder(r_k, l_{k-1}, h_t) \quad (64)$$

Here F means convolutional filter; V^S, V^H, V^F are weight parameters of multilayer perceptron which can be adjusted, r_k is a context vector. Attention mechanism's posterior probability is found as follows:

$$P_{attention}(S|X) = \prod_k P(s_k, |s_1, \dots s_{k-1}) \quad (65)$$

Loss functions for attention mechanism and CTC, for total case are defined next way:

$$J_{CTC} = -\ln P_{CTC}(S|X) \quad (66)$$

$$J_{attention} = -\ln P_{attention}(S|X) \quad (67)$$

$$J_{total} = \gamma J_{CTC} + (1-\gamma) J_{attention}, 0 \le \gamma \le 1 \quad (68)$$

here γ serves as a weight for CTC.

3.4.4 Experiments

This section discusses the dataset characteristics, transfer learning experiment, comparison of transfer approach with baseline methods. Audio files of wav format were used for experiments with frequency 44.1kHz and bit depth equal to 16 bits. For Kazakh language 400 hours (200 hours of spontaneous phone conversation, 200 hours of regular speech) and for Azerbaijani language 70 hours of speech were used. Dev (training) dataset took 80% of the dataset, while 20% was given to test dataset.

The end-to-end models which were built for this experiments had used during training 32 phonemes for Azerbaijani language and 28 for Kazakh language, which in sum equal to 60. Total size of dataset to train contained 470 hours of speech signals. CTC was trained firstly, then attention-based model. CTC model had six-layer Bi-LSTM, each layer had 256 cells inside. Attention in the encoder is a three-layer directional Bi-LSTM, the number of cells inside is the same as in CTC layers. The decoder is LSTM which has 256 cells and only one layer. Dropout values were chosen as next: encoder - 0.2, attention - 0.5 and decoder 0.1. The algorithm Adam was used as an optimizer. CTC_weight was presetted as 0.3. The width for beam search was set to 15. Acoustically similar words of Kazakh and Azerbaijani languages were not distinguished until epoch 45 during the training process. The corpus used in training had 71.649 similar words. The model after epoch 45 was chosen as a final model as it has the best accuracy. The accuracy and loss function values of the training over epochs are depicted in Figure 37.

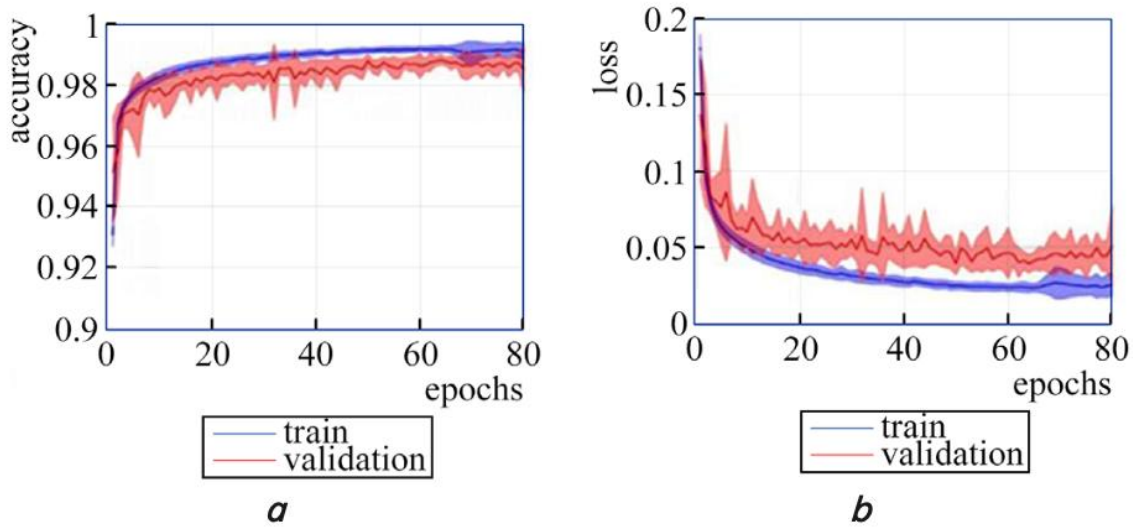

Figure 37 – The accuracy and the loss values of transfer learning process

The phoneme error rate was used as system's purpose was to recognize phonemes and this error rate was calculated using the distance of Levenshtein [158]. The result of the experiment was compared with the results of other studies and it was determined that the result of the current study outperforms others found in the state of the art, as shown in Figure 38.

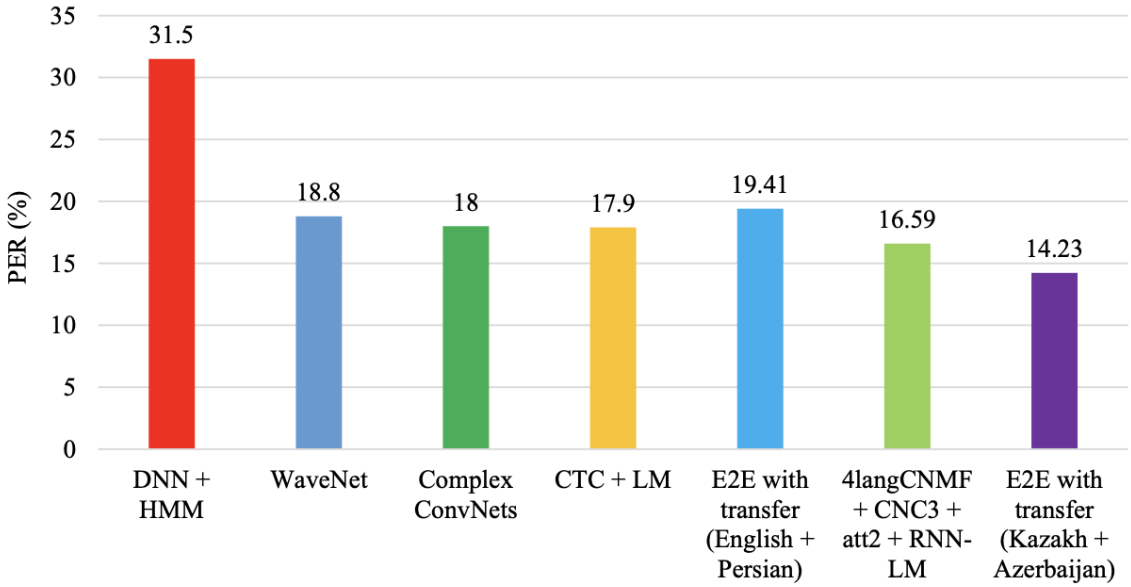

Figure 38 – Transfer learning result in comparison with other studies

3.4.5 Discussion

The last stage of the experiment contains the comparison of obtained model in the result of transfer learning with other models, built without application of transfer approach. The model considered model was compared with basic models like DNN-HMM which were described in [153, p. 397] and . Another end-to-end method with complex encoders is dedicated to construct a model for presenting the raw data from input as a sequence of audio or characters [160, 161]. Some of the studies used jointly

CTC and attention, which consists of shallow RNNs. In the result of experiments it was proved that the approach proposed for transfer learning outperformed all mentioned end-to-end methods: for example, the result obtained with DNN-HMM in was 31.5%, the result taken with WaveNet in [159, p. 472] was 18.8%, the result with CTC-LM in [153, p. 397] was 17.9%, end-to-end with the use of transfer for English and Persian languages was 19.41%. The comparison of the results of the current study is depicted in

Figure 3838. According to this figure it can be concluded that end-to-end joint transfer model's error rate is the least among other similar researches, reaching the value equal to 14.23%. But the taken result is still cannot reach the accuracy of human level and is significantly slow for real-time recognition.

3.4.6 Conclusion

The performance characteristics of ASR system critically depends on the quality and diversity of the datasets used for training. Therefore, speech corpuses were carefully collected for both of Kazakh and Azerbaijani languages were collected. Collected speech contains audio data from open sources and phone conversation speech, which can be determined as spontaneous. In the result, 400 hours of Kazakh speech and 70 hours of Azerbaijani speech were combined.

