ISSN 2007-9737

Highly Language-Independent Word Lemmatization Using a Machine-Learning Classifier

Iskander Akhmetov[1,2], Alexandr Pak[1], Irina Ualiyeva[1], Alexander Gelbukh[3]

[1] Institute of Information and Computational Technologies, Almaty, Kazakhstan

[2] Kazakh-British Technical University, Almaty, Kazakhstan

[3] Instituto Politécnico Nacional, CIC, Mexico City, Mexico

i.akhmetov@ipic.kz

Abstract. Lemmatization is a process of finding the base morphological form (lemma) of a word. It is an important step in many natural language processing, information retrieval, and information extraction tasks, among others. We present an open-source language-independent lemmatizer based on the Random Forest classification model. This model is a supervised machine-learning algorithm with decision trees that are constructed corresponding to the grammatical features of the language. This lemmatizer does not require any manual work for hard-coding of the rules, and at the same time it is simple and interpretable. We compare the performance of our lemmatizer with that of the UDPipe lemmatizer on twenty-two out of twenty-five languages we work on for which UDPipe has models. Our lemmatization method shows good performance on different languages from various language groups, and it is easily extensible to other languages. The source code of our lemmatizer is publicly available.

Keywords. Lemmatization, natural language processing, text preprocessing, Random Forest classifier, Decision Tree classifier.

1 Introduction

Lemmatization is an important data preparation step in many Natural language Processing (NLP) tasks such as Information Extraction (IE) and Information Retrieval (IR), among others. The aim of lemmatization is to determine the base form of a word (lemma) [11].

A number of approaches have been developed for lemmatization, ranging from those relying on rule-based techniques [19] and simple statistical-based methods [39] to the modern deep-learning methods: see, for example, the Stanford CoreNLP [27], a neural lemmatizer for Bengali [8] and for German, Czech and Arabic [24].

In our work, lemmatization is treated by building tree classification models [14], i.e., by supervised machine learning with decision trees that are constructed corresponding to the grammatical features of the language.

Researchers have faced with difficulties while lemmatizing words by different approaches. The main difficulty of a rule-based word lemmatization is that it is challenging to adjust existing rules to new classification tasks [32]. When social media texts are processed, it can be impractical to collect a predefined dictionary due to the fact that the language variation is high [22].

Concerning low-resource languages, it is hard to collect corpora and compile dictionaries for such languages [23]. Part-of-Speech (POS)-tagging, as one of the preliminary steps of lemmatization, is also difficult because some languages have up to

30 different word forms for the same normalized words [32].

Our method is a direct supervised approach of building word lemma classification. Our approach estimates the possibility of computing syntactic models using only datasets in the form of wordform–lemma dictionaries. We present an open-source[1] multilingual Random Forest Classifier-based lemmatizer that has been shown to support twenty-five languages. This lemmatizer is a continuation of our previous work [1], where we used Decision Tree Regression method. That model caused a character shift errors leading to poor accuracy; this does not happen in the lemmatizer presented in this paper because of using a classification algorithm instead of regression.

We compare our lemmatizer with UDPipe, an open-source tool for lemmatization.[2] Our evaluation shows that our classification tree-based lemmatizer achieves much better results than UDPipe does when our algorithm is provided with sufficient amount of training data.

This paper is organized as follows. We begin from a brief review of related works on lemmatization in Section 2. In Section 3, we describe a dataset, explain the method of generating vectors from the words in the dataset based on character co-occurrence matrix and TF-IDF vectorizer [31], present our approach based on Decision Tree and Random Forest Classifiers and give the steps of our lemmatization algorithm. In Section 4, we present the obtained results. Section 5 concludes the paper and outlises future extensions and possible research directions.

2 Related Work

To identify papers related with the present research, we have searched Google Scholar and Semantic Scholar. Our query terms included *language-independent word lemmatization*, *neural architectures for lemmatization*, and *machine learning for lemmatization*, among others. We arranged the resulting papers from each query by citation count and took at least top three. We considering a paper only if it introduced original ideas of a method or an algorithm.

2.1 Rule-based Approaches

Conventional algorithms for text lemmatization are based on rules. It is worth mention that rules can be expressed by the apparatus of fuzzy [13] or predicate [37] logic. The logical rules applied to finite-state transducers, with the help of a lexicon, define morphotactic and orthographic alternations.

As a result, a system based on such rules can solve several tasks, such as stemming, lemmatization, and full morphological analysis [2, 10]. The advantages of such an approach include transparency of the algorithm's outcome and the possibility of fine-tuning.

However, there are also disadvantages, such as the so-called problem of out-of-vocabulary (OOV) words, that leads to the need of intensive manual support for the vocabulary of many thousands of words.

In addition, there exist approaches that automatically generate rules from the dataset of pairs of the word and its normal form. For instance, [25] with the help of a decision tree predicted particular letters of the transformed word based on the letters in the form of the past tense.

Another approach relies on relational learning with decision lists applied to English verbs in the past tense [30].

