Normalization of Indonesian-English Code-Mixed Twitter Data

Anab Maulana Barik, Rahmad Mahendra, Mirna Adriani
Faculty of Computer Science, Universitas Indonesia
Depok, Indonesia
maulanaanab@gmail.com, {rahmad.mahendra, mirna}@cs.ui.ac.id

Abstract

Twitter is an excellent source of data for NLP researches as it offers a tremendous amount of textual data. However, processing tweet to extract meaningful information is very challenging, at least for two reasons: (i) using non-standard words as well as informal writing manner, and (ii) code-mixing issues, which is combining multiple languages in single tweet conversation. Most of the previous works have addressed both issues in isolated different task. In this study, we work on normalization task in code-mixed Twitter data, more specifically in Indonesian-English language. We propose a pipeline that consists of four modules, i.e tokenization, language identification, lexical normalization, and translation. Another contribution is to provide a gold standard of Indonesian-English code-mixed data for each module.

1 Introduction

Twitter has gained interest from Natural Language Processing (NLP) researchers over the last decade because it offers various textual data, such as public opinions, conversation, and breaking news, in a huge number. However, tweets are mostly noisy texts as they contain a lot of typos, slang terms, and non-standard abbreviations. This noisy data results dropping in the accuracy of the past NLP systems (Liu et al., 2011).

Another common phenomenon found in social media, including Twitter, is that people tend to alternate between multiple languages in one utterance. The embedding of linguistic units such as phrases, words, and morphemes of one language into the usage of other different languages is known as code-mixing (Myers-Scotton, 1993). The phenomenon of code-switching causing grief for NLP systems due to the grammar and spelling variations.

Indonesia, the most fourth populous country in the world, is bilingual[1]. While Bahasa Indonesia is the only official language, English is also used in formal education and business. Nowadays, the Indonesian young generation gets used to mix both languages in daily life. Code-mixing is frequently found in social media conversation in Indonesia.

In this paper, we design standardization system for Indonesian-English code-mixed Twitter data. Our solution is a pipeline of 4 modules, i.e. tokenization, language identification, lexical normalization, and translation.

1. **Tokenization**: The tweet is tokenized into several tokens. Each token may represent a word, an idiom, an interjection (e.g. haha, hehe, wkwkw), numbers, emoticon, punctuation marks, and tweet entity (i.e link, hashtag, mention). In this study, the name of entities i.e movies, people, etc. is considered as one single token.
2. **Language Identification**: Every token within tweet is labeled with corresponding language tag. The label 'en' is assigned for English token, 'id' for Indonesian token, and 'rest' for the token that not clearly belongs to either English or Indonesian (e.g. proper name, number).
3. **Normalization**: Tokens with label 'id' or 'en' are normalized into standard form. To reduce the number of token variations in the data set, we reduce character repetition to be not more than two (e.g. the interjection token "*heheeee*" is standardized into "*hehee*"). The tokens with label "rest" are left as they are

[1]https://blog.swiftkey.com/celebrating-international-mother-language-day/

4. **Translation**: We merge the sequence of normalized tokens back into complete tweet, and translate the tweet into Indonesian, with the exception of the name entities that are kept in original language (i.e term ”The Lion King” is not translated into ”Raja Singa”).

To our knowledge, this is the first attempt to normalize Indonesian-English code-mixed language. For our experiment, we build the data set consisting of 825 tweets.

2 Related Work

Text normalization has been studied for Twitter data using a variety of supervised or unsupervised methods. Liu et al. (2011) modelled lexical normalization as a sequence labelling problem, by generating letter-level alignment from standard words into nonstandard variant words, using Conditional Random Field (CRF).

Beckley (2015) performed English lexical normalization task in three steps. First, compiled a substitution list of nonstandard into standard words, then built a rule-based components for -ing and duplication rule as it is often found in the data set. Last, applied sentence-level re-ranker using bigram Viterbi algorithm to select the best candidate among all the candidates generated from first and second steps.

Sridhar (2015) proposed an unsupervised approach for lexical normalization by training a word embedding model from large English corpora. The model is used to create a mapping between non-standard into standard words. Hanafiah et al. (2017) approached lexical normalization for Indonesian language using the rule-based method with the help of a dictionary and a list of slang words. If the token is OOV, then they create a candidate list based on the consonant skeleton from the dictionary.