An end-to-end transfer model was proposed for these languages, in which the first stage performed feature extraction using NMF algorithm, and the second stage trains joint CTC-attention model using these features. Transfer learning had two levels, like bilingual and multitasking. The results of experiments proved that the model proposed outperforms modern speech recognition approaches. The final phoneme error rate was 14.23% which is relatively small in comparison with results of other modern systems.

4 DISCUSSION

All types of machine learning tasks need adequate datasets to be trained. Automatic speech recognition also needs hundreds and thousands of hours of transcribed data. It is one of the fields which mainly uses ML after the introduction of end-to-end approaches. But the process of collecting and merging the data, collected in the real environment, audio-text pair forming and making it usable to train is one of the most complex tasks. The 396-hour marked database collected and edited by the author in more than 2 years is a valuable contribution of the author to the improvement of speech recognition area for Kazakh language. This material can be used in experiments for the creation of speech recognition systems in the Kazakh language, and can be used to expand other databases and datasets. After merging 195 hours of speech collected in real time conditions database with 283 hours of previously collected data, the total volume of data reached 396 hours, after cleaning and removing duplicates. The experiment, performed with this data by training ASR on conformer architecture, proves the suitability of this data to use in machine learning tasks. Besides, the script written for converting data of different encodings into UTF-8 encoding (UTF-8, UTF-16, rk1048) can be applied in different tasks from various fields of science.

The results of the study with multilingual training showed that current experiment outperforms previous studies with these languages. For example, in [126, p. 4221] the results of transfer learning for Tatar over English character error rate was 26.42%, while in our experiments it showed five times less errors: 5.30%. The results achieved in [125, p. 524] by training languages from different language groups, showed improvements for all languages chosen for experimentation. But the results did not have dramatic improvement as in the current study: the result of a multilingual experiment with languages from one language family and common scripts gave promising results for languages which have critically low-resource data: test set WER for Kazakh language decreases from 124.20 to 64.30, test set CER decreases from 67.70 to 19.30. The chosen architecture: CTC+attention+LM can also be applied distinctly to languages of Turkic family.

The language model trained on “Big Text” gives improved and advanced word representation for speech units (words) of a language. This external language model is used in the decoding stage. Thereby, inclusion of distinct language models trained on the big raw text, which is bigger and different from the entire text of the dataset, can decrease the error rate of the ASR system. Experiments proved that it has a significant impact on word error rate. Training of so called “Big Text” with Sequential RNNLM reduced WER by 5%, with transformer language model reduced by 7.20%. By the results of experiments, it is evident that Transformer LM is effective in comparison with basic RNNLM model, as it decreased perplexity value and increased the number of trainable parameters. Moreover, appropriate choice of language model weight value in decoding can decrease word error rate by 10% for 15 hours of transcribed data.

The comparison of the model obtained with transfer learning with other models, built without application of the transfer approach, proved the efficiency of our approach for transfer learning. The model considered was compared with basic models like DNN-HMM, which were described in [153, p. 394] and [159, p. 472]. Another end-to-end method with complex encoders is dedicated to construct a model for presenting the raw data from input as a sequence of audio or characters [160; 161, p. 5510]. Some of studies used jointly CTC and attention which consists of shallow RNNs. In the result of experiments it was proved that the proposed approach for transfer learning outperformed all mentioned end-to-end methods: for example, the result obtained with DNN-HMM in [162, p. 4562] was 31.5%, the result taken with WaveNet in [159, p. 472] was 18.8%, the result with CTC-LM in [153, p. 397] was 17.9%, end-to-end with the use of transfer for English and Persian languages [163, p. 3] was 19.41%. The comparison of the results of the current study is depicted in

Figure 3838. According to this figure it can be concluded that end-to-end joint transfer model's error rate is the least among other similar researches, reaching the value equal to 14.23%. But the taken result still cannot reach the accuracy of human level and is significantly slow for real-time recognition.

According to the results discussed above it could be stated out that research questions listed out in the INTRODUCTION were answered:

1. A relevant architecture was selected for Turkic languages.
2. It was proved that transfer learning is effective for languages from one language family.
3. It was proved that multilingual training for Turkic languages with common scripts decreases ASR error rates for each language included in the experiment.
4. Transcribed audio corpus for Kazakh language Enlarged.
5. A language model trained on Transformer architecture with enhanced text corpus decreases error rates in end-to-end ASR.
6. A new automatic speech recognition model was trained for Kazakh langage
7. The programming product was constructed, which can translate speech to text.

Results of researches were presented and discussed in different conferences and seminars and some of them were published. Moreover, the author was awarded with certificates as a seminar speaker, for the best presentation:

1. O. Mamyrbayev, D. Oralbekova, A. Kydyrbekova, T. Turdalykyzy and A. Bekarystankyzy, "End-to-End Model Based on RNN-T for Kazakh Speech Recognition," 3rd International Conference on Computer Communication and the Internet (ICCCI) (Tokyo, 2021 – 25-27 June).
2. Certificate to the seminar speaker on the topic "Improved Speech Recognition for Agglutinative languages", Coimbra Institute of Engineering (ISEC), (Coimbra, 2023 – 21 April).
3. Certificate for the best presentation speech, "Improve Automatic Speech Recognition for Kazakh Language using Extended Language Model", "ACeSYRI Young Researchers School" (Almaty, 2023 – 5-10 June).

4. A. Bekarystankyzy, O. Mamyrbayev, “Improve Automatic Speech Recognition for Kazakh Language Using Extended Language Model”, 21 scientific conference, (Riga, 2023 – 20-21 April).

5. Automatic Speech Recognition Improvement for Kazakh Language with Enhanced Language Model // Recent Challenges in Intelligent Information and Database systems. ACIIDS 2023. Part of The Communications in Computer and Information Science book series. – 2023. - Vol. 1, - P.538-545 (Springer, Cham).

Main results of the dissertation research were published in four papers, one of which is published in a periodical journal with non-zero impact-factor and indexed by databases Scopus and Web of Science, three papers published in the journals recommended by the Control Committee in the sphere of education and science of MHES RK. One of the studies was published as book chapter in LNAI Book series. Autor’s certificates were taken for scripts and programming products, developed during research:

1. M. Orken, A. Keylan, O. Dina, B. Akbayan and Z. Bagashar. Identifying the influence of transfer learning method in developing an end-to-end automatic speech recognition system with a low data level // Eastern-European Journal of Enterprise Technologies. – 2022. - Vol. 1, №115. - P. 84-92 // https://doi.org/10.15587/1729-4061.2022.252801 (Scopus, percentile 34).

2. Bekarystankyzy A. and Mamyrbayev O. Integrated Automatic Speech Recognition System for Agglutinative Languages // News of the National academy of sciences of the republic of Kazakhstan. - 2023. - Vol. 1, №345. - P. 37-49 // https://doi.org/10.32014/2022.2518-1726.167.

3. Bekarystankyzy A., Mamyrbayev O., Oralbekova D., Zhumazhanov B. Transfer learning for an integrated low-data automatic speech recognition system // Scientific and technical journal "Bulletin of the Almaty University of Power Engineering and Telecommunications". - 2023. - Vol. 1, №60. - P. 185-198 // https://doi.org/10.51775/2790-0886_2023_60_1_185.

4. Bekarystankyzy A. and Mamyrbayev O. End-to-end speech recognition systems for agglutinative languages // Scientific Journal of Astana IT University. - 2023. - Vol. 13. - P. 86-92 // DOI: 10.37943/13IMII7575.

5. Author’s certificate "Software Product UniCodeKaz" №38545 from 21.08.2023 (Bekarystankyzy A.)

6. Author’s certificate "System of transcribing audio files to text" №38833 from 31.08.2023 (Bekarystankyzy A., Mamyrbayev O., Duisenkhan B.).

CONCLUSION

The thesis achieved contributions to the improvement of automatic speech recognition for agglutinative languages from Turkic family, especially for Kazakh language. The first step of investigations was dedicated to data collection in order to enlarge existing datasets of Kazakh language, because training of robust ASR models needs sufficient amount of data. In the end it was possible to produce a satisfactory dataset with 396 hours of speech and without duplicates. Moreover, a script was implemented which can convert the information in Kazakh language from any encoding to UTF-8.

