NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS

Ilia Afanasev

A HYBRID LEMMATISER FOR OLD CHURCH SLAVONIC

BASIC RESEARCH PROGRAM

WORKING PAPERS

SERIES: LINGUISTICS
WP BRP 106/LNG/2021

Ilia Afanasev[1]

A HYBRID LEMMATISER FOR OLD CHURCH SLAVONIC

The article considers a lemmatiser that is developed specifically for Old Church Slavonic (OCS). The introduction underlines the problem of the lack of lemmatisers that might deal with different datasets of the OCS. The review gives a short description of previous attempts and current trends in lemmatisation. The lemmatiser is hybrid-based and uses the advantages of linguistic rules for specific cases (fragmentary tokens, punctuation, or digits), a dictionary for the most common tokens, and a sequence-to-sequence (seq2seq) neural network with an attention mechanism for the rest of material. The model achieves an 85% overall accuracy score, which is lower than one of the previous models for the Universal Dependencies(UD) dataset. However, when specific tokens are taken into consideration, the model outperforms the previous ones with the help of its rule-based part. Possible further directions of the research include the use of more sophisticated architectures, such as BART.

JEL Classification: Z.

Keywords: lemmatisation, Old Church Slavonic, hybrid approach, natural language processing, seq2seq.

[1]1 Federal State Budgetary Educational Institution of Higher Education "Saint-Petersburg State University". Philological Faculty, Department of Slavic Studies. Graduate student; E-mail: szrnamerg@gmail.com

Introduction

Lemmatisation is the transformation of a word into its lemma or base form. It is similar to stemming, but stemming is more naive as an algorithm, generally only defining the first *x* common letters of the set of word forms, while lemmatisation provides the common word form for a given set of word forms. For instance, there is no possible way to stem Old Church Slavonic (OCS) *мене* 'I-GEN' from *азъ* 'I-NOM', however, the lemma for both of this forms is *азъ* 'I'.

Lemmatisation is widely used in linguistics. It helps to make dictionaries out of corpora, and is a natural way of making dictionaries in general. It is also a stage in corpus design, the results of which are helpful when working with the finished corpus. Lemmatisation is helpful in entity recognition and text summarising. It may be used as an additional way of tagging corpora [Hardie et al., 2014] [Schryver and Nabierye, 2018] [Camps et al., 2020]. Lemmatised corpora are easier to search through and analyse.

Automatic lemmatisation as a complex NLP task may be defined differently, depending on the particular approach that a researcher uses. The first definition is based on the older approach presented in works that treat lemmatisation as a three-step hybrid task [Mills, 1998] [Chrupała, 2006] [Plisson et al., 2008], [Gesmundo and Samardžić, 2012] [Radziszewski, 2013]. First, a researcher should describe all the lemmatisation rules applying to the words of a particular language. Second, a specific tool is trained for a multi-class classification task. This task matches a lemmatisation model and a given word given. The tool that performs this task might differ from gradient boosting algorithms, such as XGBoost [Tianqi and Carlos, 2016], or statistical methods, for instance, Hidden Markov Model (HMMs) [Jiampojarman et al., 2008]. Third, when the rule is determined, it is applied to the word, and the lemma is acquired. Thus, automatic lemmatisation with this approach may be defined as a learning task of determining of a lemmatising rule on the basis of a given word, and using it to acquire the lemma of the given word.

The second definition relies on the newer approach that appeared during the last decade in which lemmatisation is made in one step [Kanerva et al., 2020]. A tool takes a given word and auxiliary information, such as the POS tag, or morphological data, or left context, and produces a lemma. This is usually achieved via decoding input information, transforming the acquired tensor, and encoding it into the output information. The choice of tools is more limited when one chooses this approach. Encoder-decoder models are usually made with advanced neural networks [Ljubešić and Dobrovoljc, 2019]. The preferred architecture is usually a sequence-to-sequence (seq2seq) model, which takes a word and optional auxiliary information as input, and outputs a lemma [Bergmanis and Goldwater, 2018]. In this sense, automatic lemmatisation may be defined as a neural transformation of a sequence that consists of a word and linguistic data about this word into a sequence that consists of a lemma.

This article investigates how seq2seq neural networks, enhanced by an attention mechanism, are able to deal with lemmatisation and how necessary it is to enhance a model, consisting only of a neural network, with a set of specifically designed linguistic rules.

The system that is the subject of this article is a hybrid lemmatiser for the corpus of the Old Church Slavonic (OCS). A hybrid system here means one built from three different modules, and each module is designed to deal with a certain class of tokens to be lemmatised. Rules define the process of lemmatisation specifically for fragments, digits, and punctuation marks. Dictionaries are used to deal with previously encountered words. A neural network processes the rest of the words.

The neural network is based on seq2seq architecture with an attention mechanism [Sutskever et al., 2014] [Cho et al., 2014]. This architecture is based on the encoder-decoder pair of models, with the

encoder taking a sequence of language and transforming it into a vector, and the decoder forming sequences from learned vectors. Sequences are generated symbol by symbol, and thus the left and right context are important. The attention mechanism, an additional neural network, helps to better extract information from the context. The neural network was implemented with the Keras API (Keras). The representation of the similar system workflow is demonstrated in the figure 1.

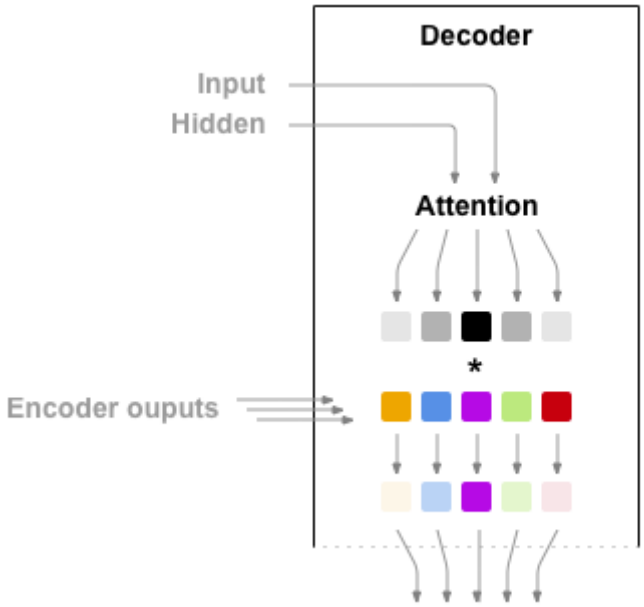

Fig. 1. One of the possible implementations of seq2seq architecture.

