Part of speech and gramset tagging algorithms for unknown words based on morphological dictionaries of the Veps and Karelian languages*

Andrew Krizhanovsky[1,2][0000−0003−3717−2079], Natalia Krizhanovskaya[1][0000−0002−9948−1910], and Irina Novak[3][0000−0002−9436−9460]

[1] Institute of Applied Mathematical Research
of the Karelian Research Centre of the Russian Academy of Sciences
[2] Petrozavodsk State University
[3] Institute of Linguistics, Literature and History
of the Karelian Research Centre of the Russian Academy of Sciences,
Petrozavodsk, Russia, andrew.krizhanovsky@gmail.com
http://dictorpus.krc.karelia.ru

Abstract. This research devoted to the low-resource Veps and Karelian languages. Algorithms for assigning part of speech tags to words and grammatical properties to words are presented in the article. These algorithms use our morphological dictionaries, where the lemma, part of speech and a set of grammatical features (gramset) are known for each word form. The algorithms are based on the analogy hypothesis that words with the same suffixes are likely to have the same inflectional models, the same part of speech and gramset. The accuracy of these algorithms were evaluated and compared. 66 thousand Karelian and 313 thousand Vepsian words were used to verify the accuracy of these algorithms. The special functions were designed to assess the quality of results of the developed algorithms. 86.8% of Karelian words and 92.4% of Vepsian words were assigned a correct part of speech by the developed algorithm. 90.7% of Karelian words and 95.3% of Vepsian words were assigned a correct gramset by our algorithm. Morphological and semantic tagging of texts, which are closely related and inseparable in our corpus processes, are described in the paper.

Keywords: Morphological analysis · Low-resource language · Part of speech tagging.

1 Introduction

Our work is devoted to low-resource languages: Veps and Karelian. These languages belong to the Finno-Ugric languages of the Uralic language family. Most Uralic languages still lack full-fledged morphological analyzers and large corpora [5].

* The study was supported by the Russian Foundation for Basic Research, grant 18-012-00117.

In order to avoid this trap the researchers of Karelian Research Centre are developing the Open corpus of Veps and Karelian languages (VepKar). Our corpus contains morphological dictionaries of the Veps language and the three supradialects of the Karelian language: the Karelian Proper, Livvi-Karelian and Ludic Karelian. The developed software (corpus manager)[4] and the database, including dictionaries and texts, have open licenses.

Algorithms for assigning part of speech tags to words and grammatical properties to words, without taking into account a context, using manually built dictionaries, are presented in the article (see Section 4).

The proposed technology of evaluation (see the section 5) allows to use all 313 thousand Veps and 66 thousand Karelian words to verify the accuracy of the algorithms (Table 1). Only a third of Karelian words (28%) and two-thirds of Veps words (65%) in the corpus texts are automatically linked to the dictionary entries with all word forms (Table 1). These words were used in the evaluation of the algorithms.

Table 1: Total number of words in the VepKar corpus and dictionary

Language	The total number of tokens in texts, 10^3	N tokens linked to dictionary automatically, 10^3	N tokens linked to lemmas having a complete paradigm, 10^3
Veps	488	400 (82%)	313 (65%)
Karelian Proper	245	111 (45%)	69 (28%)

Let us describe several works devoted to the development of morphological analyzers for the Veps and Karelian languages.

- The Giellatekno language research group is mainly engaged in low-resource languages, the project covers about 50 languages [4]. Our project has something in common with the work of Giellatekno in that (1) we work with low-resource languages, (2) we develop software and data with open licenses. A key role in the Giellatekno infrastructure is given to formal approaches (grammar-based approach) in language technologies. They work with morphology rich languages. Finite-state transducers (FST) are used to analyse and generate the word forms [4].
- There is a texts and words processing library for the Uralic languages called UralicNLP [2]. This Python library provides interface to such Giellatekno tools as FST for processing morphology and constraint grammar for syntax. The UralicNLP library lemmatizes words in 30 Finno-Ugric languages and

[4] See https://github.com/componavt/dictorpus

dialects including the Livvi dialect of the Karelian language (*olo* – language code).

2 Data organization and text tagging in the VepKar corpus

Automatic text tagging is an important area of research in corpus linguistics. It is required for our corpus to be a useful resource.

The corpus manager handles the dictionary and the corpus of texts (Fig. 1). The texts are segmented into sentences, then sentences are segmented into words (tokens). The dictionary includes lemmas with related meanings, word forms, and **sets** of **gram**matical features (in short – **gramsets**).

