Rule-based morphological parser for Shughni language: nouns, verbs and adjectives

Elen Kartina
National Research University Higher School of Economics

Project Proposal

Research Advisor: George Moroz

March 2025

Contents

1 Introduction **3**
1.1 Shughni . . . 3
1.2 Morphology parsing . . . 4

2 Literature review **4**
2.1 Shughni . . . 4
2.2 Morphology modeling . . . 5
2.2.1 Neural approach . . . 5
2.2.2 Rule-based approach . . . 5

3 Methods **6**

4 Expected outcomes **9**

5 Conclusion **10**

References **11**

Abstract

In this work I propose a rule-based morphological analysis tool based on Helsinki Finite-State Technology (HFST) for the Shughni language (ISO: sgh; glottocode: shug1248), a language of the Iranian branch of the Indo-European family, a member of 'Pamiri' areal language group. While one existing rule-based parser exists for Shughni (Melenchenko, 2021), it does not utilize finite-state transducer technology. This work proposes the first HFST-based morphological parser implementation for Shughni, offering the advantages of this well-established framework for morphological analysis. The tool is planned to be presented in two types: a morphological parser that breaks each word-form into stem and morphemes and assigns morphological tags to each one of them; a morphological generator that outputs word-forms taking a stem and morphological tags as an input. This is a continuation my previous work, where nouns, pronouns, prepositions and numerals were implemented (Osorgin, 2024). This project aims to cover at least two more main parts of speech: verbs and adjectives.

1 Introduction

1.1 Shughni

The Shughni language (ISO: sgh; glottolog: shug1248) is a low-resource language. It belongs to the Iranian branch of the Indo-European family (Plungian, 2022, p. 12), and it is spoken by circa 80 000–100 000 people (Edelman & Dodykhudoeva, 2009) on the territories of regions of Tajikistan and Afghanistan. Both regions have a subregion, where Shughni language is the most spoken native language, the subregions are called 'Shughnon' in Tajikistan and 'Shughnan' in Afghanistan (Parker, 2023, p. 2), see Figure 1 for details. Shughni is one of 'Pamiri' languages, which is an areal group of languages spoken in the Pamir Mountains, primarily along the Panj river. Shughni has a mixed morphological typology type (Parker, 2023, p. 94), which means that grammatical meanings can be carried by morphemes, words and clitics. There are three scripts for Shughni language: Latin, Cyrillic and Arabic. The Arabic script is used on the territory of Badakhshan Province of Afghanistan, and Cyrillic and Latin scripts are used in the Mountainous Badakhshan Autonomous Region of Tajikistan. The Latin script was created and gained popularity in 1930s, after it was set as the primary script for teaching in schools on the Shughni-speaking territory of Tajikistan. Later in 1980s a Cyrillic script was created (Edelman & Dodykhudoeva, 2009, p. 788).

