Structuring Lexical Data and Digitising Dictionaries

Grammatical Theory, Language Processing and Databases in Historical Linguistics

Edited by
Javier Martín Arista and
Ana Elvira Ojanguren López

BRILL

Structuring Lexical Data and Digitising Dictionaries

Language and Computers

STUDIES IN DIGITAL LINGUISTICS

VOLUME 85

The titles published in this series are listed at *brill.com/lc*

Structuring Lexical Data and Digitising Dictionaries

Grammatical Theory, Language Processing and Databases in Historical Linguistics

Edited by

Javier Martín Arista
Ana Elvira Ojanguren López

BRILL

LEIDEN | BOSTON

Library of Congress Cataloging-in-Publication Data

Names: Martín Arista, Javier, editor. | Ojanguren López, Ana Elvira, editor.
Title: Structuring lexical data and digitising dictionaries : grammatical theory, language processing and databases in historical linguistics / edited by Javier Martín Arista, Ana Elvira Ojanguren López.
Description: Leiden ; Boston : Brill, 2024. | Series: Language and computers, 0921-5034; volume 85 | Includes bibliographical references and index.
Identifiers: LCCN 2024032464 (print) | LCCN 2024032465 (ebook) | ISBN 9789004702653 (hardback ; acid-free paper) | ISBN 9789004702660 (ebook)
Subjects: LCSH: Lexicography–Data processing. | Language and languages–Etymology–Lexicography. | LCGFT: Essays.
Classification: LCC P327.5.D37 S77 2024 (print) | LCC P327.5.D37 (ebook) | DDC 413.0285–dc23/eng/20240917
LC record available at https://lccn.loc.gov/2024032464
LC ebook record available at https://lccn.loc.gov/2024032465

Typeface for the Latin, Greek, and Cyrillic scripts: "Brill". See and download: brill.com/brill-typeface.

ISSN 0921-5034
ISBN 978-90-04-70265-3 (hardback)
ISBN 978-90-04-70266-0 (e-book)
DOI 10.1163/9789004702660

This book is printed on acid-free paper and produced in a sustainable manner.

Contents

PART 1

Lexical Databases and Language Processing in Digital Historical Lexicography

PART 2
Structuring Historical Lexicons for Lexicography and Corpus Analysis

CHAPTER 2

String Similarity Measures for Evaluating the Lemmatisation in Old Church Slavonic

Ilia Afanasev and Olga Lyashevskaya

1 Introduction

Modern historical lexicography faces the need of adopting NLP methods, with an automatic corpus/dictionary system being a possible pipeline. This approach requires a scalable lemmatiser, as it should deal with heterogeneous data, both in-domain and out-of-domain.

One of the biggest challenges in creating a scalable lemmatiser model is the results assessment: which indicators signify the correctness and the quality of its performance? How well or poorly might the model perform on a specific task at the current stage of NLP development? The latter question is especially relevant, because despite our desire to obtain 100 % accuracy for automatic English part-of-speech tagging or 97 % accuracy for Finnish lemmatisation, the models for the former are more likely to overcome these barriers (Gambäck et al., 2009) than those designed for the latter (Howell et al., 2020).

The assessment might become even harder due to some additional factors. For instance, in tasks such as lemmatisation or machine translation, the degree of required accuracy from a single instance processing is often uncertain. Thus, a model may perfectly guess most of the items in a homogeneous dataset, but demonstrate weak scalability. The other model may scale very well, but not give a single fully correct prediction. Some metrics prove the first kind of models to be more efficient; others favour the second one. Some datasets, inherently heterogeneous, require the use of the latter approach.

We present one of such datasets, the Old Church Slavonic (OCS) texts. This is a very small collection that dates back to the 9th–11th centuries AD. Some researchers state that the number of OCS texts is close to 40, while others are more cautious, restricting the number to about 20 texts. OCS is a language of religion, so the set of texts is genre-restricted. The closest living relatives of OCS (having omitted the modern Church Slavonic languages, as their living status is questionable) are Bulgarian and Macedonian, as OCS emerged on the basis of Bulgarian-Macedonian dialects. The key issue with OCS is that the language is both low-resourced and heterogeneous, which researchers state (Mathiesen, 1984; Polivanova, 2013; Kamphuis, 2020; Lyashevskaya, Afanasev, 2021).

© ILIA AFANASEV AND OLGA LYASHEVSKAYA, 2025 | DOI:10.1163/9789004702660_003

Despite a relatively low number of texts, OCS has not yet been efficiently digitised; its annotated comprehensive corpus is yet to appear. The latest comprehensive dictionary of OCS dates back to 1994, which raises the need for creating a new dictionary. The best option would be a simultaneous creation of a corpus and a corpus-based dictionary via a fully functional corpus-to-dictionary system. The crucial element of such a system is a lemmatiser, a tool that performs a transformation of a token into its dictionary form, which is called lemma.

Given the heterogeneous nature of Old Church Slavonic, one should test whether the lemmatiser that performs this task should be scalable between different variations. This requires a particular kind of metric, which is at the focus of this study. This should be a metric that shows to what degree the model under consideration is robust against variation. The same metric should also effectively evaluate sequence character-by-character generation, giving the scholar insights into the inner workings of such a model.

In general, virtually any metrics may suit this role. For instance, accuracy score is simple in implementation and is already widely used in publications on lemmatisation (Milintsevich, Sirts, 2020; Akhmetov et al., 2020). By comparing out-of-domain accuracy and in-domain accuracy, we get the scalability potential of a model.

There is, however, a significant obstacle. Lemmatisation often is not a classification (where a model is either correct, or not) but a transformation problem (where a model can make as many mistakes as there are letters in a lemma) (Kestemont, de Gussem, 2017). In the latter case, the cost of a mistake set by the accuracy metrics is unjustly higher than for the classification method. In addition, accuracy does not provide insights into the exact nature of how a sequence-to-sequence model works. At best, researchers get the error list, for which they then manually perform string similarity measurement. The efficient metric should decrease the cost of mistakes for the model as well as provide more linguistically interpretable results.

The work is organised as follows. After the brief introduction in this section, we give a review of the literature on the topic. Then we characterise the model, the data, and the metrics. The following stage includes experiments, linguistic interpretation of their results, and an overview of the dictionary, created with the lemmata, produced by the model. The conclusion provides the summary of the results of the research, and outlines the future directions.

2 Related Work

Lemmatisation has been a prominent NLP task for the last few decades (Hann, 1974). Generally, lemmatisation tools are divided into universal ones (Straka et al., 2017) (Bergmanis and Goldwater, 2018) (Kanerva et al., 2020) and specific ones, designed to perform a specific task (Džeroski and Erjavec, 2001) (Groenewald, 2007) (Farkas et al., 2008) (Tamburini, 2013) (Kosch, 2016) (Fernández, 2020).

