Initial Normalization of User Generated Content: Case Study in a Multilingual Setting

Bagdat Myrzakhmetov, Zhandos Yessenbayev and Aibek Makazhanov
National Laboratory Astana
53 Kabanbay batyr ave., Astana, Kazakhstan
E-mail: {bagdat.myrzakhmetov, zhyessenbayev, aibek.makazhanov}@nu.edu.kz

***Abstract*—We address the problem of normalizing user generated content in a multilingual setting. Specifically, we target comment sections of popular Kazakhstani Internet news outlets, where comments almost always appear in Kazakh or Russian, or in a mixture of both. Moreover, such comments are noisy, i.e. difficult to process due to (mostly) intentional breach of spelling conventions, which aggravates data sparseness problem. Therefore, we propose a simple yet effective normalization method that accounts for multilingual input. We evaluate our approach extrinsically, on the tasks of language identification and sentiment analysis, showing that in both cases normalization improves overall accuracy.**

***Index Terms*—user generated content, normalization, code switching, transliteration**

I. INTRODUCTION

User generated content (UGC) generally refers to any type of content, i.e. photo, video, audio, text, created by Internet users. In computational linguistics (CL) and natural language processing (NLP) communities UGC is often associated with user generated text, and particularly, noisy text, such as tweets and user comments. UGC is notoriously difficult to process due to prompt introduction of neologisms, e.g. *esketit* (stands for *let's get it*, pronounced [ɛɕˈkerɛ]), and peculiar spelling, e.g. *b4* (stands for *before*). Moreover speakers of more than one language tend to mix them in UGC (a phenomenon commonly referred to as code-switching) and/or use transliteration (spelling in non-national alphabets). All of this increases lexical variety, thereby aggravating the most prominent problems of CL/NLP, such as out-of-vocabulary lexica and data sparseness.

It has been repeatedly shown that NLP methods struggle when applied to UGC directly [1]-[4] and that certain preprocessing is required for them to work properly. Such preprocessing is commonly referred to as lexical normalization or simply normalization. To this end, research on UGC normalization is of utmost interest to NLP community and for the past three years there have been held three shared task competitions in three consecutive WNUT workshops [5]-[7].

Kazakhstani segment of Internet is not except from noisy UGC and the following cases are the usual suspects in wreaking the "spelling mayhem":

- *spontaneous transliteration* – switching alphabets, respecting no particular rules or standards, e.g. Kazakh word "*біз*" (*we* as pronoun; *awl* as noun) can be spelled in three additional ways: "*биз*", "*быз*", and "*biz*";
- *use of homoglyphs* – interchangeable use of identical or similar looking Latin and Cyrillic alphabets, e.g. Cyrillic letters "*е*" (U+0435), "*с*" (U+0441), "*і*" (U+0456), and "*р*" (U+0440) in the Kazakh word «*есірткі*» (*drugs*) can be replaced with Latin homoglyphs "*e*" (U+0065), "*c*" (U+0063), "*i*" (U+0069), and "*p*" (U+0070), which, although appear identical, have different Unicode values;
- *code switching* – use of Russian words and expressions in Kazakh text and vice versa;
- *word transformations* – excessive duplication of letters, e.g. "*керемееет*" instead of "*керемет*" (*great*), or segmentation of words, e.g. "*к е р е м е т*" or "*к-е-р-е-м-е-т*".

In this work we propose an approach for *initial* normalization of UGC. Here an important distinction must be drawn. Unlike with lexical normalization [1], for initial normalization we do not attempt to recover standard spelling of ill-formed words, in fact, we do not even bother detecting those. All that we really care about at this point is to provide an intermediate representation of the input UGC that will not necessarily match its lexically normalized version, but will be less sparse. Thus, we aim at improving performance of downstream applications by reducing vocabulary size (effectively, parameter space) and OOV rate. To this end, initial normalization does two things: (i) converts the input into a common script (Russian Cyrillic based alphabet with some omissions); (ii) recovers word transformations and does various minor replacements. Difference between lexical and initial normalization is depicted by the example in Table I. Notice how for a given Kazakh text lexical normalization increases and initial normalization decreases the number of unique characters.

Our approach amounts to successive application of three straightforward procedures: (i) homoglyph resolution, (ii) common script transliteration, (iii) replacement and transformation. To assess the extent of data sparseness reduction we calculate the basic statistics, such as vocabulary size, token-type ration, and OOV rate, for raw and normalized data and show that our approach substantially reduces lexical variety. In addition to that we perform extrinsic evaluation of our approach testing it in the framework of language identification and sentiment analysis tasks. In both cases we report improvement in terms of per-language and overall accuracy.

TABLE I.