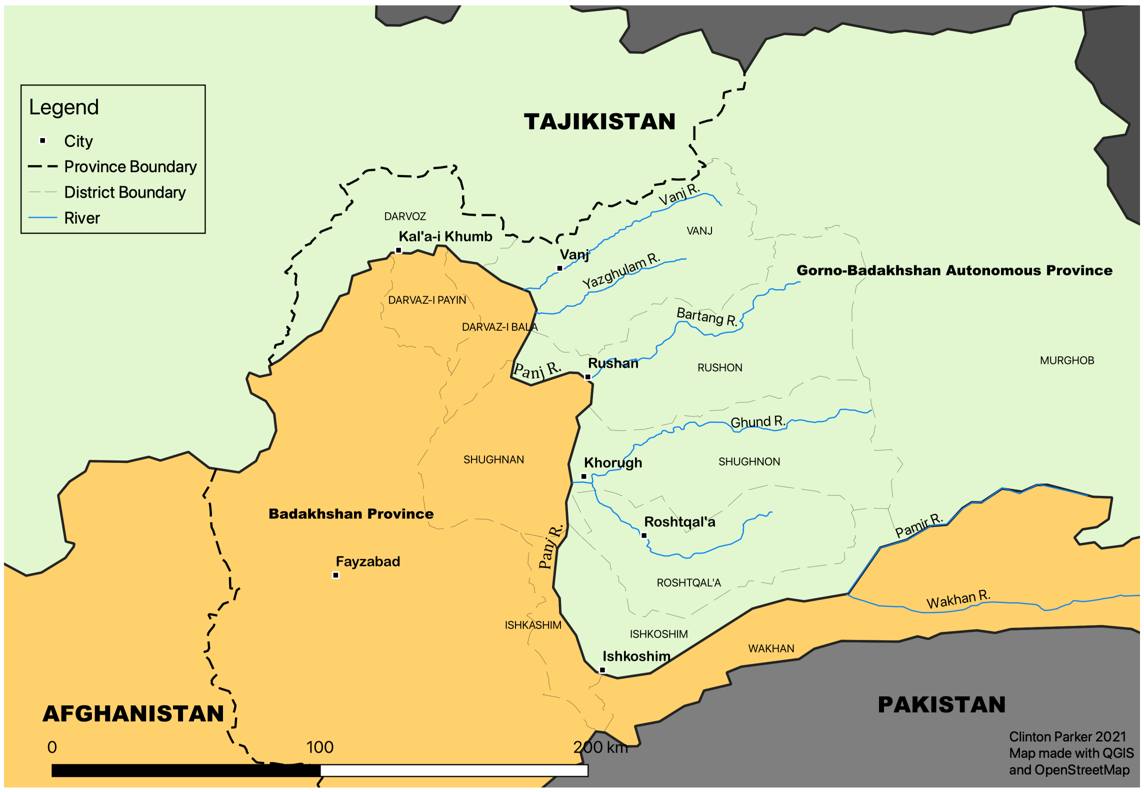

Figure 1: Mountainous Badakhshan Autonomous Province of Tajikistan and Badakhshan Province of Afghanistan, (Parker, 2023, Fig 1.1)

Morphology analysis tools in question will focus on the variation of Shughni that is spoken in Tajikistan. Cyrillic and Latin script will be supported: the core analysis tool will be implemented in Cyrillic script, and Latin script support will be implemented via transliteration. Clitics will be relevant to this work, since in target Cyrillic and Latin scripts they often attach to words either with a hyphen or through direct affixation and therefore enter the scope of morphology.

1.2 Morphology parsing

Morphological parser is a fundamental tool, a wide range of computational linguistics' tasks such as POS-tagging, lemmatization and machine translation rely on some form of morphological model. For morphologically rich languages it is close to impossible to list and manually define all the possible word-forms. An alternative solution is to model the language's morphology instead.

For high-resource languages morphology modeling today is usually approached using deep learning (DL) models such as BERT (Devlin et al., 2019), which are trained on large amounts of data. This method is not always available for low-resource languages that lack digital textual data, for such languages linguists apply rule-based approach.

Shughni is a low-resource language with very few data available, which leaves us the rule-based option. In this work Helsinki Finite-State Technology (HFST) will be used, which is a tool set that allows creating and manipulating finite-state transducers (FSTs). As it will be described below, FSTs can be used to model natural language's morphology and create morphological parsers. Modeling Shughni morphology using FST should not bring challenges. Its morphology is mostly agglunative and there are very few morphonological rules, the biggest challenge seems to be presented by clitics, which attach to words becoming part of morphology and which may carry information that exceeds the scope of morphology. Nouns, numerals, pronouns and preposition were implemented into a HFST morphological parser in my previous work (Osorgin, 2024). This work will fill the crucial gaps: verbs and adjectives.