Lemmatisers may also differ in approach. One family of methods starts with a manual creation of rules to transform a token into a lemma. Then the classifier model matches each token in the text with a specific rule, and afterwards applies the rule (Mills, 1998; Chrupała, 2006; Plisson et al., 2008; Gesmundo and Samardžić, 2012; Radziszewski, 2013).

A new approach, adopted during the last decade, requires only one step (Kanerva et al., 2020). A single model receives a token and possible additional data, like part-of-speech, morphology, or word context. The model processes this data and outputs a lemma. Starting with 2018, researchers switched to a sequence-to-sequence (seq2seq) encoder-decoder model (Bergmanis and Goldwater, 2018) (Ljubešić and Dobrovoljc, 2019).

The OCS lemmatisation witnessed a surge of interest during the last decade. The models tend to use the seq2seq architecture, which also benefits from an added attention mechanism (Sutskever et al., 2014) (Cho et al., 2014). The best result for a single text lemmatisation is UDPipe on Codex Marianus, having achieved a 95–97 % accuracy score (Straka et al., 2017) (Bergmanis and Goldwater, 2018) (Kanerva et al., 2020). However, the UD 2.7 dataset (Zeman et al., 2017) lacks punctuation marks, fragments, and digits. Thus, a lemmatiser is not able to recognise these tokens when they occur in a new text. A possible solution is to use a hybrid model, such as the one that is applied for Old East Slavonic and Middle Russian (Berdičevskis et al., 2016).

The efficiency of lemmatisation is of utmost importance when the lemmatisation results are utilised for an automatic collection by the given criteria from the tagged corpus (Rundell and Kilgariff, 2011) (Kilgariff & Kosem, 2012). The automatic collection tools appeared in the early 2010s (Schryver & Taljard, 2007) (Sangawa et al., 2009) (Granger and Paquot, 2015), together with systems that facilitate the manual compilation of dictionaries from corpora (Bugakov, 2009) and systems that integrate parallel dictionaries with corpus queries (Sangawa and Erjavec, 2012; Paquot, 2012).

Generally, an accuracy score is the only metric the scholars use to evaluate the lemmatisation (Kanerva et al., 2020). The recent years, though, have witnessed changes for similar tasks evaluation, such as multi-hypothesis technology for machine translation (Fomicheva et al., 2020).

3 Data

We conduct the experiments on two datasets, the first one being the UD 2.7 OCS dataset (Zeman et al., 2020). We start with training the model on its *train* part (37 432 tokens). We perform the validation with *dev* (10 100 tokens) and preliminary tests with *test* (10 031 tokens). Before lemmatising the corpus, we perform the final training with both *train* and *dev* (47 532 tokens). UD 2.7 OCS consists of a tagged *Codex Marianus*, one of the largest and most widely accepted OCS canon texts.

The second dataset is the *Kyiv Folia* (also referred to as Kiev Folia, or Kiev Missal), which is significantly smaller (1 342 tokens), highly dissimilar both to other OCS texts (Kamphuis, 2020) and other OCS datasets, possessing two additional classes of tokens. These are punctuation marks and alphabetical digits (∽*Б*∽ ‘2’). The *Kyiv Folia* latest edition (Kyiv Folia) also required some preprocessing, such as ASCII > Old Church Slavonic Unicode transformation (Afanasev, 2020), and PoS tagging (Uludoğan, 2018) (Lyashevskaya, Afanasev, 2021). The *Kyiv Folia* language significantly differs from the language of *Codex Marianus* by some phonetic features, such as **dj* and **tj* reflexes (Kamphuis, 2020): this also complicates the seq2seq lemmatisation, as the tokens that the model is aware of from *Codex Marianus* become out-of-vocabulary (OOV), and generally data becomes more noisy. We present the summary on datasets in table 2.1.

TABLE 2.1 Summary on the datasets

Dataset	Training tokens number	Validation tokens number	Test tokens number	Overall tokens number	**tj/*dj* reflexes	Punctuation marks	Digits
Codex Marianus (UD 2.7 OCS)	37432	10100	10031	47532	South Slavic	Non-present	Non-present
Kyiv Folia	–	–	1342	1342	East Slavic	Present	Present

4 Experiment Settings

The model we employ is a hybrid system that joins rule-based, dictionary-based, and neural network (RNN) modules, each working together. The neural network is a recurrent seq2seq one, enhanced with an attention mechanism,

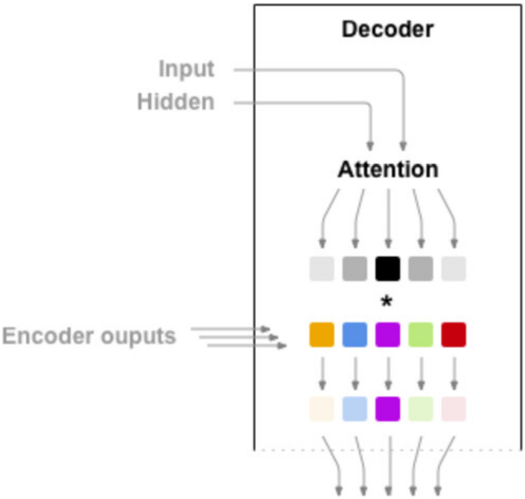

FIGURE 2.1
General implementation of seq2seq architecture
KERAS

originally used for machine translation (Sutskever et al., 2014) (Cho et al., 2014). The neural network encoder and decoder parts consist of long short-term memory (LSTM) layers (Hochreiter and Schmidhuber, 1997). The model workflow is in figure 2.1. The model decoder gets input (token), transforms it into tensors, which then are processed by hidden layers with attention, and the encoder transforms the results into output (lemma). We implement this architecture via Keras API (Keras). The rules, the dictionary, and the neural network are presumed to cover most of the tokens of the datasets described in the previous section. We publish the source code for reproducibility purposes.[1]

The model behaviour may be changed with the tuning of its hyperparameters, such as epochs number (defaults to 40), batch size (defaults to 128), hidden dimensions number (defaults to 256), optimiser (the proposed is RMSprop (Hinton et al., 2012)), loss (categorical cross entropy) and activation (softmax (Goodfellow et al., 2016)) functions, as well as the patience of the early stopping callback function (Prechelt, 1998) (by default, no early stopping).

A user may configure the model training in certain ways. For instance, the model may split tokens into n-grams, with *n* given by a user (by default we do not perform the split). The same may be executed for a lemma, if the user provides the specific instruction (again, by default the model does not split lemmata). For instance, if *n* equals 2, the word *корабь* ‘ship-ACC’ (the lemma is *корабль* ‘ship’) is to be split into 2-grams *ко*, *ор*, *ра*, *аб* and *бь*. If the user does

[1] https://github.com/The-One-Who-Speaks-and-Depicts/OCS-corpus-lemmatiser

not select a lemma split, the training pairs are *ко—корабль*, *ор—корабль*, and so forth until *бь—корабль*. If the user selects a lemma split, the lemma is also split into 2-grams *ко*, *ор*, *ра*, *аб*, *бл*, *ль*, and the pairs are *ко—ко*, *ор—ор*, and so forth until *бь—бл*. As a lemma is longer than a token, the last pair is ь# (with # meaning empty symbol)—*ль*.