For code-mixed data, Singh et al. (2018) created clusters of words based on embedding model (pre-trained on large corpora) for the semantic features and Levenshtein distance for lexical features. Then, one word is picked to become the parent candidate for each cluster, and other words in each cluster are normalized into the parent candidate.

Mave et al. (2018) worked a language identification task on code-mixed data. The experiment shown that Conditional Random Field (CRF) model outperformed Bidirectional LSTM.

Types	**Count**
Number of tweets	825
Number of tokens	22.736
Number of ’id’ tokens	11.204
Number of ’en’ tokens	5.613
Number of ’rest’ tokens	5.919

Table 1: Data Set Detail

Dhar et al. (2018) augmented existing machine translation by using Matrix Language-Frame Model proposed by Myers-Scotton (1997) to increase the performance of the machine translations apply on code-mixed data.

Bhat et al. (2018) presented a universal dependency parsing with the Hindi-English dataset using a pipeline comprised of several modules such as language identification, back-transliteration, normalization using encoder-decoder framework and dependency parsing using neural stacking. It is found that normalization improves the performance of POS tagging and parsing models.

3 Data Set

We utilize three kinds of Twitter corpora in this study i.e. English, Indonesian, and Indonesian-English code-mixed corpus. We obtain 1.6M English tweets collection from ’Sentimen40’ (Go et al., 2009) and 900K Indonesian tweets from Adikara (2015). We collect 49K Indonesian-English code-mixed tweets by scrapping them using Twitter API. To harvest those tweets, first we take 100 English and Indonesian stopwords from wiktionary[2]. To fetch code-mixed tweets, we use the stopwords as query term and set the language filter as the opposite of the stopword one (e.g. Indonesian tweets are collected using a English stopword as a query, vice versa).

We select 825 tweets randomly from code-mixed corpus for the experiment. The data is labeled by two annotators. The gold standard is constructed for four individual tasks.

The inter-annotator agreement has 97,92% for language identification and 99,23% for normalization. Our data has a code-mixing index (Das and Gambäck, 2014) of 33.37, means that the level of mixing between languages in the data is quite high (see Table 1 for detail)

[2]https://en.wiktionary.org/wiki/Wiktionary:Frequency lists

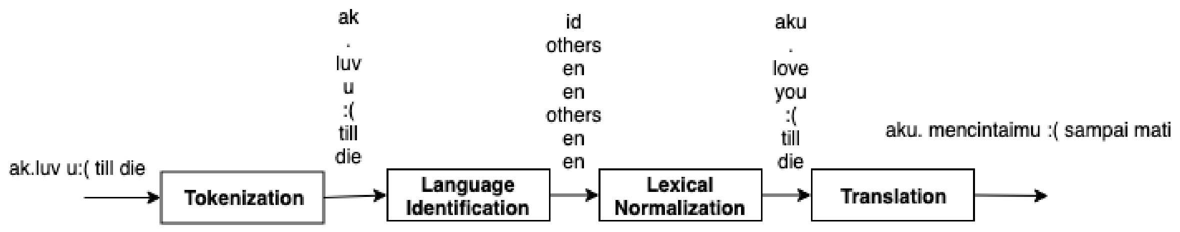

Figure 1: An Input-Output Example of Pipeline Model

4 Methods

The main objective of this study is to standardize the lexical form of code-mixed tweets. To achieve this, we propose a pipeline which is composed of four modules, tokenization, language identification, lexical normalization, and translation. The pipeline takes the code-mixed Indonesian-English tweet as the input, runs each module sequentially, and produces tweet in well-formed Indonesian language as output. Figure 1 depicts the pipeline model with an example of input and output.

4.1 Tokenization

The tokenization module takes a single tweet as input, and produces a sequence of tokens (one token can be a multiword expression) as output. Default delimiter in tokenization, i.e. whitespace, does not work well for social media domain due to non-standard writing style, for example, space disappearance (*of course not..absolutely not*) and space excesses (*E N D*). More specific problem found in the Indonesian is writing inaccuracy of morphological affix ”di” as a preposition, and vice versa. While whitespace is needed to separate preposition ”di” and the next word (e.g. *”di kelas”*, in English: *”at class”*); this does not apply for affixation case (e.g. *”dimakan”*, in English: *”eaten”*).