2.2 Statistical Approaches

Various approaches to NLP have been influenced by ideas from statistics methods, such as Hidden Markov Model (HMM) and Conditional Random Fields (CRF), among others.

Researchers adopted HMM for POS tagging and approximation of language model for speech recognition systems. These methods have difficulties in estimating transitional probabilities on a small amount of data.

Besides, for good accuracy performance of such methods, there is a need for the large

[1] The source code of our lemmatizer is publicly available on `https://github.com/iskander-akhmetov/Highly-Language-Independent-Word-Lemmatization-Using-a-Machine-Learning-Classifier`.

[2] `http://ufal.mff.cuni.cz/udpipe`

Table 1. List of 25 languages we used

Language	Code	Language group	Word pairs	Source
Asturian	ast	Romance	108,792	Lemmatization lists
Bulgarian	bg	Slavic/Baltic	30,323	Lemmatization lists
Catalan	ca	Romance	591,534	Lemmatization lists
Czech	cs	Slavic/Baltic	36,400	Lemmatization lists
English	en	Germanic	41,649	Lemmatization lists
Estonian	et	Ural/Altaic	80,536	Lemmatization lists
Farsi	fa	Iranian	6,273	Lemmatization lists
French	fr	Romance	223,999	Lemmatization lists
Galician	gl	Romance	392,856	Lemmatization lists
German	de	Germanic	358,473	Lemmatization lists
Hungarian	hu	Ural/Altaic	39,898	Lemmatization lists
Irish	ga	Gaelic	415,502	Lemmatization lists
Italian	it	Romance	341,074	Lemmatization lists
Manx Gaelic	gv	Gaelic	67,177	Lemmatization lists
Portuguese	pt	Romance	850,264	Lemmatization lists
Romanian	ro	Romance	314,810	Lemmatization lists
Russian	ru	Slavic/Baltic	2,657,468	Zaliznjak dictionary
Scottish Gaelic	gd	Gaelic	51,624	Lemmatization lists
Slovak	sk	Slavic/Baltic	858,414	Lemmatization lists
Slovenian	sl	Slavic/Baltic	99,063	Lemmatization lists
Spanish	es	Romance	496,591	Lemmatization lists
Swedish	sv	Germanic	675,137	Lemmatization lists
Turkish	tr	Ural/Altaic	1,337,898	Zargan dictionary
Ukrainian	uk	Slavic/Baltic	193,704	Lemmatization lists
Welsh	cy	Gaelic	359,224	Lemmatization lists

manually annotated corpora to approximate the probabilities [16, 12, 4].

2.3 Neural Approaches

Nowadays, neural approaches are prevailing over a great variety of algorithms in the task of text lemmatization. The advantage of artificial neural networks can be explained by the simplicity of development, the possibility of multi-task learning, and application in multi-criterial optimization.

Conventional language models can be easily presented in terms of a universal neural estimator.

The most popular idea in this field is the sequence-to-sequence model (S2S), which can be used for contextual lemmatization. The main idea behind the S2S model is the attention mechanism, which leads to good accuracy performance and to reducing the number of parameters to be computed [28].

3 Methods and Data

3.1 Datasets

For this research, we used Lemmatization lists [29] for 23 languages publicly available under the Open Data Base License (ODbL);[3] see Table 1.

Additionally, for Russian language we used Zaliznjak's dictionary [41] and for Turkish we used Zargan dictionary [18].

The language group representation of our data is unbalanced, with the majority of languages being Romance and Slavic / Baltic, followed by the Gaelic and Germanic languages. The distribution of the

[3] https://opendatacommons.org/licenses/odbl/1-0/

ISSN 2007-9737

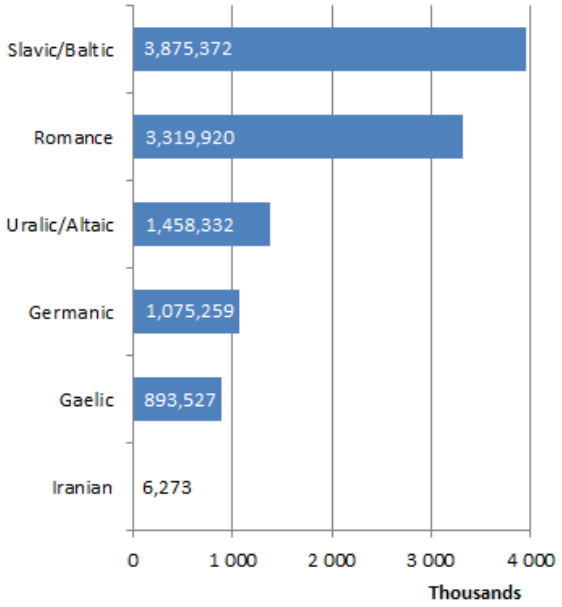

Fig. 1. Language groups representation in the data we used: total word pairs

Table 2. Number of words by language group

Language groups	Total number of words
Slavic/Baltic	3,875,372
Romance	3,319,920
Ural/Altaic	1,458,332
Germanic	1,075,259
Gaelic	893,527
Iranian	6,273
Total	10 628 683

data we have collected by the number of words for a language group is presented in Fig. 1 and Table 2.