If the model trains on *n*-gram pairs, it may form a prediction in two different ways. The first one is taking the first letter of each predicted *n*-gram except the last, which is taken as a whole (*back* approach). The second is vice versa, taking the whole first *n*-gram, and then the final letter of all the following *n*-grams (a *forward* approach, the default one). In such a way, the predictions *жи*, *им*, *ла* are more accurate for getting the lemma *жила* 'vein' with the *backwards* approach (here the model yields a correct option *жила*, and not *жима*). Another set of predictions, *жи*, *ил*, *ма* is going to perform better with the *forward* option switched on (yet again the model yields *жила* as compared to *жима*).

A quasi-stemming technique is stripping both the word and its lemma of every letter but the last stem letter and the inflection (for instance, for the pair *вашихъ* 'yours-GEN'—*вашь* 'yours' the result is the pair *шихъ—шь*), these being used for training. We prohibit the use of quasi-stemming, if a token is formally equal to its lemma (for instance, *рабъ* 'slave-ACC'—*рабъ* 'slave'). The model does not perform quasi-stemming by default.

The model may also employ the technique of using additional information, a token PoS tag, acquired via an HMM-based hybrid tool presented in Lyashevskaya and Afanasev (2021). For instance, the pair *врача* 'doctor-GEN'—*врачь* 'doctor' transforms into *врачаNOUN—врачь*.

Unfortunately, there is no possibility to enhance the model architecture with linguistically-aware custom loss functions, such as Damerau-Levenshtein distance (Damerau, 1964) (Levenshtein, 1966). The loss function in Keras does not deal with the input and the output directly, but with its transformed tensor representation, and there is not much sense in using any string similarity measures with these objects. The same nuances of Keras's internal structure block the possibility of using the Levenberg-Marquardt algorithm presented and enhanced in (Levenberg, 1944) (Marquardt, 1963).

The attempts at using a new activation function, mellowmax (Kavosh and Littman, 2017), also fail. Mellowmax is primarily used in reinforcement learning cases, and the case presented in the paper is not one of them. In addition, mellowmax is tested against a very specific architecture, designed for very specific purposes, and it is not a seq2seq transformation.

5 Evaluation Methodology

There are two different ways that we evaluate the model performance. There is no gold tagging for the *Kyiv Folia* dataset, so we evaluate its lemmatisation manually. On the contrary, the UD dataset possesses the gold tagging, and we score its efficiency automatically via two steps.

For the first step, we acquire the accuracy score, the simplest of all metrics to implement (Milintsevich, Sirts, 2020) (Akhmetov et al., 2020). We employ this metric mostly for the comparison with the other lemmatisation models, like the OCS lemmatisers that already exist (Podtergera, 2016), as well as the universal lemmatisers (Straka et al., 2017) (Bergmanis and Goldwater, 2018) (Kanerva et al., 2018). But we do not treat its results, though helpful for general assessment of model performance, as a clear-cut solution.

Accuracy score, despite its ability to push any model to its limits, lacks insightfulness in three critical aspects. Accuracy score is strictly binary. If a model generalises better (capturing most of token change paradigms), but is more likely to produce insignificant errata, the accuracy score evaluates it worse than the model that generalises poorer, but fits better for just the most frequent token change paradigms. Accuracy score fines a seq2seq model that generates output character-by-character even if most mistakes are one letter substitution, addition, deletion, or transposition. The final issue is that accuracy score does not provide a researcher with insights on what are the most common and/or the most critical erroneous patterns that the model reproduces. All these issues constitute the critique of the existing evaluation methods, having sparked in recent years in NLP in general (Ethayarajh and Jurafsky, 2020), and some specific branches (Stringham and Izbicki, 2020) (Gupta et al., 2020).

We propose alternatives to the accuracy score for lemmatisation evaluation: string similarity measures. These metrics compare gold string and predicted string character-by-character, which is a more fine-grained approach. There are different string similarity measures. Some of them, like Hamming distance (Hamming, 1950) that compares equal strings, are not fit for evaluating lemmatisation (lemma and token are rarely of equal size). Others, like the longest common substring, require significant enhancements, like attention to the position of the match between the prediction and the gold lemma.

The string similarity distance that we use for evaluation is Levenshtein distance (Levenshtein, 1966). It counts additions, deletions, and substitutions between two strings. Some works have already implemented the Levenshtein distance (Kanerva et al., 2020) (Metheniti et al., 2020) (Zalmout and Habash, 2020), with some attempts of enhancing it (Znamenskij. 2017) (Ganesh et al., 2020). However, the enhancements have not yet been adopted and tested as a full-fledged metric, so we cautiously avoid their implementation. The only enhancement we use is the other well-established metric, Damerau-Levenshtein distance, that adds transposition to the possible set of operations (Damerau, 1964).

As an alternative, we propose Jaro-Winkler distance (Jaro, 1989) (Winkler, 1990), which is similar to Levenshtein distance but penalises a model that generates incorrect beginnings of strings. This helps to check whether the model understands the concepts of lemmatisation (via strong fines of the randomly generated strings) and suppletion (a model will score lower, if it uses regular token change rules for irregular paradigms).

To evaluate the results of a model on the dataset, we use the mean for each of the metrics, including accuracy score. For string similarity measures, we also normalise the results by getting rid of outliers (Grubbs, 1969). Normalisation is helpful for the evaluation of poorer performing models. It helps to test whether their performance is due to the lack of training data, with an overall score being low but normalised string similarity measures being close to the non-normalised, or due to the overfitting, with overall score being high, but normalised string similarity measures being much lower that the non-normalised ones. In the latter case, the model clearly memorised the exact input-output pairs.

We provide the short summary of methods and their contribution to overall evaluations in table 2.2.

TABLE 2.2 Evaluation methods and their contribution

Method	**Short description**	**Contribution**
Accuracy score	Binary comparison of whether the gold and predicted string match	The most popular metric that allows for comparison against other models
Levenshtein distance	Counting additions, deletions, and substitutions between gold and predicted string	A more fine-grained evaluation that provides the researcher with insights on the particular lemmatisation patterns that a model under consideration produces
Damerau-Levenshtein distance	Counting additions, deletions, substitutions, and transpositions between gold and predicted string	Provides the researcher with a more detailed account as compared to the Levenshtein distance

TABLE 2.2 Evaluation methods and their contribution (*cont.*)

Method	Short description	Contribution
Jaro-Winkler distance	Counting transpositions between gold and predicted string with prefix index that controls for the correct beginning	Checks how well the model captures the essence of lemmatisation including suppletion phenomenon
Normalisation	Counting the mean metric score without outliers	Divides the models that learn input-output pairs from the models that learn lemmatisation rules.