The original purpose of this network was to perform machine translation. However, with the emergence of Universal Dependency (UD) lemmatisers [Straka et al., 2017] [Bergmanis and Goldwater, 2018] [Kanerva et al., 2018], it was proved that it could also be used for lemmatisation [Metheniti et al., 2020], e.g. in the case of *Lematus* [Bergmanis and Goldwater, 2018].

There are two proposed solutions for enhancing the efficiency of the seq2seq-based models. The first one uses more morphological information from the dataset provided and the left context. This may be useful when testing part of the UD dataset, but in the OCS corpus the only previous tagging is POS. Thus, the whole purpose of preparing the lemmatiser for the corpus is defeated. The second way to improve the results is utilising more sophisticated models used for machine translation [Lewis et al., 2020]. This is the next step of the research.

The rest of the work is organised as follows. Previous attempts to build a lemmatiser are reviewed. The model architecture and possible alternates are presented. The datasets, the UD and OCS corpora, are characterised. The experimental settings are provided, the results are given, compared to the previous models, and discussed. The outline for the further development of the system concludes the research.

Previous Work

Lemmatisation has been of interest in NLP for the last few decades [Hann, 1974]. Since then, tools for lemmatisation have been divided into universal lemmatisers [Straka et al., 2017] [Bergmanis and Goldwater, 2018] [Kanerva et al., 2020] and specific lemmatisers designed to execute a particular task, for instance, for a particular language [Džeroski and Erjavec, 2001] [Groenewald, 2007] [Tamburini, 2013] or for a particular POS [Prinsloo, 2012] [Gouws and Prinsloo, 2012] [Nthambeleni and Musehane, 2014], or a group of words within a POS [Fernández, 2020], or a class of

words with a very specific behaviour, such as words within fixed expressions [Farkas et al., 2008] [Mulhall, 2008] [Kosch, 2016]. One approach unites both lemmatiser and tagger in a single model [Spyns, 1996] [Aduriz et al., 1998]. Some lemmatisers, developed for a single language, may be fine-tuned to be used for the other ones [Groenewald, 2009].

Automatic lemmatisation was initially based mostly on the linguistic rule-based method [Evans, 2006] [Jursic, 2010]. Later, there was a shift to basic machine learning models that rely on statistics [Mzamo et al., 2015]. These days, the most successful models are universal [Straka et al., 2017] [Bergmanis and Goldwater, 2018] [Kanerva et al., 2018]. Usually the corpus used for the training, validation and testing of this model is UD [Straka et al., 2017], and they are the part of the common UD pipeline [Straka et al., 2016].

Interest in lemmatisation for OCS has grown during the last five years. The models tend to use the seq2seq architecture with an attention mechanism [Sutskever et al., 2014] [Cho et al., 2014]. The results have been satisfying since the first published works [Podtergera, 2016]. However, for a particular part of the OCS canon, *Codex Marianus*, which is the part of the UD, these results have been bested by universal lemmatisers which have gradually achieved an overall accuracy score of 95–97% [Straka et al., 2017] [Bergmanis and Goldwater, 2018] [Kanerva et al., 2020]. These models are mostly UD-based, and the UD dataset for OCS lacks the methods to deal with punctuation marks, fragments and digits. A possible solution is to use a hybrid model, such as the one that is used for Old Eastern Slavonic and Middle Russian [Berdičevskis, 2016].

Datasets

Two datasets are used for the task. The first one is the UD OCS dataset [Zeman et al., 2020], on which the model was trained, validated, and preliminarily tested. The UD OCS dataset contains POS tagged tokens of *Codex Marianus,* one of the largest and most widely accepted as a part of the OCS canon texts. The second dataset is the raw text of the *Kiev Folia* [Kiev Folia]. This is significantly smaller, does not have the prepared lemma options, but is considered to be one of the core texts of the OCS canon. Its dissimilarity to the UD OCS dataset and plans to make it the first text of a projected OCS corpus are what make this text suitable for the out-of-domain testing of the model. The results of processing the latter with the model are going to be implemented into the upcoming OCS corpus after manual correction.

The UD OCS dataset consists of 57,563 tokens [Haug and Jøhndal, 2008]. It is taken from the PROIEL project [Haug and Jøhndal, 2008]. Additional information in the dataset consists of POS tag, morphological tagging, and the particular place of the word in the text, with the possibility of extracting left and right contexts from the dataset. The text used is *Codex Marianus*, one of the largest Gospels written in OCS. The dataset is split into three parts. The largest one, *train*, consists of 37,432 tokens. The second and the third, *dev* and *test*, consist of approximately equal amount of tokens, 10,100 and 10,031 respectively. *Test* is more lexically diversified than *dev* or *train*. Thus, *dev* is added to *train* during the model training. The resulting splits are *train_dev* with a token count of 47,532, and *test* with 10,031. *Train_dev* is used for training and validating, while *test* is used for evaluation. The combination of model parameters which proved to be the most efficient on *test* are then is used for the lemmatisation of the corpus.

Kiev Folia is taken from the TITUS collection [Kiev Folia]. The version is based on the latest edition of *Kiev Folia* [Schaeken, 1987]. The TITUS version of the text is presented in electronic form via ASCII. It is impossible to use this version directly because of the seq2seq architecture of the

model. The model is trained on Cyrillic text of *Codex Marianus*, presented in the UD OCS dataset. For the model to see correlations between the sequences of particular symbols, *Kiev Folia* should be converted into Cyrillic script, because when facing Latin symbols, the model generates arbitrary sequences, and thus the whole process of training would become useless. Thus, it is important to prepare the dataset of *Kiev Folia* for processing, which means the transliteration from the ASCII script to Cyrillic. The version that is used for the corpus was the Cyrillic version of the TITUS text, which was made with specific transliteration rules for conversion from ASCII to Cyrillic [Afanasev, 2020]. After that it was enriched with POS tagging, based on the HMM model [Uludoğan, 2018]. The resulting dataset consists of 1,342 tokens. Each word is treated as a separate token, as in the UD OCS dataset. However, *Kiev Folia* contains two more classes of tokens. The first is punctuation marks, the second is digits (in the alphabetical number system, such as ⌣*Б*⌣ '2'). OCS corpus texts may also contain fragmentary tokens, marked with sign '=', however, in *Kiev Folia* these are not present. Tokens are taken from the corpus, lemmatised by the model, and then returned to the corpus, containing the predicted lemmas for all the words given as input. *Kiev Folia* is not split into parts, it is processed as a whole because the model is not trained or validated on this text, it only predicts the lemmas for its words.