We approach this task as a sequence labeling at the character level with inside/outside chunk representation, as introduced by Ramshaw and Marcus (1999). We use three different tags: **B** represents the beginning of a new chunk, **I** means that the current token is inside the chunk, and **O** indicates that the current token is outside of any chunks. For instance, a sentence ”*this car*” will be encoded into ”*BIIIOBII*”.

The features set used in tokenization comprises of morphological information of the character i.e current character, is alphabet, is digit, is uppercase, and n-window character (n = 5). Sequence labeling model is trained using Conditional Random Field.

4.2 Language Identification

This module takes a sequence of tokens outputted from the tokenization module as an input, and identifies the language for each token. The language tag is one of these labels: ’en’, ’id’, ’rest’.

The features set for language identification consist of the current token in which the language to be determined, morphological information of current token, n-neighbor tokens, and n-gram character of the current token (n = 5). For morphological information, we use the binary features such as is alphabet, is digit, is capital, contains alphabet, contains digit, and has apostrophes. We do not include Part-Of-Speech (POS) information in our selected features, as recommended by Mave et al. (2018). The model is trained using Conditional Random Field.

4.3 Lexical Normalization

This module takes a sequence of tokens along with their language tags as input, and normalize the token one by one independently by substituting out-of-vocabulary (OOV) tokens into their standard equivalents that exist in the vocabulary. We identify nine common types of OOV token within code-mixed data, as details in Table 2.

Our approach is unsupervised. We create a mapping between OOV tokens into their standard forms using word distribution. However, some types are specific to a language, and cannot be solved with the word distribution information only. Thus, we combine the distributional approach with the rule-based method. Our method is outlined as follows.

1. OOV types which are encountered at both languages is tackled using translation dictionary built from word distribution approaches.

2. For ”reduplication with 2” problem, we removes character 2, apply the reduplication, and insert the hyphen in the text.

Types	**Language**	**Example / Explanation**
Non-standard spelling/typo	'en', 'id'	*lvie* or *youuuu*
Informal Abbreviations	'en', 'id'	"lht" for "lihat" (in English: "see"), "ppl" for "people"
Slang words	'en', 'id'	"epic" or "gue" for "saya" (in English: "I")
Spelled phonetically	'en', 'id'	"plis" for "please", "kalo" for "kalau" (in English: "if")
Reduplication with 2	'id'	In Indonesian, plural form of noun is written with a hyphen, e.g. "orang-orang" (in English: "people"). However, informal writing style often use the number "2" to indicate this (e.g. "orang2")
Hyphenless reduplication	'id'	On the other hand, the hyphen is sometimes not written (e.g. "orang orang")
Contracted words	'en'	"im" for "i am"
Combining English word with Indonesian prefix (nge-)	'en'	Indonesian people tend to use an informal prefix (nge-) before the words to stress that the word is a verb. For instance, the word "vote" is written as "ngevote"
Combining English word with Indonesian suffix (-nya)	'en'	Suffix "-nya" in Indonesian means the possessive pronoun (e.g. "miliknya" similar to "hers" or "his"). Suffix "-nya" can also refers to the definite article (in English: "the"). Informally, the suffix is used to follow English word usage in Indonesian conversation, e.g. "jobnya" (in English: "the job")

Table 2: Type of OOV Tokens

3. For "reduplication without hyphen" problem, we check whether the token consists of multiple words, then replace the space delimiter with "-" if it is a reduplication case.

4. For "contracted words" problem, we normalize them by utilizing the list provided by Kooten [3].

5. For problem of "combining English word with Indonesian prefix (nge-)", we remove the prefix (-nge) from the word.

6. For problem of "combining English word with Indonesian suffix (-nya)", we remove the suffix (-nya) and add the word "the" before the word.

There are two sub tasks in lexical normalization module. First, create mapping between OOV tokens to their standard forms. Then, build the system to incorporate the rule-based method with the distributional semantics.