We can observe that Uralic / Altaic group, represented by only two languages, is greater than such groups as Germanic and Gaelic by the number of wordform–lemma pairs.

This is because of enormous Turkish language data. Same effect can be observed for Slavic / Baltic language group, mainly because of Russian language data.

3.2 Method

3.2.1 Character Co-occurence Embeddings

For converting words in wordform–lemma pairs to vectors, we used the following method for building the character co-occurrence matrix.

Calculating TF-IDF All the words were converted to vectors at a character level by the TF-IDF vectorizer in scikit-learn implementation of the method based on the works [26] and [21]:

1. Calculate the term frequency (in our case, character frequency cf) as

$$cf(c, w) = f_{c,w}/len(w), \tag{1}$$

where c stands for a character, w for a word and $len(w)$ for the length of the word in characters.

2. Calculate the inverse document frequency (inverse word frequency iwf, in our case) as

$$iwf(c, W) = \log \frac{N}{|\{ w \in W : c \in w \}|}, \tag{2}$$

where N is the total number of words in the corpus, and $|\{ w \in W \mid c \in w \}|$ is the number of words where the character c appears $(cf(c, w) \neq 0)$.

3. Calculate the term frequency-inverse document frequency (character frequency-inverse word frequency, in our case) as

$$cfiwf(c, w, W) = cf(c, w) \times iwf(c, W). \tag{3}$$

4. As a result we get a sparse matrix:

$$mx = (n_w, n_c), \tag{4}$$

where the size of the matrix is defined by the number n_w of words in the corpus and the number n_c of unique characters found in all of the words in the corpus.

Calculating the Co-occurrence Matrix We multiplied the transpose of the matrix mx by the matrix itself to find the cooccurence matrix as

$$cooc\_mx = mx^T \times mx, \quad (5)$$

which yields a matrix of size (nc, nc), every row or column of which is serving as the embedding for a corresponding character.

The character co-occurrence embeddings can store the character semantic distribution information [18] in the word context for a given language, reflecting the phonetic patterns and their similarity [34].

3.2.2 Decision Tree Classifier

For lemmatization, we used the Decision Tree Classification as the base, extending it to an ensemble method called Random Forest Classifier as explained below.

Our selection of this classifier was based on the fact that only K-Nearest Neighbors Classifier [3], Radius Neighbors Classifier and tree algorithms support multiclass-multioutput [31] or multitask classification. However, the first two algorithms require reducing the number of features used to less than ten and have a complexity of $O(N^2)$ or $O(N \times log(N))$, whereas tree algorithms do not require dimensionality reduction and have a complexity of $O(n_{samples} n_{features} \log(n_{samples}))$. The reason behind the selection of Random Forest technique out of tree algorithms is explained in Section 4.1.

The Decision Tree method is well known from ancient times [5]. It was first formalized by Hoveland and Hunt in late 1950s and further elaborated in [36]. The classifier builds a tree starting from the root question: the feature that separates the elements into two groups according to a criterion (Gini coefficient, entropy, or variance) [15, 35, 33]; in our case, entropy or Information Gain criterion was used, such that each group contains similar elements. The process continues iteratively for each group until a stopping criterion is met, which can be:

— the specified depth of the tree is achieved,

— all the items on a leaf are of the same class or one item is left on a leaf,

— more than N elements are left on a leaf, or

— further branching does not enhance the homogeneity of items on a leaf beyond certain value.

In our case, we go for multiclass classification and the data is represented in the form of

$$(x, Y) = (x_1, x_2, x_3, \ldots x_k, Y_1, Y_2, Y_3, \ldots Y_m), \quad (6)$$

where x_{1-k} are the independent variables associated with the features and Y_{1-m} are the dependent variables or targets. The information gain (IG) criterion is based on the concept of entropy heavily used by physicists in thermodynamics [9] and introduced for information by Shannon [35]. It is defined as follows:

$$H(T) = I_e(p_1, p_2, \ldots, p_j) = \sum_{i=1}^{j} p_i \log_2 p_i, \quad (7)$$

where $p_1, p_2, ..., p_j$ are fractions that sum up to 1 and show the share of each class presence in the child node that results from a split in the tree [40]. So, the formal criterion can be calculated as

$$IG(T, a) = H(T) - H(T \mid a), \quad (8)$$

where $H(T)$ is the entropy of the parent node and $H(T \mid a)$ entropy of a child.

3.2.3 Random Forest Classifier Method

To counter some of the disadvantages of the Decision Tree Classifier, which include easy overfitting and non-robustness [20], we exploited the Random Forest ensemble technique [6]. Here, the method implies using a random subset from the training set with replacements; the most discriminative thresholds are drawn at random for each subset and the best of these randomly-generated thresholds is picked as the splitting rule (thus we employed a heuristic methodology similar to Variable Neighborhood Search [17]). Despite relatively low classification power of each individual tree in the forest,

the cumulative classification power is increased through averaging (by canceling out the errors) and voting processes [31]. This usually leads to the reduction of the model variance, at the expense of a modest increase in the bias.