A second step considered multilingual training of languages from one language group and which have similar scripts. This approach gave for the language with critically low resource an improved error rate. Training languages like Bashkir, Kazakh, Kyrgyz, Tatar and Saha from Commonvoice reduced WER for Kazakh language to half, decreasing its value for one hour of dataset from 124.2 to 64.3.

A third part of the studies contains a method which can improve ASR performance without the need of collecting and transcribing audio data. Use of only raw external text, so called “Big Text” decreased word error rate for 15 hours of Kazakh speech by 10%.

The last investigation contains the study of transfer learning with Turkic languages, like Kazakh and Azerbaijani. Here firstly were extracted audio features by NMF algorithm. Further these features were trained by CTC-attention joint architecture for 60 phonemes, and the final phoneme error rate was the least in comparison with other similar studies. To sum up all, it can be concluded that the thesis studies and make contributions to improve ASR for Turkic languages, especially for Kazakh language. If first stage considers data collection, all other experiments and studies are dedicated to improve the ASR performance for low-resource languages with data pooling methods, like multilingual training and transfer learning and improving word representations with external text data. The proposed pooling methods easily can be applied in the situations where only low-resource data is available. But for some of agglutinative languages from Turkic family even these approaches can be inadequate, because these methods also requires at least more than 15 hours of transcribed data.

The main contributions of the research were published in three international conferences, four journal papers and in one book as a chapter. Also, author’s certificates were taken for scripts and programming products developed under research.

Future work can be dedicated for the building of specific architectures for Turkic languages and finding metaparameters for tuning existing architectures for agglutinative language, more precisely for Kazakh language, because any type of training for Kazakh language has incorrect accuracy graph due to binding words so called “shylau” and the similar words which have similar body and different meanings.

REFERENCES

1 Ahmad A., Rizwana I., Jiechao G. et al. E2E-DASR: End-to-end deep learning-based dysarthric automatic speech recognition // Expert Systems with Applications. – 2023. – Vol. 222. – P. 119797.

2 Jasper O., Laura T., Bernd T.M. Self-conducted speech audiometry using automatic speech recognition: Simulation results for listeners with hearing loss // Computer Speech and Language. – 2023. – Vol. 78. – P. 101447.

3 Ablimit M., Neubig G., Mimura M. et al. Uyghur morpheme-based language models and ASR // In procced. IEEE 10th internat. conf. on signal processing proceedings. – Beijing, 2010. – P. 581-584.

4 Frank Z., Yongqiang W., Xiaohui Z. et al. Faster, Simpler and More Accurate Hybrid ASR Systems Using Wordpieces // Procced. conf. «Interspeech 2020». – Shanghai, 2020. – P. 976-980.

5 Hilal Ö., Emin E.K. Transmorph: a transformer based morphological disambiguator for Turkish // Turkish Journal of Electrical Engineering and Computer Sciences. – 2022. – Vol. 30, Issue 5. – P. 1897-1913.

6 Shweta B., Shambhu S., Shyam S.A. Study of Speech Recognition System Based on Transformer and Connectionist Temporal Classification Models for Low Resource Language // Procced. 24th internat. conf. «Speech and Computer» (Specom 2022). – Gurugram, 2022. – P. 56-63.

7 Jain A. Finnish language modeling and ASR with Deep Transformer Models // https://www.researchgate.net/publication/340223686. 12.04.2023.

8 Orken M., Keylan A., Dina O. et al. Identifying the influence of transfer learning method in developing an end-to- end automatic speech recognition system with a low data level // Eastern-European Journal of Enterprise Technologies. – 2022. – Vol. 1, Issue 9(115). – P. 84-92.

9 SONIX A short history of speech recognition // https://sonix.ai/history-of-speech-recognition. 12.04.2023.

10 Fredrik N., Zelal Y. The implementation of Voice Command in smart Homes: thes. … bachelor of science – Stockholm, 2018. – 36 p.

11 IBM Shoebox // https://www.ibm.com/ibm/history. 12.04.2023.

12 Lowerre B., Reddy R. The Harpy Speech Recognition System: performance with large vocabularies // The Jounal of the Acoustical Society of America. – 1976. – Vol. 60, Issue 1. – P. S10-S11.

13 Kouemou G. History and Theoretical Basics of Hidden Markov Models // In book: Hidden Markov Models, Theory and Applications. – Ulm, 2011. – P. 1-26.

14 Mark G., Steve Y. The Application of Hidden Markov Models in Speech Recognition. – Hanover, 2008. – 113 p.

15 Ambuj M., Navonil M., Rishabh B. et al. A Review of Deep Learning Techniques for Speech Processing // https://arxiv.org/abs/2305.00359. 12.04.2023.

16 Hojjat S., Julianne B., Sharan S. et al. Recent Advances in Recurrent Neural Networks // https://arxiv.org/abs/1801.01078. 13.04.2023.

17 Hinton G., Osindero S., Teh Y.-W. A Fast Learning Algorithm for Deep Belief Nets // Neural computation. – 2006. – Vol. 18. – P. 1527-1554.

18 Bengio Y., Boulanger-Lewandowski N., Pascanu R. Advances in optimizing recurrent networks // Procced. IEEE internat. conf. on Acoustics, Speech and Signal Processing. – Vancouver, 2013. – P. 8624-8628.

19 Song W., Guanyu L. Overview of end-to-end speech recognition // Journal of Physics: Conference Series. – 2019. – Vol. 1187, Issue 5. – P. 052068-1-052068-5.

20 Ruchao F., Wei C., Peng C. et al. A CTC Alignment-based Non-autoregressive Transformer for End-to-end Automatic Speech Recognition // IEEE Transactions on Audio, Speech, and Language Processing. – 2023. – Vol. 31. – P. 1436-1448.

21 Dzmitry B., Kyunghyun C. et al. Neural Machine Translation by Jointly Learning to Align and Translate // https://arxiv.org/abs/1409.0473. 13.04.2023.

22 Pengbin F., Daxing L., Huirong Y. LAS-Transformer: An Enhanced Transformer Based on the Local Attention Mechanism for Speech Recognition // Information. – 2022. – Vol. 13, Issue 5. – P. 250-1-250-13.

23 Sehoon K., Amir G. Squeezeformer: An Efficient Transformer for Automatic Speech Recognition // https://arxiv.org/abs/2206.00888. 13.04.2023.

24 Anmol G., James Q., Chung-Cheng C. et al. Conformer: Convolution-augmented Transformer for Speech Recognition // Procced. conf. «Interspeech 2020». – Shanghai, 2020. – P. 5036-5040.

25 Helmer S., Roeland V.H., Catia C. et al. Automatic Speech Recognition and Pronunciation Error Detection of Dutch Non-native Speech: cumulating speech resources in a pluricentric language // Speech Communication. – 2022. – Vol. 144. – P. 1-9.

26 Bence M.H., Siyuan F., Rob V.S. et al. Low-resource automatic speech recognition and error analyses of oral cancer speech // Speech Communication. – 2022. – Vol. 141. – P. 14-27.

27 Alicia B.W., Cady G., Isabel B. Uneven success: automatic speech recognition and ethnicity-related dialects // Speech Communication. – 2022. – Vol. 140. – P. 50-70.

28 Karol N., Michal P., Kyoko M. et al. Adapting Multilingual Speech Representation Model for a New, Underresourced Language through Multilingual Fine-tuning and Continued Pretraining // Information Processing & Management. – 2023. – Vol. 60, Issue 2. – P. 103148-1-103148-14.

29 Xu J., Pan J., Yan Y. Agglutinative Language Speech Recognition Using Automatic Allophone Deriving // Chinese Journal of Electronics. – 2016. – Vol. 25, Issue 2. – P. 328-333.

30 Muhammadjon M., Saida M., Ilyos K. et al. USC: An Open-Source Uzbek Speech Corpus and Initial Speech Recognition Experiments // https://arxiv.org/abs/2107.14419v1. 18.04.2023.

31 Alakbar V., Natavan A. et al. Development of Speech Recognition Systems in Emergency Call Centers // Symmetry. – 2021. – Vol. 13. – P. 634-1-634-17.