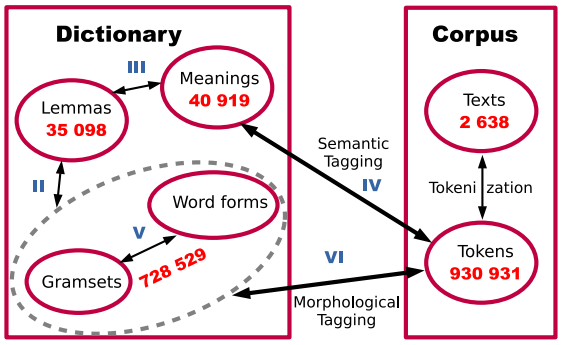


Fig. 1: Data organization and text tagging in the VepKar corpus.
Total values (e.g. number of words, texts) are calculated for all project languages.

Text tokens are automatically searched in the dictionary of lemmas and word forms, this is the first stage (I) of the text tagging, it is not presented at Fig. 1.

1. **Semantic tagging**. For the word forms found in the dictionary, the lemmas linked with them are selected (II at Fig. 1), then all the meanings of the lemmas are collected (III) and semantic relationships are established between the tokens and the meanings of the lemmas (marked "not verified") (IV).
   The task of an expert linguist is to check these links and confirm their correctness, either choose the correct link from several possible ones, or manually add a new word form, lemma or meaning.

$$\text{tokens (words)} \xleftrightarrow{\text{I}} \text{word forms} \xleftrightarrow{\text{II}} \text{lemmas} \xleftrightarrow{\text{III}} \text{meanings}$$
$$\text{IV not verified}$$

   When the editor clicks on the token in the text, then a drop-down list of lemmas with all the meaning will be shown. The editor selects the correct lemma and the meaning (Fig. 2).

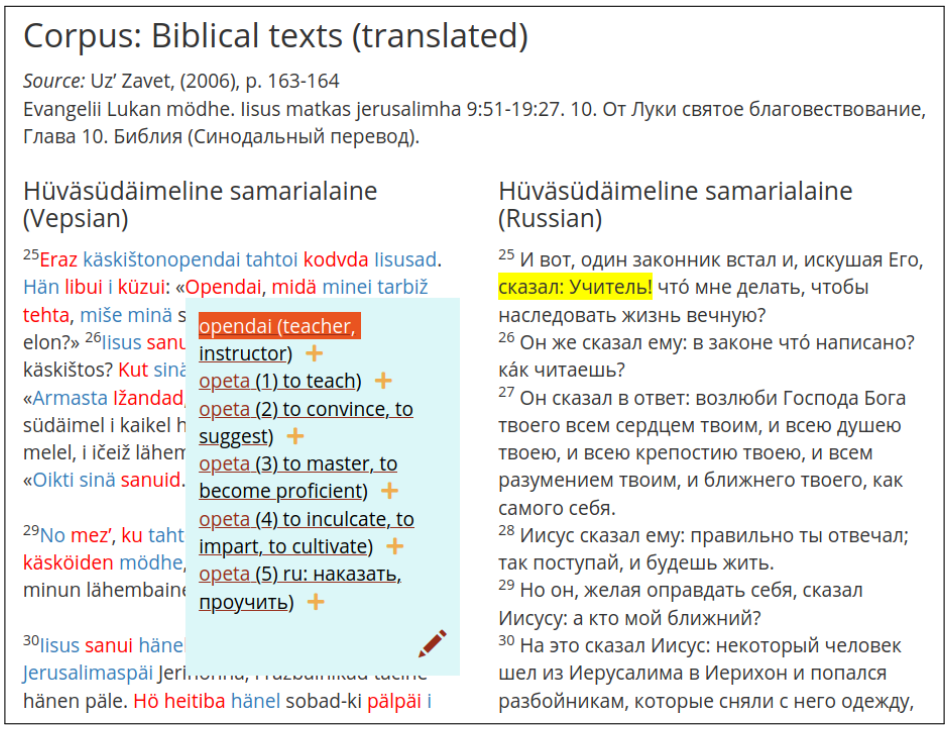
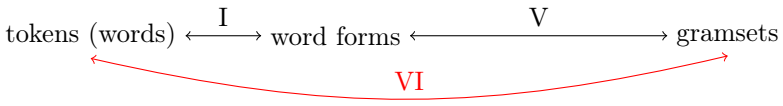


Fig. 2: Vepsian and Russian parallel Bible translation[‡] in the corpus. The editor clicks the word "Opendai" in the text, a menu pops up. This menu contains a list of meanings collected automatically for this token, namely: the meaning of the noun "teacher" ("opendai" in Veps) and five meanings of the Veps verb "opeta". The noun "opendai" and the verb "opeta" have the same wordform "opendai". If the editor selects one of the lemma meanings in the menu (clicks the plus sign), then the token and the correct meaning of the lemma will be connected (IV stage is verified).

[‡] See full text online at VepKar: http://dictorpus.krc.karelia.ru/en/corpus/text/494

2. **Morphological tagging**. For the word forms found in the dictionary, the gramsets linked with them are selected (V) and morphological links are established (VI) between the tokens and the pairs "word form – gramset" (Fig. 1). The expert's task is to choose the right gramset.