2 Literature review

2.1 Shughni

There are two main dictionaries of the Shughni language: one by Zarubin (1960) and one by Karamshoev (1988–1999), both are written using Cyrillic script and include Russian translations. Some early dictionaries are 'Brief grammar and dictionary of Shughni' (Tumanovich, 1906), that is also using Cyrillic and translates to Russian, and 'Shughni dictionary by D. L. Ivanov' (Salemann, 1895), that translates to Russian but uses Arabic script alongside Cyrillic transcriptions for Shughhi word-forms.

Several Shughni grammar descriptions were written throughout the years, starting from basic grammar description done by D. L. Ivanov (Salemann, 1895, pp. 274–281). An important mention is a work by Karamshoev (1963), which was the most detailed Shughni grammar description of its time. Latest significant works were 'Shughni language' (Edelman & Yusufbekov, 1999, pp. 225–242), 'Comparative Grammar of Eastern Iranian Languages' (Edelman, 2009) and 'A grammar of the Shughhi language' by Parker (2023), which is the biggest existing grammar, the most detailed and the most recent one.

A significant contribution to the Shughni NLP field is 'Digital Resources for the Shughni language' (Makarov et al., 2022). The authors, among other tools and resources, developed a rule-based morphological analysis tool for the Shughni language Melenchenko (2021). The parser proposed in this work, while also being rule-based, differs in its implementation through the use of Helsinki Finite-State Technology (HFST).

2.2 Morphology modeling

2.2.1 Neural approach

One of the most recent and widely adopted approaches to morphology modeling involves the use of the Transformer-based deep learning models. This approach usually requires large amounts of training data in form of manually tagged word-forms, which is not available for Shughni. There are texts available to me, that were manually tagged as part of 'Digital Resources for the Shughni Language' project (Makarov et al., 2022), which consist of 3453 tokens in total. While this amount of training data is small, it is worth mentioning that it is possible to train a Transformer-based model with such small datasets, as shown by the developers of `UDify` model (Kondratyuk & Straka, 2019). Their approach includes fine-tuning a pre-trained multilingual BERT model (Devlin et al., 2019), authors conclude, that multilingual learning is most beneficial for low-resource languages, even ones that do not possess a training set.

The neural approach is not the main target of this work and is not planned to be implemented. Working with BERT models is a highly resource-demanding task. The authors of `UDify` state, that the training of their model for a new language would require at least 16 Gigabytes of RAM and at least 12 Gigabytes of GPU video memory, and the training process would take at least 20 days depending on the video card model. While a deep learning approach would be interesting to explore, such computational resources are not available for this project.

2.2.2 Rule-based approach

Finite-state technology (FST) is a finite-state machine with two tapes, one for input strings and one for output strings. The machine maps the alphabet of the first string to the alphabet of the second string, this concept was first proposed by Mealy (1955) and Moore (1956).

Eventually linguists noticed this technology and started applying it to model natural languages' grammar. Woods (1970) suggested Recursive Transition Networks (RTN) for sentence structure parsing, RTN essentially is a finite-state machine applied to syntax.

Koskenniemi (1983) created a model, which introduced an explicit formalism named Two-level morphology (TWOL) for describing morphological and morphonological paradigms. This model was capable of word-form recognition and production, but it was not yet compilable into finite-state machines, it was working at runtime and was known for being slow. Then Karttunen et al. (1987) at Xerox Research Center developed a Two-level rule Compiler (`twolc`), which compiled TWOL rules into a finite-state machine. Later a separate compiler for lexicon definitions was introduced named `lexc` (**Lex**icon **C**ompiler) (Karttunen, 1993), it came with its own formalism language for describing lexicon and morphotactics. The standard approach to modeling a language at that point was using `lexc` to describe lexicon and morphology and `twolc` to describe morphonology, which stayed almost the same to this day.

HFST is a set of tools for creating and working with languages' morphology models in form of transducers (Lindén et al., 2009). It includes both `hfst-lexc` and `hfst-twolc` compilers, as well as command line interface commands for mathematical and other miscellaneous operations with transducers. One of the latest recent advances was the release of the `lexd` lexicon compiler (Swanson & Howell, 2021). It is based on the `hfst-lexc` compiler but is claimed to be much faster in the compilation time.