32 Povey D., Ghoshal A., Gilles B. et al. The Kaldi Speech Recognition Toolkit // Procced. conf. Workshop on Automatic Speech Recognition and Understanding (IEEE 2011). – Waikoloa, 2011. – P. 1-4.

33 Samir R., Natavan A., Alakbar V. Automatic Speech Recognition in Taxi Call Service Systems // Procced. internat. conf. Emerging Technologies in Computing (ICETIC 2019). – London, 2019. – P. 243-253.

34 Huang X., Acero A., Hon H.-W. Spoken Language Processing: A Guide to Theory, Algorithm, and System Development. – New Jersey: Prentice Hall., 2001. – 980 p.

35 Ren Z., Nurmemet Y., Wushour S. et al. Improving Hybrid CTC/Attention Architecture for Agglutinative Language Speech Recognition // Sensors. – 2022. – Vol. 22, Issue 19. – P. 7319-1-7319-17.

36 Common Voice // https://commonvoice.mozilla.org/en/datasets. 18.04.2023.

37 Mamyrbayev O., Oralbekova D., Alimhan K. et al. A study of transformer-based end-to-end speech recognition system for Kazakh language // Scientific Reports. – 2022. – Vol. 12, Issue 1. – P. 8337-1-8337-11.

38 Orken Z.M., Dina O., Keylan A. et al. Hybrid end-to-end model for Kazakh speech recognition // International Journal of Speech Technology. – 2022. – Vol. 26. – P. 261-270.

39 Guo T., Yolwas N., Slamu W. Efficient Conformer for Agglutinative Language ASR Model Using Low-Rank Approximation and Balanced Softmax // Applied Sciences. – 2023. – Vol. 13, Issue 7. – P. 4642-1-4642-16.

40 Amirgaliyev B., Kuanyshbay D., Baimuratov O. et al. Development of Automatic Speech Recognition for Kazakh Language using Transfer Learning // International Journal of Advanced Trends in Computer Science and Engineering. – 2020. – Vol. 9, Issue 4. – P. 5880-5886.

41 Nicholas P. Introduction to Altaic Linguistics. – Wiesbaden: Harrassowitz, 1965. – 212 p.

42 Lars J., Csató É.Á. The Turkic Languages. – London: Routledge, 2015. – 504 p.

43 Muhamedova R. Kazakh: A Comprehensive Grammar. – London: Routledge, 2015. – 323 p.

44 Dotton Z., Wagner J.D. A Grammar of Kazakh // https://slaviccenters.duke.edu/sites/slaviccenters.duke.edu/files/file-attachments/kazakh. 18.04.2023.

45 Aslı G., Celia K. Turkish: A Comprehensive Grammar. – London: Routledge, 2005. – 580 p.

46 Rehm G., Uszkoreit. The Finnish Language in the Digital Age. – Berlin, 2012. – 81 p.

47 Durrell M. Hammer's German Grammar and Usage. – NY.: Routledge, 2016. – 632 p.

48 Jaehoon Y., Lucien B., Korean: A Comprehensive Grammar. – London: Routledge, 2011. – 496 p.

49 Jurafsky D., Martin J. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. – Ed. 3rd. – New Jersey, 2009. – 988 p.

50 Bird S., Klein E., Loper E. Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. – Sebastopol, 2009. – 504 p.

51 Loper E., Bird S. NLTK: the Natural Language Toolkit // https://archive.org/details/arxiv-cs0205028. 29.09.2023.

52 Smith R. An Overview of the Tesseract OCR Engine // Procced. Ninth internat. conf. on Document Analysis and Recognition (ICDAR 2007). – Curitiba, 2007. – P. 629-633.

53 LeCun Y., Bengio Y., Hinton G. Deep learning // Nature. – 2015. – Vol. 521. – P. 436-444.

54 Yingying Z., Cong Y., Xiang B. Scene text detection and recognition: recent advances and future trends // Frontiers of Computer Science. – 2015. – Vol. 10. – P. 19-36.

55 Sutskever I., Oriol V., Quoc V.L. Sequence to sequence learning with neural networks // https://arxiv.org/abs/1409.3215. 29.09.2023 y.

56 Ashish V., Noam S., Niki P. et al. Attention Is All You Need // https://arxiv.org/abs/1706.03762. 29.09.2023.

57 Radford A., Wu J., Child R. et al. Language Models are Unsupervised Multitask Learners // https://paperswithcode.com/paper/language. 29.09.2023.

58 Lewis M., Liu Y., Goyal N. et al. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension // Proceed. conf. of the 58th Annual Meeting of the Association for Computational Linguistics. – Stroudsburg, 2020. – P. 7871-7880.

59 Tomas M., Kai C., Greg S.C. and Jeffrey D. Efficient Estimation of Word Representations in Vector Space // https://arxiv.org/abs/1301.3781. 29.09.2023y.

60 Огнев И В. and Парамонов П.А. Распознавание речи методами скрытых марковских моделей в ассоциативной осцилляторной среде // Технические науки. Информатика, вычислительная техника. – 2013. – №3(27). – С. 115-126.

61 Bhatt S., Jain A., Dev A. Feature Extraction Techniques with Analysis of Confusing Words for Speech Recognition in the Hindi Language // Wireless Personal Communications. – 2021. – Vol. 118. – P. 3303-3333.

62 Yang C.-H. et al. Decentralizing Feature Extraction with Quantum Convolutional Neural Network for Automatic Speech Recognition // Proocced. (ICASSP 2021) IEEE internat. conf. on Acoustics, Speech and Signal Processing. – Toronto, 2021. – P. 6523-6527.

63 Qin L., Yuze Y., Tianxiang L. et al. MSP-MFCC: Energy-Efficient MFCC Feature Extraction Method With Mixed-Signal Processing Architecture for Wearable Speech Recognition Applications // IEEE Access. – 2020. – Vol. 8. – P. 48720-48730.

64 Apeksha A., Akshat S., Ajay A. et al. Two-Way Feature Extraction for Speech Emotion Recognition Using Deep Learning // Sensors. – 2022. – Vol. 22, Issue 6. – P. 2378-1-2378-11.

65 Naderi N., Nasersharif B. Robust sub-band speech feature extraction using multiresolution convolutional neural networks // Journal of Electrical Engineering. – 2019. – Vol. 49, Issue 3. – P. 89-1-89-13.

66 Panikos H., Yasser F.O.M., Akio Y. Deep Convolutional Neural Networks for Feature Extraction in Speech Emotion Recognition // Human-Computer Interaction. Recognition and Interaction Technologies: procced. conf. – Orlando, 2019. – P. 117-132.

67 Becchetti C., Ricotti L.P. Speech Recognition: Theory and C++ Implementation. – Hoboken, NJ, 1999. – 428 p.

68 Zhang Q., Lu H., Sak H. et al. Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss // https://arxiv.org/abs/2002.02562. 07.05.2023.

69 Watanabe S., Hori T., Kim S. Hybrid CTC/Attention Architecture for End-to-End Speech Recognition // IEEE Journal of Selected Topics in Signal Processing. – 2017. – Vol. 11, Issue 8. – P. 1240-1253.

70 Jonas A., Marc P., Noah R. et al. Introducing a Virtual Assistant to the Lab: A Voice User Interface for the Intuitive Control of Laboratory Instruments // SLAS Technology. – 2018. – Vol. 23, Issue 5. – P. 476-482.

71 Gupta M., Rajnish K., Bhattacharjee V. Impact of Parameter Tuning for Optimizing Deep Neural Network Models for Predicting Software Faults // Scientific Programming. – 2021. – Vol. 2021, Issue 3. – P. 1-17.

72 Hughes J. The Problem with Word Error Rate (WER) // https://www.speechmatics.com/company/articles-and-news/the-problem. 07.05.2023.

73 Leug K. Evaluate OCR Output Quality with Character Error Rate (CER) and Word Error Rate (WER) // https://towardsdatascience.com. 07.05.2023.

74 Wang S., Schuurmans D., Peng F. et al. Combining Statistical Language Models via the Latent Maximum Entropy Principle // Machine Learning. – 2005. – Vol. 60. – P. 229-250.

75 Rabiner L. and Juang B. Speech Recognition // In book: Springer Handbook of Speech Processing. – Heidelberg: Springer Berlin, 2008. – P. 873-902.

