Normalization of Spelling Variations in Code-Mixed Data

Krishna Yadav[1], Md Shad Akhtar[1], Tanmoy Chakraborty[2]
[1]IIIT Delhi, India; [2]IIT Delhi, India.
{krishna19039, shad.akhtar}@iiitd.ac.in, tanchak@ee.iitd.ac.in

Abstract

Code-mixed text infused with low resource language has always been a challenge for natural language understanding models. A significant problem while understanding such texts is the correlation between the syntactic and semantic arrangement of words. The phonemes of each character in a word dictates the spelling representation of a term in low resource language. However, there is no universal protocol or alphabet mapping for code-mixing. In this paper, we highlight the impact of spelling variations in code-mixed data for training natural language understanding models. We emphasize the impact of using phonetics to neutralize this variation in spelling across different usage of a word with the same semantics. The proposed approach is a computationally inexpensive technique and improves the performances of state-of-the-art models for three dialog system tasks *viz.* intent classification, slot-filling, and response generation. We propose a data pipeline for normalizing spelling variations irrespective of language.

1 Introduction

There are around 6,500 languages spoken in the world today(Wikipedia contributors, 2021). *English, Mandarin, Chinese* tops the list with over 2 billion speakers around the globe hence are highly resourceful languages. On the other hand, there are resource-scarce languages such as *Polish, Odia, Hindi and, many more* with few million speakers only. Due to lack of resources understanding such languages poses a great challenge for the research community. Natural Language Understanding (NLU) means extracting the semantic schema of the utterance to re-act according to the intent of the utterance. NLU is crucial for any human-to-machine interaction-based system such as chatbots, virtual assistants, and many more. Now the mode of communication is restricted by the speaker's language and innate understanding of language systems.

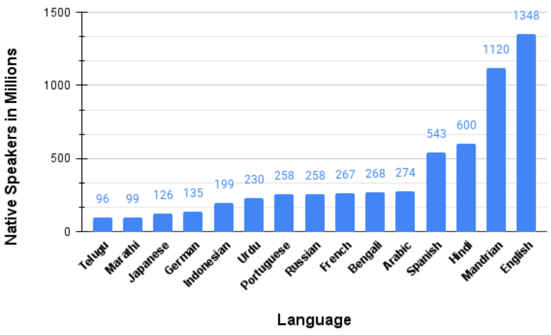

Figure 1: Native Speakers Count

The Goal-Oriented Dialogue System (Young, 2000) was first introduced based on dialog state tracking (Williams et al., 2013) and gave a new direction to the NLU tasks. A number of datasets are available in diverse domains for like-wise downstream goal-oriented conversational data such as ATIS (Hemphill et al., 1990), SNIPS (Coucke et al., 2018), DSTC (Williams et al., 2013), WOZ (Budzianowski et al., 2018), etc. However, all of them are monolingual, i.e., available in the English language only. Code-mixing is the form of linguistics where the conveyer uses two (or more than two) languages together so that some words of the low resource language replace the words from high resource language or vice versa. The usual trend is to mix English with any other regional language such as Hindi (Hindi + English $\rightarrow$ Hinglish), Bengali (Bengali + English $\rightarrow$ Benglish), Tamil (Tamil + English $\rightarrow$ Tamilish), and many more. With the increase in multi-lingual speakers, code-mixing is very common in online platforms, social media, and day-to-day life (Gumperz, 1982; Gysels, 1992; Durán, 1994; Moyer, 2002). Most common activities such as shopping, restaurant reservations, booking tickets, and so on all involve extensive use of Code-mixing. For example, a Hindi speaking

Language	Utterances
English	**Speaker 1**: Hi, Can you help me in booking a table at this restaurant? **Speaker 2**: Sure, would you like something in cheap, moderate or expensive price range?
Hinglish	**Speaker 1**: Hi, kya tum is restaurant mein ek table book karne mein meri help karoge? **Speaker 2**: Sure, kya aap cheap, moderate ya expensive price range mein kuch like karenge?

Table 1: Sample Code-Mixed Utterances.

user looking to book a restaurant would typically ask, ”Kya tum is restaurant mein ek table book karne mein meri help karoge?” (”Can you help me in booking a table at this restaurant?”) (Banerjee et al., 2018).

A significant proportion of the population worldwide is using code-mixed language over online platforms (Singh et al., 2018). The prime complication with linguistic diversity is such that there is no convention or protocol to refer to when it comes to code-mixing. Hence, it depends on the writer’s perception and knowledge of the phonics of the source language. (Kukich, 1992) grouped writing errors into two classes. First one is typographical that occurs when a character is substituted by the wrong character whose key is nearby in the keyboard or interchanging of the character order, for instance, *merw paas(mere paas), kimd(kind), kys krna ha(kya krna hai), and more variations due to different reasons.* Other class is of cognitive errors that occur when the writer is unaware of the native spelling and semantics of that word. In this case, the wrongly spelled word is phonetically close to the correct word (Toutanova and Moore, 2002). We assume that the chances of typographical errors are less since the writer intends to avoid making such errors. This paper mainly focuses on cognitive errors and covers a major portion of typographical errors as well.

In the case of code-mixed data, there are no rules that can lead towards achieving the correct spelling because there is no correct spelling. We can assume the most commonly used representation as correct and normalized text to get some contextual meaning. The introduction of external knowledge can also help to improve the results of spelling correction. We propose a computationally inexpensive novel technique to normalize spelling variations irrespective of the language of the bilingual speaker.

