The 6th Intl. Workshop on Spoken Language Technologies for Under-Resourced Languages
29-31 August 2018, Gurugram, India

Mining Training Data for Language Modeling Across the World's Languages

Manasa Prasad, Theresa Breiner, Daan van Esch

Google LLC, Mountain View (CA), United States
pbmanasa@google.com, tbreiner@google.com, dvanesch@google.com

Abstract

Building smart keyboards and speech recognition systems for new languages requires a large, clean text corpus to train n-gram language models on. We report our findings on how much text data can realistically be found on the web across thousands of languages. In addition, we describe an innovative, scalable approach to normalizing this data: all data sources are noisy to some extent, but this situation is even more severe for low-resource languages. To help clean the data we find across all languages in a scalable way, we built a pipeline to automatically derive the configuration for language-specific text normalization systems, which we describe here as well.
Index Terms: speech recognition, keyboard input, low-resource languages, data mining, language modeling, text normalization

1. Introduction

Despite the advances in language technology over recent years, there are still many language communities that cannot use smart keyboards (with auto-correction and next-word prediction) or speech technology in their own languages. For some of the world's 6,000+ language communities, access to technology remains elusive. However, many places that even a few years ago had virtually no internet connectivity or technology are now coming online at a very rapid pace. In 2017 alone, around 180 million people newly gained access to the internet, with the highest growth rate in places like Asia and Africa [1]. Many speakers in these communities gain access only to find that their native language is not well supported by technology.

The usefulness of keyboards is obvious: they enable everyday input. Automatic speech recognition (ASR) can also be extremely helpful for users who are using technology for the first time. These users may find it hard to use virtual keyboards in their language (especially if their language uses a complex script), and may prefer to leverage voice input to interact with their friends (e.g. on social media) and the web more quickly than when typing.

ASR systems typically use n-gram language models trained on target-language text data [2]. Text data is often hard to find for low-resource languages. In this paper, we present our methods for mining text data in many languages and normalizing it to make it useable for training language models that could be used for building smart keyboards or speech recognition systems. Sharing models between keyboard and voice input is especially easy if the keyboard decoder is built in the finite-state transduction paradigm (e.g. [3]).

Specifically, we have gathered data sets across hundreds of languages that can be used to train n-gram language models using the following steps:

1. Identifying sentence and wordlist data for as many languages as possible
2. Merging the data into consistent language codes
3. Automatically deriving a preliminary normalization configuration
4. Normalizing the data to reduce noise levels

We will describe these in more detail below. Our main findings are that:

- There are quite a few resources that can be used to train language models, across a surprisingly large number of languages
- Even if noise levels are relatively high, automatic approaches to normalization can significantly clean up the resulting text corpora.

2. Mining data

2.1. Open-source resources on the web

There are a variety of open-source resources available on the web containing data in low-resource languages, which we will briefly describe here. Tatoeba is a collaborative site that gathers sentences and translations for over 300 languages [4]. Wikipedia currently hosts editions in 291 languages [5], and also provides an Incubator for languages whose content is growing but is not yet ready to be published as a complete Wikipedia site, covering 279 languages as of June 2018 [6]. There are additional Wikimedia resources that we did not incorporate ourselves in our work but could provide more sentence data such as Wikibooks, Wikinews, Wikiquote, Wiktionary, and Wikisource. These sources mostly cover on the order of tens of languages rather than hundreds; Wiktionary (a set of dictionaries) covers the most languages after Wikipedia with 173 published and 115 Incubator languages. However, its MediaWiki markup syntax makes it quite difficult to parse [7].

There are also translations of the Universal Declaration of Human Rights (UDHR) available through the Unicode Consortium in over 400 languages, and with more being added [8]. A quick human inspection revealed the UDHR translations, while short, are fairly clean in terms of content. While the Wikipedia data clearly came in larger quantities, data quality seemed lower, containing auto-generated pages with very little or noisy content, perhaps even with data from other languages mixed in.

Another type of resource that covers hundreds of low-resource languages is religious texts. Bibles.org has data

10.21437/SLTU.2018-13

in over 900 languages [9], while jw.org contains Jehovah's Witnesses materials in over 800 languages [10]. The data sets in this domain may be less applicable for language modeling tasks for general-purpose speech recognition systems, as they are tied to a domain with its own domain-specific vocabulary and frequency counts. However, these data sets may still be useful to infer common words, as well as for character-based language modeling. In addition, due to their typically parallel nature, they can be leveraged for translation research, as well as for large-scale linguistic typology [11] [12] [13] [14] [15].