76 Ронжин А., Карпов А., Ли И. Речевой и многомодальный интерфейсы. – М., 2006. – 170 с.

77 Jelinek F. Continuous speech recognition by statistical methods // In Proceedings of the IEEE. – 1976. – Vol. 64, Issue 4. – P. 532-556.

78 Hinton G., Deng L., Yu D. et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups // IEEE Signal Processing Magazine. – 2012. – Vol. 29, Issue 6. – P. 82-97.

79 Yu D., Deng L. Automatic Speech Recognition: A Deep Learning Approach. – London: Springer-Verlag, 2015. – 321 p.

80 Deng L. Deep learning: from speech recognition to language and multimodal processing // APSIPA Transactions on Signal and Information Processing. – 2016. – Vol. 5. – P. 1-15.

81 Grezl F., Karafiat M., Kontar S. et al. Probabilistic and Bottle-Neck Features for LVCSR of Meetings // Proceed. conf. Acoustics, Speech and Signal Processing (ICASSP). – NY., 2007. – P. IV-757-IV-760.

82 Andrew L.M., Peng Q., Ziang X. et al. Building DNN acoustic models for large vocabulary speech recognition // Computer Speech & Language. – 2017. – Vol. 41. – P. 195-213.

83 Miao Y. Kaldi+ PDNN: building DNN-based ASR systems with Kaldi and PDNN // https://arxiv.org/abs/1401.6984. 29.09.2023.

84 Sainath T.N., Mohamed A.-r., Kingsbury B. et al. Deep convolutional neural networks for LVCSR // Proceed. of IEEE internat. conf. on Acoustics, Speech and Signal Processing (ICASSP). – Vancouver, 2013. – P. 8614-8618.

85 Delcroix M., Kinoshita K., Ogawa A. et al. Context Adaptive Neural Network Based Acoustic Models for Rapid Adaptation // IEEE/ACM Transactions on Audio, Speech, and Language Processing. – 2018. – Vol. 26, Issue 5. – P. 895-908.

86 Гапочкин А.В. Нейронные сети в системах распознавания речи // Science Time. – 2014. – №1. – С. 29-36.

87 Кипяткова И.С., Карпов А.А. Разновидности глубоких искусственных нейронных сетей для систем распознавания речи // Тр. СПИИРАН. – 2016. – Вып. 49. – С. 80-103.

88 Waibel A., Toshiyuki H., Hinton G., Shikano K. and Lang J.K. Phoneme recognition using time-delay neural networks // IEEE Transactions on Acoustics, Speech and Signal Processing. – 1989. – Vol. 37, Issue 3. – P. 328-339.

89 Peddinti V., Povey D., Khudanpur S. A time delay neural network architecture for efficient modeling of long temporal contexts // Procced. conf. «Interspeech 2015». – Dresden, 2015. – P. 3212-3218.

90 Тампель И. Автоматическое распознавание речи – основные этапы за 50 лет // Научно-технический вестник информационных технологий, механики и оптики. – 2015. – Т. 15, №6. – С. 957-968.

91 Geiger J.T., Zhang Z., Weninger F. et al. Robust speech recognition using long short-term memory recurrent neural networks for hybrid acoustic modelling // Procced. conf. «Interspeech 2014». – Singapore, 2014. – P. 631-635.

92 Zhang Y., Pezeshki M., Brakel P. et al. Towards End-to-End Speech Recognition with Deep Convolutional Neural Networks // https://arxiv.org/abs/1701.02720. 29.09.2023.

93 Graves A. et al. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks // Appearing in Proceedings of the 23 rd International Conference on Machine Learning. – Pittsburgh, 2016. – P. 1-9.

94 Graves A., Jaitly N. Towards End-To-End Speech Recognition with Recurrent Neural Networks // Proceed. of the 31st internat. conf. on Machine Learning (PMLR). – Beijing, 2014. – P. 1764-1772.

95 Adnan A., Javeria N., Meer H. Voice Reminder Assistant based on Speech Recognition and Speaker Identification using Kaldi // Pakistan Journal of Scientific Research. – 2017. – Vol. 1, Issue 2. – P. 13-18.

96 Hori T., Watanabe S., Hershey J.R. Multi-level language modeling and decoding for open vocabulary end-to-end speech recognition // Procced. 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). – Okinawa, 2017. – P. 287-293.

97 Rabiner L. A tutorial on hidden Markov models and selected applications in speech recognition // Proceed. of the IEEE. – 1989. – Vol. 77, Issue 2. – P. 257-286.

98 Bishop C.M. Pattern Recognition and Machine Learning. – NY., 2006. – 738 p.

99 Baum L.E., Petrie T.S.G., Weiss N. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains // The Annals of Mathematical Statistics. – 1970. – Vol. 41, Issue 1. – P. 164-171.

100 Технологии распознавания речи. Системы искусственного интеллекта, распознающие речь, прошли большой путь развития от появления в 1970-х годах до наших дней // https://ai-news.ru/2018/02/tehnologii _raspoznavaniya_rechi_sistemy_iskusstvennogo_intellekta_raspoznau. 29.09.2023.

101 Марковников Н.М., Кипяткова И.С. Аналитический обзор интегральных систем распознавания речи // Тр. СПИИРАН. – 2018. – Вып. 58. – С. 77-110.

102 Mikolov T., Karafiat M., Burget L. et al. Recurrent neural network based language model // Procced. conf. Interspeech 2010. – Chiba, 2010. – P. 1045-1048.

103 Rao K., Peng F., Sak H. et al. Grapheme-to-phoneme conversion using Long Short-Term Memory recurrent neural networks // Procced. 2015 IEEE internat. conf. on Acoustics, Speech and Signal Processing (ICASSP). – South Brisbane, 2015. – P. 4225-4229.

104 Маковкин К.А. Гибридные модели – Скрытые марковские модели/Многослойный персептрон и их применение в системах распознавания речи. Обзор // Речевые технологии. – 2012. – №3. – С. 58-83.

105 Jaitly N., Hinton G. Learning a better representation of speech soundwaves using restricted boltzmann machines // Procced. 2011 IEEE internat. conf. on Acoustics, Speech and Signal Processing (ICASSP). – Prague, 2011. – P. 5884-5887.

106 Rumelhart D.E., McClelland J.L. Information Processing in Dynamical Systems: Foundations of Harmony Theory // Procced. Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations. – Cambridge: MIT Press, 1987. – P. 194-281.

107 Haykin S. Neural Networks: A Comprehensive Foundation. – New Jersey: Prentice Hall, 1998. – 842 p.

108 Shrott R.Sequence Models by Andrew Ng – 11 Lessons Learned // https://towardsdatascience.com/sequence-models-by-andrew-ng-11. 15.05.2023.

109 Vanilla Recurrent Neural Network // https://calvinfeng.gitbook.io /machine-learning-notebook/supervised-learning/recurrent-neural.18.05.2023y.

110 RNN, LSTM, GRU и другие рекуррентные нейронные сети // http://vbystricky.ru/2021/05/rnn_lstm_gru_etc.html. 18.05.2023.

111 Lipton Z.C. and Berkowitz J. A Critical Review of Recurrent Neural Networks for Sequence Learning // https://arxiv.org/abs/1506.00019. 20.05.2023.

112 Williams R.J., Zipser D. Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity // In book: Backpropagation: theory, architectures, and applications. – London: Psychology Press, 1995. – P. 433-486.

113 Saxena S. Introduction to Gated Recurrent Unit (GRU) // https://www.analyticsvidhya.com/blog/2021/03/introduction-to-gated. 20.05.2023.

114 Sayre K.M. Machine recognition of handwritten words: A project report // Pattern Recognition. – 1973. – Vol. 5, Issue 3. – P. 213-228.

115 Bridle J.S. Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition // In book: Neurocomputing. – Heidelberg: Springer-Verlag Berlin, 1990. – P. 227-236.

116 Cho K., Merriënboer B.V., Gulcehre C. et al. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation // Proceed. of the 2014 conf. on Empirical Methods in Natural Language Processing (EMNLP). – Doha, Qatar, 2014. – P. 1724-1734.

