Normalizing Non-canonical Turkish Texts Using Machine Translation Approaches

Talha Çolakoğlu
Istanbul Technical University
Istanbul, Turkey
colakoglut@itu.edu.tr

Umut Sulubacak
University of Helsinki
Helsinki, Finland
umut.sulubacak@helsinki.fi

A. Cüneyd Tantuğ
Istanbul Technical University
Istanbul, Turkey
tantug@itu.edu.tr

Abstract

With the growth of the social web, user-generated text data has reached unprecedented sizes. Non-canonical text normalization provides a way to exploit this as a practical source of training data for language processing systems. The state of the art in Turkish text normalization is composed of a token-level pipeline of modules, heavily dependent on external linguistic resources and manually-defined rules. Instead, we propose a fully-automated, context-aware machine translation approach with fewer stages of processing. Experiments with various implementations of our approach show that we are able to surpass the current best-performing system by a large margin.

1 Introduction

Supervised machine learning methods such as CRFs, SVMs, and neural networks have come to define standard solutions for a wide variety of language processing tasks. These methods are typically data-driven, and require training on a substantial amount of data to reach their potential. This kind of data often has to be manually annotated, which constitutes a bottleneck in development. This is especially marked in some tasks, where quality or structural requirements for the data are more constraining. Among the examples are text normalization and machine translation (MT), as both tasks require parallel data with limited natural availability.

The success achieved by data-driven learning methods brought about an interest in user-generated data. Collaborative online platforms such as social media are a great source of large amounts of text data. However, these texts typically contain non-canonical usages, making them hard to leverage for systems sensitive to training data bias. Non-canonical text normalization is the task of processing such texts into a canonical format. As such, normalizing user-generated data has the capability of producing large amounts of serviceable data for training data-driven systems.

As a denoising task, text normalization can be regarded as a translation problem between closely related languages. Statistical machine translation (SMT) methods dominated the field of MT for a while, until neural machine translation (NMT) became more popular. The modular composition of an SMT system makes it less susceptible to data scarcity, and allows it to better exploit unaligned data. In contrast, NMT is more data-hungry, with a superior capacity for learning from data, but often faring worse when data is scarce. Both translation methods are very powerful in generalization.

In this study, we investigate the potential of using MT methods to normalize non-canonical texts in Turkish, a morphologically-rich, agglutinative language, allowing for a very large number of common word forms. Following in the footsteps of unsupervised MT approaches, we automatically generate synthetic parallel data from unaligned sources of *"monolingual"* canonical and non-canonical texts. Afterwards, we use these datasets to train character-based translation systems to normalize non-canonical texts[1]. We describe our methodology in contrast with the state of the art in Section 3, outline our data and empirical results in Sections 4 and 5, and finally present our conclusions in Section 6.

2 Related Work

Non-canonical text normalization has been relatively slow to catch up with purely data-driven

[1]We have released the source code of the project at `https://github.com/talha252/tur-text-norm`

learning methods, which have defined the state of the art in many language processing tasks. In the case of Turkish, the conventional solutions to many normalization problems involve rule-based methods and morphological processing via manually-constructed automata. The best-performing system (Eryiğit and Torunoğlu-Selamet, 2017) uses a cascaded approach with several consecutive steps, mixing rule-based processes and supervised machine learning, as first introduced in Torunoğlu and Eryiğit (2014). The only work since then, to the best of our knowledge, is a recent study (Göker and Can, 2018) reviewing neural methods in Turkish non-canonical text normalization. However, the reported systems still underperformed against the state of the art. To normalize noisy Uyghur text, Tursun and Cakici (2017) uses a noisy channel model and a neural encoder-decoder architecture which is similar to our NMT model. While our approaches are similar, they utilize a naive artificial data generation method which is a simple stochastic replacement rule of characters. In Matthews (2007), character-based SMT was originally used for transliteration, but later proposed as a possibly viable method for normalization. Since then, a number of studies have used character-based SMT for texts with high similarity, such as in translating between closely related languages (Nakov and Tiedemann, 2012; Pettersson et al., 2013), and non-canonical text normalization (Li and Liu, 2012; Ikeda et al., 2016). This study is the first to investigate the performance of character-based SMT in normalizing non-canonical Turkish texts.