Method

The model connects rule-based and dictionary-based modules to recurrent neural network (RNN), creating a hybrid system that is able to cover most of the cases that are met in the datasets described earlier.

The system may use naive stemming, additional information (the POS tag of the word analysed), the dictionary created in the training process, and n-gram predictions. In addition, some rules regulating the defining of a word's attribution to the specific POS are implemented. The system is trained and tested on *Codex Marianus*, an OCS text that is the part of the UD [Zeman et al., 2020]. The architecture that achieved the best results is then tested once again on the material of the *Kiev Folia* dataset [Kiev Folia]. The code and the models, the results of which are presented in the article, are open source published [Project Repository].

The input to the model is a file that contains linguistic data, with a combination of settings that define the workflow of the model. A user must define the file as there is no default option. A user may define the mode in which the model is going to work (training mode for the model to learn, accuracy mode to validate it on the test data, and prediction mode to add data to the previously untagged dataset) The default mode is training. A user may provide a data configuration in a number of ways. A word may be split into *n*-grams, with *n* given by a user (by default the word is not split). The same may be performed for a lemma, if the specific instruction is provided (lemmas are not split by default). For instance, if *n* equals 2, the word *корабь* 'ship-ACC' is to be split into 2-grams *ко*, *ор*, *ра*, *аб* and *бь*. If the user selects lemma split, the lemma *корабль* 'ship' is split into 2-grams *ко*, *ор*, *ра*, *аб*, *бл*, *ль*. The 2-gram *ь#* is added to the word 2-gram set, to equalise the number of 2-gram training pairs.

A quasi-stemming may be done when the word and its lemma are stripped of every letter but the last letter of their stem and inflection, with the latter being used for training. By default, quasi-stemming is not performed, the same happens when word is formally equal to lemma (for instance *рабъ* 'slave-ACC' – *рабъ* 'slave'). For instance, the result of this action for the pair *вашихъ* 'yours-GEN' – *вашь* 'yours' is the pair *шихъ* – *шь*.

Each of these steps may be accompanied by adding POS information to the word. Thus, the pair *врача* 'doctor-GEN' – *врачь* 'doctor' becomes *врачаNOUN – врачь*.

A user may also adjust some settings for the neural network of the model, such as the number of epochs (40 by default), batch size (128 by default), hidden dimension number (256 by default), optimiser (RMSprop by default [Hinton et al., 2012]), loss function (categorical cross entropy by default), activation function (softmax by default [Goodfellow et al., 2016]), and the patience of early stopping callback function [Prechelt, 1998] (no early stopping, patience is equal to the number of epochs by default).

While in prediction or accuracy mode, a user may set how the neural network of the model forms its prediction, if it is trained on *n*-grams. It may take the first letter of each predicted *n*-gram before the latter one, which is taken as a whole (*back* option), or take as a whole the first *n*-gram, and later take the final letter of each of the following *n*-grams (*forward* option by default). For instance, predictions *жи, им, ла* are going to give a better result for lemma *жила* 'vein' with the *back* option (forming a correct sequence *жила* instead of an incorrect one *жима*), while predictions *жи, ил, ма* are going to give a better result for the same lemma with the *forward* option (*жила* as compared to *жима*).

A user may also set name of the model (*seq2seq* by default) and directory (where file with the model is located by default) while in the training mode, and should repeat these steps for the prediction and accuracy mode runs of the model.

In the training mode, the model takes a .conllu file in UD format and processes it according to the user requirements, forming a dataset and creating a dictionary file in the directory provided by user: *lemmatized_NAME.txt*, where *NAME* is the given model name. The dataset is then processed once again into an array of word-lemma training pairs. On the basis of the latter, the file *train_NAME.txt* is created. The parameters of the neural network part of the model are then set and the model is trained on this material. The weights are saved in the file *NAME.h5*.

The neural network is a seq2seq model with an attention mechanism, first presented for the task of machine translation [Sutskever et al., 2014] [Cho et al., 2014]. Both the encoder and decoder parts of the model consist of a long short-term (LSTM) layer [Hochreiter and Schmidhuber, 1997].

The workflow of the prediction and accuracy modes is similar at its core, although it is different in detail. The dataset for the prediction mode is formed from a .json file, and its data should be returned with information on the lemmas. The result for the accuracy mode is information on the evaluation of the model results, and an *errors_NAME.csv* file containing data on outliers and especially poor results of the system workflow according to the metrics.

Otherwise, when the dataset is processed, the model reinstantiates the word-lemma dictionaries via the previously formed files, and the neural network parameters via loading weights.

After the neural network is reinstantiated, the process of prediction or evaluation starts.

First, the rule-based part is used. If the POS of a given word is FRAG (fragment), the lemma is ===. If the POS is PUNCT (punctuation mark) or DIGIT (digit), the convention is that the word is its own lemma. The rules are implemented and used for the Kiev Folia dataset.

In the following step, the dictionary part is used. If the word is in the dictionary file, then its lemma is automatically set from the dictionary.

If the rules do not apply, and there is no dictionary entry, the neural network makes a prediction.

Every prediction is evaluated (according to the metrics discussed below) and then the average prediction accuracy for each metric is calculated, with the outliers identified and saved into a separate dataset.

Experiment Settings

The training parameters of the model were varied for the experiments. The default configuration for the model is 40 epochs, a batch size of 128, 256 hidden dimensions, no early stopping, RMSprop optimiser, categorical cross entropy loss function, softmax activation function. POS is not joined to the word, lemmas are not split, stemming is not done, and words are not split into *n*-grams. The default model forms the final prediction by going *forward*. While training, the number of epochs is set to 60.

The baseline for the model is formed via simple substitution of the lemma by the word itself. Thus, each time the model sees the word, it immediately makes its own lemma. It was decided to make the baseline this way as earlier models tended to solve this task on a very restricted dataset type, the UD treebanks, and using them as a baseline was going to be incorrect.