EXAMPLE OF INITIAL VS LEXICAL NORMALIZATION

Normalization	Text	Vocabulary size (chars)
No normalization	*Пенсияга ерте шыгады, б**і**здег**і**деи емес*	17
Lexical	*Пенсияға ете шығады, біздегідей емес*	18
Initial	*пенсыяга ерте шыгады быздегыдеы емес*	15

II. INITIAL NORMALIZATION

Initial normalization occurs in three stages: (1) homoglyph resolution; (2) common script transliteration; (3) replacement and transformation. In this section we describe all of these stages in greater detail.

Homoglyph resolution as a problem for Kazakh NLP was first discussed by Assylbekov et al. [8]. Following their description of the task, we develop the following relatively simple algorithm:

```
given: text, T;
(1) split T into words by spaces;
(2) for each word w ∈ T:
    (3) L = LAT(w); # number of latin
                      letters in w;
    (4) C = CYR(w); # number of cyrillic
                      letters in w;
    (5) if L==0 or C==0, then go to (2);
    (6) H = number of homoglyphs in w;
    (7) if H==0, then go to (2);
    (8) A = number of alphabetic characters
          in w;
    (9) if H==A, then go to (2);
   (10) w1 = HCYR(w); # replace all
                   homoglyphs with cyrillic
                   analogues
   (11) if CYR(w1)==A, then w = w1,
                            go to (2);
   (12) w2 = HLAT(w); # replace all
                   homoglyphs with latin
                   analogues
   (13) if LAT(w2)==A, то w = w2.
```

The algorithm is performed in a linear time proportional to the number of words in the text. Homoglyphs are listed in Table III (symbols with an asterisk), which contains common script transliteration rules.

In Kazakhstan Kazakh is written in Cyrillic alphabet that uses all letters of the Russian alphabet and 9 additional national letters. Kazakh Cyrillic is often gets transliterated into Russian Cyrillic and Latin by users (especially on mobile); hence the script needs to be brought into some common form. Our transliteration procedure translates symbols of the Latin alphabet and national symbols of the Kazakh Cyrillic alphabet into Russian Cyrillic, the alphabet chosen as a common script. Note that the term «common script», we have chosen, has nothing to do with the formal rules of spelling or the reform of the alphabet of the Kazakh language. The common script in our case is just a common denominator for the three alphabets used in the Kazakh-Russian environment. We do not claim correctness or uniqueness of the proposed transliteration rules.

Technically, the transliteration procedure is implemented by a simple algorithm that reads in the input (character at a time) and, upon matching any of the characters listed in the «Latin» and «Kazakh Cyrillic» columns, replaces it by a common script analogue; other characters are ignored.

TABLE II.

THE COMMON SCRIPT TRANSLITERATION RULES

Latin	Common script	Kazakh Cyrillic		Latin	Common script	Kazakh Cyrillic
A	а*	ә		p	п	
B	б			q	к*	
C	с*			r	р*	
D	д			s	с*	
E	е*			t	т *	
F	ф			u	у*	Ү, Ұ
G	г	ғ*		v	в*	
H	х*	h*		w	ш	
I	ы	і*		x	х*	
G	ж			y	ы	і*
K	к	қ		z	з	
L	л			ch	ч	
M	м			kh	х*	h*
N	н*	ң		sh	ш	
O	о*	ө		zh	ж	
	ё → е				щ → ш	
	и, й → ы				ь, ъ → «»	

TABLE III.

DATA SET STATISTICS

Language	Positive	Neutral	Negative	Total
Kazakh	3660	8929	4592	17181
Russian	2167	4632	2541	9340
Mixed	168	271	276	715
Total	5995	13832	7409	27236

Lastly we perform the replacement and transformation procedure. To this end, we use regular expressions to reduce duplications and adjoin segmentations. In the former case we replace two or more consecutive occurrences of the same letter with just one occurrence, e.g. "*керемееет*" becomes "*керемет*". In the latter case we delete spaces (any number) between three or more single letters, e.g *"к е р е м е т"* becomes "*керемет*".

III. EXPERIMENTS AND EVALUATION

In this section we describe our experiments and discuss the results. We begin by describing our data set and providing intrinsic evaluation in the form of per-token statistics. We then proceed to report on extrinsic evaluation in the form of language identification and sentiment analysis tasks.

A. Data Set

We have collected a total of 27 236 comments from the comments sections of the three of the most popular Kazakhstani online news outlets, namely `nur.kz`, `tengrinews.kz`, and `zakon.kz`. Bearing in mind the need to perform language identification and sentiment analysis on the collected data, we set up a semi-automatic annotation process, where a standard `Python scikit-learn` [9] implementation of Naïve Bayes classifier [10] was trained over unigram and bigram character sequences on randomly selected and manually labeled 1100 comments. This model showed perfect accuracy on a 10-fold cross validation on the training set. With the help of this model the remainder of the data was automatically labeled for language ID. After that four annotators were instructed to manually label the data for sentiment polarity (on a three class scale) and at the same time correct possible errors of the language identifier. Thus, we have obtained the data set whose statistics is given in Table III.