We compare the results with the baseline that utilises the lemma-as-token approach. With the use of the baseline, the lemma for *pacnʌma* 'crucified' is *pacnʌma* itself, not the expected *pacnʌmu* 'crucify'.

The second step of the evaluation process is the errata analysis. We restrict ourselves to the most significant errata that we extract with outlier recognition methods (Grubbs, 1969) from the results of Damerau-Levenshtein and Jaro-Winkler distances. During the testing process, we save them into a separate file and manually classify and analyse them on the basis of the linguistic information.

6 Results and Analysis

The baseline is a lemma-as-token approach, a script that takes each token as its own lemma. Table 2.3 presents the results of how the baseline has been evaluated by all the proposed metrics.

TABLE 2.3 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 rounded to 2 decimal places. A = Accuracy score, L = Levenshtein distance, D-L = Damerau-Levenshtein distance, J-W = Jaro-Winkler distance, R = raw, and N = normalised.

TABLE 2.4 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	*47.76%*	11.3	11.3	9.37	9.37	0.57	0.57
2	*47.76%*	11.64	11.64	9.71	9.71	0.55	0.55

Notes: The best results for each metrics are highlighted in bold. If two or more models share the best results, these are *italicised*.

The metrics demonstrate disagreement. Accuracy score and Levenshtein distance demonstrate low efficiency of the baseline method. Damerau-Levenshtein distance, on the other hand, shows that character transposition causes a high percentage of errata, which in baseline conditions means that a lot of word changing paradigms involve two symbols being swapped. Jaro-Winkler distance metric is relatively low: suppletive forms, like *мѧ* 'I-ACC' from *азъ* 'I', are quite frequent in the dataset. Normalised metrics support the hypothesis.

We start to prepare the seq2seq model that we are going to test against the baseline. We do this via the selection of the training parameters. The initial configuration does not employ any particular heuristics, like early stopping, quasi-stemming, or prediction with PoS information. The batch hyperparameter equals 128. The models form lemmata by 2-grams. The lemma generation priority is the forward one: we take the first 2-gram as a whole, and then add the second character of each new generated 2-gram. The second configuration has the backwards lemma generation priority: we obtain the first character from each generated n-gram, starting with the first, and then we use the final 2-gram as a whole.

We present the results in table 2.4.

The configurations fail by each metric. The accuracy score comparison helps to decide whether to use forward or backwards lemma generation priority for the later experiments.

Each metric judged the results of the models quite harshly. The accuracy score suggests that the models carried out the task similarly, and it is unclear whether the backwards lemma generation priority actually enhanced or damaged the performance. Models do not stop generalising, as the same values for raw and normalised metrics show. However, they do not generalise enough even for this dataset, they just generate random sequences: the extremely low Jaro-Winkler distance value is a proof of such behaviour. Overall, string similarity measures demonstrate slightly worse behaviour of a model with a backward

TABLE 2.5 The evaluation results of models 1 and 3

Metrics	A	L(R)	L(N)	D-L(R)	D-L(N)	J-W(R)	J-W(N)
1	47.76%	11.3	11.3	9.37	9.37	0.57	0.57
3	18.3%	21.07	24.98	20.33	24.98	0.33	0.33

TABLE 2.6 The evaluation results of models 1 and 4 to 10

Metrics	A	L(R)	L(N)	D-L(R)	D-L(N)	J-W(R)	J-W(N)
1	47.76%	11.3	11.3	9.37	9.37	0.57	0.57
4	51.88%	9.91	9.91	7.81	7.81	0.65	0.65
5	56.71%	12.79	12.79	10.5	10.5	0.67	0.67
6	62.1%	11.1	11.1	8.59	8.59	0.69	0.69
7	67.77%	11	11	8.26	8.26	0.74	0.74
8	73.05%	9.34	5.41	6.39	1.76	0.79	0.79
9	77.47%	8.75	4.02	5.61	24.92	0.81	0.43
10	84.52%	4.45	4.4	1.03	6.66	0.85	0.77

lemma generation priority. For the further experiments, we will use the forward lemma generation priority.

The second experiment introduces an alternative method of predicting the lemma from each *n*-gram of the given token. Here, each metric significantly decreases, which shows that the *n*-grams themselves do not provide enough information for the seq2seq model. String similarity measures show that, generally, the model demonstrates extremely poor results. The almost exact coincidence between Levenshtein and Damerau-Levenshtein distance scoring results highlight these. It means that there are hardly any transposition errors. The models either add or delete too many symbols or generate completely wrong output. The rare cases of transpositions errors (as the results of the Damerau-Levenshtein distance measurement are slightly better than the results of the Levenshtein distance measurement) are positive outliers: the normalised Levenshtein and Damerau-Levenshtein distance scores are identical. We show the results in table 2.5.

Table 2.6 demonstrates an incrementing of *n* in n-grams parameter. We start with minimal *n*=2 (setup 1). We increase *n* by 1 for the following 6 (4 to 9) experiments, achieving maximal *n*=8 (that captures almost every word in the dataset as a whole). The last experiment in the series (10) uses n=length(t), where t is a token that we lemmatise.

We see the general tendency for all the metrics to score higher with the incrementation of n: an accuracy score gets closer to 100 %, Jaro-Winkler distance—to 1, and Levenshtein and Damerau-Levenshtein distances—to 0.

However, the only metric that behaves in a robust way is the accuracy score. This is due to the hybrid nature of the lemmatiser, mainly its dictionary part. The model gets to remember significantly more words with each increase in the maximal length of n-grams. So, the question of which neural network module of model 10 actually performs better than the others, is left open. Models 1 to 7, despite demonstrating robust behaviour (there were no outliers), have not achieved great success, demonstrating mediocre results.

Model 10 (which does not split words into n-grams of any kind) demonstrates the best results in accuracy score and each of the raw string similarity measurements. However, normalised string similarity metrics show that the model has quite a number of positive outliers in the results. It seems to be overfitting, and, probably, relies only on the inner dictionary.

Model 9 (that splits words into 8-grams) is the best one by normalised Levenshtein distance, and the normalised Levenshtein distance score for it is far less than the raw one, which means that at least some of the outliers are negative ones. Yet, its normalised Damerau-Levenshtein distance measurements are very high, and the Jaro-Winkler ones are very low, both being among the worst in this series of experiments. This may signal that the model does not behave in a very robust way. Moreover, it actually performs worse than the accuracy score shows, as this metric does not take outliers into consideration. In addition, very high mean raw Jaro-Winker distance suggests that the positive outliers are the words for which the starting n-grams are predicted better. It is clearly the dictionary module that helps this model to achieve its high accuracy score. It is a crucial fallacy for the task of lemmatisation of the Old Church Slavonic-like languages as they are very heterogeneous.