117 Chorowski J.K., Bahdanau D., Serdyuk D. et al. Attention-Based Models for Speech Recognition // Advances in Neural Information Processing Systems. – 2015. – Vol. 9. – P. 577-585.

118 Mnih V., Heess N., Graves A. and Kavukcuoglu K. Recurrent Models of Visual Attention // Proceed. of the 27th internat. conf. on Neural Information Processing Systems. – Montreal, 2014. – P. 2204-2212.

119 Трансформеры как графовые нейронные сети // https://habr.com/ru/articles/491576/.10.06.2023.

120 Yifan P., Siddharth D., Ian L. and Shinji W. Branchformer: Parallel MLP-Attention Architectures to Capture Local and Global Context for Speech Recognition and Understanding // Procced. internat. conf. on Machine Learning. – Baltimore, 2022. – P. 17627-17643.

121 Liu H., Dai Z., So D.R. et al. Pay Attention to MLPs // https://deepai.org/publication/pay-attention-to-mlps. 10.06.2023.

122 Mussakhojayeva S., Khassanov Y., Varol H.A. KSC2: An Industrial-Scale Open-Source Kazakh Speech Corpus // Procced. Interspeech 2022. – Incheon, 2022. – P. 1367-1371.

123 Akbayan B., Orken M. End-to-end speech recognition systems for agglutinative languages // Scientific Journal of Astana IT University. – 2023. – Vol. 13, Issue 13. – P. 86-92.

124 Du W., Maimaitiyiming Y., Nijat M. et al. Automatic Speech Recognition for Uyghur, Kazakh, and Kyrgyz: An Overview // Applied Sciences. – 2023. – Vol. 13, Issue 1. – P. 326-1-326-25.

125 Cho J., Baskar M.K., Li R. et al. Multilingual Sequence-to-Sequence Speech Recognition: Architecture, Transfer Learning, and Language Modeling // Procced. 2018 IEEE Spoken Language Technology Workshop (SLT). – Athens, 2018. – P. 521-527.

126 Ardila R., Branson M., Kelly D. et al. Common Voice: A Massively-Multilingual Speech Corpus // Procced. 12 th internat. conf. on Language Resources and Evaluation. – Marseille, 2020. – P. 4218-4222.

127 Heigold G., Vanhoucke V., Senior A. et al. Multilingual acoustic models using distributed deep neural networks // Procced. 2013 IEEE internat. conf. on Acoustics, Speech and Signal Processing. – NY., 2013. – P. 8619-8623.

128 Çarki K., Geutner P. and Schultz T. Turkish LVCSR: towards better speech recognition for agglutinative languages // Procced. 2000 IEEE internat. conf. on Acoustics, Speech, and Signal Processing. – Istanbul, 2000. – P. 1563-1566.

129 Żelasko P., Feng S., Velázquez L.M. et al. Discovering phonetic inventories with crosslingual automatic speech recognition // Computer Speech & Language. – 2022. – Vol. 74. – P. 101358-1-101358-58.

130 Tachbelie M.Y., Abate S.T., Schultz T. Multilingual speech recognition for GlobalPhone languages // Speech Communication. – 2022. – Vol. 140. – P. 71-86.

131 Chowdhury S.A., Hussein A., Abdelali A. et al. Towards One Model to Rule All: Multilingual Strategy for Dialectal Code-Switching Arabic ASR // Procced. of the 22nd Annual conf. of the internat. Speech Communication Association. – Brno, 2021. – P. 1-5.

132 Ankit K., Aggarwal R.K. An Investigation of Multilingual TDNN-BLSTM Acoustic Modeling for Hindi Speech Recognition // International Journal of Sensors Wireless Communications and Control. – 2022. – Vol. 12. – P. 19-31.

133 Mussakhojayeva S., Khassanov Y., Atakan V. A Study of Multilingual End-to-End Speech Recognition for Kazakh, Russian, and English // Lecture Notes in Computer Science. – 2021. – Vol. 12997 LNAI. – P. 448-459.

134 Watanabe S., Hori T., Karita S. et al. ESPnet: End-to-End Speech Processing Toolkit // Procced. «Interspeech, 2018». – Hyderabad, 2018. – P. 1-6.

135 Watanabe S., Boyer F., Chang X. et al. The 2020 ESPnet Update: New Features, Broadened Applications, Performance Improvements, and Future Plans // Procced. 2021 IEEE Data Science and Learning Workshop (DSLW). – Toronto, 2021. – P. 1-6.

136 Guo P., Boyer F., Chang X. et al. Recent Developments on Espnet Toolkit Boosted By Conformer // Procced. IEEE internat. conf. on Acoustics, Speech and Signal Processing (ICASSP). – Toronto, 2021. – P. 5874-5878.

137 Qin C.-X., Qu D., Zhang L.-H. Towards end-to-end speech recognition with transfer learning // EURASIP Journal on Audio, Speech, and Music Processing. – 2018. – Vol. 2018. – P. 18-1-18-9.

138 Kimanuka U., Buyuk O. Turkish Speech Recognition Based On Deep Neural Networks // Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi. – 2018. – Cil. 22. – S. 319-329.

139 Xiao Z., Ou Z., Chu W. et al. Hybrid CTC-Attention based End-to-End Speech Recognition using Subword Units // Procced. 11th internat. sympos. on Chinese Spoken Language Processing (ISCSLP). – Taipei, 2018. – P. 146-150.

140 Karita S., Kubo Y., Bacchiani M. et al. A Comparative Study on Neural Architectures and Training Methods for Japanese Speech Recognition // Procced. Interspeech - 2021. – Brno, 2021. – P. 2092-2096.

141 Hannun A., Case C., Casper J. et al. Deep Speech: Scaling up end-to-end speech recognition // https://arxiv.org/abs/1412.5567. 30.06.2023.

142 Yang H., Nam H. Hyperparameter experiments on end-to-end automatic speech recognition // Phonetics Speech Sci. – 2021. – Vol. 13. – P. 45-51.

143 Chuang S.-P., Liu A.H. et al. Improving Automatic Speech Recognition and Speech Translation via Word Embedding Prediction // IEEE/ACM Transactions on Audio, Speech, and Language Processing. – 2020. – Vol. 29. – P. 93-105.

144 Kubo Y., Karita S., Bacchiani M. Knowledge Transfer from Large-scale Pretrained Language Models to End-to-end Speech Recognizers // Procced. IEEE internat. conf. on Acoustics, Speech and Signal Processing (ICASSP). – Singapore, 2022. – P. 8512-8516.

145 Ronny H.W., Peyser C., Sainath T.N. et al. Sentence-select: large-scale language model data selection for rare-word speech recognition // Procced. Interspeech, 2022. – Incheon, 2022. – P. 689-693.

146 Mukherji K., Pandharipande M., Kopparapu S.K. Improved Language Models for ASR using Written Language Text // Procced. 2022 National conf. on Communications (NCC). – Mumbai, 2022. – P. 362-366.

147 Mussakhojayeva S., Dauletbek K., Yeshpanov R. et al. Multilingual Speech Recognition for Turkic Languages // Information. – 2023. – Vol. 14, Issue 2. – P. 74-1-74-18.

148 Amirgaliyev Y., Kuanyshbay D., Yedilkhan D. et al. Automatic speech recognition system for kazakh language using connectionist temporal classifier // Journal of Theoretical and Applied Information Technology. – 2020. – Vol. 98, Issue 4. – P. 703-713.

149 Kun J., Jungang X. A Survey on Neural Network Language Models // https://www.researchgate.net/publication/333678905_A_Survey_on. 10.07.2023.

150 Bengio Y., Senecal J.-S. Quick Training of Probabilistic Neural Nets by Importance Sampling // Proceed. of the Ninth International Workshop on Artificial Intelligence and Statistics. – Florida, 2003. – P. 1-8.

151 Rao K., Peng F., Sak H. et al. Grapheme-to-phoneme conversion using Long Short-Term Memory recurrent neural networks} // Procced. IEEE internat. conf. on Acoustics, Speech and Signal Processing (ICASSP). – South Brisbane, 2015. P. 4225-4229.

152 Alsayadi H.A., Abdelhamid A.A., Hegazy I. et al. Arabic speech recognition using end-to-end deep learning // IET Signal Processing. – 2021. – Vol. 15, Issue 8. – P. 521-534.