The evaluation of model is done in two different ways. The *Kiev Folia* dataset has not been lemmatised previously, thus, its evaluation is made approximately and manually, paying attention mostly to the work of the rule-based part of the model. The efficiency of the model on the UD dataset is evaluated automatically. The process of evaluation consists of two steps.

First, the information on prediction accuracy for each word is acquired. For the accuracy score, Jaro-Winkler [Jaro, 1989] [Winkler, 1990] and Damerau-Levenshtein [Damerau, 1964] [Levenshtein, 1966] distances are used. The precise accuracy score was used for comparison with the other models that are used for evaluating the lemmatisation results. However, when evaluating the results of the seq2seq model, which generates output letter by letter, one might get very poor results, which in fact consist of many mistakes by one letter. Thus, the accuracy score is not applicable in every case, although it is still widely used due to its implementation simplicity [Milintsevich, Sirts, 2020] [Akhmetov et al., 2020]. The other option is to use string similarity metrics, such as the Levenshtein distance [Damerau, 1964] [Levenshtein, 1966]. The latter metrics help to better understand how precise the system is. Some works implement it as the main approach to analyse model efficiency [Kanerva et al., 2020] [Metheniti et al., 2020] [Zalmout and Habash, 2020]. However, none of these implemented the Damerau-Levenshtein distance and Jaro-Winkler distance, as is done in this work.

The most significant errata of the system are detected using outliers [Grubbs, 1969] of the results of Jaro-Winkler and Damerau-Levenshtein distances for each particular prediction, which are then classified and analysed on the basis of the linguistic data. Each measurement is added to the respective metric results.

Second, after the whole dataset is processed, raw and normalized means are calculated from the results. Outliers are detected and saved in a separate file to be analysed manually. The results are discussed in comparison with the existing lemmatisers of OCS [Podtergera, 2016] and UD [Straka et al., 2017] [Bergmanis and Goldwater, 2018] [Kanerva et al., 2018], with a baseline of making each word its own lemma.

There were attempts to enhance the architecture of this model. The purpose of the first was to implement the Damerau-Levenshtein distance as a loss function. Due to the nuances of Keras's internal structure, however, it was proven to lack usefulness, because the loss function in Keras does not have a direct access to input, and there is not much sense in implementing the Damerau-Levenshtein distance for tensors of numbers. Architectural restrictions also blocked the use of the Levenberg–Marquardt algorithm [Levenberg, 1944] [Marquardt, 1963].

The second attempt used mellowmax [Kavosh and Littman, 2017] as an activation function. This implementation failed as mellowmax is mainly used in reinforcement learning, which is not the case for the workflow presented in this paper. Mellowmax is used to increase the efficiency of a very

specific architecture, designed for very specific purposes, thus is unsuitable for the purposes of this paper.

There were attempts to further modify the Levenshtein distance, and its advanced version, the Damerau-Levenshtein distance [Znamenskij. 2017] [Ganesh et al., 2020]. However, these have not yet acquired the same level of use as their earlier counterparts, and one needs to be cautious while implementing them for the evaluation of model's results.

Results and Analysis

This section consists of three parts. First, the baseline for the model on the UD dataset is given, and a series of experiments (1 - 7) with different model parameters on the UD dataset is described. For the final experiment, the description and analysis of the outliers are provided. At the end of the section, the results of the experiment on the *Kiev Folia* dataset are given.

A preliminary evaluation is made on the material of the UD dataset. It contains a series of experiments, the purpose of which is to define which parameters were the most effective, compared to the baseline results (Table 1). The baseline results were acquired via replacing the neural network module of the model with a rule that states that the lemma of each word form is this word form. For instance, the lemma for *гръси* 'sin-PL-NOM' is *гръси* 'sin-PL-NOM', and not *гръхъ* 'sin'.

Tab. 1. The baseline evaluation results

Metrics	A	L(R)	L(N)	D-L(R)	D-L(N)	J-W(R)	J-W(N)
Baseline	29.24%	3.86	3.73	1.77	1.74	0.77	0.86

Notes: Results are to 2 decimal places. A = Accuracy score, L = Levenshtein distance, D-L = Damerau-Levenshtein distance, J-W = Jaro-Winkler distance, R = raw, and N = normalized.

The experiment 1 deals with quasi-stemming efficiency. The changes from the basic parameters are using 3-grams, an increased batch size of 256, early stopping with patience of 3, and splitting lemmas. Model 1 does quasi-stemming, model 2 does not. The results are presented in Table 2. Quasi-stemming slightly reduces the efficiency of the model, and thus is to be discarded. The possible reason for the reduced efficiency is that when the model performs quasi-stemming, it loses the ability to predict the stems of the lemmas from the stems of the respective words, it generates only inflections. However, it is important to underline that the models' results differ only slightly, and drawing general conclusions would be premature.

Tab. 2. The evaluation results of models 1 and 2

Metrics	A	L(R)	L(N)	D-L(R)	D-L(N)	J-W(R)	J-W(N)
1	53.94%	5.41	5.31	3.21	3.2	0.67	0.67
2	**53.96%**	**5.4**	5.31	**3.2**	3.2	**0.68**	**0.68**

Notes: The best results for each metrics are highlighted in bold.

Experiment 2 establishes whether the addition of information on POS helps in the prediction of the correct lemma. The changes for the basic parameters remained the same as in experiment 1. Model 3 adds POS to the word, and its results are compared to the results of model 2 from experiment 1. The comparison is given in Table 3. Information on POS does not increase model efficiency. What is more, some metrics show a slight decrease, and therefore adding POS to the word is not included as a basic strategy. A possible conclusion from that decrease is that information on POS has a little to no effect on machine comprehension of how one sequence transforms to another. This might be resolved with adding more morphological information, however, due to the restrictions presented by the datasets, its use is not a possibility to be explored in the research described in this article.

Tab. 3. The evaluation results of models 2 and 3

Metrics	A	L(R)	L(N)	D-L(R)	D-L(N)	J-W(R)	J-W(N)
2	53.96%	5.4	**5.31**	**3.2**	**3.2**	**0.68**	**0.68**
3	53.96%	5.4	5.32	3.21	3.21	0.66	0.66

Experiment 3 deals with batch size regulation. It is important to check whether an increase from 128 to 256, and from 256 to 512, or a decrease from 128 to 64, makes any significant difference. Models, 4 with default batch size of 128, 5 with batch size of 64, and 6 with batch size of 512, are compared to model 2 from the previous experiments. The comparison is given in Table 4. There is no strong correlation between batch size and model efficiency. The results for all the metrics seem to change almost randomly, with similar results for models 2 and 6, when taking into consideration Levenshtein and Damerau-Levenshtein distances, and constantly changing, when taking into consideration Jaro-Winkler distance. As changing the batch size does not significantly change the speed of training, and because the best overall results for all the metrics (despite a slight decrease in overall accuracy score) were demonstrated by the model of 256 batch size, the latter is the best choice for the subsequent experiments.