TABLE IV.

INTRINSIC EVALUATION STATISTICS

Data	Voc. size	# tokens	TTR, %	OOV, %
Kazakh-R	58942	242068	24.35	17.89
Kazakh-N	39566	244952	16.15	11.12
Russian-R	32915	159341	20.66	15.51
Russian-N	28332	149020	19.01	13.88
Mixed-R	7271	12863	56.53	48.37
Mixed-N	6009	12970	46.33	37.91
Total-R	92006	414272	22.21	16.77
Total-N	67179	406942	16.51	11.83

TABLE V.

ACCURACY OF LANGUAGE IDENTIFICATION

	Kazakh	Russian	Overall
Raw data	99.48	98.41	99.06
Normalized data	99.83	98.58	99.35

Lastly we perform the replacement and transformation

As can be seen from Table III, there were a total of 27,236 comments collected and annotated for language and sentiment. More than half (13832) of all comments were neutral. Positive and negative comments amounted to 5995 (22%) and 7409 (27.2%) respectively. In terms of languages, most comments were gathered in Kazakh (63%), and mixed comments (i.e. code-switched between Kazakh and Russian) constituted a minority of 715 (or 2.6%).

B. *Intrinsic Evaluation*

To assess the extent of data sparseness reduction we calculate basic statistics before and after normalization. We use standard indicators of data sparseness, namely, vocabulary size (as number of words counted only ones), type-token ratio (TTR, as a ratio of vocabulary size to total word token count), and OOV rate (as a ratio of out-of-vocabulary words). To calculate OOV rate we randomly split the data into 10 equal sets. For each such set we count words that do not appear in the other nine sets and divide by the number of words in the given set. We then report the average ratio for all 10 sets.

We calculate aforementioned statistics on the entire data set, as well as on per language basis. The results of the experiment are given in Table IV. Suffix –N next to a language indicates that data was normalized, and suffix –R indicates the opposite. As it can be seen normalization greatly reduces values across all metrics and languages, and for the entire data set (represented as Total in the table) totals to 27% reduction in vocabulary size, and 5.7% and 4.9% net reduction in TTR and OOV rate. Thus, our initial intuition in regarding data sparseness reduction was correct and normalization does indeed reduce sparseness significantly.

C. *Language Identification*

For the language identification experiment we use the Naïve Bayes classifier that we on a small subset of the data. To assess the impact of initial normalization, we run the classifier on raw and normalized data. As we have not trained our classifier to identify mixed language comments we do not evaluate it on those. For the evaluation metric we use standard accuracy as percent of correctly identified documents (comments). The results are given in Table V. As it can be seen normalization has a marginal effect on language identification, and on provides only 0.3% accuracy gain overall.

TABLE VI.

ACCURACY OF SENTIMENT ANALYSIS (BINARY SCALE)

	NB	NB-N	LSTM	LSTM-N
Kazakh	91.7	92.7	71.9	75.6
Russian	85.3	86.5	69.9	72.6
Mixed	84.2	85.6	70.5	77.3
Overall	89.5	90.5	71.3	73.6

TABLE VII.

ACCURACY OF SENTIMENT ANALYSIS (TRINARY SCALE)

	NB	NB-N	LSTM	LSTM-N
Kazakh	67.1	69.0	63.3	67.2
Russian	61.7	61.9	58.5	62.4
Mixed	55.5	57.6	56.3	57.7
Overall	65.0	66.7	61.7	65.2

A. *Sentiment Analysis*

We use two models for analyzing the sentiment of the text: (1) the machine learning model based on the naive Bayesian classifier [10] (hereinafter NB) and the deep learning model based on recurrent networks with LSTM cell [11] (hereinafter LSTM). For the software implementation of models, we use the Python programming language in combination with the `scikit-learn` libraries (for NB) and `keras` [12] (for LSTM). Parameters NB are the frequencies (not the presence) of unigrams and bigrams in the documents. For LSTM, the standard keras architecture was used with the following hyper-parameters: the dimension of the insert vectors was 32; the size of the dictionary is 3000.

The experiments were carried out in the context of two polarity scales: binary (positive and negative) and trinary (positive, negative, and neutral) on a per-language and overall basis. In all cases, 80% of data were used for training, 10% for testing, and another 10% for tuning model parameters. To assess the quality of sentiment analysis, we use the simplest metric - accuracy, as a percentage of correctly analyzed comments. The results are given in tables VI and VII for the experiments on binary and trinary scales respectively. Models trained on normalized data have the prefix N-.