FST (and HFST specifically) is widely applied when it comes to creating rule-based morphological models. Some of the latest examples of FST-based morphological tools are: morphological parser for the Tamil language by Sarveswaran et al. (2021), a morphological transducer for Kyrgyz by Washington et al. (2012), a morphological parser for Andi by Buntyakova (2023) and a morphological parser for the Chamalal language by Budilova (2023).

3 Methods

Shughni has multiple dictionary options, as we learned in the previous section. My choice fell for the dictionary made by Karamshoev (1988–1999) since it is the latest and the biggest dictionary. And moreover, researchers from the 'Digital Resources for the Shughni Language' project (Makarov et al., 2022) created an online Karamshoev's dictionary of the Shughni language. They granted me access to the underlying database from where it is easily possible to export all the needed lexemes and their meta information like POS tags or verb's tense and aspect.

Regarding the choice of the grammatical description, as the main source I have selected the one provided by Parker (2023) as the most suitable foundation for this study. This work offers an exceptionally thorough analysis of Shughni morphology including a very detailed inflectional paradigm description. In addition, I also will be using the inside materials of the 2019 – 2024 HSE expeditions to Tajikistan.

The morphological parser will be developed using Helsinki Finite-State Technology (HFST) (Lindén et al., 2009), which is a tool set for working with rule-based morphology models in a form of transducers. Finite-state transducer (FST) is a finite-state machine that works with two tapes, following the terminology for Turing machines (Turing, 1937): it reads strings of text from the input tape and writes strings of text to the output tape. When FST receives an input string, it walks along its characters from one state to another as long as there is a valid transition from the current state with an upcoming letter, see Figure 2 for visualization. It stops when it reaches an end state (marked as a double-lined circle on the Figure 2) and then the output tape's string is considered to be a valid output. If there is no valid path to an end state then the output tape's string is discarded. There are usually multiple end states present, not just one, as on the Figure 2.

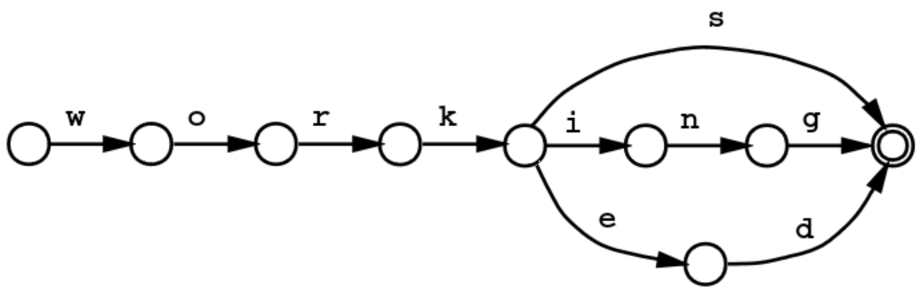

Figure 2: An example of FST for a language where only three words exist: *works*, *working* and *worked*. The word *worker*, for example will not be recognized as a valid word by this FST, since there is no 'd' transition at state *worke*. The only way from *worke* state is via 'd' transition, which corresponds to the *worked* word. (Beesley & Karttunen, 2002)

As FST walks through states (on Fig.2 states are graph's nodes) on each transition (on Fig.2 transitions are graph's edges) it can read an input tape and/or write to the output tape. A syntax for transitions is *'x:y'*, which means 'read *x* from input, write *y* to output'. Transition will happen only if input matches. With this in mind we can turn FST from Figure 2 into a morphological analyser by modifying some transitions (See Fig. 3). Syntax *'x:'* means that FST must write nothing to the output tape while still making a transition to the next state if input matches.