3 Corpus tagging peculiarities

In this section, we describe why the word forms with white spaces and analytical forms are not taken into account in the search algorithm described below. Analytical form is the compound form consisting of auxiliary words and the main word.

The ultimate goal of our work is the morphological markup of the text, previously tokenized into words by white spaces and non-alphabetic characters (for example, brackets, punctuation, numbers). Therefore, analytical forms do not have markup in the texts.

Although we store complete paradigms in the dictionary, including analytical forms, such forms do not used in the analysis of the text, because each individual word is analyzed in the text, not a group of words.

For example, we take the Karelian verb "pageta" (leave, run away). In the dictionary not only the negative form of indicative, presence, first-person singular "en pagene" is stored, but also connegative (a word form used in negative clauses) of the indicative, presence "pagene", which is involved in the construction of five of the six forms of indicative, presence. Thus, in the text the word 'en' (auxiliary verb 'ei', indicative, first-person singular) and 'pagene' (verb 'pageta', connegative of indicative, presence) are separately marked.

4 Part of speech and gramset search by analogy algorithms

The proposed algorithms operate on data from a morphological dictionary. The algorithms are based on the analogy hypothesis that words with the *same suffixes* are likely to have the same inflectional models and the same sets of grammatical information (part of speech, number, case, tense, etc.). The *suffix* here is a final segment of a string of characters.

Let the hypothesis be true, in that case, if the suffixes of new words coincide with the suffixes of dictionary words, then part of the speech and other grammatical features of the new words will coincide with the dictionary words. It should be noted that the length of the suffixes is unpredictable and can be various for different pairs of words [1, p. 53].

The POSGuess and GramGuess algorithms described below use the concept of "suffix" (Fig. 3), the GramPseudoGuess algorithm uses the concept "pseudo-ending" (Fig. 4).

4.1 The POSGuess algorithm for part of speech tagging with a suffix

Given the set of words W, for each word in this set the part of speech is known. The algorithm 1 finds a part of speech pos_u for a given word u using this set.

Algorithm 1: Part of speech search by a suffix (POSGuess)

Data: P – a set of part of speech (POS),
$W = \{w \mid \exists \text{pos}_w \in P\}$ – a set of words, POS is known for each word,
$u \notin W$ – the word with unknown POS,
$len(u)$ – the length (in characters) of the string u.

Result:

$$u_z : \begin{cases} len(u_z) \xrightarrow[z=2,\ldots,len(u)]{} \max, & \texttt{//Longest suffix} \\ \exists w \in W : w = w_{\text{prefix}} + u_z & \texttt{//Concatenation of strings} \end{cases}$$

$$\text{Counter}\left[\text{pos}^k\right] = c^k,\ k = \overline{1,m},\ \text{where :}$$
$$c^k \in \mathbb{N},\ c^1 \geq c^2 \geq \ldots \geq c^m,$$
$$\exists w_i^k \in W : w_i^k = w_{\text{prefix}_i^k} + u_z \Rightarrow c^k = |\text{pos}_{w_i^k}^k|,$$
$$i = \overline{1,c^k},$$
$$\forall i : \text{pos}_{w_i^k}^k = \text{pos}^k \in P, \quad a \neq b \Leftrightarrow \text{pos}^a \neq \text{pos}^b$$
m – the number of different POS of found words w_i^k

```
1  z = 2 // The position in the string u
2  z_found = FALSE
3  while z ≤ len(u) and ¬z_found do
       // The suffix of the word u from z-th character
4      u_z = substr (u, z)
5      foreach w ∈ W do
           // If the word w has the suffix u_z (regular expression)
6          if w =~ m/u_z$/ then
7              Counter [pos_w] + +
8              z_found = TRUE // Only POS of words with this u_z suffix
               will be counted. The next "while" loop will break, so the
               shorter suffix u_{z+1} will be omitted.
9          end
10     end
11     z = z + 1
12 end
   // Sort the array in descending order, according to the value
13 arsort( Counter [ ] )
```

In Algorithm 1 we look for in the set W (line 5) the words which have the same suffix u_z as the unknown word u. Firstly, we are searching for the longest substring of u, that starts at index z. The first substring $u_{z=2}$ will start at the second character (line 1 in Algorithm 1), since $u_{z=1} = u$ is the whole string (Fig. 3).

Then we increment the value z decreasing the length of the substring u_z in the loop, while the substring u_z has non-zero length, $z \leq len(u)$. If there are words in W with the same suffix, then we count the number of similar words for each part of the speech and stop the search.

The Fig. 3 shows the idea of the algorithm 1: for a new word (*kezaman*), we look for a word form in the dictionary (*raman*) with the same suffix (*aman*).