The scikit-learn [31] implementation of the Random Forest allows for bootstrapping, using a random subset of the dataset for estimator instance training leading to a leaner and more robust model, and using the parallelization in computations to increase the effectiveness of the training process. The module also uses averaging of the estimators probabilistic predictions [31], contrary to the original paper's method of each classifier voting for a single class [6].

3.2.4 Lemmatization Algorithm

These steps we used for each language can be described as follows:

— given the dictionary of wordform–lemma pairs, assign them as independent (X) and dependent (Y) variables for applying the machine learning approach;

— prepare the character co-occurrence matrix where each row or column will serve as an embedding vector for the corresponding symbol;

— encode the words in X by the character embeddings, producing the vectors of length of the longest word in the corpus and flattening it;

— encode words in Y by the character ordinal number, to carry out multiclass classification task;

— split the dataset $90/10\%$ for training and testing;

— train the Random Forest Classifier model employing the bootstrapping technique, 10 estimators and using entropy as a criterion;

— test the model.

We compare our lemmatization algorithm with the UDPipe system as a baseline. The baseline UDPipe system [38] is an updated version of the UDPipe. Both UDPipe versions have a lemmatizer based on the edit-tree classification method.

We use UDPipe Future as one of the top performing entries in the lemmatization evaluation. Its performance in the CoNLL 2018 UD Shared task was ranked 1st, 3rd and 3rd in the three official metrics: MLAS, LAS and BLEX, respectively.

Table 3. Test sample accuracy score of models per language

Language	ExtraTrees	ExtraTree	Decision Tree	Random Forest
Manx Gaelic	**0.39**	0.33	**0.39**	**0.39**
Farsi	**0.40**	0.28	0.31	0.38
Scottish Gaelic	**0.47**	0.38	0.44	0.45
Estonian	0.45	0.36	0.41	**0.47**
Czech	0.46	0.40	0.44	**0.48**
Bulgarian	0.48	0.42	0.45	**0.50**
English	**0.50**	0.31	0.40	0.48
Hungarian	0.50	0.44	0.46	**0.51**
Asturian	**0.71**	0.63	0.66	**0.71**
Irish	0.73	0.66	**0.75**	0.73
Slovenian	**0.74**	0.67	0.70	**0.74**
German	**0.76**	0.68	0.69	0.74
Romanian	0.78	0.68	0.77	**0.79**
Russian	**0.79**	0.75	0.77	**0.79**
French	**0.81**	0.74	0.77	**0.81**
Portuguese	0.86	0.81	0.84	**0.87**
Spanish	**0.87**	0.80	0.84	**0.87**
Welsh	0.87	0.85	0.87	**0.88**
Galician	**0.89**	0.83	0.86	**0.89**
Catalan	**0.89**	0.86	0.87	**0.89**
Swedish	**0.89**	0.81	0.84	0.88
Ukrainian	0.90	0.85	0.88	**0.91**
Italian	**0.91**	0.86	0.88	**0.91**
Slovak	**0.92**	0.85	0.89	**0.92**
Turkish	0.95	0.90	**0.96**	**0.96**
Average	**0.72**	0.65	0.69	**0.72**

Table 4. Weighted average test sample accuracy score of algorithms by the number of words in data-sets

Language	Num. words	Weight	ExtraTrees	ExtraTree	DecisionTree	RandomForest
Manx Gaelic	67,177	0.6%	0.0025	0.0021	0.0025	0.0025
Farsi	6,273	0.1%	0.0002	0.0002	0.0002	0.0002
Scottish Gaelic	51,624	0.5%	0.0023	0.0019	0.0021	0.0022
Estonian	80,536	0.8%	0.0034	0.0027	0.0031	0.0035
Czech	36,400	0.3%	0.0016	0.0014	0.0015	0.0016
Bulgarian	30,323	0.3%	0.0014	0.0012	0.0013	0.0014
English	41,649	0.4%	0.0020	0.0012	0.0016	0.0019
Hungarian	39,898	0.4%	0.0019	0.0017	0.0017	0.0019
Asturian	108,792	1.0%	0.0072	0.0064	0.0068	0.0072
Irish	415,502	3.9%	0.0286	0.0258	0.0294	0.0284
Slovenian	99,063	0.9%	0.0069	0.0062	0.0065	0.0069
German	358,473	3.4%	0.0257	0.0229	0.0233	0.0250
Romanian	314,810	3.0%	0.0230	0.0200	0.0228	0.0233
Russian	2,657,468	25.0%	0.1982	0.1867	0.1932	0.1987
French	223,999	2.1%	0.0171	0.0156	0.0163	0.0171
Portuguese	850,264	8.0%	0.0690	0.0644	0.0669	0.0693
Spanish	496,591	4.7%	0.0405	0.0373	0.0392	0.0405
Welsh	359,224	3.4%	0.0295	0.0286	0.0295	0.0297
Galician	392,856	3.7%	0.0328	0.0306	0.0316	0.0329
Catalan	591,534	5.6%	0.0494	0.0476	0.0484	0.0495
Swedish	675,137	6.4%	0.0567	0.0518	0.0536	0.0562
Ukrainian	193,704	1.8%	0.0165	0.0156	0.0160	0.0165
Italian	341,074	3.2%	0.0291	0.0275	0.0284	0.0291
Slovak	858,414	8.1%	0.0742	0.0684	0.0722	0.0743
Turkish	1,337,898	12.6%	0.1201	0.1135	0.1203	0.1207
Total	10 628 683	100.0%	0.8396	0.7813	0.8183	0.8405