153 Mamyrbayev O., Alimhan K., Zhumazhanov B. et al. End-to-End Speech Recognition in Agglutinative Languages // Procced. conf. Intelligent Information and Database Systems. – Yogyakarta, 2019. – P. 391-401.

154 Asefisaray B., Haznedaroğlu A., Erden M. et al. Transfer learning for automatic speech recognition systems // Procced. 26th Signal Processing and Communications Applications conf. (SIU). – Izmir, 2018. – P. 1-4.

155 Zou W., Jiang D., Zhao S. et al. Comparable Study Of Modeling Units For End-To-End Mandarin Speech Recognition // Procced. 11th internat. sympos. on Chinese Spoken Language Processing (ISCSLP). – Taipei, 2018. – P. 369-373.

156 Markovnikov N., Kipyatkova I. Investigating Joint CTC-Attention Models for End-to-End Russian Speech Recognition // Procced. conf. Speech and Computer. – SPb., 2019. – P. 337-347.

157 Zeng Z., Pham V.T., Xu H. et al. Leveraging Text Data Using Hybrid Transformer-LSTM Based End-to-End ASR in Transfer Learning // Procced. 12th internat. sympos. on Chinese Spoken Language Processing (ISCSLP). – Hong Kong, 2021. – P. 1-5.

158 Thilo von N., Christoph B., Kinoshita K. et al. On Word Error Rate Definitions and their Efficient Computation for Multi-Speaker Speech Recognition Systems // https://api.semanticscholar.org/CorpusID. 29.09.2023.

159 Mamyrbayev O., Turdalyuly M., Mekebayev N. et al. Automatic Recognition of Kazakh Speech Using Deep Neural Networks // Procced. conf. Intelligent Information and Database Systems. – Yogyakarta, 2019. – P. 465-474.

160 Oord A.V.D., Dieleman S., Zen H. et al. WaveNet: A Generative Model for Raw Audio // https://arxiv.org/abs/1609.03499. 15.07.2023.

161 Zeghidour N., Usunier N., Kokkinos I. et al. Learning Filterbanks from Raw Speech for Phone Recognition // Procced. IEEE internat. conf. on Acoustics, Speech and Signal Processing (ICASSP). – Calgary, 2018. – P. 5509-5513.

162 Schuller B., Weninger F., Wöllmer M. et al. Non-negative matrix factorization as noise-robust feature extractor for speech recognition // Procced. IEEE internat. conf. on Acoustics, Speech and Signal Processing. – Dallas, 2010. – P. 4562-4565.

163 Kermanshahi M.A., Akbari A., Nasersharif B. Transfer Learning for End-to-End ASR to Deal with Low-Resource Problem in Persian Language // Procced. 26th internat. Computer conf., Computer Society of Iran (CSICC). – Tehran, 2021. – P. 1-5.

APPENDIX A

Certificates to a speaker of the seminars “Improved Speech Recognition for Agglutinative languages”
https://satbayev.university/en/news/the-acesyri-summer-school-began-with-the-itm-2023-conference

Instituto Sup
de Engenhari
Politécnico de Coimbr

Erasmus+

CERTIFICATE
CERTIFICADO

THIS CERTIFICATE IS AWARDED TO
O PRESENTE CERTIFICADO É ATRIBUÍDO A

PhD student Akbayan Bekarystankyzy, from Satbayev Technical University, Almaty - Kazakhstan
Aluna de Doutoramento Akbayan Bekarystankyzy da Universidade Técnica Nacional, Almaty - Cazaquistão

AS A SPEAKER OF THE SEMINAR
ENQUANTO ORADORA DO SEMINÁRIO

"Improved Speech Recognition for Agglutinative languages"
"Reconhecimento melhorado do discurso para Línguas aglutinativas"

On 21st April 2023, at Coimbra Institute of Engineering (ISEC)
No dia 21 de abril de 2023, no Instituto Superior de Engenharia de Coimbra (ISEC)

ISEC International Relations Coordinator
Coordenador das Relações Internacionais do ISEC
Prof. Luís Castro, PhD

ACeSYRI

Co-funded by the Erasmus+ Programme of the European Union

CERTIFICATE

issued to

Akbayan Bekarystankyzy

for the best presentation speech in the
"ACeSYRI Young Researchers School"
organized by **Satbayev University**, Almaty,
from June 05, 2023 until June 10, 2023.

10.06.2023
DATE

Elena Zaitseva

"Advanced Centre for PhD Students and Young Researchers in Informatics" (reg.no. 610166-EPP-1-2019-1-SK-EPPKA2-CBHE-JP)

APPENDIX B

Author’s certificates of government registration for intellectual object

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ

РЕСПУБЛИКА КАЗАХСТАН

АВТОРЛЫҚ ҚҰҚЫҚПЕН ҚОРҒАЛАТЫН ОБЪЕКТІЛЕРГЕ ҚҰҚЫҚТАРДЫҢ МЕМЛЕКЕТТІК ТІЗІЛІМГЕ МӘЛІМЕТТЕРДІ ЕНГІЗУ ТУРАЛЫ

КУӘЛІК

2023 жылғы «21» тамыз № 38545

Автордың (лардың) жөні, аты, әкесінің аты (егер ол жеке басын куәландыратын құжатта көрсетілсе):
БЕКАРЫСТАНҚЫЗЫ АҚБАЯН

Авторлық құқық объектісі: **ЭЕМ-ге арналған бағдарлама**

Объектінің атауы: **Программное обеспечение UniCodeKaz**

Объектіні жасаған күні: **17.11.2022**

Құжат түпнұсқалығын http://www.kazpatent.kz/ru сайтының "Авторлық құқық" бөлімінде тексеруге болады https://copyright.kazpatent.kz

Подлинность документа возможно проверить на сайте kazpatent.kz в разделе «Авторское право» https://copyright.kazpatent.kz

ЭЦҚ қол қойылды

Е. Оспанов

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ

РЕСПУБЛИКА КАЗАХСТАН

АВТОРЛЫҚ ҚҰҚЫҚПЕН ҚОРҒАЛАТЫН ОБЪЕКТІЛЕРГЕ ҚҰҚЫҚТАРДЫҢ МЕМЛЕКЕТТІК ТІЗІЛІМГЕ МӘЛІМЕТТЕРДІ ЕНГІЗУ ТУРАЛЫ

КУӘЛІК

2023 жылғы «31» тамыз № 38833

Автордың (лардың) жөні, аты, әкесінің аты (егер ол жеке басын куәландыратын құжатта көрсетілсе):
БЕКАРЫСТАНҚЫЗЫ АҚБАЯН, Мамырбаев Оркен, Дуйсенхан Бейбитжан

Авторлық құқық объектісі: **ЭЕМ-ге арналған бағдарлама**

Объектінің атауы: **Система автоматического транскрибирование аудио файлов в текст**

Объектіні жасаған күні: **15.08.2023**

ҚР «Ұлттық зияткерлік меншік институты» РМК
РГП «Национальный институт интеллектуальной собственности» МЮ РК

Құжат түпнұсқалығын http://www.kazpatent.kz/ru сайтының "Авторлық құқық" бөлімінде тексеруге болады https://copyright.kazpatent.kz

Подлинность документа возможно проверить на сайте kazpatent.kz в разделе «Авторское право» https://copyright.kazpatent.kz