We begin to search in the dictionary for word forms with the suffix $u_{z=2}$. If we have not find any words, then we are looking for $u_{z=3}$ and so on. The longest suffix $u_{z=4}$="aman" with $z = 4$ is found.

Then we find all words with the suffix $u_{z=4}$ and count how many of such words are nouns, verbs, adjectives and so on. The result is written to the array $Counter[\,]$. In Fig. 3 the *noun* "raman" was found, therefore we increment the value of $Counter[noun]$.

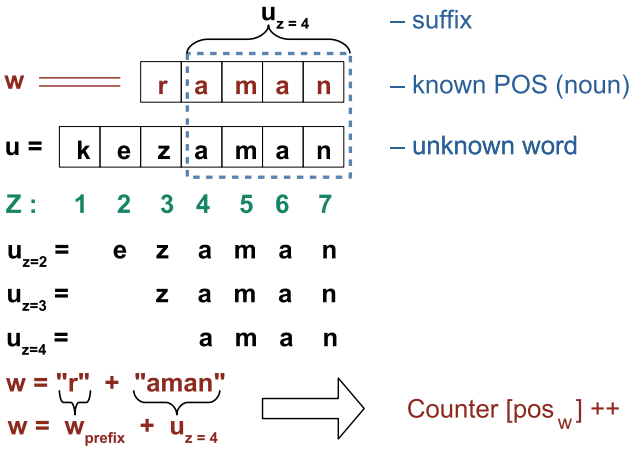


Fig. 3: Veps nouns in the genitive case "kezaman" ("kezama" means "melted ground") and "raman" ("rama" means "frame"). The word u with an unknown part of speech is "kezaman". The word w from the dictionary with the known POS is "raman". They share the common suffix u_z, which is "aman".

4.2 The GramGuess algorithm for gramset tagging with a suffix

The GramGuess algorithm is exactly the same as the POSGuess algorithm, except that it is needed to search a subset of gramsets instead of parts of speech. That is in the set W the gramset is known for each word. The gramset is a set of morphological tags (number, case, tense, etc.).

4.3 The GramPseudoGuess algorithm for gramset tagging with a pseudo-ending

Let us explain the "pseudo-ending" used in the algorithm GramPseudoGuess.

All word forms of one lemma share a common invariant substring. This substring is a *pseudo-base* of the word (Fig. 4). Here the pseudo-base is placed at the start of a word, it suits for the Veps and Karelian languages. For example, in Fig. 4 the invariant substring "huuk" is the pseudo-base for all word forms of the lemma "huukkua". The Karelian verb "huukkua" means "to call out", "to holler", "to halloo".

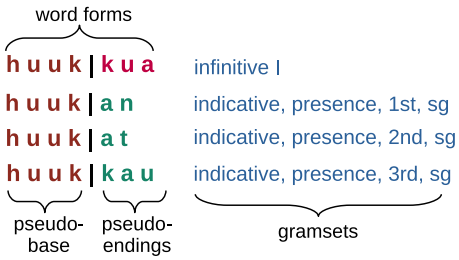


Fig. 4: Wordforms of the Karelian verb "huukkua" (it means "to call out", "to holler", "to halloo"). All word forms have the same pseudo-base and different pseudo-endings for different set of grammatical attributes (gramsets).

Given the set of words W, for each word in this set a gramset and a pseudo-ending are known. The algorithm 2 finds a gramset g_u for a given word u using this set.

In Algorithm 2 we look for in the set W (line 5) the words which have the same pseudo-ending u_z as the unknown word u. Firstly, we are searching for the longest substring of u, that starts at index z.

Then we increment the value z decreasing the length of the substring u_z in the loop, while the substring u_z has non-zero length, $z \leq len(u)$. If there are words in W with the same pseudo-ending, then we count the number of similar words for each gramset and stop the search.

Algorithm 2: Gramset search by a pseudo-ending (GramPseudoGuess)

Data: G – a set of gramsets,
$W = \{w \mid \exists\, g_w \in G, \exists\, \text{pend}_w : w = w_{\text{prefix}} + \text{pend}_w\}$ – a set of words, where gramset and pseudo-ending (pend) are known for each word,
$u \notin W$ – the word with unknown gramset,
$len(u)$ – the length (in characters) of the string u.