Tab. 4. The evaluation results of models 2, 4, 5, and 6

Metrics	A	L(R)	L(N)	D-L(R)	D-L(N)	J-W(R)	J-W(N)
2	53.96%	**5.4**	**5.31**	**3.2**	**3.2**	**0.68**	**0.68**
4	**53.97%**	5.54	5.5	3.34	3.34	0.67	0.67
5	**53.97%**	5.47	5.45	3.27	3.27	**0.68**	**0.68**
6	**53.97%**	5.43	5.32	3.23	3.23	0.67	0.67

Experiment 4 investigates early stopping, and its effect on the efficiency of model training. Model 7 has no early stopping, and is compared to model 2, which has early stopping of 3. The comparison is given in Table 5. A larger number of epochs results in a slight increase in the overall accuracy. Despite that, all other metrics demonstrate a slight decrease when the model is trained during the larger number of epochs. Therefore, and taking into consideration that the time consumed is significantly larger, if early stopping is not performed, it is better to save early stopping with patience of 3, in order to save time for further experiments, and to have better efficiency by most of the metrics.

Tab. 5. The evaluation results of models 2, and 7

Metrics	A	L(R)	L(N)	D-L(R)	D-L(N)	J-W(R)	J-W(N)
2	53.96%	5.4	5.31	3.2	3.2	0.68	0.68
7	53.97%	5.43	5.32	3.23	3.23	0.67	0.67

Experiment 5 looks at how splitting lemmas influences the overall efficiency of the model. Lemma splitting is done during the data preparation process. It is used when the model is trained on *n*-grams. The lemma splitting parameter defines whether the *n*-gram in a word is matched to the *n*-gram in a lemma during the training process, and whether a single lemma is predicted via a combination of *n*-grams or each *n*-gram in a word is matched to a lemma during the training process, and a set of lemmas is predicted for each *n*-gram during the prediction phase.

For instance, during the 3-gram training with splitting lemmas the word *вѣрѫ* ‘faith-ACC-SG’ is split into 3-grams *вѣр* and *ѣрѫ*. When the lemma splitting parameter is active, lemma *вѣра* ‘faith’ is split into 3-grams *вѣр* and *ѣра*. The training pairs would be *вѣр – вѣр* and *ѣрѫ – ѣра*. When the lemma splitting parameter is inactive, the training pairs would be *вѣр –вѣра* and *ѣрѫ – вѣра*. The prediction results are expected to be *вѣра* and *вѣра#вѣра,* with # sign being the sign of split between two predicted lemmas for each of 3-grams.

Model 8 does not split lemmas, while model 2 does it in both the training and accuracy modes. The results of this comparison are given in Table 6.

The conclusion is clear: it is significantly more effective to split lemmas in both the training and accuracy modes (i.e. model 2). The reason is that the efficiency of prediction in the *n*-gram by *n*-gram mode is clearly a better option, than the efficiency of prediction in the *n*-gram to word mode, because in the former a model better and faster adapts to the requirements that it faces. Another reason for the increase in efficiency is that the model is confused by needing to predict a single lemma from completely different *n*-grams.

Tab. 6. The evaluation results of models 2, and 8

Metrics	A	L(R)	L(N)	D-L(R)	D-L(N)	J-W(R)	J-W(N)
2	53.96%	5.4	5.31	3.2	3.2	0.68	0.68
8	24.15%	6.96	6.73	5.97	5.97	0.38	0.38

Experiment 6 decides whether it is more effective to have the *forward* form of the final prediction, rather than the *back* one. The differences between them are described above. Model 9 uses the *back* forming priority, while the model 2 uses the default *forward* forming priority. The results are shown in Table 7. The order in which the model forms the output has little to no negative effect. As with most of the previous experiments, the difference is not significant. Every metric but the overall accuracy score shows that model 2 demonstrates better results than model 9, and for the overall accuracy score the difference negligible. The reason is that rate of mistakes remains high enough, and whether the model constructs the word from the beginning or the end, the possible positive effect is reduced to zero. At this stage it is safe to say that the forming priority can stay on the default *forward* option.

Tab. 7. The evaluation results of models 2, and 9

Metrics	A	L(R)	L(N)	D-L(R)	D-L(N)	J-W(R)	J-W(N)
2	53.96%	**5.4**	**5.31**	**3.2**	**3.2**	**0.68**	**0.68**
9	**53.97%**	5.46	5.36	3.26	3.25	0.67	0.67

Experiment 7 investigates an increase of *n* in *n*-grams that are given to the model as input to understand whether the synchronous split of the word and its lemma in the training phase provides a sig-

nificant increase in the efficiency of the model. For the part of experiment 7, *n* is not defined, so the model does not split either words or their respective lemmas. The *n* used are 4, 5, and 8. The first two options are to see whether there is a correlation between an increase in *n* and an increase of model efficiency. *n*=8 is one of the biggest word lengths in the OCS UD dataset, so it is used to check, whether the tendency remains. The models are 2 (3-grams), 10 (4-grams), 11 (5-grams), 12 (8-grams), and 13 (no *n*-gram split). The results are presented in Table 8.

By most metrics model 13 demonstrates the best results. The only exceptions are the normalised average Damerau-Levenshtein distance, and the normalised average Jaro-Winkler distance. This is because with a large *n* in *n*-gram the resulting array becomes abnormally distributed with most of the metrics values tending to be 0 for Damerau-Levenshtein distance, and 1 for Jaro-Winkler distance.

Such a dramatic increase in the model results may be because the dictionary part performs better, being formed from whole words not from *n*-grams. However, the dictionary part is in fact formed from whole words in each of the models that are under the consideration. A more probable reason is that model is better trained, when given the correlations between bigger (probably, more meaningful) sequences of symbols.

Thus, the model for final test against the baseline is 13.