Model 8 (that splits words into 7-grams) is the least successful by the criterion of accuracy among those three. However, its behaviour seems to be more robust. Its raw and normalised Jaro-Winkler distances are identical, so there are no outliers by these metrics, and, probably, the model is equally good at predicting n-grams at the end and at the beginning of the word. Its normalised mean Jaro-Winkler distance is the best among all the models. Its raw Levenshtein and Damerau-Levenshtein distances values are quite low. But the most important fact here is that the outliers among these models' predictions are exclusively negative and quite rare. Normalised mean Damerau-Levenshtein distance is the lowest one in this series of experiments.

TABLE 2.7 The evaluation results of models 8, 10, and their modifications

Metrics	A	L(R)	L(N)	D-L(R)	D-L(N)	J-W(R)	J-W(N)
8	73.05%	9.34	5.41	6.39	1.76	0.79	*0.79*
10	*84.52%*	4.45	4.4	1.03	6.66	0.85	0.77
11	73.05%	9.96	9.96	7.01	7.01	0.75	0.75
12	*84.52%*	6.94	*4.01*	3.52	22.7	0.87	0.48
13	73.05%	8.85	8.85	5.9	5.9	0.77	0.77
14	*84.52%*	6.2	4.4	2.78	17.95	*0.9*	0.45
15	73.05%	5.87	4.87	2.92	1.66	0.79	*0.79*
16	*84.52%*	6.5	*4.01*	3.08	19.92	*0.9*	0.45

Thus, the two most advanced metrics show that mode 8l is the most robust if not the best one. However, by most metrics and by accuracy score the best model is the 10th one. There is no obvious choice, and in the next series of experiments we attempt at both stabilising model 10 and enhancing the functionality of the 8th one. Model 11 implements joining PoS tag to model 8, the 12th one is the same augmentation implementation for model 10. Models 13 and 14 introduce quasi-stemming for the models in the same order, and models 15 and 16 join these two techniques together. The results are in table 2.7.

It still seems to be unclear which architecture performs better.

The models that we trained on full tokens still demonstrate a much better accuracy score. However, they show an increasing lack of stability, with the difference between raw and normalised metrics going significantly upwards with possible modifications. What is more, the actual instability of the model, similar to the 8-gram one, is revealed by the Damerau-Levenshtein distance and the Jaro-Winkler distance measurement results, with normalised metric values getting much higher than the raw ones.

On the contrary, the model that learns on 7-grams becomes more robust, with differences between raw and normalised Levenshtein and Damerau-Levenshtein distances becoming smaller. In the case of Damerau-Levenshtein distance, it eventually provides the lowest result among all the models. Jaro-Winkler distance measurements do not contain any outliers, and the normalised one remains the best among all the experiments.

Thus, the model that trains on 7-grams with PoS addition and quasi-stemming implementation demonstrates the most robust behaviour with the tendency of becoming more robust and effective. We use this architecture in further experiments.

TABLE 2.8 The evaluation results of model 15 and its enhancements

Metrics	A	L(R)	L(N)	D-L(R)	D-L(N)	J-W(R)	J-W(N)
15	*73.05%*	5.87	4.87	2.92	1.66	0.79	0.79
17	*73.05%*	*5.13*	5.08	*2.17*	*1.45*	0.74	0.74
18	*73.05%*	*5.13*	5.07	*2.17*	1.47	0.74	0.74
19	*73.05%*	8.98	8.98	6.02	6.02	0.77	0.77
20	*73.05%*	8.62	8.62	5.67	5.67	0.77	0.77
21	*73.05%*	9.08	9.08	6.12	6.12	0.76	0.76
22	*73.05%*	8.99	8.99	6.03	6.03	0.8	0.8
23	*73.05%*	*5.13*	5.08	2.18	*1.45*	0.74	0.74

These experiments consist of adjusting the models' hyperparameters, namely batch size increase and decrease and the early stopping implementation. The results are in table 2.8.

The experiments 17 to 20 introduced the changing of the batch size to 64, 256, 512, and 32 respectively. Only the 256 and 64 batch models show some sort of growth above the results of the original model with the batch size of 128. They show some decrease in the Jaro-Winkler distance measurement results, while being compared to the original model, and normalised Levenshtein distance is higher for them. However, they also demonstrate a decrease in both raw and normalised Damerau-Levenshtein distance measurement results. It is impossible to tell which model actually performs better.

To resolve this issue, we introduce early stopping for each model. We use the early stopping of 2, 3, and 4 epochs, with the former not having enough time to fire out during the course of the experiment. Thus, experiment 21 is the early stopping of 2 epochs for the model with batch size 128, the experiment 22—with the batch size of 64, and the experiment 23—with the batch size of 256. These experiments do not demonstrate any sort of increase. At their best, the models either grow by just some metrics or remain robust. Model 22 actually shows an increase in the Jaro-Winkler distance, which, in addition to other string similarity metrics results worsening, shows that it was better in predicting only the beginning of the lemma. Model 21 scores the worst among three. The model 23 score demonstrates an insignificant decrease in the normalised Levenshtein distance and the raw Damerau-Levenshtein distance results. However, it performs better, according to the normalised Damerau-Levenshtein distance results. There is no best model here, only the most robust one, the batch size of which is 256. As it demonstrates better results by the

TABLE 2.9 The evaluation results of the baseline model and model 18

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
18	73.05 %	5.13	5.07	2.17	1.47	0.74	0.74

stricter Levenshtein distance metrics without early stopping, we compare it with the baseline in table 2.9.

Model 18 shows a better accuracy score and a slightly better normalised Damerau-Levenshtein distance score. However, for every other metric there is a significant loss in quality, if we compare the model with the baseline. As the metric of normalised Damerau-Levenshtein distance shows better results, we may not be sure that the hybrid model is relying only on its dictionary part. Higher Jaro-Winkler distance scores of the baseline show that the baseline is (naturally) better at predicting the beginnings of the lemmata. It just generates them in any case but the suppletive ones. Model 18 is not perfect, but it seems to be robust and effective enough for tests on the heterogeneous datasets.