3 Methodology

Our guiding principle is to establish a simple MT recipe that is capable of fully covering the conventional scope of normalizing Turkish. To promote a better understanding of this scope, we first briefly present the modules of the cascaded approach that has defined the state of the art (Eryiğit and Torunoğlu-Selamet, 2017). Afterwards, we introduce our translation approach that allows implementation as a lightweight and robust data-driven system.

3.1 Cascaded approach

The cascaded approach was first introduced by Torunoğlu and Eryiğit (2014), dividing the task into seven consecutive modules. Every token is processed by these modules sequentially (hence *cascaded*) as long as it still needs further normalization. A transducer-based morphological analyzer (Eryiğit, 2014) is used to generate morphological analyses for the tokens as they are being processed. A token for which a morphological analysis can be generated is considered fully normalized. We explain the modules of the cascaded approach below, and provide relevant examples.

Letter case transformation. Checks for valid non-lowercase tokens (*e.g.* “ACL”, “Jane”, “iOS”), and converts everything else to lowercase.

Replacement rules / Lexicon lookup. Replaces non-standard characters (*e.g.* ‘ß’→‘b’), expands shorthand (*e.g.* “slm”→“selam”), and simplifies repetition (*e.g.* “yaaaaa”→“ya”).

Proper noun detection. Detects proper nouns by comparing unigram occurrence ratios of proper and common nouns, and truecases detected proper nouns (*e.g.* “umut”→“Umut”).

Diacritic restoration. Restores missing diacritics (*e.g.* “yogurt”→“yoğurt”).

Vowel restoration. Restores omitted vowels between adjacent consonants (*e.g.* “olck”→“olacak”).

Accent normalization. Converts contracted, stylized, or phonetically transcribed suffixes to their canonical written forms (*e.g.* “yapcem”→“yapacağım”)

Spelling correction. Corrects any remaining typing and spelling mistakes that are not covered by the previous modules.

While the cascaded approach demonstrates good performance, there are certain drawbacks associated with it. The risk of error propagation down the cascade is limited only by the accuracy of the ill-formed word detection phase. The modules themselves have dependencies to external linguistic resources, and some of them require rigorous manual definition of rules. As a result, implementations of the approach are prone to human error, and have a limited ability to generalize to different domains. Furthermore, the cascade only works on the token level, disregarding larger context.

3.2 Translation approach

In contrast to the cascaded approach, our translation approach can appropriately consider sentence-level context, as machine translation is a

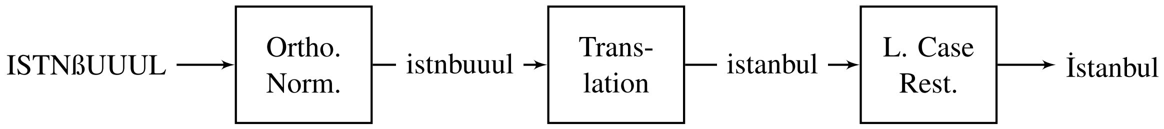

Figure 1: A flow diagram of the pipeline of components in our translation approach, showing the intermediate stages of a token from non-canonical input to normalized output.

sequence-to-sequence transformation. Though not as fragmented or conceptually organized as in the cascaded approach, our translation approach involves a pipeline of its own. First, we apply an orthographic normalization procedure on the input data, which also converts all characters to lowercase. Afterwards, we run the data through the translation model, and then use a recaser to restore letter cases. We illustrate the pipeline formed by these components in Figure 1, and explain each component below.

Orthographic normalization. Sometimes users prefer to use non-Turkish characters resembling Turkish ones, such as $\mu \rightarrow u$. In order to reduce the vocabulary size, this component performs lowercase conversion as well as automatic normalization of certain non-Turkish characters, similarly to the replacement rules module in the cascaded approach.

Translation. This component performs a lowercase normalization on the pre-processed data using a translation system (see Section 5 for the translation models we propose). The translation component is rather abstract, and its performance depends entirely on the translation system used.

Letter case restoration. As emphasized earlier, our approach leaves truecasing to the letter case restoration component that processes the translation output. This component could be optional in case normalization is only a single step in a downstream pipeline that processes lowercased data.

4 Datasets

As mentioned earlier, our translation approach is highly data-driven. Training translation and language models for machine translation, and performing an adequate performance evaluation comparable to previous works each require datasets of different qualities. We describe all datasets that we use in this study in the following subsections.