Tab. 8. The evaluation results of models 2, 10, 11, 12, and 13

Metrics	A	L(R)	L(N)	D-L(R)	D-L(N)	J-W(R)	J-W(N)
2	53.96%	5.4	5.31	3.2	3.2	0.68	0.68
10	58.67%	5.36	5.26	2.96	2.96	0.72	0.72
11	63.86%	5.18	5.07	2.58	**2.5**	0.74	**0.74**
12	78.81%	4.77	4.7	1.56	–	0.82	–
13	**85.67%**	**4.37**	**4.34**	**0.88**	–	**0.9**	–

The results of a comparison between the baseline model and 13 are presented in Table 9. Model 13 beats the baseline by a significant margin in 3 out of 4 metrics: overall accuracy score, average Damerau-Levenshtein distance, and average Jaro-Winkler distance. The baseline demonstrated better results in average Levenshtein distance, which may be explained by the nuances of the implementation of the Levenshtein distance measurement, or the metric itself, and its differences to, for instance, Damerau-Levenshtein distance. The probable reasons for Leveshtein distance metrics being significantly better for the baseline are that the baseline results are much more normally distributed, while results for the Damerau-Levenshtein and Jaro-Winkler distance metrics are hesitating between completely misleading lemmas and the target lemmas.

Tab. 9. The evaluation results of model 13, while compared to the baseline

Metrics	A	L(R)	L(N)	D-L(R)	D-L(N)	J-W(R)	J-W(N)
13	**85.67%**	4.37	4.34	**0.88**	–	**0.9**	–
Baseline	29.24%	**3.86**	**3.73**	1.77	**1.74**	0.77	**0.86**

The outliers, acquired during the testing of model 13, are recorded into the separate dataset. The outliers are important, because they highlight the most significant system flaws, design issues, and failures, shown by different metrics. The outliers also give insights into the times when a model works abnormally well, which is impossible for the metric data.

First, the analysis of outliers is done for the most common lemmas. These are the same for Damerau-Levenshtein distance and Jaro-Winkler distance: *имѣти* 'to have', *глаголати* 'to speak', *клѧти* 'to curse', *приѣти* 'to come', *мънасъ* 'face', *исходити* 'to come out of', *чловѣчьскъ* 'human', *оумыти* 'wash', *рыба* 'fish', *въходити* 'to come in'. For Levenshtein distance they are completely different: *съвѣдѣтельствовати* 'to tell about smth smb witnessed', *пророчьствовати* 'to declare a prophecy', *законооучитель* 'the one who is the teacher of Law', *отъдесѧтьствовати* 'to tax', *сѵрофѵникиссаныни* 'smb from the Syriac territory who is of Phoenician background', *дальманоуфаньскъ* 'smb from the Dalmanufa territory', четврьтовластьць 'tetrarch', *благословлѥнъ* 'the blessed one'. The difference is due to the fact that Levenshtein distance calculations demonstrate a higher level of normalisation in general, so outliers are rarer, and the least common ones for Damerau-Levenshtein and Jaro-Winkler distances are the most common ones for Levenshtein distance.

The most common outliers are more similar, when comparing the three metrics. The most common are *и* 'and', *новъ* 'new' for each of the metrics. The Levenshtein distance calculations have only three more outliers: *абоѥ* (meaning is unclear), *оудъ* 'body part', *къ* 'to'. The Damerau-Levenshtein and Jaro-Winkler distance calculations share the remaining eight, *божии* 'God's', *w* 'about', *быти* 'to be', посълати 'to send', *рѫбъ* (meaning is unclear), *тъ* 'that', *оу* 'of', *исоусъ* 'Jesus'.

The model tends to make mistakes mostly by generating shorter outputs for longer inputs. This can be seen in that the average lemma is longer than average outlier (7.14, 7.14, 16 versus 4.02, 4.02, 2.1 for Jaro-Winkler distance, Damerau-Levenshtein distance, and Levenshtein distance respectively). In addition to this, the maximum lemma length is 18, while maximum outlier length is 12.

The UD OCS dataset is not the only one that can be used to test the model and it should not be as it consists only of one text, *Codex Marianus*. Any deviation might influence the model's efficiency when it meets tokens which are completely new for it, the kind that are simply not present in the UD OCS. These are fragments, punctuation marks, and digits. In addition to this, it is import to check whether a larger number of different words, with sometimes contrasting regular letter sequences (such as the reflexes of **tj* and **dj*, that are different between some of the OCS texts [Kamphuis, 2020]), influences the overall accuracy of the model. In other words, the model needed out-of-domain testing [Mathis et al., 2020].

For that specific purpose, the *Kiev Folia* dataset was used. This dataset consists of the single text. *Kiev Folia* was chosen specifically because its key linguistic features are very different from the other texts of the OCS canon [Kamphuis, 2020]. *Kiev Folia* tokens include punctuation marks, fragments, and digits.

To regulate these, the model was enhanced with two additional rules. One states that if the token was POS tagged as fragment, then its lemma is always three marks of lost graphemes '==='. The second states that if the token is POS tagged as a digit or a punctuation mark then the lemma for the token is the token itself. The examples of the model results are given in the table 10.

Tab. 10. Examples of tokens, lemmatised by rule-enhanced model 13

Token	Lemma
======*абъ*	===
:.:	*:.:*

Thus, the system, having been slightly modified, demonstrated that it is able to lemmatise tokens and work with the ones that it had been previously trained to work with.

Conclusion

In this paper a lemmatiser for OCS was presented. New evaluation metrics were implemented, and new methods of the enhance model training were proposed.

The system seems to lack overall accuracy, compared to previous UD lemmatisers [Straka et al., 2017] [Bergmanis and Goldwater, 2018] [Kanerva et al., 2018]. Overall, it was only about 85% accurate. However, it is better adapted to the texts from the OCS corpus that is being formed. Using the *Kiev Folia* dataset, the model demonstrated the ability to lemmatise unknown words in general and fragmentary tokens, punctuation marks, and digits in particular.

The analysis of the most serious errata of the system shows that it tends to generate lemmas that are shorter than the average lemma length of the dataset. This is probably due to the fact that shorter words are met far more often in the training phase. That leads to the conclusion that model is going to need additional training on the datasets that contain longer words, when the latter are available for OCS. Another option is to change the architecture of the neural network of the model from basic seq2seq with an attention mechanism to something more advanced, for instance, BART.