[3]https://github.com/kootenpv/contractions

4.3.1 Build OOV and normal word mapping

Word embedding can be used to cluster words because it can model word relatedness, both syntactically and semantically. We use embedding model to construct mapping between OOV and its normal form. The word embedding model is trained by using skip-gram architecture (Mikolov et al., 2013). The procedure is described as follows:

1. Collect Indonesian and English vocabulary from Kateglo[4] and Project Gutebnberg[5].

2. For each word $normal\_word$ in the vocabulary, get 100 most similar words from social media corpus by using embedding model.

3. For each most similar words w_i to $normal\_word$
   - If w_i exists in the dictionary, it means that w_i is already normalized.
   - otherwise, w_i is OOV, then add a mapping instance between OOV word w_i into $normal\_word$.

[4]http://kateglo.com/

[5]http://www.gutenberg.org/ebooks/3201

4. If an OOV w_j is mapped into several $normal\_word$ entries, choose one which has the highest lexical similarity with w_j.

   We use the lexical similarity function (Hassan and Menezes, 2013). The function is based on the Longest Common Subsequence Ration (LCSR), which is the ratio of the length of the Longest Common Subsequence (LCS) and the length of the longer token (Melamed, 1999). The lexical similarity function defined as:

$$lex\_sim(s_1,\ s_2) = \frac{LCSR(s_1,\ s_2)}{ED(s_1,\ s_2)} \quad (1)$$

$$LCSR(s_1,\ s_2) = \frac{LCS(s_1,\ s_2)}{MaxLength(s_1,\ s_2)} \quad (2)$$

We apply static mapping as our mapping mechanism instead of finding the replacement online because of the execution time and memory performance. It is much faster to look up at the mapping rather than calculate it from the word embedding model. Furthermore, the memory needed to store the mapping is much smaller than the word embedding model.

4.3.2 Combine rules with mapping list

Normalization system employs combination of hand-crafted rules and mapping of words as follows.

1. Skip this procedure when the input is not a word (e.g hastag, mention, link, emoticon).

2. If the token is a multiword (there are more than one single word that is separated by white space), split the token into the list of single words. Try to normalize each word independently by applying the rules, and merge them back into one token. For instance, put the hyphen for reduplication case.

3. If the input is a word, then transform it into lowercase format. Reduce character repetition to at most two and consider any possible transformation. For instance, word "*Pleassssseee*" is transformed into "*pleassee*", "*pleasee*", "*pleasse*", and "*please*". Check whether one of generated transformed word is a normal form. If not, apply rule-based strategy. Last, use the mapping list created by utilizing word embedding.

4.4 Translation

This module aims to merge the list of tokens processed in previous modules back into one tweet and translates the tweet into Indonesian grammatical language. The module needs the Machine Translation (MT) system which is specific to the Indonesian-English code-mixed text domain. However, such MT system is not available at this moment. Dhar et al. (2018) found similar problem and tackled this by augmenting Matrix Language-Frame (MLF) model on top of the existing state-of-the-art MT system.

In the MLF model proposed by Myers-Scotton (1997), the code-mixed sentence can be splitted into the dominant language (the matrix language) and the embedded language. The matrix language grammar sets the morphosyntactic structure for the code-mixed sentence, while the embedded language borrows words from its vocabulary.

Thus, in this module, first, we merge the list of tokens into one tweet. Then, we separate the tweet into several sentences using sentence delimiter such as period (.), comma (,), the question mark (?), the exclamation mark (!), etc. For each sentence, we decide the language of the sentence (English or Indonesian sentence). To do that, we count how many words are English and how many words are Indonesian. The language which has a bigger frequency is the dominant language and also the language of the sentence, and the language which has a smaller frequency is the embedded language. After we decide the language of the sentence, we translate all the words into the language of the sentence. Last, if the language of the sentence in English, we translate the whole sentence into Indonesian. We use Microsoft Machine Translation[6] as the MT system. Figure 2 shows an example of input and output of translation module.

5 Experiments and Evaluation

We evaluate the performance of our implementation for each module and the pipeline model as whole process. First two modules (tokenization and language identification) are evaluated in supervised way using 4-fold cross validation setting.

Tokenization is evaluated at both character-level and token-level. The character-tagging achieves **98.70** for F1-score, while token identification obtains **95.15** for F1-score.