4.1 Training data

OpenSubs$_{Filtered}$ As a freely available large text corpus, we extract all Turkish data from the OpenSubtitles2018[2] (Lison and Tiedemann, 2016) collection of the OPUS repository (Tiedemann, 2012). Since OpenSubtitles data is rather noisy (e.g. typos and colloquial language), and our idea is to use it as a collection of well-formed data, we first filter it offline through the morphological analyzer described in Oflazer (1994). We only keep subtitles with a valid morphological analysis for each of their tokens, leaving a total of ∼105M sentences, or ∼535M tokens.

Train$_{ParaTok}$ In order to test our translation approach, we automatically generate a parallel corpus to be used as training sets for our translation models. To obtain a realistic parallel corpus, we opt for mapping real noisy words to their clean counterparts rather than noising clean words by probabilistically adding, deleting and changing characters. For that purpose, we develop a custom weighted edit distance algorithm which has a couple of new operations. Additional to usual insertion, deletion and substitution operations, we have defined duplication and constrained-insertion operations. Duplication operation is used to handle multiple repeating characters which are intentionally used to stress a word, such as *geliyoooooorum*. Also, to model keyboard errors, we have defined a constrained-insertion operation that allows to assign different weights of a character insertion with different adjacent characters.

To build a parallel corpus of clean and ill-formed words, firstly we scrape a set of ∼25M Turkish tweets which constitutes our noisy words source. The tweets in this set are tokenized, and non-word tokens like hashtags and URLs are eliminated, resulting ∼5M unique words. The words in OpenSubs$_{Filtered}$ are used as clean words source. To obtain an ill-formed word candidate list for each clean word, the clean words are matched with the noisy words by using our custom weighted edit

[2]http://www.opensubtitles.org/

Datasets	# Tokens	# Non-canonical tokens
Test$_{IWT}$	38,917	5,639 (14.5%)
Test$_{2019}$	7,948	2,856 (35.9%)
Test$_{Small}$	6,507	1,171 (17.9%)

Table 1: Sizes of each test datasets

distance algorithm, Since the lists do not always contain relevant ill-formed words, it would've been mistake to use the list directly to create word pairs. To overcome this, we perform tournament selection on candidate lists based on word similarity scores.

Finally, we construct Train$_{ParaTok}$ from the resulting ∼5.7M clean-noisy word pairs, as well as some artificial transformations modeling tokenization errors (*e.g.* "birşey"→"bir şey").

Huawei$_{MonoTR}$ As a supplementary collection of canonical texts, we use the large Turkish text corpus from Yildiz et al. (2016). This resource contains ∼54M sentences, or ∼968M tokens, scraped from a diverse set of sources, such as e-books, and online platforms with curated content, such as news stories and movie reviews. We use this dataset for language modeling.

4.2 Test and development data

Test$_{IWT}$ Described in Pamay et al. (2015), the ITU Web Treebank contains 4,842 manually normalized and tagged sentences, or 38,917 tokens. For comparability with Eryiğit and Torunoğlu-Selamet (2017), we use the raw text from this corpus as a test set.

Test$_{Small}$ We report results of our evaluation on this test set of 509 sentences, or 6,507 tokens, introduced in Torunoğlu and Eryiğit (2014) and later used as a test set in more recent studies (Eryiğit and Torunoğlu-Selamet, 2017; Göker and Can, 2018).

Test$_{2019}$ This is a test set of a small number of samples taken from Twitter, containing 713 tweets, or 7,948 tokens. We manually annotated this set in order to have a test set that is in the same domain and follows the same distribution of non-canonical occurrences as our primary training set.

Val$_{Small}$ We use this development set of 600 sentences, or 7,061 tokens, introduced in Torunoğlu and Eryiğit (2014), as a validation set for our NMT and SMT experiments.

Table 1 shows all token and non-canonical token count of each test dataset as well as the ratio of non-canonical token count over all tokens.

5 Experiments and results

The first component of our system (i.e. Orthographic Normalization) is a simple character replacement module. We gather unique characters that appear in Twitter corpus which we scrape to generate Train$_{ParaTok}$. Due to non-Turkish tweets, there are some Arabic, Persian, Japanese and Hangul characters that cannot be orthographically converted to Turkish characters. We filter out those characters using their unicode character name leaving only characters belonging Latin, Greek and Cyrillic alphabets. Then, the remaining characters are mapped to their Turkish counterparts with the help of a library[3]. After manual review and correction of these characters mappings, we have 701 character replacement rules in this module.