References

Aduriz, I & Aldezabal, I & Artola, X & Ezeiza, N & Urizar, R 1998, "EUSLEM: A lemmatiser/tagger for Basque. ", Proceedings of the 7th EURALEX International Congress, Novum Grafiska AB, Göteborg, Sweden, pp. 27-35.

Akhmetov, I & Pak, A & Ualiyeva, I & Gelbukh, A 2020, "Highly Language-Independent Word Lemmatization Using a Machine-Learning Classifier. ", Computacion y Sistemas vol. 24, pp. 1353-1364.
Afanasev, I 2020, "Korpus staroslavianskogo iazyka: nedostaiushchee zveno v diakhronicheskoi slavistike", Slavica iuvenum XXI : sbornik trudov mezhdunarodnoi nauchnoi konferentsii Slavica iuvenum 2020, 31.3. – 1.4.2020, Ostravskii universitet, Ostrava, pp. 13-21.
Berdičevskis, A 2016, "The beginning of a beautiful friendship: rule-based and statistical analysis of Middle Russian", Komp'yuternaya lingvistika i intellektual'nye tekhnologii. Trudy mezhdunarodnoj konferencii «Dialog», Moscow, Russia, pp. 99–111.
Bergmanis, T & Goldwater, S 2018, "Context sensitive neural lemmatization with Lematus". Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), Association for Computational Linguistics, New Orleans, Louisiana, vol. 1, pp. 1391–1400.
Camps, J-B & Gabay, S & Fièvre, P & Clérice, T & Cafiero, F 2020, "Corpus and Models for Lemmatisation and POS-tagging of Classical French Theatre", viewed 5 December 2020, < https://arxiv.org/pdf/2005.07505.pdf >
Cho, K. & Merrienboer, BV & Gülçehr Ç & Bahdanau, D & Bougares, F & Schwenk, H & Bengio Y 2014, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation", Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), ACL, Doha, Quatar, pp. 1724-1734.
Chrupała, G 2006, "Simple data-driven context-sensitive lemmatization", Procesamiento del Lenguaje Natural, vol. 37, pp. 121-127.
Damerau, FJ 1964, "A technique for computer detection and correction of spelling errors", Communications of the ACM vol. 7 no. 3, pp. 171–176.
Džeroski S, Erjavec T 2000, "Learning to Lemmatise Slovene Words", in: Cussens J., Džeroski S. (eds) Learning Language in Logic. LLL 1999. Lecture Notes in Computer Science, vol 1925. Springer, Berlin, Heidelberg, pp. 69-88.
Evans, R & Brown, D & Corbett, G & Tiberius, C 2006, Russian Lemmatisation with DATR, University of Brighton.
Farkas, R & Vincze, V & T, I & Ormándi, R & Szarvas, G & Almási, A 2008, "Web-Based Lemmatisation of Named Entities", Proceedings of TSD 2008, Brno, Czech Republic, pp. 53-60.
Fernández, L 2020, "A contribution to Old English lexicography: Utgangan, wiðhealdan, ofersceadan, onbefeallan and ongangan", NOWELE. North-Western European Language Evolution vol. 73 no. 2, pp. 236-251.
Ganesh, D & Kumar, T & Kumar, M 2020, "Optimised Levenshtein centroid cross-layer defence for multi-hop cognitive radio networks", IET Communications, vol. 15 no. 2, pp. 245-256.
Gesmundo, A & Samardžić, T 2012, "Lemmatisation as a tagging task", Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics vol. 2, ACL, Jeju Island, Korea, pp. 368-372.
Goodfellow, I & Bengio, Y & Courville, A 2016, "6.2.2.3 Softmax Units for Multinoulli Output Distributions", in Deep Learning, MIT Press, pp. 180–184.
Gouws, RH, & Prinsloo, D 2012, "Lemmatisation of Adjectives in Sepedi", Lexikos vol. 7, pp. 45-57.
Groenewald, H 2007, Automatic lemmatisation for Afrikaans, North-West University, Potchefstroom Campus.

Groenewald, H 2009, "Using technology transfer to advance automatic lemmatisation for Setswana", Proceedings of the First Workshop on Language Technologies for African Languages, ACL, Stroudsburg, PA, pp. 32-37.
Grubbs, FE 1969, "Procedures for detecting outlying observations in samples", Technometrics vol. 11, no. 1, pp. 1–21.
Hann, M 1974. "Principles of Automatic Lemmatisation", ITL Review of Applied Linguistics vol. 23 no. 1, pp. 3-22.
Hardie, A & Lohani, R& Yadava, Y 2014, "Extending corpus annotation of Nepali: advances in tokenisation and lemmatisation", Himalayan Linguistics vol. 10, no. 1, pp. 151-165.
Haug, DTT & Jøhndal ML 2008, "Creating a Parallel Treebank of the Old Indo-European Bible Translations", Proceedings of the Second Workshop on Language Technology for Cultural Heritage Data (LaTeCH 2008), ACM, New York, NY, pp. 27-34.
Hinton G & Srivastava N & Swersky K 2012, "Lectures on Machine Learning", viewed 5 December 2020, <http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf>
Hochreiter S & Schmidhuber J 1997, "Long short-term memory", Neural Computation vol. 9 no. 8, pp. 1735—1780.
Jaro, MA 1989, "Advances in record linkage methodology as applied to the 1985 census of Tampa Florida". Journal of the American Statistical Association vol. 84 (406), pp. 414–20.
Jiampojamarn S & Cherry C & Kondrak G 2008, "Joint processing and discriminative training for letter-to-phoneme conversion", Proceedings of ACL-08: HLT. Association for Computational Linguistics, Columbus, Ohio, pp. 905-913.
Jursic, M & Mozetic, I & Erjavec, T & Lavrac, N 2010, " LemmaGen: Multilingual Lemmatisation with Induced Ripple-Down Rules", JUCS vol. 16, pp. 1190-1214.
Kamphuis, J 2020, Verbal Aspect in Old Church Slavonic, Brill, Leiden.
Kanerva, J & Ginter, F & Salakoski, T 2020, "Universal Lemmatizer: A sequence-to-sequence model for lemmatizing Universal Dependencies treebanks,", Natural Language Engineering, Cambridge University Press, pp. 1–30.
Kavosh A & Littman M 2017, "An Alternative Softmax Operator for Reinforcement Learning. ", Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, pp. 243-252.
Ki-
ev Folia, viewed 5 December 2020, <http://www.schaeken.nl/lu/research/online/editions/kievfol.html>
Kosch, IM 2016, "Lemmatisation of Fixed Expressions: The Case of Proverbs in Northern Sotho", Lexikos vol. 26, pp. 145-161.
Levenberg, K 1944, "A Method for the Solution of Certain Non-Linear Problems in Least Squares", Quarterly of Applied Mathematics vol. 2, pp. 164–168.
Levenshtein, VI 1966, "Binary codes capable of correcting deletions, insertions, and reversals", Soviet Physics Doklady vol. 10 no. 8, pp. 707–710.
Lewis, M & Liu, Y & Goyal, N & Ghazvininejad, M & Mohamed, A & Levy, O & Stoyanov, V & Zettlemoyer, L 2020, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension", Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL, Online, pp. 7871-7880.
Ljubešić, N & Dobrovoljc, K 2019, "What does Neural Bring? Analysing Improvements in Morphosyntactic Annotation and Lemmatisation of Slovenian, Croatian and Serbian", Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing, ACL, Florence, Italy, pp. 29-34.