There are different ways of classifying model 18 outliers. These may be the value (from 11 to 18), the metrics themselves (Levenshtein vs. Damerau-Levenshtein, there are no outliers by the accuracy score, or the Jaro-Winkler distance), or the PoS distribution. The model makes mistakes only in nouns (such as *иѥроусалимъ* 'Jerusalem'), verbs (*прѣдълагати* 'to suggest'), including participles (*благословлѥнъ* 'blessed'), adjectives (*самарѣньскъ* 'Samarian'). There is even one compound numeral (*дъва.на.десѧте* '12'). The one possible explanation is that there is some sort of correlation between syntactic freedom of words and the model error rate. However, we propose a much more trivial and, at the same time, probable explanation. These words are long, they are no less than 10 characters in length. The tokens themselves are outliers by some metrics, so are the evaluation results of the model that processes them. This may be supported by the evidence of predictions. They are split into two kinds: the empty sequence (cf. gold *съвѣдѣтельствовати* 'to witness' / predicted (*empty*)), and the sequence of symbol *ш*, continued by the different number of point punctuation marks (cf. gold *наставьникъ* 'teacher' / predicted *ш./ш../ш…/ш…./ш…..* '(meaningless)'). We may see that the predictions themselves are quite short, the longest prediction is less than 10 symbols in length. This leads to the conclusion that the model struggles with generating long lemmata from longer input tokens.

To test this hypothesis, we use the new dataset, *Kyiv Folia*. To work with this data that contains new kinds of tokens, such as punctuation marks and numbers, we enhance the model with rules that treated punctuation marks and numbers as their own lemmata, which seems to be the sole correct solution linguistically, as there are no full gold data for this corpus that is both accessible and machine-readable. The best option is the Manuscript project (Manuscript), which human researchers may refer to, while evaluating the model results, but not the machine itself. Thus, the further analysis is more qualitative than quantitative.

After we lemmatise *Kyiv Folia*, we load it into a special system for automatically creating a dictionary from a chosen set of texts that are present in the corpus. This system is written in C# (web application backend, ASP.NET Core, Razor Pages) and JavaScript (web application frontend, mostly jQuery, with some VanillaJS elements) and may integrate different modules in Python. This kind of fully automated conversion from dictionary to corpus has not been widely attempted by linguists previously.

The created thesaurus helps to shape the data to the better form for analysis. With words, grouped by the lemmata, one may observe, where the main mistakes are made and of what types they are. An example of the thesaurus unit is provided in the listing below:

The dictionary unit *къ* ‘towards’.
- Къ
- *къ* ADP
- *къ* ADP
- *къ* ADP

Generally, the model demonstrates some sort of ability to generalise (as in *къ* ‘towards’ in listing 1), and the rules overall tend to work with accuracy, close to 100 % (with digits like ~*Б*~ ‘2’, or punctuation marks, such as :.: ‘five points’).

However, there are occasional fallacies, when the model produces output similar to *лвеннѳѳшгвѫѫѫлѫѫлцццццц* ‘(meaningless)’. This implies that the model does not know exactly how to react for the inputs that are significantly different from the train dataset. What is more, this reaction does not depend on the input token length.

There is a pattern: the overfit model tends to produce shorter wrong predictions on the test data, and much longer wrong predictions on the out-of-domain data (the *Kyiv Folia* is certainly out-of-domain for Old Church Slavonic (Lyashevskaya and Afanasev, 2021)). The possible explanation is that the algorithm is stopping earlier when the neural network recognizes the wrong pattern, and later, when it does not know a lot of patterns within the data.

7 Conclusion

The paper presents a set of new possible evaluation metrics for the Old Church Slavonic lemmatiser model. We use the metrics to tune it to be the most robust and scalable of all the possible options.

Each of the introduced metrics proves its usefulness in evaluation of the neural network part of the model. Generally, Damerau-Levenshtein distance is the one that motivates the decision to choose this or another architecture. Levenshtein distance helps to better see the failures of the model. Jaro-Winkler distance helps to deduce, whether the model captures the essence of the lemmatisation concept for languages like Old Church Slavonic.

The results of the metrics help to pick the best model to predict on the heterogeneous data, which assists in building the corpus-based dictionary of an Old Church Slavonic document, *Kyiv Folia*. This dictionary, having been built automatically, highlights the typical model errors.

Overall, the implementation of the new metrics is successful. Figure 2.2 shows their dynamics and how the model has been transforming into more and more robust, complex architecture. The system seems to lack an overall accuracy compared to previous UD lemmatisers (Straka et al., 2017) (Bergmanis and Goldwater, 2018) (Kanerva et al., 2018). However, it is better adapted to the out-of-domain texts from the OCS corpus. Overall accuracy on *Kyiv Folia* goes over 50 %, with significantly (up to 60 %) higher results for punctuation marks, digits, and fragmentary tokens that get to 100 %. UD lemmatisers stay under the 50 % threshold, as they are unable to recognize new classes. Thus, the rule-based part of the system partly eliminates one of the most significant failures of the encoder-decoder neural network part of the system, the inability to work with different inflection models, none of which is prevalent in the language (McCurdy et al., 2020).

The main shortcoming of the system is generating sequences precisely character by character, without actually learning inflection paradigms. One of the main proofs is the correlation between the increase of accuracy by all the metrics with the increase of the characters' number in the generated sequences. A possible future solution is to use character-level generalised embeddings, retrieved from *Codex Marianus*, thus enabling the model to benefit from the previously inaccessible enhancement (Jinman et al., 2020). Otherwise, the system needs complete redesign and augmentation with the possibility to gen-