2.2. Crawling the web for more data

While the above resources are already labeled by language, there is much more data available to be mined from a general crawl of the internet, as long as some method is available to determine the language of the websites in the crawl. We used an internal web crawler similar to open-source projects like [16] [17]. While these crawlers target about 40 and 200 languages respectively, we applied an internal language classifier that can identify text in over 100 languages, similar to other language identification modules such as [18] [19] [20] [21]. We used this labeling to mine data for languages directly from a web crawl. Due to limitations on the language classifier, this data doesn't cover as many languages as the other resources, but it does tend to contain much more data per language compared to the above resources. One issue is that our language identification tool may be unable to differentiate between a language it supports and a similar one it does not, meaning that the data labeled for a certain language may contain data in other related languages as well. We leave the training of a language classification module on more languages based on the data sources described above to future work.

2.3. Open-source wordlists

The resources mentioned in the previous sections provide full sentence data for many languages, but we also identified some open-source projects that provide lists of words in hundreds or thousands of languages. Some of these projects also involved mining the web looking for data in specific languages, including data from some of the above resources, which was then processed to create word lists. The Crúbadán project from St. Louis University provides mined, somewhat noisy word lists and n-grams in about 2,500 languages [22]. The Unilex project is a subproject of the Unicode Consortium that also provides mined lists of words for almost 1,000 languages, along with frequency data for the words [23]. The PanLex project is a lexical translation database covering 5,700 languages, with dictionaries for 2,500 of them [24].

While these word list resources cannot be used to train full language models, we were able to leverage them in our process to automatically clean up and normalize the data sets, discussed below. Word lists could also be combined with isolated-word audio databases such as [25] and [26] for other ASR applications, for example in supporting voice input for a dictionary app lookup or a lexical translation app. There is ongoing research in ASR for isolated words in low-resource languages [27] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] as well as for accessibility [38].

Table 1: *Language Coverage per Web Resource*

Resource	**Languages (Sentences)**	**Languages (Words)**
Tatoeba	313	–
Wikipedia	570	–
– Published	291	–
– Incubator	279	–
UDHR	549	–
Bibles.org	923	–
JW.org	882	–
Crubadan	–	2,500
Unilex	–	998
PanLex	–	5,700

There are also many linguistic resources available online that provide helpful information about the world's languages: to name but a few, we found Glottolog [39], Omniglot [40], and ScriptSource [41] quite valuable.

3. Processing data sets

3.1. Merging the data

While it was relatively easy to mine this data for so many languages once these resources were identified, merging all the data from the same language across these various resources required a number of steps to standardize the language labels and clean up the data itself.

An unexpectedly large issue in merging so many resources was inconsistencies in the labeling conventions. Many resources use BCP-47 codes to refer to data set in different languages, containing ISO 639 codes to describe the language variety, ISO 15924 codes to describe the script, and ISO 3166-1 codes to describe the region [42]. While the included resources used these codes overall, there were some exceptions, and some cases of existing BCP-47 codes being used for different language data. For example, the edition of Wikipedia that is labeled with the code "als" (at https://als.wikipedia.org) is actually written in "gsw", the ISO 639 code shared by Swiss German and Alsatian, rather than the language assigned to ISO 639 code "als", Tosk Albanian.

There were some languages for which we still found only minimal amounts of data even after combining data from all these resources. We were concerned that the merged data sets would not provide representative data or even full coverage of the language's character set. We discarded a number of languages for this reason.

Overall, after merging the data, we had word lists in more than 2,200 languages, and over 1,700 of those languages also had full-sentence data. For many languages the only source of sentence data was the parallel religious corpora in JW.org and Bibles.org, but even excluding those resources still yielded sentence data for over 600 languages. These numbers do not include data from PanLex, which we did not use in this project.

3.2. Noise in the data

Once the data was merged using standard language codes, a quick human inspection showed that noise levels in the text were, as expected, quite high. For example,

there were non-standard tokens like “fr!3nd”, and characters from different scripts appearing together, like “This is நல்ல technology.” There were also two types of encoding challenges. First, not all resources were encoded in the same flavor of Unicode. This was easily addressed across all languages by converting all data to Unicode Normalization Form C (NFC) [43], to avoid storing multiple representations of the same words and phrases.