As it can be seen, normalization improves sentient analysis accuracy across languages regardless of the scale of polarity used. In terms of models, NB which is faster to train and easier to implement consistently outperforms LSTM across languages and polarity scales. We believe that this is due to the fact that we had to (due to limited computing resources) reduce the size of the LSTM dictionary, and also reduce the size of the insertion vectors. In the future, we plan to improve the accuracy of NB by using the morphological representation of the insertion vectors [13]. Finally, in the context of the comparison by language, it is observed that the accuracy for Kazakh is consistently higher than for Russian language. We explain this by the fact that in terms of length (in words) Kazakh are shorter than Russian ones (14.2 and 15.9 words per comment, respectively), and also have smaller type-token ratio (16% against 19%), i.e. less diverse.

IV. RELATED WORK

Most of the resent works on normalization employ variety of methods ranging from supervised and deep learning to machine translation [1]-[7].

Eryiğit and Torunoğlu-Selamet [2] develop a cascaded approach for normalizing Turkish social media data, which aims at solving the following tasks: (1) letter case transformation, (2) replacement rules & lexicon lookup, (3) proper noun detection, (4) diacritic restoration, (5) vowel restoration, (6) accent normalization and (7) spelling correction. The authors use various methods and techniques ranging from a simple look-up to morphological analysis and tagging.

Tursun and Cakici [4] approach normalization of Uyghur UGC, using a noisy channel model and a neural encode-decoder architecture. The first model approaches the task as a spellchecking problem and the latter as machine translation. Performing experiments on a range of data sets the authors achieve encouraging results for both models.

Assylbekov et al. [8] perform homoglyph resolution in the context of bitext extraction task. The authors report improvement in sentence alignment of a Kazakh-Russian parallel corpus.

V. CONCLUSIONS

We have experimented with initial normalization of user generated content in a multilingual environment. Our approach amounted to successive application of three straightforward procedures: (i) homoglyph resolution, (ii) common script transliteration, (iii) replacement and transformation. It has been shown that initial normalization substantially reduces vocabulary size and OOV rate, therefore reducing data sparseness. It has also been shown that initial normalization improves overall accuracies of language identification and sentiment analysis tasks.

ACKNOWLEDGMENT

This work has been supported by Nazarbayev University research grant 144-2018//010-2018 and the Committee of Science of the Ministry of Education and Science of the Republic of Kazakhstan under the research grant AP05134272.

REFERENCES

[1] B. Han and T. Baldwin, “Lexical normalisation of short text messages: Makn sens a #twitter,” in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*: Human Language Technologies. Portland, Oregon, USA: ACL, June 2011, pp. 368–378.

[2] G. Eryiğit and D. Torunoğlu-Selamet, “Social media text normalization for Turkish,” *Natural Language Engineering*, vol. 23, no. 6, pp. 835–875, 2017.

[3] K. Adali and G. Eryiğit, “Vowel and diacritic restoration for social media texts,” in *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)*. Gothenburg, Sweden: ACL, April 2014, pp. 53–61.

[4] O. Tursun and R. Cakici, “Noisy uyghur text normalization,” in *Proceedings of the 3rd Workshop on Noisy User-generated Text*. Copenhagen, Denmark: Association for Computational Linguistics, September 2017, pp. 85–93.

[5] L. Derczynski, E. Nichols, M. van Erp, and N. Limsopatham, “Results of the WNUT-2017 shared task on novel and emerging entity recognition,” in *Proceedings of the 3rd Workshop on Noisy User-generated Text*. Copenhagen, Denmark: ACL, September 2017, pp. 140–147.

[6] B. Strauss, B. Toma, A. Ritter, M.-C. de Marneffe, and W. Xu, “Results of the WNUT-16 named entity recognition shared task,” in *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*. Osaka, Japan: The COLING 2016 Organizing Committee, December 2016, pp. 138–144.

[7] T. Baldwin, M.-C. de Marneffe, B. Han, Y.-B. Kim, A. Ritter, and W. Xu, “Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition,” in *Proceedings of the Workshop on Noisy User-generated Text*. Beijing, China: Association for Computational Linguistics, July 2015, pp. 126–135.

[8] Z. Assylbekov, B. Myrzakhmetov, and A. Makazhanov, “Experiments with Russian to Kazakh Sentence Alignment,” Izvestija KGTU im.I.Razzakova, vol. 38, no. 2, pp. 18–23, 2016.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[10] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. New York City, USA: Cambridge University Press, 2008.

[11] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in Advances in neural information processing systems. NIPS, 2014, pp. 3104–3112.

[12] Keras: The Python Deep Learning library. — Online: Available https://keras.io.

[13] A. Toleu, G. Tolegen, and A. Makazhanov, “Character-aware neural morphological disambiguation,” in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Vancouver, Canada: Association for Computational Linguistics, 2017, pp. 666–671.