HFST is a modern tool set, it allows working with a wide range of compilers in a single shared FST format. Instead of built-in `hfst-lexc` I will be using `lexd` compiler. Even thought it is not included into the HFST, it still produces transducers in HFST format. The choice between `lexc` and `lexd` was made in the latter's favor since it is a newer, more optimized and faster version of `lexc`. The other option is Xerox's `xfst` compiler, which is a quite outdated option and was rarely used outside of Xerox. The advantage of `lexc` is its tag system, which allows tagging stems and affixes, allowing to tune the combination of the lexemes with specific tags.

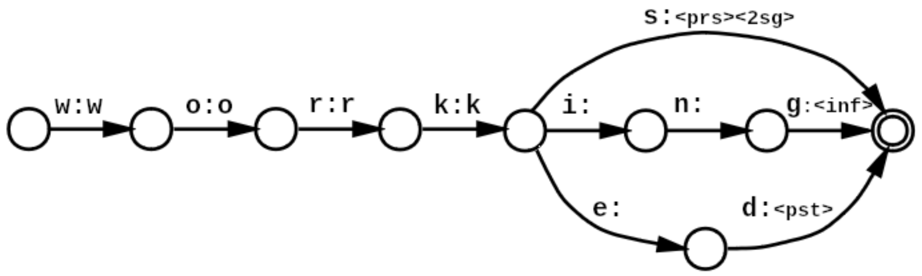

Figure 3: A modified version of Figure 2 which takes as input *works*, *working*, *worked* and outputs *work<prs><2sg>*, *work<inf>*, *work<pst>* respectively.

This feature allows us to minimize overgeneration of unwanted FST paths.

For the matter of the morphonology part, HFST's implementation of TWOLC (`hfst-twolc`) will be used. Shughni is not rich for morphonology, but it still has some rules like the insertion of *'y'* on the phoneme border between two vowels, which is not possible to model using only a lexicon compiler.

The quality of the product morphological parser will be measured using two metrics: *coverage* and *accuracy*. The *coverage* is calculated as follows:

$$coverage = \frac{N_{rec}}{N_{total}}$$

where N_{rec} is the amount of tokens, that the parser was able to recognize and return any output, and N_{total} is the total amount of tokens. As of now, texts for *coverage* evaluation are the Luke book of the Bible, a fairy tale and a collection of short miscellaneous texts with the total amount of 187149 tokens. All these texts were provided to me by Makarov et al. (2022). The formula for *accuracy* is:

$$accuracy = \frac{N_{correct}}{N_{rec}}$$

where $N_{correct}$ is the amount of tokens for which the parser gave the same set of morphological tags as a human linguist did manually and N_{total} is the total amount of recognized tokens. The manually tagged texts were also provided by 'Digital resources for the Shughni Language' project. The tagged texts consist of 3453 tokens in total.

The core of the model will be written around Cyrillic stems and affixes. Then a separate FST for transliteration will be attached to the Cyrillic FSTs to make transducers that are able to work with Latin script.

The format of the tagged version of token *'working'* will be *'work<v>><inf>'*. Each tag is encapsulated into angular brackets and the morpheme border will be represented by a single 'greater than' or 'right angular bracket' sign. The tagged token *'work<v>><inf>'* is an equivalent of *'work.V-INF'* in standard linguistic notation. The motivation for morpheme '>' separator is that it is a standard for Apertium's systems and this format may be assumed by the outside user. This decision is not final since it brings complications with parsing of this format because this separator matches morpheme border bracket.

4 Expected outcomes

The main product of this work is a set of finite-state transducers capable of morphological analysis and morphological generation. This tool set then can be used for different NLP tasks such as POS-tagging, spell-checking, morphological tagging or even translation. In the Table 1 the planned input-output formats of the FSTs are presented. 'Plain' analysers and generators work with the word-forms as they are occurring in plain text, while 'morph' analysers and generators work with word-forms that have morphemes separated by '>' symbol. The latter word-form format is useful to linguists in research, while the first format is easily applicable to plain texts.