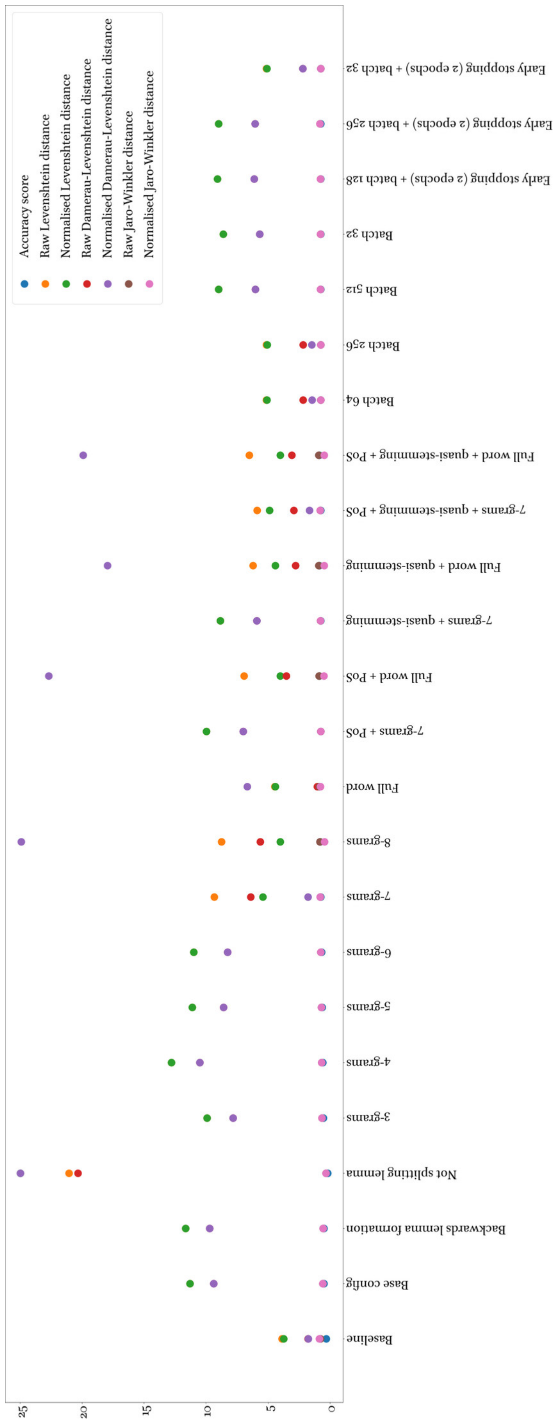

FIGURE 2.2 Results of the models by metrics

erate sequences of character n-grams, not just characters themselves (Qi et al., 2020). It is likely that the emerging technique of historical text normalisation improves the results of the system as well (Makarov and Clematide, 2020).

References

Akhmetov, I., & Pak, A., & Ualiyeva, I., & Gelbukh, A. (2020). Highly Language-Independent Word Lemmatization Using a Machine-Learning Classifier. *Computacion y Sistemas*, 24, 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*, 13–21.

Berdičevskis, A., & Eckhoff, H., & Gavrilova, T. (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"*, 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*, 1 (1), 1391–1400.

Bugakov, O. (2009). Using Ukranian National Linguistic Corpus in Lexicography. *Proceedings of the 5th open workshop "Research Infrastructure for Digital Lexicography"*, 120–124/

Cho, K., & Merrienboer, B.V., & 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*), 1724–1734.

Chrupała, G. (2006). Simple data-driven context-sensitive lemmatization. *Procesamiento del Lenguaje Natural*, 37, 121–127.

Damerau, F.J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM* , 7(3), 171–176.

Džeroski S., Erjavec T. (2000). Learning to Lemmatise Slovene Words. In J. Cussens & S. Džeroski (Eds.), *Learning Language in Logic. LLL 1999. Lecture Notes in Computer Science* (pp. 69–88). Springer.

Ethayarajh, K., & Jurafsky, D. (2020). Utility is in the Eye of the User: A Critique of NLP Leaderboards. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 4846–4853.

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*, 53–60.

Fernández, L. (2020). A contribution to Old English lexicography: Utgangan, wiðhealdan, ofersceadan, onbefeallan and ongangan. *NOWELE. North-Western European Language Evolution*, 73(2), 236–251.

Fomicheva, M., & Specia, L., & Guzmán, F. (2020). Multi-hypothesis Machine Translation Evaluation. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 1218–1232.

Gambäck B., & Olsson F., & Argaw A.A., & Asker L. (2009). Methods for Amharic Part-of-Speech Tagging. *Proceedings of the EACL 2009 Workshop on Language Technologies for African Languages—AfLaT 2009*, 104–111.

Ganesh, D., Kumar, T., Kumar, M. (2020). Optimised Levenshtein centroid cross-layer defense for multi-hop cognitive radio networks. *IET Communications*, 15(2), 245–256.

Gesmundo, A., & Samardžić, T. (2012). Lemmatisation as a tagging task. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, 2, 368–372.

Goodfellow, I., & Bengio, Y. & Courville, A. (2016) 6.2.2.3 Softmax Units for Multinoulli Output Distributions. In J.D. Kelleher (Ed.), *Deep Learning* (pp. 180–184). MIT Press.

Granger, S., & Paquot, M. (2015). Electronic lexicography goes local: Design and structures of a needs-driven online academic writing aid / Die elektronische Lexikographie wird spezifischer: Das Design und die Struktur einer auf die Benutzerbedürfnisse berzogenen akademischen Online-Schreibhilfe / La lexicographie électronique devient plus spécifique : conception et structure d'une aide à l'écriture académique. *Lexicographica*, 31(1), 118–141. https://doi.org/10.1515/lexi-2015-0007

Groenewald, H. (2007). Automatic lemmatisation for Afrikaans. North-West University, Potchefstroom Campus.

Grubbs, F.E. (1969). Procedures for detecting outlying observations in samples. *Technometrics*, 11(1), 1–21.

Gupta, A., Krishna, A., Goyal, P & Hellwig, O. (2020). Evaluating Neural Morphological Taggers for Sanskrit. *Proceedings of the Seventeenth SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, 198–203.

Hann, M. (1974). Principles of Automatic Lemmatisation. *ITL Review of Applied Linguistics*, 23(1), 3–22.

Hamming, R.W. (1950). Error detecting and error correcting codes. *The Bell System Technical Journal*, 29 (2), 147–160. https://doi.org/10.1002/j.1538-7305.1950.tb00463.x.

Hinton, G., & Srivastava, N., & Swersky K. (2012). Lectures on Machine Learning, viewed 5 December 2020, available at http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.

Howell, N., Bibaeva, M., & Tyers, F.M. (2020). Effort versus performance tradeoff in Uralic lemmatisers. *Proceedings of the 6th International Workshop on Computational Linguistics of Uralic languages*, 9–14.

Jaro, M.A. (1989). Advances in record linkage methodology as applied to the 1985 census of Tampa Florida. *Journal of the American Statistical Association*, 84, 414–420.

Jinman, Z., & Zhong, S., & Zhan, X., & Liang, Y. (2020). PBoS: Probabilistic Bag-of-Subwords for Generalizing Word Embedding. *Findings of the Association for Computational Linguistics: EMNLP 2020*, 596–611.

Kamphuis, J. (2020). *Verbal Aspect in Old Church Slavonic*. Brill.

Kanerva, J., & Ginter, F., & Salakoski, T. (2020). Universal Lemmatizer: A sequence-to-sequence model for lemmatizing Universal Dependencies treebanks. *Natural Language* Engineering, 1–30.

Kavosh, A., & Littman, M. (2017). An Alternative Softmax Operator for Reinforcement Learning. *Proceedings of the 34th International Conference on Machine Learning*, 243–252.

Kestemont, M., & de Gussem, J. (2017). Integrated Sequence Tagging for Medieval Latin Using Deep Representation Learning. *Journal of Data Mining and Digital Humanities, Special Issue on Computer-Aided Processing of Intertextuality in Ancient Languages*, 1–17.

Kyiv Folia, viewed 5 December 2020, available at http://www.schaeken.nl/lu/research/online/editions/kievfol.html

Kilgariff, A., & Kosem, I. (2012). Corpus tools for lexicographers. In S. Granger & M. Paquot (Eds.), *Electronic Lexicography* (pp. 31–55). Oxford University Press.