Second, there were also rendering related problems, such as characters appearing using an unexpected encoding. For example, there may be a sentence written with Cyrillic characters where a Latin <W> (Unicode U+0057: LATIN CAPITAL LETTER W) also appears. There is a Cyrillic <Ԝ> character (Unicode U+051C: CYRILLIC CAPITAL LETTER WE) that looks almost identical to the Latin <W>, so this poses no problem to the human eye. However, machine-learning language modeling systems require consistency, and the Cyrillic encoding should be used for data in the Cyrillic script. There were also other rendering-related characters such as the NO-BREAK SPACE (Unicode U+00A0) and the RIGHT-TO-LEFT OVERRIDE (Unicode U+202E) that would typically need to be removed prior to training models. However, given the number of languages, it was not feasible to do a manual human inspection for each language individually, so we turned to automation.

3.3. Automatically generating normalizer configurations

Our goal in automating the normalization process was to reduce noise levels in the data as much as possible, while still proceeding automatically without using any language-specific knowledge, in order to maintain scalability. Prior to training language models, we wanted to remove sentences containing noisy examples like those above as well as:

- Words with attached or internal punctuation
- Sentences containing out-of-language characters
- Words that contain only digits
- Tokens like email addresses (for privacy purposes)

We were able to use the gathered data to generate rule-based normalizers automatically, which could in turn be applied to clean up the same data. Specifically, we used Thrax-based normalizers which allow grammar rules to be written in an accessible domain-specific language, which is then compiled into finite-state transducers [44]. Our system allows us to use a core shared grammar template with only limited amounts of configuration needed for each individual language, containing the characters allowed in that language and specific rewrite rules if necessary [45].

In order to produce a normalizer configuration extension for a given language, we first ran through all of the data that we gathered for that language, keeping track of the frequencies of every character we came across, as well as which Unicode block that character belonged to (for example “Basic Latin”, “Cyrillic”, or “Devanagari”). This information for a given character can be found using the Python ICU library [46]. Based on this analysis, we were able to make a fairly accurate guess as to which script the language data should be in; it would typically be the most commonly used Unicode block. If there were other Unicode blocks (excluding Latin) that were used for more than 20% of the data, we also considered that block to be acceptable for the language. We also made a note of additional individual characters if they appeared an unusual number of times, informing potential future human review by linguists.

When we had determined the target script of the language data, we could add the characters that we had seen in the data belonging to the corresponding Unicode blocks to the normalizer configuration, and ignore any other characters that were seen. Seen characters that belonged to the “punctuation” Unicode category (identified as such by the Python ICU module) were also gathered and included as acceptable initial or final punctuation options, depending on the context in which they were found. The appearance of characters from a specific set of confusable characters [47], such as variations of the hyphen character (HYPHEN MINUS, M DASH, N DASH, etc) triggered addition of a configuration rule to reformat those characters to one chosen standard (HYPHEN MINUS, for example). We also added numeric characters present as a separate set.

With these language-specific normalizer extensions automatically generated, we were ready to run through the data again, dropping any malformed sentences or out-of-set characters that did not pass the normalizer.

Table 2: *Comparison of quality based on percent kept after normalization across a selection of resources. Does not include 29 language outliers in UDHR data with over 60% of data rejected.*

Resource	Mined	Accepted	% Kept
Web Crawler	~29B	~18B	62%
Wikipedia	~68M	~46M	68%
UDHR	~33K	~29K	88%

4. Cleaned data sets for training

In examining how much of our mined data successfully passed the normalization phase, we confirmed our initial impressions around the cleanliness of the data: as can be seen in Table 2, the UDHR data, which was carefully curated by humans, is much cleaner than the data we crawled from the web at large and Wikipedia. The UDHR data set has an average acceptance rate of 88%, which is similar to high-quality internally curated data sets, with an average acceptance rate of 94%. Some examples of issues in the generally high-quality UDHR data that our systems detected automatically are:

- the inadvertent appearance of a string ”#x06C1” instead of the ARABIC LETTER HEH GOAL in Western Panjabi (ISO 639: pnb)
- the incorrect use of LATIN W instead of CYRILLIC WE in Northern Yukaghir (ISO 639: ykg)
- the seemingly stray appearance of a question mark in the word ”jahuequ¿scamabi”, which may have been intended to be ”jahuequescamabi” in Shipibo-Conibo (ISO 639: shp)

We compiled a list of all detected issues and shared it with the UDHR database maintainers. While it may be

tempting to include even incorrectly encoded data, especially in a low-resource setting, in our experience discarding is the right choice to prevent complexities elsewhere (e.g. with grapheme-to-phoneme models, which may be confused by unexpected encodings), and given the low return on investment for writing rules to repair data.