4 Results

4.1 Model Selection

For selecting the best model, we compared four tree classifier algorithms: Extra Trees [14], Extra Tree, Decision Tree [7] and Random Forest [6].

As was already mentioned, we compare only tree classification algorithms, because only these algorithms and the K-Nearest Neighbors with Radius Neighbors algorithms are compatible with multiclass-multioutput tasks in the Python sklearn module implementation [31]. However, the K-Nearest Neighbors algorithms requires feature dimensionality reduction and significant amount of time to test on large datasets, so we omitted them.

As Table 3 shows, Random Forest Algorithm holds the majority of the leading testset accuracy results, and followed immediately by the Extra Trees algorithm. On average, both algorithms result in the same 0.72 accuracy score.

To make an informed choice between the two algorithms, we calculated the weighted average testset accuracy score, weighing by the number of words available for each language.

Table 4 shows that the leader is Random Forest with its weighted average test sample accuracy score of 0.8405, leaving the Extra Trees algorithm behind with its 0.8396 score.

To measure the affinity of the compared algorithms for each language, we calculated the correlation coefficient between the results of each pair of algorithms for test sample accuracy scores

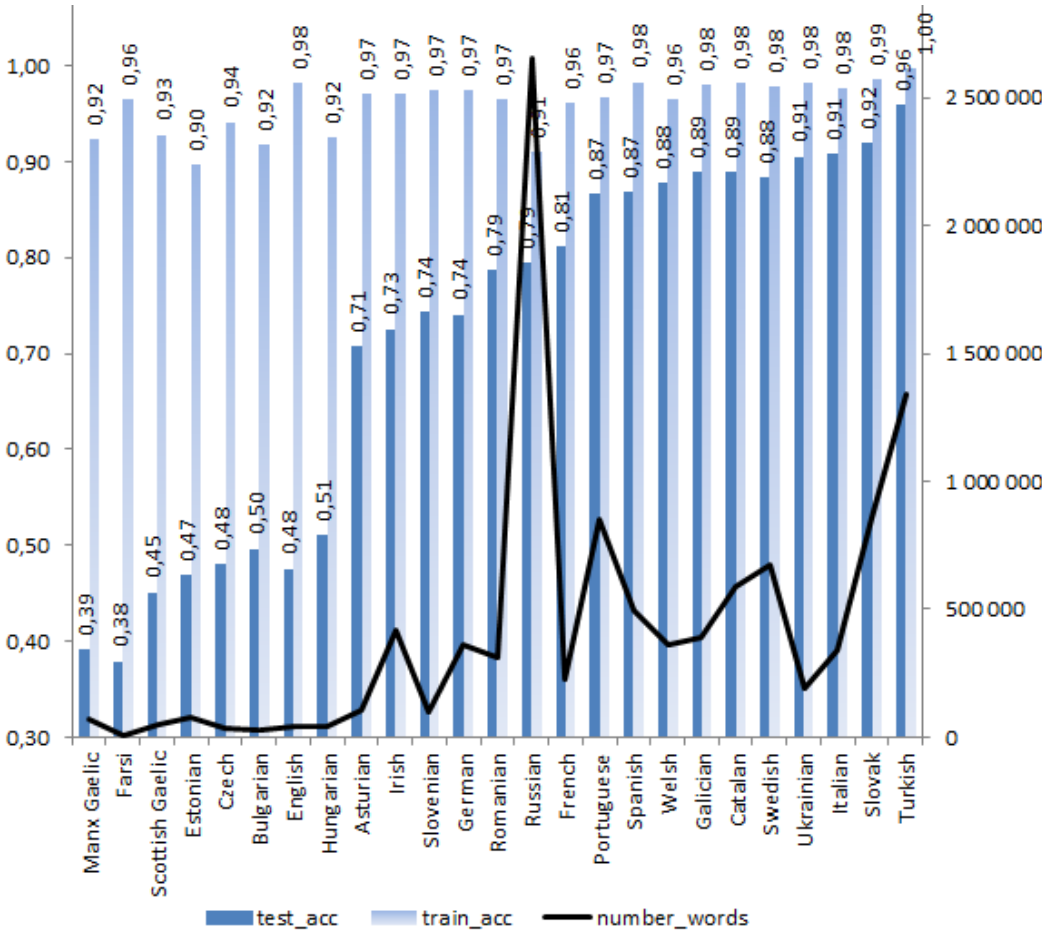

Fig. 2. Train / test accuracy and dataset volume

by language as

$$Correl(X,Y) = \frac{\sum(x-\bar{x}) \times (y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2 \times \sum(y-\bar{y})^2}}, \quad (9)$$

where $\bar{x}$ and $\bar{y}$ are the mean values of the arrays of the compared algorithms' results.