Kosch, I.M. (2016). Lemmatisation of Fixed Expressions: The Case of Proverbs in Northern Sotho. *Lexikos*, 26, 145–161.

Levenberg, K. (1944). A Method for the Solution of Certain Non-Linear Problems in Least Squares. *Quarterly of Applied Mathematics*, 2, 164–168.

Levenshtein, V.I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8), 707–710.

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*, 29–34.

Lyashevskaya, O., & Afanasev, I. (2021). An HMM-based PoS Tagger for Old Church Slavonic. *Jazykovedny Casopis*, 72(2), 556–567.

Makarov, P., & Clematide, S. (2020). Semi-supervised Contextual Historical Text Normalization. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7284–7295.

Manuscript, viewed 5 December 2020, available at http://manuscripts.ru/

Martins, P., & Abbasi, M. (2020). End to end distance measurement algorithms in biology sequences. *2020 15th Iberian Conference on Information Systems and Technologies* (*CISTI*), 1–6.

Mathiesen, R. (1984). The Church Slavonic Language Question: An Overview (IX–XX centuries). In R. Picchio & H. Goldblatt & S. Fusso S. (Eds.), *Aspects of the Slavonic languages* (pp. 45–56). Yale Concilium on International and Area Studies: New Haven.

Marquardt, D. (1963). An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *SIAM Journal on Applied Mathematics*, 11(2), 431–441.

Mathis, A., & Yüksekgönül, M., & Rogers, B., & Bethge, M., & Mathis, M.W. (2020). Pretraining boosts out-of-domain robustness for pose estimation (arXiv preprint), viewed 5 December 2020, available at https://arxiv.org/pdf/1909.11229.pdf

McCurdy, K., & Goldwater, S., & Lopez, A. (2020). Inflecting When There's No Majority: Limitations of Encoder-Decoder Neural Networks as Cognitive Models for German Plurals. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 1745–1756.

Metheniti, E., & Neumann, G., & Genabith, J. (2020). Linguistically inspired morphological inflection with a sequence to sequence model (arXiv preprint), viewed 5 December 2020, available at https://arxiv.org/pdf/2009.02073v1.pdf

Milintsevich, K., & Sirts K. (2020). Lexicon-Enhanced Neural Lemmatization for Estonian. *Human Language Technologies—The Baltic Perspective*, 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*. 1–6.

Nerbonne, J. (1992). Natural language disambiguation and taxonomic reasoning. *Proceedings of DFKI Workshop on Taxonomic Reasoning*, 40–47.

Qi, W., & Yan, Y., & Gong, Y., & Liu, D., & Duan, N., & Chen, J., & Zhang, R., & Zhou, M. (2020). ProphetNet: Predicting Future N-gram for Sequence-to-Sequence Pretraining. *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2401–2410.

Paquot, M. (2012). The LEAD dictionary-cum-writing aid: an integrated dictionary and corpus tool. In S. Granger & M. Paquot (Eds.), *Electronic Lexicography* (pp. 163–185), Oxford University Press.

Plisson, J., & Lavrac, N., & Mladenić, D & Erjavec, T. (2008). Ripple Down Rule learning for automated word lemmatisation. *AI Communications*, 21, 15–26.

Podtergera, I. (2016). SlaVaComp-Lemmatizer: a Lemmatization Tool for Church Slavonic. *Proceedings of El'Manuscript-2016: Textual Heritage and Information Technologies*, 212–221.

Polivanova, A. (2013). *Staroslavianskij jazyk. Grammatika. Slovari.* Russkij Fond Sodejstvija Obrazovaniju i Nauke.

Radziszewski, A. (2013). Learning to lemmatise Polish noun phrases. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 1, 701–709.

Rundell, M. & Kilgariff, A. (2011). Automating the creation of dictionaries: where will it all end? In F. Meunier, & S. De Cock & G. Gilquin & M. Paquot (Eds), *A Taste for Corpora. A tribute to Sylviane Granger* (pp. 257–281). Benjamins.

Sangawa, K.H., & Erjavec, T., & Kawamura, Y. (2009) Automated Collection of Japanese Examples from a Parallel and a Monolingual Corpus. *eLEX 2009 Book of abstracts, Lexicography in the 21st century: New challenges, new applications*, 105–107.

Sangawa, K.H., & Erjavec, T. (2012). JaSlo: Integration of a Japanese-Slovene Bilingual Dictionary with a Corpus Search System. *Acta Linguistica Asiatica*, 2(3), 125–140.

Schryver, G.-M. de, & Taljard, E. (2007). Compiling a Corpus-based Dictionary Grammar: An Example of Northern Sotho. *Lexikos*, 17, 37–55.

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*, 88–99.

Stringham, N., & Izbicki, M. (2020). Evaluating Word Embeddings on Low-Resource Languages. *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems (Eval4NLP)*, 176–186,

Sutskever, I., & Vinyals, O., & Le, Q.V. (2014). Sequence to Sequence Learning with Neural Networks. *NIPS*, 1–9.

Tamburini, F. (2013). The AnIta-Lemmatiser: A Tool for Accurate Lemmatisation of Italian Texts. *Proceedings of EVALITA 2012*, 266–273.

Uludoğan, G. (2018), HMM POS tagger, viewed 5 December 2020, available at https://github.com/gokceuludogan/hmm-pos-tagger

Winkler, W.E. (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*. 354–359.

Zalmout, N. & Nizar H. (2020). Joint Diacritization, Lemmatization, Normalization, and Fine-Grained Morphological Tagging. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 8297–8307.

Index of Terms

In order to guarantee open access and full searchability, research in historical lexicography and lexicology must follow the same directions as the evolution of the Internet, which has moved from hypertext-based resources to more significative services and products stored and disseminated through databases and, more recently, through knowledge bases. Against this background, this book addresses specific questions like What is involved in the digitisation of linguistic data? What annotation systems can give rise to datasets compatible with knowledge bases? What standards are needed to reach full searchability? What sources and methods can be used to gather the lemmas of a historical dictionary? What determines the obsolescence of lexicographical resources?

Javier Martín Arista is Professor of Old English and Linguistics at the University of La Rioja. He is the principal investigator of a research project in the corpus linguistics and lexicography of Old English with methods of digital humanities, computational linguistics, natural language processing, and artificial intelligence.

Ana Elvira Ojanguren López is Lecturer in English at the University of La Rioja. She earned her doctorate with a thesis dealing with the syntax and semantics of Old English within the framework of Role and Reference Grammar. She has published in several specialised journals, like *Journal of Historical Linguistcs*, and is the author of the book *Predications in competition and the rise of serial verbs in English* (Peter Lang, 2024).

ISSN 0921-5034
brill.com/lc

ISBN 978-90-04-70265-3
9 789004 702653

Language
and
Computers
Studies in Digital Linguistics