Similar issues appear frequently throughout the web crawl and Wikipedia data, and many seem to be due to visual confusion and/or the unavailability of keyboards with the correct keys. For example, in Akan (ISO 639: ak), the character <Ɔ> (Unicode U+03FD: CAPITAL GREEK REVERSED LUNATE SIGMA) is often used where <Ɔ> (Unicode U+0186: OPEN O) should be. Our pipeline automatically detects these issues, flags them for review, and drops the sentences from the set until a language expert can be consulted. This way, we create high-quality data sets automatically at large scale, while further improvements can be made if human language experts are available.

Figure 1 gives a sense of the scale of the data sets with an overview of the number of unique words per language across more than 2,000 languages. This does not include data from our web crawl or Wikipedia; including these sources would show an even more optimistic picture.

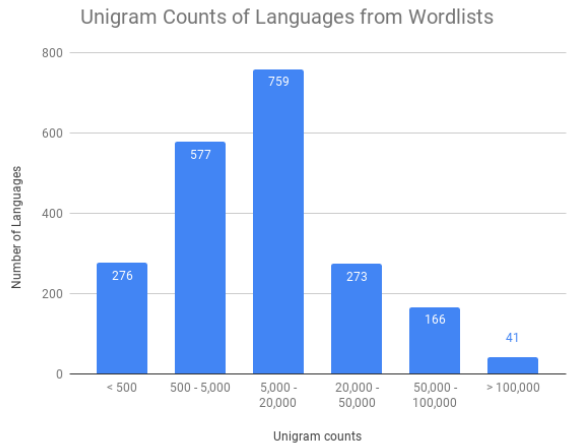

Figure 1: *Unigram counts pulled from Crúbadán, Incubator, Tatoeba, UDHR and Unilex. We split the languages into 6 buckets.*

5. Conclusion

We presented an overview of freely available text data resources across hundreds and even thousands of languages. We described challenges encountered in merging and standardizing these data sets, and the application of automatic scalable text normalization for consistency, which we find critical for machine learning applications such as language modeling. Our automatic normalization system scales effortlessly to any number of languages, enabling the creation of clean text corpora, while also optionally allowing human experts to provide further language-specific information. This clean text data can be used to train n-gram language models, a critical building block for smart keyboards and for eventually building ASR systems. Of course, much more work is needed to build automatic speech recognition systems, particularly in acoustic and pronunciation modeling. Our main conclusions are that text data, previously thought to be a serious bottleneck for extending language technology to more languages, is more readily available than expected, in larger quantities than we had assumed we would find; and that it is possible to clean up this text data automatically at scale across languages.

6. References

[1] P. Biggs, "The state of broadband: Broadband catalyzing sustainable development," International Telecommunications Union and Broadband Commission for Sustainable Development and UNESCO, Tech. Rep., 2017.

[2] M. Mohri, F. Pereira, and M. Riley, "Speech recognition with weighted finite-state transducers," in *Handbook of Speech Processing*, Y. H. Jacob Benesty, Mohan Sondhi, Ed. Springer, 2008, pp. 559–582.

[3] T. Ouyang, D. Rybach, F. Beaufays, and M. Riley, "Mobile keyboard input decoding with finite-state transducers," *CoRR abs/1704.03987*, 2017.

[4] T. Ho. Tatoeba. Accessed: Oct. 14, 2017. [Online]. Available: https://tatoeba.org/eng

[5] List of Wikipedias. Wikimedia. Accessed: June 4, 2018. [Online]. Available: https://meta.wikimedia.org/wiki/List_of_Wikipedias

[6] Incubator: Wikis. Wikimedia. Accessed: June 4, 2018. [Online]. Available: https://incubator.wikimedia.org/wiki/Incubator:Wikis

[7] T. Tresoldi, "Extracting translation data from the Wiktionary project," *Computer-Assisted Language Comparison in Practice: Tutorials on Computational Approaches to the History and Diversity of Languages*, 06 2018. [Online]. Available: https://calc.hypotheses.org/32

[8] Translations of the universal declaration of human rights. Unicode, Inc. Accessed: Feb. 10, 2017. [Online]. Available: http://www.unicode.org/udhr/translations.html

[9] Biblesearch. American Bible Society. Accessed: Apr. 10, 2017. [Online]. Available: http://bibles.org/eng-GNTD/Gen/1

[10] Jw.org. Watch Tower Bible and Tract Society of Pennsylvania. Accessed: June 4, 2018. [Online]. Available: https://www.jw.org/en/

[11] R. Östling and J. Tiedemann, "Neural machine translation for low-resource languages," *CoRR abs/1708.05729*, 2017.