Description	Input example	Output example
Cyrillic plain analyser	дарйойен	дарйо<n>><pl>
Cyrillic morph analyser	дарйо>йен	дарйо<n>><pl>
Latin plain analyser	daryoyen	дарйо<n>><pl>
Latin morph analyser	daryo>yen	дарйо<n>><pl>
Cyrillic plain generator	дарйо<n>><pl>	дарйойен
Cyrillic morph generator	дарйо<n>><pl>	дарйо>йен
Latin plain generator	дарйо<n>><pl>	daryoyen
Latin morph generator	дарйо<n>><pl>	daryo>yen

Table 1: A full list of planned HFST transducers

The title of this work mentions only verbs, nouns and adjectives, however this morphological parser is planned to cover all parts of speech. The title is focused on the verbs, nouns and adjectives since their morphological paradigms are the biggest. In terms of parser's quality I aim to reach 80% *coverage*, the best case scenario would be more than 90%. The existing non HFST-based parser for Shughni created by Melenchenko (2021) has final recall score of 90%, which formula is equivalent to my *coverage* and Melchenko's *accuracy* score is 69%. For HFST-based parsers for other languages that were mentioned earlier *coverage* varies from 40% (Buntyakova, 2023) to 90% (Ivanova et al., 2022). My target *coverage* values are motivated by the results of existing solution for Shughni and by the results of my previous work (Osorgin, 2024) where I have reached the *coverage* value of 45% by implementing nouns, pronouns, numerals and prepositions.

The source code and the compiled `.hfst` files will be uploaded to the public GitHub repository for anyone to use and contribute to.

5 Conclusion

In this work I will continue the development of HFST-based morphological parser for Shughni started in my prevoius work (Osorgin, 2024). In the first version of the parser only nouns, numerals, pronouns and prepositions were implemented. In this work I aim to fill the crucial gaps: verbs and adjectives. The addition of such important parts of speech will make the parser much more complete.

The hope is to introduce an alternative to the existing solution and advance the development of natural language processing tools for the Shughni language. Morphological parser can later be utilized by other NLP tools such as spell-checkers, POS-taggers, syntactic parsers, lemmatizers and more.

References

Beesley, e. R., & Karttunen, L. (2002). *Finite-state morphology: Xerox tools and techniques*. CSLI, Stanford.

Budilova, Z. A. (2023). *Создание морфологического парсера для чамалинского языка в системе lexd и twol [Morphological parser of Chamalal in lexd and twol]*, NRU HSE. https://www.hse.ru/edu/vkr/837214661

Buntyakova, V. A. (2023). *Создание морфологического парсера андийского языка в системе lexd и twol [Morphological parser of Andi in lexd and twol*, NRU HSE. https://www.hse.ru/edu/vkr/837214826

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)* (pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

Edelman, D. I. (2009). Сравнительная грамматика восточноиранских языков [Comparative Grammar of Eastern Iranian Languages].

Edelman, D. I., & Yusufbekov, S. (1999). Shughni language. In *Языки мира: Иранские языки. III: Восточноиранские языки [Languages of the world: Iranian languages. III. Eastern Iranian languages]*.

Edelman, D. I., & Dodykhudoeva, L. R. (2009). Shughni. In G. Windfuhr (Ed.), *The iranian languages* (pp. 787–824). London & New York: Routledge.

Ivanova, S., Washington, J., & Tyers, F. (2022). A free/open-source morphological analyser and generator for sakha. In N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, J. Odijk, & S. Piperidis (Eds.), *Proceedings of the Thirteenth Language Resources and Evaluation Conference* (pp. 5137–5142). European Language Resources Association. https://aclanthology.org/2022.lrec-1.550/

Karamshoev, D. (1963). *Баджувский диалект шугнанского языка [Badzhuvskij dialect of the Shughni language]*. изд-во АН Тадж. ССР. https://books.google.ru/books?id=8q1GXwAACAAJ

Karamshoev, D. (1988–1999). *Шугнанско-русский словарь [Shughni-Russian dictionary]*. Izd-vo Akademii nauk SSSR.