ЭЦҚ қол қойылды

Е. Оспанов

APPENDIX C

Script code for collecting the data of different Unicodes to one file

```
#The current program is for the preparation of data to machine learning task, exactly for automatic sppech recognition.
#This program writes the contents of text files to one file.

import codecs
import re
import sys
import os

import sys


def find_txt_files(directory):
    names = []
    for subdir, dirs, files in os.walk(directory):
        for file in files:
            full = os.path.join(subdir, file)
            index = file.find(".txt")
            if index != -1:
                names.append(file[0: index])
    return names


def find_title(titles, title):
    for index, item in enumerate(titles):
        if item == title:
            return index
    return -1


def _detect_encoding(s):
    if s.startswith(codecs.BOM_UTF16_BE):
        return 'utf-16-be'
    if s.startswith(codecs.BOM_UTF16_LE):
        return 'utf-16-le'
    if s.startswith(codecs.BOM_UTF32_BE):
        return 'utf-32-be'
    if s.startswith(codecs.BOM_UTF32_LE):
        return 'utf-32-le'
    if s.startswith(codecs.BOM_UTF8):
```

```
        return 'utf-8'
    m = re.match(br'\s*<\?xml\b.*\bencoding="([^"]+)"', s)
    if m:
        return m.group(1).decode()
    m = re.match(br"\s*<\?xml\b.*\bencoding='([^']+)'", s)
    if m:
        return m.group(1).decode()
    return 'utf-8'


def func(value):
    return ' '.join(value.split())


def remove_in_brackets(mystring):
    result = mystring
    while (True):
        print(result)
        start = result.find("(!")
        if start ==-1:
            start = result.find("( !")


        end = result.find(").")

        if start == -1 or start>end:
            return result

        if start != -1 and end != -1:
            result = result[0: start] + result[end+1:]
            result = func(result)

    return result


#This function throws the files of different codings, except UTF-8

def main_func():
    print(sys.argv[1])
    print(sys.argv[2])
    print(sys.argv[3])
    source_dir = sys.argv[1]
    target_dir = sys.argv[2]
    broken_dir = sys.argv[3]
```

```
    txt_files = find_txt_files(source_dir)

    validated_file = open(os.path.join(target_dir, "validated.tsv"), "w")

validated_file.write("client_id\tpath\tsentence\tup_votes\tdown_votes\tage\tgender\ta
ccents\tlocale\tsegment\n")

    main_title = ""
    sentence = ""
    for subdir, dirs, files in os.walk(source_dir):
        for file in files:
            full = os.path.join(subdir, file)
            if file.find(".wav") != -1:
                main_title = file[0: file.find(".wav")]
                txt_file_path = os.path.join(subdir, main_title + ".txt")

                if os.path.exists(txt_file_path):
                    with open(str(txt_file_path), 'br') as reader:
                        bytes = reader.read()



                        try:

                            sentence = str(bytes, 'UTF-8')
                            if sentence == "":
                                os.system("rm " + full)
                                os.system("rm " + os.path.join(subdir, main_title + ".txt"))
                                print(full)
                            else:
                                after_bytes = sentence.encode('UTF-8')
                                print("After_bytes: ", after_bytes)
                                if after_bytes.startswith(b'\xef\xbb\xbf'):
                                    after_bytes = after_bytes[3:]
                                    print(after_bytes)
                                    sentence = after_bytes.decode('UTF-8')
                                sentence = func(sentence)
                                os.system("cp " + full + " " + target_dir)

                                spker = main_title
                                if main_title.find("spk_") != -1:
                                    title_part = main_title[len("spk_"):]
                                    print(title_part)
                                    spker = title_part[0: title_part.find("_T")]
                                    print(spker)
```

```
                    print(main_title, "Senten: "+sentence)
                    validated_file.write(str(spker) + "\t" + str(
                        main_title + ".wav") + "\t" + sentence + "\t\t\t\t\t\t" + "kk" +
"\n")
                except:
                    txt_file_path = str(subdir + "/" + main_title + ".txt")
                    os.system("cp " + full + " " + broken_dir)
                    os.system("cp " + txt_file_path + " " + broken_dir)


#This function processes the files from exception block

def collect_from_broken():

    print(sys.argv[2])
    print(sys.argv[3])
    print(sys.argv[4])
    target_dir = sys.argv[2]
    broken_dir = sys.argv[3]
    after_broken_dir = sys.argv[4]


    validated_file = open(os.path.join(target_dir, "validated.tsv"), "a")

    main_title = ""
    sentence = ""
    for subdir, dirs, files in os.walk(broken_dir):
        for file in files:
            full = os.path.join(subdir, file)
            if file.find(".wav") != -1:
                main_title = file[0: file.find(".wav")]
                txt_file_path = os.path.join(subdir, main_title + ".txt")

                if os.path.exists(txt_file_path):
                    with open(str(txt_file_path), 'br') as reader:
                        bytes = reader.read()
                        try:
                            if bytes.startswith(b'\xef\xbb'):
                                bytes = bytes.replace(b'\xef\xbb', b'')
                                print(bytes)
                                sentence = str(bytes, 'UTF-8')
                            elif bytes.startswith(b'\xff\xfe'):
                                bytes = bytes.replace(b'\xff\xfe', b'')
                                print(bytes)
```

```
                    sentence = str(bytes, 'UTF-16')
                else:
                    sentence = str(bytes, 'rk1048')

                if sentence == "":
                    os.system("rm " + full)
                    os.system("rm " + os.path.join(subdir, main_title + ".txt"))
                    print(full)
                else:
                    sentence = func(sentence)

                    os.system("cp " + full + " " + target_dir)

                    spker = main_title
                    if main_title.find("spk_") != -1:
                        title_part = main_title[len("spk_"):]
                        print(title_part)
                        spker = title_part[0: title_part.find("_T")]
                        print(spker)
                    print(main_title, "\nSenten: "+sentence)
                    validated_file.write(str(spker) + "\t" + str(
                        main_title + ".wav") + "\t" + sentence + "\t\t\t\t\t\t" + "kk" +
"\n")
            except:
                txt_file_path = str(subdir + "/" + main_title + ".txt")
                os.system("cp " + full + " " + after_broken_dir)
                os.system("cp " + txt_file_path + " " + after_broken_dir)


main_func()
collect_from_broken()
```

APPENDIX D

Source code for Telebot

```
import subprocess as s
import os

import string
import soundfile
from espnet_model_zoo.downloader import ModelDownloader
from espnet2.bin.asr_inference import Speech2Text


# BEST MODEL:
tag = "Shinji Watanabe/librispeech_asr_train_asr_transformer_e18_
raw_bpe_sp_valid.acc.best"
# SECOND BEST MODEL:
#tag = 'Shinji Watanabe/spgispeech_asr_train_asr_conformer6_n_
fft512_hop_length256_raw_en_unnorm_bpe5000_valid.acc.ave'
# EXTREMELY POOR MODEL:
#tag = "kamo-naoyuki/wsj"
config = "exp/asr_train_asr_conformer5_raw_kk_char_sp/config.yaml"

model = "exp/asr_train_asr_conformer5_raw_kk_char_sp/valid.acc.ave_10best.pth"
speech2text = Speech2Text(config, model)
#d = ModelDownloader()
# speech2text = Speech2Text(
#    "model/valid.acc.best.pth",
#     device="cpu", #cuda if gpu
#     minlenratio=0.0,
#     maxlenratio=0.0,
#     ctc_weight=0.3,
#     beam_size=10,
#     batch_size=0
# )
#Strips text of punctuation and makes it uppercase
def text_normalizer(text):
    text = text.upper()
    return text.translate(str.maketrans('', '', string.punctuation))


# Generates and returns transcript given audio file path
def get_transcript(path):
    speech, rate = soundfile.read(path)
```

```
    nbests = speech2text(speech)
    text, *_ = nbests[0]
    return text, rate

# Set necessary paths
path = os.path.join(os.getcwd(), 'egs')
files = os.listdir(path+'/audio')
# For every file in audio directory
for file in files:
   # Get transcript, converting to .wav if not a wav
    if not file.endswith('.wav'):
        os.chdir(path+'/audio')
        s.run(f"ffmpeg -i {file} {file.split('.')[0]}.wav", shell=True, check=True,
universal_newlines=False)
        os.chdir('../..')
        file = file.split('.')[0]+'.wav'
        text, est_rate = get_transcript(f'{path}/audio/{file}')
        os.remove(f'{path}/audio/{file}')
    else:
        text, est_rate = get_transcript(f'{path}/audio/{file}')

    # Fetch true transcript
    #print(sentence)
    label_file = open(file.split('.')[0]+'.txt', "w")
    label = file.split('.')[0]+'.txt'
    #writes result of regicnition to the file
    label_file.write(text)
    # with open(f'{path}/text/{label}', 'r') as f:
    #   true_text = f.readline()
    # Print true transcript and hypothesis
    #print(f"\n\nReference text: {true_text}")
    #print(f"ASR hypothesis: {text_normalizer(text)}\n\n")
```