[12] T. Mayer and M. Cysouw, "Creating a massively parallel Bible corpus," in *Ninth International Conference on Language Resources and Evaluation, LREC 2014, 26-31 May 2014, Reykjavik, Iceland*. Research Unit Quantitative Language Comparison, Philipps University of Marburg, 2014, pp. 3158–3163.

[13] E. Asgari and H. Schütze, "Past, present, future: A computational investigation of the typology of tense in 1000 languages," *CoRR*, vol. abs/1704.08914, 2017. [Online]. Available: http://arxiv.org/abs/1704.08914

[14] C. Christodouloupoulos and M. Steedman, "A massively parallel corpus: the Bible in 100 languages," *Language Resources and Evaluation*, vol. 49, pp. 375–395, 2015.

[15] Z. Agic, D. Hovy, and A. Søgaard, "If all you have is a bit of the Bible: Learning POS taggers for truly low-resource languages," in *ACL*. Center for Language Technology, University of Copenhagen, Denmark, 2015.

[16] Common crawl. Common Crawl Foundation. Accessed: June 4, 2018. [Online]. Available: https://www.commoncrawl.org

[17] U. L. Abteilung Automatische Sprachverarbeitung. Corpora collection. Deutscher Wortschatz. Accessed: June 4, 2018. [Online]. Available: http://corpora.uni-leipzig.de/en?corpusId=deu_newscrawl_2011

[18] E. Grave. (2017, October) Language identification. Accessed: June 4, 2018. [Online]. Available: https://fasttext.cc/blog/2017/10/02/blog-post.html

[19] M. Danilak. (2016, October) langdetect 1.0.7. Accessed: June 4, 2018. [Online]. Available: https://pypi.org/project/langdetect/

[20] (2018, May) Natural language toolkit. NLTK Project. Accessed: June 4, 2018. [Online]. Available: https://www.nltk.org/

[21] D. Tankersley. (2016, July) Wikipedia seeks to speak your language. Wikimedia Foundation. Accessed: June 4, 2018. [Online]. Available: https://blog.wikimedia.org/2016/07/27/wikipedia-language-search/

[22] K. P. Scannell, "The crúbadán project: Corpus building for under-resourced languages," in *3rd Web as Corpus Workshop, 2007, Louvain-la-Neuve, Belgium*, 2007.

[23] Unilex: Lexical data at Unicode. Unicode, Inc. Accessed: June 4, 2018. [Online]. Available: https://github.com/unicode-org/unilex

[24] Panlex. The Long Now Foundation. Accessed: June 4, 2018. [Online]. Available: https://panlex.org/

[25] S. E. Ouahabi, M. Atounti, and M. Bellouki, "A database for Amazigh speech recognition research: Amzsrd," in *2017 3rd International Conference of Cloud Computing Technologies and Applications (CloudTech), 24-26 Oct. 2017, Rabat, Morocco*, 2017.

[26] N. D. Londhe and G. B. Kshirsagar, "Chhattisgarhi speech corpus for research and development in automatic speech recognition," *International Journal of Speech Technology*, vol. 21, pp. 193–210, 2018.

[27] R. Singh and A. Sharma, "Speech recognition methods applied for different Punjabi language accents: A review," in *International Conference on Recent Innovations in Science, Agriculture, Engineering and Management, Nov. 20, 2017, Bathinda, Punjab, India*, 2017. [Online]. Available: http://data.conferenceworld.in/GKU/128.pdf

[28] A. P. Kandagal and V. Udayashankara, "Speaker independent speech recognition using maximum likelihood approach for isolated words," *International Journal of Computer Application*, vol. 7, no. 6, pp. 72–83, 2017. [Online]. Available: https://rspublication.com/ijca/2017/dec17/10.pdf