We experiment with both SMT and NMT implementations as contrastive methods. For our SMT pipeline, we employ a fairly standard array of tools, and set their parameters similarly to Scherrer and Erjavec (2013) and Scherrer and Ljubešić (2016). For alignment, we use MGIZA (Gao and Vogel, 2008) with grow-diag-final-and symmetrization. For language modeling, we use KenLM (Heafield, 2011) to train 6-gram character-level language models on OpenSubs$_{Filtered}$ and Huawei$_{MonoTR}$. For phrase extraction and decoding, we use Moses (Koehn et al., 2007) to train a model on Train$_{ParaTok}$. Although there is a small possibility of transposition between adjacent characters, we disable distortion in translation. We use Val$_{Small}$ for minimum error rate training, optimizing our model for word error rate.

We train our NMT model using the OpenNMT toolkit (Klein et al., 2017) on Train$_{ParaTok}$ without any parameter tuning. Each model uses an attentional encoder-decoder architecture, with 2-layer LSTM encoders and decoders. The input embeddings, the LSTM layers of the encoder, and the inner layer of the decoder all have a dimensionality of 500. The outer layer of the decoder has a dimensionality of 1,000. Both encoder and decoder LSTMs have a dropout probability of 0.3.

[3]The library name is *Unidecode* which can be found at `https://pypi.org/project/Unidecode/`

Model	Test$_{IWT}$	Test$_{2019}$	Test$_{Small}$
Eryiğit et al.	95.78%	80.25%	92.97%
(2017)	93.57%	75.39%	86.20%
SMT	**96.98%**	**85.23%**	**93.52%**
	95.21%	**78.10%**	**89.59%**
NMT	93.90%	74.04%	89.52%
	92.20%	67.87%	85.77%

Table 2: Case-insensitive (top) and case-sensitive (bottom) accuracy over all tokens.

Model	Test$_{IWT}$	Test$_{2019}$	Test$_{Small}$
Eryiğit et al.	79.16%	66.18%	74.72%
(2017)	70.54%	56.44%	53.80%
SMT	**87.43%**	**74.02%**	**76.00%**
	84.70%	**66.35%**	**68.40%**
NMT	71.34%	50.84%	58.67%
	68.91%	45.03%	51.84%

Table 3: Case-insensitive (top) and case-sensitive (bottom) accuracy scores over non-canonical tokens.

In our experimental setup, we apply a naïve tokenization on our data. Due to this, alignment errors could be caused by non-standard token boundaries (*e.g.* "A E S T H E T I C"). Similarly, it is possible that, in some cases, the orthography normalization step may be impairing our performances by reducing the entropy of our input data. Regardless, both components are frozen for our translation experiments, and we do not analyze the impact of errors from these components in this study.

For the last component, we train a case restoration model on Huawei$_{MonoTR}$ using the Moses recaser (Koehn et al., 2007). We do not assess the performance of this individual component, but rather optionally apply it on the output of the translation component to generate a recased output.

We compare the lowercased and fully-cased translation outputs with the corresponding ground truth, respectively calculating the case-insensitive and case-sensitive scores shown in Tables 2 and 3. We detect tokens that correspond to URLs, hashtags, mentions, keywords, and emoticons, and do not normalize them[4]. The scores we report are token-based accuracy scores, reflecting the percentages of correctly normalized tokens in each test set. These tables display performance evaluations on our own test set as well as other test sets used in the best-performing system so far Eryiğit and Torunoğlu-Selamet (2017), except the Big Twitter Set (BTS), which is not an open-access dataset.

The results show that, while our NMT model seem to have performed relatively poorly, our character-based SMT model outperforms Eryiğit and Torunoğlu-Selamet (2017) by a fairly large margin. The SMT system demonstrates that our unsupervised parallel data bootstrapping method and translation approach to non-canonical text normalization both work quite well in the case of Turkish. The reason for the dramatic underperformance of our NMT model remains to be investigated, though we believe that the language model we trained on large amounts of data is likely an important contributor to the success of our SMT model.