As can be seen from Table 5, the results yielded by each individual algorithm we compared are highly correlated between each other, so there is no any obvious preference of the algorithms for any specific language.

4.2 Experiments

The results we have obtained using the Random Forest classifier models for 25 languages are shown in Fig. 2. As can be seen from the figure, there is a clear dependency of test accuracy from the volume of the dataset the model was trained on, thus Manx Gaelic, Farsi, Scottish Gaelic, Estonian, Czech, Bulgarian, English and Hungarian languages score the lowest for the test accuracy (from 0.39 to 0.51) and at the same time for these languages we had less data available.

We can also observe that despite the largest dataset available, we obtained relatively low test accuracy score on Russian language (more than 2.3 million word pairs, and test accuracy of 0.79), which we can attribute to the grammar complexity of the Russian language. Another fact is that in the top five languages by test accuracy, there are

Table 5. Correlation coefficient between test sample accuracy scores of algorithms

	ExtraTrees	**ExtraTree**	**DecisionTree**	**RandomForest**
ExtraTrees	1	0.9905	0.9921	0.9983
ExtraTree		1	0.9941	0.9935
DecisionTree			1	0.9947
RandomForest				1

Table 6. Correlation matrix between accuracy scores and language features

	Test accuracy	**Train accuracy**	**Max word length**	**Num of letters**	**Word pairs number**
Test accuracy	1	**0.64**	**0.28**	-0.03	**0.49**
Train accuracy		1	0.05	0.10	-0.01
Max word length			1	0.11	0.21
Number of letters				1	-0.28
Word pairs number					1

two Slavic languages, namely Ukrainian (193 704 word pairs, test accuracy 0.91) and Slovak (858 414 word pairs, test accuracy 0.92), sharing the group with Czech (low amount of data: 36,400 word pairs, test accuracy score 0.48) and Russian (complex grammar) that they overrun significantly.

Considering the Turkish language as having the best result for test accuracy score (0.96) and having substantial amount of data (1.3 million word), we must note that the dictionary we had for this language was essentially wordform–stem dictionary, and that is why we can disqualify it but making the point that our algorithm might be exceptionally good for stemming tasks. It is also worth mentioning that our baseline, UDPipe, when used for Turkish language, instead of lemma gives the stem.

Another factor that might be affecting the test accuracy we have is the maximum length of the word for a given language. In Table 6, one can observe the correlation coefficient (9) of 0.28 between this length and test accuracy score. The maximum length of the word in a language may indicate the presence of a set of grammar rules regulating the construction of words, and these rules can be generalized by the Random Forest Classifier Algorithm if they do not have many exceptions.

Again, for this explanation we have a contradicting Romanian language which has relatively large dataset of more than 300 thousand word pairs, longest word of 53 letters (longer than the Turkish language longest word of 50 letters) and scoring only 0.79 for test accuracy, neighboring with the Russian language on the scale. Other Romance group languages such as Italian, Catalan, Galician, Spanish, Portuguese and French scored on the range of 0.81 to 0.91 for test accuracy, and we can conclude that Romance languages are positively responsive to our lemmatization method except for the Romanian language.

4.3 Comparison with UDPipe

Comparing the test samples scoring with lemmatization results over the UDPipe API yielded the results shown in Fig. 3.