Result:

$$u_z : \begin{cases} len(u_z) \xrightarrow[z=2,\ldots,len(u)]{} \max, \texttt{ //Longest substring} \\ \exists w \in W : \text{pend}_w = u_z \end{cases}$$

$$\text{Counter}\left[g^k\right] = c^k,\ k = \overline{1,m},\ \text{where :}$$

$$c^k \in \mathbb{N},\ c^1 \geq c^2 \geq \ldots \geq c^m,$$

$$\exists w_i^k \in W : \text{pend}_{w_i^k} = u_z \Rightarrow c^k = |g_{w_i^k}^k|,$$

$$i = \overline{1,c^k},$$

$$\forall i : g_{w_i^k}^k = g^k \in G, \quad a \neq b \Leftrightarrow g^a \neq g^b$$

m – the number of different gramsets of found words w_i^k

```
1  z = 2 // The position in the string u
2  z_found = FALSE
3  while z ≤ len(u) and ¬z_found do
       // The substring of the word u from z-th character
4      u_z = substr (u, z)
5      foreach w ∈ W do
           // If the word w has the pseudo-ending u_z
6          if pend_w == u_z then
7              Counter [g_w] + +
8              z_found = TRUE // Only gramsets of words with the
               pseudo-ending u_z will be counted. The next "while" loop
               will break, so the shorter u_{z+1} will be omitted.
9          end
10     end
11     z = z + 1
12 end
   // Sort the array in descending order, according to the value
13 arsort( Counter [ ] )
```

5 Experiments

5.1 Data preparation

Lemmas and word forms from our morphological dictionary were gathered to *one set* as a search space of part of speech tagging algorithm. This set contains unique pairs "word – part of speech".

In order to search a gramset, we form the set consisting of (1) lemmas without inflected forms (for example, adverbs, prepositions) and (2) inflected forms (for example, nouns, verbs). This set contains unique pairs "word – gramset". For lemmas without inflected forms the gramset is empty.

We put on constraints for the words in both sets: strings must consist of more than two characters and must not contain whitespace. That is, analytical forms and compound phrases have been excluded from the sets (see section 3).

5.2 Part of speech search by a suffix (POSGuess algorithm)

For the evaluation of the quality of results of the searching algorithm POSGuess the following function eval(pos^u) was proposed:

$$\text{eval}\left(\begin{matrix}\text{pos}^u,\\ \text{Counter}\left[\text{pos}^k\right] \to c^k,\\ \forall k = \overline{1,m}\end{matrix}\right) = \begin{cases}\text{The array Counter[] do not contain the correct pos}^u.\\ 0, \quad \text{pos}^u \neq \text{pos}^k, \forall k = \overline{1,m},\\ \\ \text{First several POS in the array can have}\\ \text{the same maximum frequency } c^1\text{, one of this POS is pos}^u.\\ 1, \quad \text{pos}^u \in \{\left[\text{pos}^1, \ldots, \text{pos}^j\right] : c^1 = c^2 = \ldots = c^j, j \leq m\},\\ \\ \frac{c^k}{\sum_{k=1}^{m} c^k}, \ \exists k : \text{pos}^k = \text{pos}^u, \ c^k < c^1\end{cases} \tag{1}$$

This function (1) evaluates the result of the POSGuess algorithm against the correct part of speech pos^u. The POSGuess algorithm counts the number of words similar to the word u separately for each part of speech and stores the result in the Counter array.

The Counter array is sorted in descending order, according to the value. The first element in the array is a part of speech with maximum number of words similar to the unknown word u.

71 091 "word – part of speech" pairs for the Proper Karelian supradialect and 399 260 "word – part of speech" pairs for the Veps language have been used in the experiments to evaluate algorithms.

During the experiments, two Karelian words were found, for which there were no suffix matches in the dictionary. They are the word "cap" (English: snap; Russian: *цап*) and the word "štob" (English: in order to; Russian: *чтобы*). That is, there were no Karelian words with the endings -p and -b. This could be explained by the fact that these two words migrated from Russian to Karelian language.

Figure 5 shows the proportion of Veps and Karelian words with correct and wrong part of speech assignment by the POSGuess algorithm. Values along the X axis are the values of the function eval(pos^u), see the formula (1). This function for evaluating the part of speech assignment takes the following values:

0 — 4.7% of Vepsian words and 9% of Karelian words ($x = 0$ in Fig. 5) were assigned the wrong part of speech. That is, there is no correct part of speech in the result array $Counter[]$ in the POSGuess algorithm. This is the first line in the formula (1).

0.1 – 0.5 — 2.92% of Vepsian words and 4.23% of Karelian words ($x \in [0.1; 0.5]$ in Fig. 5) were assigned the partially correct POS tags. That is, the array $Counter[]$ contains the correct part of speech, but it is not at the beginning of the array. This is the last line in the formula (1).

1 — 92.38% of Vepsian words and 86.77% of Karelian words ($x = 1$ in Fig. 5) were assigned the correct part of speech. The array $Counter[]$ contains the correct part of speech at the beginning of the array.