[4]The discrepancy between the reproduced scores and those originally reported in Eryiğit and Torunoğlu-Selamet (2017) is partly because we also exclude these from evaluation, and partly because the original study excludes all-uppercase tokens from theirs.

6 Conclusion and future work

In this study, we proposed a machine translation approach as an alternative to the cascaded approach that has so far defined the state of the art in Turkish non-canonical text normalization. Our approach is simpler with fewer stages of processing, able to consider context beyond individual tokens, less susceptible to human error, and not reliant on external linguistic resources or manually-defined transformation rules. We show that, by implementing our translation approach with basic pre-processing tools and a character-based SMT model, we were able to outperform the state of the art by a fairly large margin.

A quick examination of the outputs from our best-performing system shows that it has often failed on abbreviations, certain accent normalization issues, and proper noun suffixation. We are working on a more detailed error analysis to be able to identify particular drawbacks in our systems, and implement corresponding measures, including using a more sophisticated tokenizer. We also plan to experiment with character embeddings and character-based composite word embeddings in our NMT model to see if that would boost its performance. Finally, we are aiming for a closer look at out-of-domain text normalization in order to investigate ways to perform domain adaptation using our translation approach.

Acknowledgments

The authors would like to thank Yves Scherrer for his valuable insights, and the Faculty of Arts at the University of Helsinki for funding a research visit, during which this study has materialized.

References

Gülşen Eryiğit. 2014. ITU Turkish NLP web service. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1–4.

Gülşen Eryiğit and Dilara Torunoğlu-Selamet. 2017. Social media text normalization for Turkish. *Natural Language Engineering*, 23(6):835–875.

Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. *Software engineering, testing, and quality assurance for natural language processing*, pages 49–57.

Sinan Göker and Burcu Can. 2018. Neural text normalization for turkish social media. In *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, pages 161–166. IEEE.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the sixth workshop on statistical machine translation*, pages 187–197. Association for Computational Linguistics.

Taishi Ikeda, Hiroyuki Shindo, and Yuji Matsumoto. 2016. Japanese text normalization with encoder-decoder model. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 129–137.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180.

Chen Li and Yang Liu. 2012. Normalization of text messages using character- and phone-based machine translation approaches. In *Thirteenth Annual Conference of the International Speech Communication Association*.

Pierre Lison and Jörg Tiedemann. 2016. OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles.

David Matthews. 2007. Machine transliteration of proper names. *Master's Thesis, University of Edinburgh, Edinburgh, United Kingdom*.

Preslav Nakov and Jörg Tiedemann. 2012. Combining word-level and character-level models for machine translation between closely-related languages. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 301–305. Association for Computational Linguistics.

Kemal Oflazer. 1994. Two-level description of turkish morphology. *Literary and linguistic computing*, 9(2):137–148.

Tuğba Pamay, Umut Sulubacak, Dilara Torunoğlu-Selamet, and Gülşen Eryiğit. 2015. The annotation process of the itu web treebank. In *Proceedings of the 9th Linguistic Annotation Workshop*, pages 95–101.

Eva Pettersson, Beáta Megyesi, and Jörg Tiedemann. 2013. An smt approach to automatic annotation of historical text. In *Proceedings of the workshop on computational historical linguistics at NODALIDA 2013; May 22-24; 2013; Oslo; Norway. NEALT Proceedings Series 18*, 087, pages 54–69. Linköping University Electronic Press.

Yves Scherrer and Tomaž Erjavec. 2013. Modernizing historical Slovene words with character-based SMT. In *BSNLP 2013-4th Biennial Workshop on Balto-Slavic Natural Language Processing*.

Yves Scherrer and Nikola Ljubešić. 2016. Automatic normalisation of the Swiss German ArchiMob corpus using character-level machine translation. In *Proceedings of the 13th Conference on Natural Language Processing (KONVENS 2016)*, pages 248–255.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Dilara Torunoğlu and Gülsen Eryiğit. 2014. A cascaded approach for social media text normalization of turkish. In *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)*, pages 62–70.

Osman Tursun and Ruket Cakici. 2017. Noisy uyghur text normalization. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 85–93.

Eray Yildiz, Caglar Tirkaz, H Bahadır Sahin, Mustafa Tolga Eren, and Omer Ozan Sonmez. 2016. A morphology-aware network for morphological disambiguation. In *Thirtieth AAAI Conference on Artificial Intelligence*.