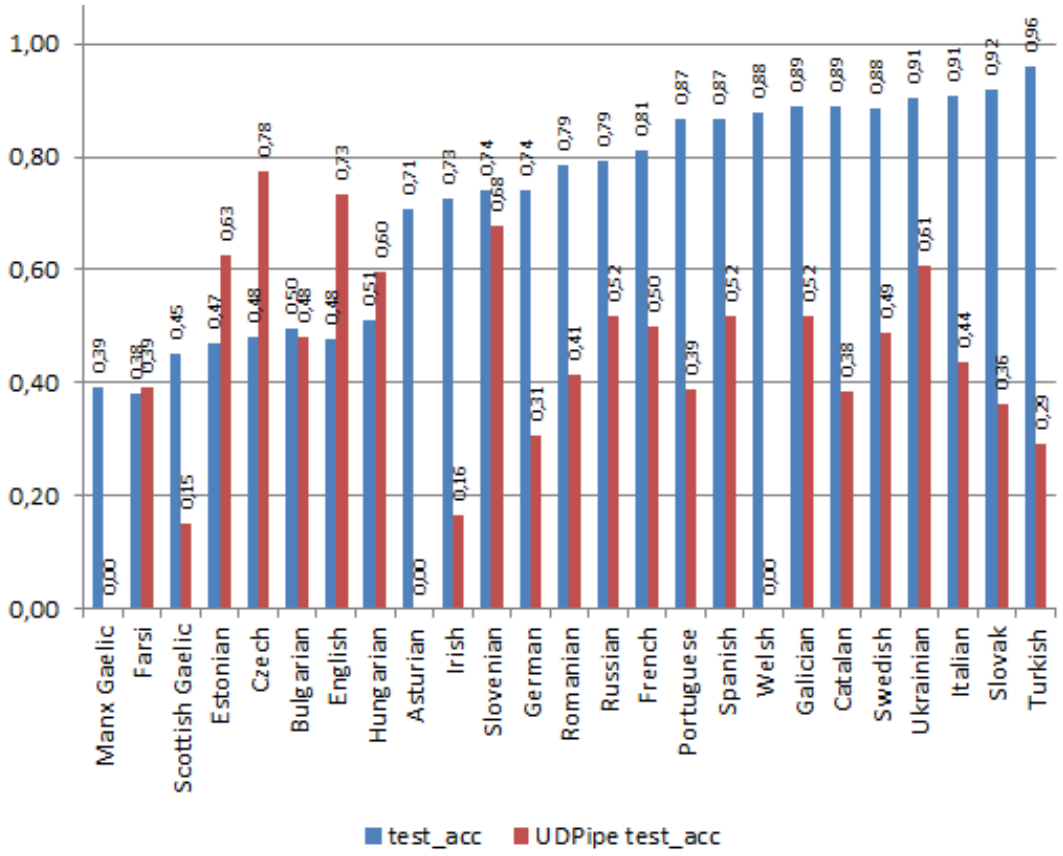

Fig. 3. Test accuracy comparison of our lemmatizer with UDPipe

UDPipe has no models for Manx Gaelic, Asturian and Welsh languages, so we were able to make comparison only on the rest of the languages.

Our lemmatizer outperformed the UDPipe models on all languages except for Farsi, Estonian, English and Hungarian. The languages on which our lemmatizer performed badly supposedly had insufficient training data (see Figure 3), which can be fixed in the future.

5 Conclusion and Future Work

The lemmatization method presented in this paper showed good potential for the use on different languages from different language groups, and is worth further development on larger datasets of the tested languages, as well as Asian and African languages.

For future work, we plan exploring the feature importance for different languages, such as what parts of the word are deemed more significant in a language for inducing a words normal form.

In addition, building and possible interpreting the decision tree diagrams built for each language by the algorithm can be a very important step towards improving the accuracy of the algorithm.

Acknowledgments

This research is conducted within the framework of the grant No. AP05132760 “Development of methods for deep learning of semantic probability inference.”