[6]https://www.microsoft.com/en-us/translator/business/machine-translation/

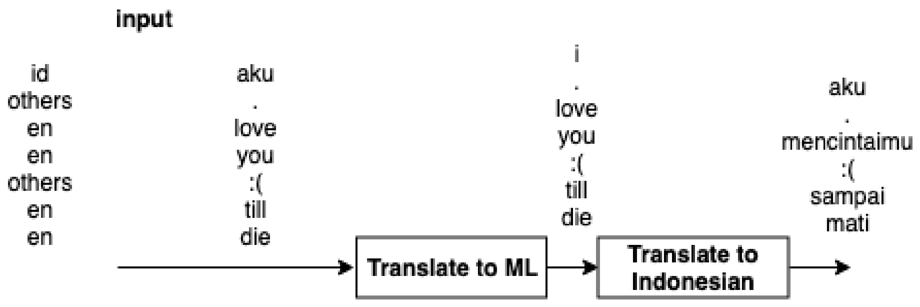

Figure 2: An Input-Output Example of Translation Module

The performance of our tokenization module excels NLTK *TweetTokenizer* tool, which scores F1 of 90.98 on evaluating token identification.

Language Identification module gets **89.58** F1-score and **90.11** accuracy. F1-score for label 'en', 'id', and 'rest' are respectively 87.07, 91.99, and 89.44 (the detail is in Table 3)

Language	**prec**	**recall**	**F1-score**
en	89.90	84.42	87.07
id	88.13	96.22	91.99
rest	94.99	83.96	89.14

Table 3: Language Identification Experiment Result

For evaluation of **lexical normalization**, we conduct a number of scenario. We test the difference of corpus source for building embedding model, i.e. combined corpora vs separated corpora. In first scenario, we only build single embedding model from merging of Indonesian and English corpus. While, in later, two distinct models are learned respective from each monolingual corpus. We also examine the contribution of rule-based strategy to enhance the word normalizer.

F1-score and accuracy are used as metric for evaluation. Those are measured across all the unique OOV words. If an OOV appears several times in the corpus, it is counted once. False positive is defined as a word that is actually a normalized form, but the system detects it as OOV word and normalizes the word incorrectly. On the contrary, false negative is a word that is OOV, but the system fails to detect it or fails to normalize the word into its standard form.

The best result is 81.31 for F1-score and 68.50 for accuracy, achieved when the mapping list of OOV and normal form is provided for separated language. A set of rules double the performance of normalization system. See Table 4) for detail.

Type	**F1-score**	**Accuracy**
Combined Corpora	47.49	31.14
Separated Corpora	48.34	31.87
Combined Corpora + Rule-based	80.96	68.01
Separated Corpora + Rule-based	81.31	68.50

Table 4: Lexical Normalization Experimental Result

Moreover, we investigate the errors by drawing sample of misclassified cases. False positive mostly happens in affixed words. In this case, affixed word is forced to transform into stem form. False negative occurs with words that supposed to be slang words, but they do exist in the vocabulary. For example, the word "aja" is commonly slang form of word "saja" (in English: "only"), but "aja" is found in Indonesia dictionary with different meanings.

When evaluating **translation** module, we test the effect of augmenting Matrix Language-Frame (MLF) Model into MT system. Incorporating MT system with MLF Model achieves better performance, **71.54** for BLEU and **19.50** for WER, as presented in Table 6.

Type	**BLEU**	**WER**
Without MLF	66.69	21.45
With MLF	71.54	19.50

Table 5: Translation Experimental Result
(BLEU: higher is better, WER: lower is better)

As integration of aforementioned modules, we evaluate the pipeline model by conducting four experiments, 1) comparing raw tweets with final tweets, 2) comparing raw tweets which have been translated (without MLF model) into Indonesian with final tweets, 3) comparing raw tweets which have been translated (with MLF Model) into Indonesian with final tweets, and 4) comparing raw tweets which have been normalized and translated (with MLF Model) into Indonesian with final tweets. From the experiments, our final pipeline obtains **54.07** for BLEU and **31.89** for WER. From Table 6, we can see that each module affects positively toward the performance of the pipeline. The pipeline model increases BLEU score for 8 points, and WER for 14 points compared to the baseline (raw tweets).