Karttunen, L. (1993). Finite-state lexicon compiler. Technical Report ISTL-NLTT-1993-04-02, Xerox Palo Alto Research Center, Palo Alto, CA.

Karttunen, L., Koskenniemi, K., & Kaplan, R. (1987). A compiler for two-level phonological rules. *tools for morphological analysis*.

Kondratyuk, D., & Straka, M. (2019). 75 languages, 1 model: Parsing universal dependencies universally.

Koskenniemi, K. (1983). Two-level Morphology: A General Computational Model for Word-Form Recognition and Production.

Lindén, K., Silfverberg, M., & Pirinen, T. (2009). HFST Tools for Morphology – An Efficient Open-Source Package for Construction of Morphological Analyzers. In C. Mahlow & M. Piotrowski (Eds.), *State of the Art in Computational Morphology* (pp. 28–47, Vol. 41). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-04131-0_3

Makarov, Y., Melenchenko, M., & Novokshanov, D. (2022). Digital Resources for the Shughni Language. *Proceedings of The Workshop on Resources and Technologies for Indigenous, Endangered and Lesser-Resourced Languages in Eurasia within the 13th Language Resources and Evaluation Conference*, 61–64. https://aclanthology.org/2022.eurali-1.9

Mealy, G. H. (1955). A method for synthesizing sequential circuits. *The Bell System Technical Journal*, 1045–1079.

Melenchenko, M. G. (2021). *Автоматический морфологический анализ шугнанского языка [Automatic full morphology analysis for Shughni]*, NRU HSE.

Moore, E. F. (1956). Gedanken-Experiments on Sequential Machines. In C. E. Shannon & J. McCarthy (Eds.), *Automata Studies* (pp. 129–154). Princeton University Press. https://doi.org/doi:10.1515/9781400882618-006

Osorgin, I. G. (2024). *Создание морфологического парсера для шугнаского языка в системе lexd и twol [Creating a morphological parser for the Shughni language using lexd and twol systems]*, NRU HSE.

Parker, C. (2023). *A grammar of the shughni language* [Doctoral dissertation, Department of Linguistics, McGill University].

Plungian, V. (2022). The study of shughni: The past and the future. *RSUH/RGGU Bulletin: "Literary Teory. Linguistics. Cultural Studies", Series. 2022;(5):11-22*.

Salemann, K. (1895). Шугнанский словарь Д.Л. Иванова [Shughni dictionary by D.L. Ivanov]. In *Восточные заметки [Eastern notes]*.

Sarveswaran, K., Dias, G., & Butt, M. (2021). ThamizhiMorph: A morphological parser for the Tamil language. *Machine Translation 35*, 37–70. https://link.springer.com/article/10.1007/s10590-021-09261-5

Swanson, D., & Howell, N. (2021). Lexd: A Finite-State Lexicon Compiler for Non-Suffixational Morphologies.

Tumanovich. (1906). *Краткая грамматика и словарь шугнанского наречия [Brief grammar and dictionary of Shughni]*.

Turing, A. M. (1937). On computable numbers, with an application to the entscheidungsproblem. https://doi.org/10.1112/plms/s2-42.1.230

Washington, J., Ipasov, M., & Tyers, F. (2012). A finite-state morphological transducer for Kyrgyz. In N. Calzolari, K. Choukri, T. Declerck, M. U. Doğan, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, & S. Piperidis (Eds.), *Proceedings of the eighth international conference on language resources and evaluation (LREC'12)* (pp. 934–940). European Language Resources Association (ELRA). https://aclanthology.org/L12-1642/

Woods, W. (1970). Transition Network Grammars for Natural Language Analysis. *Computational Linguistics*. https://cse.buffalo.edu/~rapaport/663/S02/woods.atn.pdf

Zarubin, I. (1960). *Шугнанские тексты и словарь [Shughni texts and dictionary]*. Izd-vo Akademii nauk SSSR.