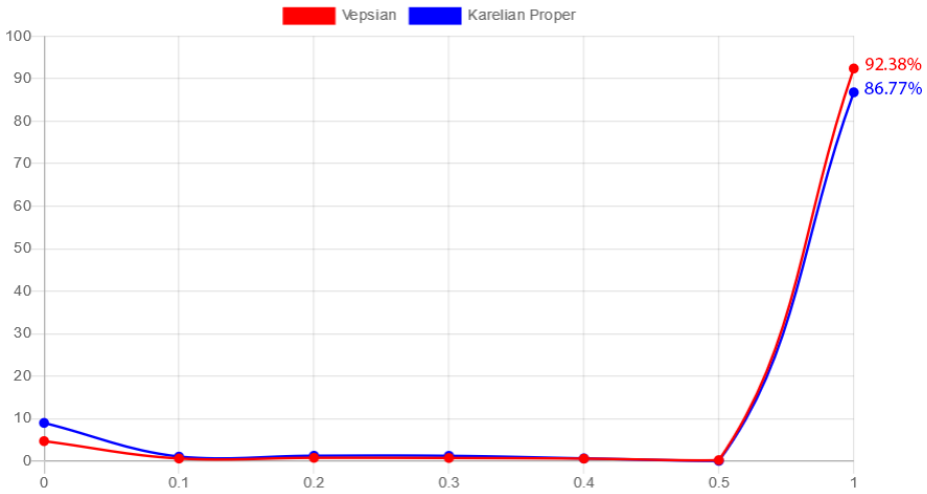


Fig. 5: The proportion of Vepsian (red curve) and Karelian (blue curve) words with correct ($x = 1$) and wrong ($x = 0$) part of speech assignment by the POSGuess algorithm with the formula (1).

Figure 5 shows the evaluation of the results of the POSGuess algorithm for all parts of speech together. Table 2 (Veps) and Table 3 (Karelian) show the evaluation of the same results of the POSGuess algorithm, but they are presented for each part of speech separately.

Table 2: Number of Vepsian words of different parts of speech used in the experiment. The evaluation of results found by POSGuess algorithm by the formula (1) and fraction of results in percent, where the column *0* means the fraction of words with incorrectly found POS, *1* – the fraction of words with correct POS in the top of the list created by the algorithm.

Veps		Fraction of not guessed (column 0), partly guessed (0.1–0.5) and guessed (1) POS, %						
POS	Words	0	0.1	0.2	0.3	0.4	0.5	1
Verb	93 047	2.12	0.52	0.55	0.47	0.36	0.01	**95.97**
Noun	240 513	2.88	0.3	0.67	0.6	0.42	0.24	**94.89**
Adjective	61 845	12.45	1.62	1.44	1.53	1.58	0.51	**80.87**
Pronoun	1244	46.54	8.12	0.56	0.64	0	0	**44.13**
Numeral	1200	44	6.25	2.33	0.67	0.33	0	**46.42**
Adverb	650	64.92	3.08	2.46	1.23	0.46	0	**27.85**

Table 3: Number of Karelian words of different parts of speech used in the experiment.

Karelian		Fraction of not guessed (column 0), partly guessed (0.1–0.5) and guessed (1) POS, %						
POS	Words	0	0.1	0.2	0.3	0.4	0.5	1
Verb	26 033	3.26	0.5	0.74	0.6	0.23	0.01	**94.67**
Noun	36 908	5.47	0.38	1.13	1.08	0.52	0.04	**91.38**
Adjective	6596	35.81	6.66	4.15	4.56	2.73	0.38	**45.71**
Pronoun	610	81.64	2.13	0.66	3.11	2.3	0	**10.16**
Numeral	582	65.81	1.72	1.03	0.17	1.03	0	**30.24**
Adverb	235	68.51	3.4	2.98	2.13	0	0	**22.98**