[29] M. K. Ssarma, A. Gajurel, A. Pokhrel, and B. Joshi, "Hmm based isolated word Nepali speech recognition," in *2017 International Conference on Machine Learning and Cybernetics, July 9-12, 2017, Ningbo, China*, 2017. [Online]. Available: IEEE Explore, https://ieeexplore.ieee.org/abstract/document/8107745/

[30] A. H. Unnibhavi and D. S. Jangamshetti, "LPC based speech recognition for Kannada vowels," in *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques, Dec. 15-16, 2017, Mysuru, India*, 2017. [Online]. Available: IEEE Explore, https://ieeexplore.ieee.org/abstract/document/8284582/

[31] P. G. Vanajakshi, M. Mathivanan, and T. S. Kumaran, "Modified Wiener filter based speech restoration with autocorrelation of isolated Kannada word speech recognition," *International Journal of Intelligent Engineering & Systems*, vol. 11, no. 2, pp. 208–216, 2017. [Online]. Available: https://rspublication.com/ijca/2017/dec17/10.pdf

[32] Y. Kumar and N. Singh, "An automatic spontaneous speech recognition system for Punjabi language," in *Speech and Language Processing for Human-Machine Communications*, ser. Advances in Intelligent Systems and Computing, S. Agrawal, A. Devi, R. Wason, and P. Bansal, Eds. Springer, vol. 664.

[33] Asadullah, A. Shaukat, H. Ali, and U. Akram, "Automatic Urdu speech recognition using Hidden Markov Model," *2016 International Conference on Image, Vision and Computing (ICIVC)*, pp. 135–139, 2016.

[34] R. Hasan, H. Hussein, P. Lazaridis, S. Khwandah, M. Ritter, and M. Eibl, "Improvement of speech recognition results by a combination of systems," *2017 23rd International Conference on Automation and Computing (ICAC)*, 2017.

[35] H. G., P. M., and T. K., "Speech recognition system for different Kannada dialects," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 2, pp. 180–188, 2017.

[36] A. Hajitabar, H. Sameti, H. Hadian, and A. Safari, "Persian large vocabulary name recognition system (FarsName)," in *2017 Iranian Conference on Electrical Engineering, ICEE 2017, 2-4 May 2017, Tehran, Iran*, 2017.

[37] Y. A. Ibrahim and T. S. Ibiyemi, "A comparative survey of DTW and HMM using Hausa isolated digits recognition in human computer interaction sytem," *Annals. Computer Science Series*, vol. 15, pp. 61–67, 2017.

[38] J. Emeeshat, "Isolated word speech recognition system for children with Down syndrome," Master's thesis, Youngstown State University, 2017. [Online]. Available: https://etd.ohiolink.edu/

[39] Glottolog 3.2. Max Planck Institute for the Science of Human History. Accessed: June 4, 2018. [Online]. Available: https://glottolog.org

[40] S. Ager. Omniglot: Writing systems and languages of the world. Accessed: June 4, 2018. [Online]. Available: https://omniglot.com/

[41] Scriptsource. Unicode, Inc. Accessed: June 4, 2018. [Online]. Available: http://scriptsource.org/cms/scripts/page.php

[42] A. Phillips and M. Davis. (2009) Tags for identifying languages. IEFT Trust. Accessed: June 4, 2018. [Online]. Available: https://tools.ietf.org/html/bcp47

[43] "Unicode standard annex #15: Unicode normalization forms," Unicode, Inc., Tech. Rep., 2018, accessed: June 4, 2018. [Online]. Available: http://www.unicode.org/reports/tr15/

[44] M. Riley. Opengrm thrax grammar development tools. Accessed: June 4, 2018. [Online]. Available: https://tools.ietf.org/html/bcp47

[45] M. Chua, D. van Esch, N. Coccaro, E. Cho, S. Bhandari, and L. Jia, "Text normalization infrastructure that scales to hundreds of language varieties," in *Proc. of the 11th edition of the Language Resources and Evaluation Conference, LREC 2018, 7-12 May 2018, Miyazaki, Japan*, 2018. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2018/pdf/8883.pdf

[46] A. Vajda. PyICU: Python extension wrapping the ICU C++ API. Accessed: June 4, 2018. [Online]. Available: https://github.com/ovalhub/pyicu

[47] "Unicode technical standard #39 unicode security mechanisms," Unicode, Inc., Tech. Rep., 2018, accessed: June 13, 2018. [Online]. Available: http://www.unicode.org/reports/tr39/#confusables