Marquardt, D 1963, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters", SIAM Journal on Applied Mathematics, vol. 11 no. 2, pp. 431–441.
Mathis, A & Yüksekgönül, M & Rogers, B & Bethge, M & Mathis, MW 2020, "Pretraining boosts out-of-domain robustness for pose estimation", arXiv preprint, viewed 5 December 2020, <https://arxiv.org/pdf/1909.11229.pdf>
Metheniti, E & Neumann, G & Genabith, J 2020, " Linguistically inspired morphological inflection with a sequence to sequence model", viewed 5 December 2020, <https://arxiv.org/pdf/2009.02073v1.pdf >
Milintsevich K & Sirts K 2020, "Lexicon-Enhanced Neural Lemmatization for Estonian", Human Language Technologies – The Baltic Perspective, pp. 158-165.
Mills, J 1998, "Lemmatisation of the Corpus of Cornish", Proceedings of the Workshop on Language Resources for European Minority Languages, LREC First International Conference on Language Resources and Evaluation, Granada, Spain, pp. 1-6.
Mulhall, C 2008, "The Lemmatisation of Lexically Variable Idioms: The Case of Italian-English Dictionary", Proceedings of the 13th EURALEX International Congress, Barcelona, Spain, pp. 1373-1378.
Mzamo L & Helberg A & Bosch S 2015, "Introducing XGL - a lexicalised probabilistic graphical lemmatiser for isiXhosa", Proceedings of 2015 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech), Port Elizabeth, pp. 142-147
Nthambeleni, M & Musehane, N 2014, "The Lemmatisation of Nouns in Tshivenda Dictionaries", Lexikos vol. 24, pp. 214-224.
Plisson, J & Lavrac, N & Mladenić, D & Erjavec, T 2008, "Ripple Down Rule learning for automated word lemmatisation". AI Commun. vol 21, pp. 15-26.
Project Repository, viewed 10 January 2021, <https://github.com/The-One-Who-Speaks-and-Depicts/OCS-corpus-lemmatiser>
Podtergera I 2016, "SlaVaComp-Lemmatizer: a Lemmatization Tool for Church Slavonic", Proceedings of El'Manuscript-2016: Textual Heritage and Information Technologies, Izhevsk, pp. 212–221.
Prechelt, L 1998, "Early stopping – but when? ", in Neural Networks: Tricks of the trade, pp. 55 – 69
Prinsloo, D 2012, "A Critical Analysis of the Lemmatisation of Nouns and Verbs in isiZulu", Lexikos vol. 21, pp. 169-193.
PyTorch Project, viewed 5 December 2020, <https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html>
Radziszewski, A 2013, "Learning to lemmatise Polish noun phrases", Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics vol. 1, Sofia, Bulgaria, pp. 701-709.
Schaeken, J 1987, Die Kiever Blätter (= Studies in Slavic and General Linguistics 9), Amsterdam.
Schryver, G-M de & Nabirye M 2018, "Corpus-driven Bantu Lexicography, Part 2: Lemmatisation and Rulers for Lusoga". Lexikos vol. 28, pp. 79-111.
Spyns, P 1996, "A tagger/lemmatiser for Dutch medical language", Proceedings of the 16th conference on Computational linguistics vol. 2, ACL, Stroudsburg, PA, pp. 1147-1150.
Straka M & Hajič J & Straková J 2016, "UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing", Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), Portorož, Slovenia.

Straka, M & Straková, J 2017, "Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe", Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Vancouver, Canada, pp. 88-99.
Sutskever, I & Vinyals, O & Le, QV 2014, "Sequence to Sequence Learning with Neural Networks", NIPS, pp. 1-9
Tamburini, F 2013, "The AnIta-Lemmatiser: A Tool for Accurate Lemmatisation of Italian Texts", Proceedings of EVALITA 2012, Rome, Italy, pp. 266–273.
Tianqi, C & Carlos, G 2016. "XGBoost: A Scalable Tree Boosting System", Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, San Francisco, CA, USA, pp. 785–794.
Ulu-
doğan, G 2018, HMM POS tagger, viewed 5 December 2020, <https://github.com/gokceuludogan/hmm-pos-tagger>
Winkler, WE 1990, "String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage", Proceedings of the Section on Survey Research Methods. American Statistical Association, Alexandria, VA, pp. 354-359.
Zalmout, N & Nizar Habash 2020, "Joint Diacritization, Lemmatization, Normalization, and Fine-Grained Morphological Tagging. ", Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL, Online, pp. 8297-8307.
Zeman, D & Nivre J & Abrams & Mitchell et al 2020, "Universal Dependencies 2.7", LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University, viewed 5 December 2020, http://hdl.handle.net/11234/1-3424
Znamenskij, S 2017, "A Choice of Similarity Measure for the Strings", pre-print, viewed 25 December 2020, <https://www.researchgate.net/publication/317823569_A_Choice_of_Similarity_Measure_for_the_Strings>

Contact Details

Ilia Afanasev
Federal State Budgetary Educational Institution of Higher Education «Saint-Petersburg State University». Philological Faculty, Department of Slavic Studies. Graduate student;
E-mail: szrnamerg@gmail.com

Disclaimer

Any opinions or claims contained in this Working Paper do not necessarily reflect the views of HSE.