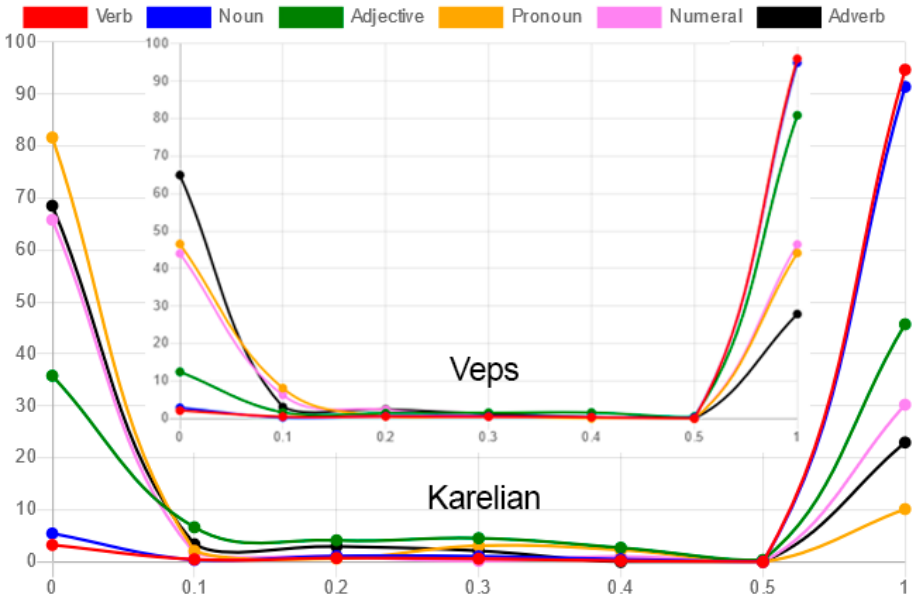


Fig. 6: Number of Vepsian and Karelian words of different parts of speech used in the experiment.

5.3 Gramset search by a suffix (GramGuess algorithm) and by a pseudo-ending (GramPseudoGuess algorithm)

73 395 "word – gramset" pairs for the Karelian Proper supradialect and 452 790 "word – gramset" pairs for the Veps language have been used in the experiments to evaluate GramGuess and GramPseudoGuess algorithms.

A list of gramsets was searched for each word. The list was ordered by the number of similar words having the same gramset.

For the evaluation of the quality of results of the searching algorithms the following function eval(g^u) has been proposed:

$$\text{eval}\left(\begin{matrix} g^u, \\ \text{Counter}\left[g^k\right] \to c^k, \\ \forall k = \overline{1,m} \end{matrix}\right) = \begin{cases} \text{The array Counter do not contain the correct gramset } g^u. \\ 0, \quad g^u \neq g^k, \forall k = \overline{1,m}, \\ \\ \text{First several gramsets in the array can have} \\ \text{the same maximum frequency } c^1\text{, one of these gramsets is } g^u. \\ 1, \quad g^u \in \{\left[g^1, \ldots, g^j\right] : c^1 = c^2 = \ldots = c^j, j \leq m\}, \\ \\ \frac{c^k}{\sum_{k=1}^{m} c^k}, \ \exists\, k : g^k = g^u, \ c^k < c^1 \end{cases} \tag{2}$$

This function (2) evaluates the results of the GramGuess and GramPseudoGuess algorithms against the correct gramset g^u.

Table 4: Evaluations of results of gramsets search for Vepsian and Karelian by GramGuess and GramPseudoGuess algorithms.

	GramGuess		GramPseudoGuess	
Evaluation	Veps	Karelian	Veps	Karelian
0	2.53	5.72	7.9	9.23
0.1	0.53	0.83	1.04	1.57
0.2	0.71	1.16	1.24	1.37
0.3	0.64	0.89	2.68	1.36
0.4	0.2	0.56	0.14	0.68
0.5	0.11	0.09	0.83	0.43
1	**95.29**	**90.74**	**86.17**	**85.36**

The table 4 shows that the GramGuess algorithm gives the better results than the GramPseudoGuess algorithm, namely:

Karelian 90.7% of Karelian words were assigned a correct gramset by GramGuess algorithm versus 85.4% by GramPseudoGuess algorithm;

Veps 95.3% of Vepsian words were assigned a correct gramset by GramGuess algorithm versus 85.4% by GramPseudoGuess algorithm.

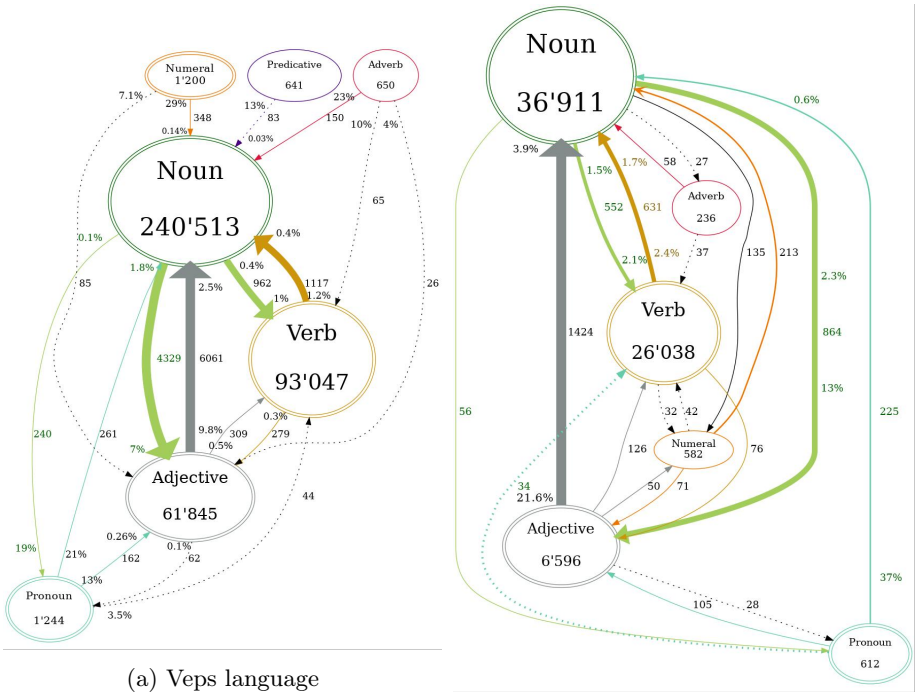
It may be suggested by the fact that suffixes are longer than pseudo-endings. In addition, the GramPseudoGuess algorithm is not suitable for a part of speech without inflectional forms.