References

1. **Akhmetov, I., Krassovitsky, A., Ualiyeva, I., Mussabayev, R., & Gelbukh, A. (2020).** Lemmatization of russian language by tree regression models. *Research in Computing Science*.
2. **Altintas, K. & Cicekli, I. (2001).** A morphological analyser for Crimean Tatar. *Proceedings of the 10th Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN'2001)*, pp. 180–189.
3. **Altman, N. S. (1992).** An introduction to kernel and nearest-neighbor non-parametric regression. *The American Statistician*, Vol. 46, pp. 175–185.
4. **Banko, M. & Moore, R. C. (2004).** Part-of-speech tagging in context. *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, ACL, pp. 556–561.
5. **Barnes, J. (2003).** *Porphyry: Introduction*. Oxford University Press UK.
6. **Breiman, L. (2001).** Random forests. *Machine Learning*, Vol. 45, pp. 5–32.
7. **Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984).** *Classification and Regression Trees*. Chapman and Hall/CRC.
8. **Chakrabarty, A., Chaturvedi, A., & Garain, U. (2019).** CNN-based context sensitive lemmatization. *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, ACM, pp. 334–337.
9. **Clausius, R. (1879).** *The mechanical theory of heat*. Macmillan.
10. **Cöltekin, C. (2010).** A freely available morphological analyzer for Turkish. *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, volume 2, European Language Resources Association (ELRA), pp. 19–28.
11. **Dave, R. & Balani, P. (2015).** Survey paper of different lemmatization approaches. *International Journal of Research in Advent Technology, ICATEST 2015*, Vol. 8, pp. 366–370.
12. **Diab, M., Hacioglu, K., & Jurafsky, D. (2004).** Automatic tagging of Arabic text: From raw text to base phrase chunks. *Proceedings of HLT-NAACL 2004: Short papers*, ACL, pp. 149–152.
13. **Gashkov, A. & Eltsova, M. (2018).** Lemmatization with reversed dictionary and fuzzy sets. *SHS Web of Conferences*, volume 55, EDP Sciences, pp. 04007.
14. **Geurts, P., Ernst, D., & Wehenkel, L. (2006).** Extremely randomized trees. *Machine Learning*, Vol. 63, No. 1, pp. 3–42.
15. **Gini, C. (1921).** Measurement of inequality of incomes. *The economic journal*, Vol. 31, No. 121, pp. 124–126.
16. **Habash, N. & Rambow, O. (2005).** Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. *Proceedings of the 43rd annual meeting of the association for computational linguistics (ACL'05)*, ACL, pp. 573–580.
17. **Hansen, P., Mladenović, N., Brimberg, J., & Pérez, J. A. M. (2019).** Variable neighborhood search. In *Handbook of metaheuristics*. Springer, pp. 57–97.
18. **Harris, Z. S. (1954).** Distributional structure. *Word*, Vol. 10, No. 2-3, pp. 146–162.
19. **Ingólfsdóttir, S. L., Loftsson, H., Daðason, J. F., & Bjarnadóttir, K. (2019).** Nefnir: A high accuracy lemmatizer for Icelandic. *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, Linköping University Electronic Press, pp. 310–315.
20. **James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013).** *An introduction to statistical learning*, volume 112. Springer.
21. **Jones, K. S. (1972).** A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, Vol. 28, No. 1, pp. 11–21.
22. **Kanerva, J., Ginter, F., & Salakoski, T. (2019).** Universal lemmatizer: A sequence to sequence model for lemmatizing universal dependencies treebanks. *CoRR*, Vol. abs/1902.00972.
23. **King, B. (2015).** *Practical Natural Language Processing for Low-Resource Languages*. Ph.D. thesis, University of Michigan.
24. **Kondratyuk, D., Gavenčiak, T., Straka, M., & Hajič, J. (2018).** LemmaTag: Jointly tagging and lemmatizing for morphologically rich languages with BRNNs. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, ACL, pp. 4921–4928.
25. **Ling, C. X. (1994).** Learning the past tense of English verbs: The symbolic pattern associator vs. connectionist models. *J. Artif. Intell. Res.*, Vol. 1, pp. 209–229.
26. **Luhn, H. P. (1957).** A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, Vol. 1, No. 4, pp. 309–317.

27. **Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., & McClosky, D. (2014).** The Stanford CoreNLP natural language processing toolkit. *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, ACL, pp. 55–60.
28. **McCarthy, A. D., Vylomova, E., Wu, S., Malaviya, C., Wolf-Sonkin, L., Nicolai, G., Kirov, C., Silfverberg, M., Mielke, S. J., Heinz, J., Cotterell, R., & Hulden, M. (2019).** The SIGMORPHON 2019 shared task: Morphological analysis in context and cross-lingual transfer for inflection. *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, ACL, pp. 229–244.
29. **Měchura, M. (2018).** Lemmatization-lists. `https://github.com/michmech/lemmatization-lists`. [Accessed: February 15, 2020].
30. **Mooney, R. J. & Califf, M. E. (1995).** Induction of first-order decision lists: Results on learning the past tense of English verbs. *J. Artif. Intell. Res.*, Vol. 3, pp. 1–24.
31. **Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011).** Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, Vol. 12, pp. 2825–2830.
32. **Plisson, J., Lavrac, N., & Mladenić, D. (2004).** A rule based approach to word lemmatization. *Proceedings of IS04*, pp. .
33. **Rutemiller, H. C. & Bowers, D. A. (1968).** Estimation in a heteroscedastic regression model. *Journal of the American Statistical Association*, Vol. 63, No. 322, pp. 552–557.
34. **Sahlgren, M. (2008).** The distributional hypothesis. *Italian Journal of Disability Studies*, Vol. 20, pp. 33–53.
35. **Shannon, C. E. (1948).** A mathematical theory of communication. *The Bell System Technical Journal*, Vol. 27, No. 3, pp. 379–423.
36. **Simon, H. A. (1967).** Experiments in induction. *The American Journal of Psychology*, Vol. 80, No. 4, pp. 651–653.
37. **Stanković, R., Krstev, C., Obradović, I., Lazić, B., & Trtovac, A. (2016).** Rule-based automatic multi-word term extraction and lemmatization. *Proceedings of the 10th International Conference on Language Resources and Evaluation, LREC 2016*, pp. 507–514.
38. **Straka, M. (2018).** UDPipe 2.0 prototype at CoNLL 2018 UD shared task. *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, Association for Computational Linguistics, Brussels, Belgium, pp. 197–207.
39. **Thomas, M., Ryan, C., Alexander, F., & Hinrich, S. (2015).** Joint lemmatization and morphological tagging with LEMMING. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, ACL, pp. 2268–2274.
40. **Witten, I. & Eibe, F. (1999).** *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.
41. **Zaliznjak, A. A. (2014).** Open grammar dictionary of the Russian language. `http://odict.ru`. Accessed: 2020-07-20.

Article received on 12/06/2020; accepted on 29/08/2020.
Corresponding author is Iskander Akhmetov.