Pipeline	**BLEU**	**WER**
Raw Tweets	46.07	45.75
Raw Tweets + Translation (1 + 4 Module)	51.02	34.37
Raw Tweets + Translation with MLF Model (1 + 2 + 4 Modules)	51.75	34.39
Raw Tweets + Normalization + Translation with MLF Model (Full pipeline)	54.07	31.89

Table 6: Pipeline Experiment Result
(BLEU: higher is better, WER: lower is better)

6 Conclusion and Future Work

In this paper, we have proposed a pipeline model comprising of four modules, i.e. tokenization, language identification, lexical normalization, and translation. In addition, we also have prepared gold standard data consisting of 825 Indonesian-English code-mixed tweets for four different tasks corresponding to the modules. The data set is freely available online for research purpose only.[7] We experiments with Indonesian-English code-mixed Twitter data and the evaluation shows that our model works satisfactorily. Overall, the pipeline yields 54.07 scores for BLEU and 31.89 scores for WER.

However, the final result is not as high as the performance at the translation module because the final result uses the output from previous modules as inputs. The error from each module will propagate to the next modules. At the normalization module, using vector representations from word embedding for Indonesian-English code-mixed data quite good and applying the rule-based approach for each language improve the performance significantly. At the translation module, there are some errors caused by the MT system, but we could not do much about it since we use the existing MT system.

Moving forward, we would like to augment more data and enhance technique in order to improve performance of the model. Currently, lexical normalization module is much dependent of handcrafted rules. While the rule-based approach can increase the performance, it still not a robust solution seen from how language evolved.

[7]https://github.com/seelenbrecher/code-mixed-normalization

Acknowledgments

The authors acknowledge the support of Universitas Indonesia through Hibah PITTA B 2019 Pengolahan Teks dan Musik pada Sistem Community Question Answering dan Temporal Information Retrieval.

References

Putra Pandu Adikara. 2015. Normalisasi kata pada pesan/status singkat berbahasa indonesia. Master's thesis, Universitas Indonesia, Depok.

Russell Beckley. 2015. Bekli: A simple approach to twitter text normalization. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 82–86.

Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Manish Shrivastava, and Dipti Misra Sharma. 2018. Universal dependency parsing for hindi-english code-switching. *arXiv preprint arXiv:1804.05868*.

Amitava Das and Björn Gambäck. 2014. Identifying languages at the word level in code-mixed indian social media text. In *Proceedings of the 11th International Conference on Natural Language Processing*, pages 378–387.

Mrinal Dhar, Vaibhav Kumar, and Manish Shrivastava. 2018. Enabling code-mixed translation: Parallel corpus creation and mt augmentation approach. In *Proceedings of the First Workshop on Linguistic Resources for Natural Language Processing*, pages 131–140.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12):2009.

Novita Hanafiah, Alexander Kevin, Charles Sutanto, Yulyani Arifin, Jaka Hartanto, et al. 2017. Text normalization algorithm on twitter in complaint category. *Procedia computer science*, 116:20–26.

Hany Hassan and Arul Menezes. 2013. Social text normalization using contextual graph random walks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1577–1586.

Fei Liu, Fuliang Weng, Bingqing Wang, and Yang Liu. 2011. Insertion, deletion, or substitution?: normalizing text messages without pre-categorization nor supervision. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 71–76. Association for Computational Linguistics.

Deepthi Mave, Suraj Maharjan, and Thamar Solorio. 2018. Language identification and analysis of code-switched social media text. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 51–61.

I Dan Melamed. 1999. Bitext maps and alignment via pattern recognition. *Computational Linguistics*, 25(1):107–130.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Carol Myers-Scotton. 1993. Common and uncommon ground: Social and structural factors in codeswitching. *Language in society*, 22(4):475–503.

Carol Myers-Scotton. 1997. *Duelling languages: Grammatical structure in codeswitching*. Oxford University Press.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Rajat Singh, Nurendra Choudhary, and Manish Shrivastava. 2018. Automatic normalization of word variations in code-mixed social media text. *arXiv preprint arXiv:1804.00804*.

Vivek Kumar Rangarajan Sridhar. 2015. Unsupervised text normalization using distributed representations of words and phrases. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 8–16.