6 Morphological analysis results

In order to analyze the algorithm errors, the results of the part-of-speech algorithm POSGuess were visualized using the Graphviz program. Part-of-speech error transition graphs were built for Veps language (Fig. 7a) and Karelian Proper supradialect (Fig. 7b).

Let us explain how these graphs were built. For example, a thick grey vertical arrow connects adjective and noun (Fig. 7b), and this arrow has labels of 21.6%, 1424 and 3.9%. This means that the POSGuess algorithm has erroneously identified 1424 Karelian adjectives as nouns. This accounted for 21.6% of all Karelian adjectives and 3.9% of nouns. This can be explained by the fact that the same lemma (in Veps and Karelian) can be both a noun and an adjective. Nouns and adjectives are inflected in the same form (paradigm).

The experiment showed that there are significantly more such lemmas (noun-adjective) for the Karelian language than for the Veps language (21.6% versus 9.8% in Fig. 7). Although in absolute numbers Veps exceeds Karelian, namely: 6061 versus 1424 errors of this kind. This is because the Veps dictionary is larger in the VepKar corpus.



(a) Veps language

(b) Karelian Proper supradialect

Fig. 7: Part-of-speech error transition graph, which reflects the results of the POSGuess algorithm.

7 Conclusion

This research devoted to the low-resource Veps and Karelian languages.

Algorithms for assigning part of speech tags to words and grammatical properties to words are presented in the article. These algorithms use our morphological dictionaries, where the lemma, part of speech and a set of grammatical features (gramset) are known for each word form.

The algorithms are based on the analogy hypothesis that words with the same suffixes are likely to have the same inflectional models, the same part of speech and gramset.

The accuracy of these algorithms were evaluated and compared. 313 thousand Vepsian and 66 thousand Karelian words were used to verify the accuracy of these algorithms. The special functions were designed to assess the quality of results of the developed algorithms.

71,091 “word – part of speech” pairs for the Karelian Proper supradialect and 399,260 “word – part of speech” pairs for the Veps language have been used in the experiments to evaluate algorithms. 86.77% of Karelian words and 92.38% of Vepsian words were assigned a correct part of speech.

73,395 "word – gramset" pairs for the Karelian Proper supradialect and 452,790 "word – gramset" pairs for the Veps language have been used in the experiments to evaluate algorithms. 90.7% of Karelian words and 95.3% of Vepsian words were assigned a correct gramset by our algorithm.

If you need only one correct answer, then all three of developed algorithms are not very useful. But in our case, the task is to get an ordered list of the parts of speech and gramsets for a word and to offer this list to an expert. Then the expert selects the correct part of speech and gramset from the list and assigns to the word. This is a semi-automatic tagging of the texts. Thus, these algorithms are useful for our corpus.

References

1. G. G. Belonogov, Yu. P. Kalinin, A. A. Khoroshilov: Computer Linguistics and Advanced Information Technologies: Theory and Practice of Building Systems for Automatic Processing of Text Information (In Russian). Russian World, Moscow (2004)
2. Hämäläinen, M.: UralicNLP: An NLP Library for Uralic Languages. Journal of open source software, **4**(37), 1345 (2019). https://doi.org/10.21105/joss.01345
3. Klyachko, E. L., Sorokin, A. A., Krizhanovskaya, N. B., Krizhanovsky, A. A., Ryazanskaya, G. M.: LowResourceEval-2019: a shared task on morphological analysis for low-resource languages. In: Conference "Dialog", 45–62. Moscow, Russia (2019). arXiv:2001.11285.
4. Moshagen, S., Rueter, J., Pirinen, T., Trosterud, T. and Tyers, F.M.: Open-source infrastructures for collaborative work on under-resourced languages. In: Collaboration and Computing for Under-Resourced Languages in the Linked Open Data Era, 71–77. Reykjavík, Iceland (2014).
5. Pirinen, T. A., Trosterud, T., Tyers, F. M., Vincze, V., Simon, E., Rueter, J.: Foreword to the Special Issue on Uralic Languages. Northern European Journal of Language Technology **4**(1), 1–9 (2016). https://doi.org/10.3384/nejlt.2000-1533.1641