2 Related Work

Divergence from the traditional spelling and having variation for the same word often carry some meaning (Sebba, 2007). In computational linguistics, while dealing with digital forms of regional text forms, it becomes helpful to map all spelling variations (semantically identical) to the same point in embedding space. (Nguyen and Grieve, 2020) highlighted a detailed analysis on the same. They analyzed that the skip-gram model, which does not consider spelling variations, encode spelling variation patterns to some extent. Also, the use of cosine similarities helped find a link between intentional variations and distance from the conventionally followed standards.

Historical writings face the identical problem of high degree variance in spellings (Reynaert et al., 2012)—every day with new findings in historical text and extending the corpora in digital form (literature). Various researches already explored the normalization approaches based on string distance measures to a reasonable extent for proposing various tools for normalization. (Reynaert et al., 2012) shows that, individually, the rule-based method (Norma Tool) performed best in the presence of a large amount of training data (Bollmann et al., 2012). A combination of normalization methods produces the best results and helps in further cleaning and processing of data. Hence, integrating simple word-to-word mappers always increase the overall performance. Methods like Edit Distance or Levenshtein distance (Levenshtein, 1965; Yujian and Bo, 2007) needs a massive corpus of universally correct data. (Bollmann and Søgaard, 2016) further gave improvisation for this problem with the use of bi-LSTM network (Schuster and Paliwal, 1997) applied on a character level. Multi-task learning with additional normalization (integration of mappers) improves the model’s performance. Their model outperformed the CRF-based models and Norma tool given in (Bollmann et al., 2012). Extending the work keeping in mind the idea of integrating, (Domingo and Casacuberta, 2018) proposed three approaches based on statistical, neural, and character-level machine translation to train the model concerning modern spelling variation standards. Their model covered a holistic view of the word-to-word mappers. Additionally, they proposed a simplistic approach of a statistical dictionary, similar to a word-to-word mapper in which they used the changing frequency of spelling on the training corpora. They also stated that the statistical machine translation approach gave better results than neural machine translation on small corpora. (Lertpiya et al., 2020) explored another low resource language which was (*Thai*). In their work, they proposed a two-staged pipeline with neural contextual attention. Using neural error corrector and Seq2Seq error corrector alleviates the problem of *overcorrection*.

The phonetically motivated approach has also been explored a little by (Downs et al., 2020) where they prioritized the phonetic key of the misspelled word over supplementary ones. A survey conducted for (Weld et al., 2021) reports an excellent survey on joint intent classification and slot filling techniques. They NLU models of over a decade and gave a detailed comparison with the pros and cons of various techniques. Concluding the state-of-the-art research, they provide multiple comparisons that best summarise the past work done along different dimensions, including the features, base approaches, and dataset domain used. Hybrid phonetic neural models (Viana-Cámara et al., 2021) and BERT (Devlin et al., 2018) models have also been explored to capture character-phonetic but they don't capture the code-mixed data. (Hládek et al., 2020) conducted a survey of spelling correction techniques. They studied the interactive process of error production and correction. All the major research assumes that the correct spelling of the miss-spelled word is native to the written language. In the case of code-mixed data, no such thesaurus exists. Additionally, there is the absence of any particular set of rules that one can use for code-mixing, and it is complex to come up with such a method or protocol to translate one language into a romanized language.

Recently, (Sengupta et al., 2021) came up with a method for sub-word level representation learning that is supplemented by the word level lexical variations in code-mixed languages. They evaluated the proposed architecture on a mix of European and Indic languages *(Spanish, Hindi, Bengali, Tamil, Telugu, and Malayalam)*. They proposed a Hierarchically attentive Transformer (HIT) framework, a novel architecture to encode text semantic and syntactic features in an embedding space with efficacy. It learns word representations at the sub-word level using a Fused Attention MEchanism (FAME). It incorporates two major attention components. An outer product attention (OPA) (Le et al., 2020) to extract higher-order character-level similarities and multi-headed self attention (MSA) (Vaswani et al., 2017), a standard transformer module that computes a scaled query-key vector pair dot product. FAME extends the MSA module by including OSA and calculates their weighted sum. The proposed model tries to embed semantically and phonetically similar words of a code-mixed language by capturing relevant information at a more granular level but lacks overall coverage of spelling variations. HIT is computationally expensive with over trainable parameters *2.7M* for sequence classification and over *1.4M* for POS tagging. It misses the essence of layman language in text utterances.

3 Dataset Used

We will use a code-mixed version of the DSTC2 dataset (Williams et al., 2013; Henderson et al., 2014a,b; Williams et al., 2014, 2016). They incorporated code-mixing in four regional languages Hindi, Bengali, Gujarati, and Tamil, romanized as English (Banerjee et al., 2018) by crowd-sourcing the data for language translation from native speakers of respective languages. The data contains 50k utterances on the restaurant reservation domain, including getting reservations done or asking for information such as restaurant address, phone, etc. Final data contains bot-to-human dialog conversations. The authors converted the raw data from audio format to text. For this task, the authors used Automatic Speech Recognition (ASR) modules.

Sentence	cheap	restaurant	south	west	mein
Slots	B-Price	O	B-Area	I-Area	O
Intent	inform				

Table 2: BIO-Tagging

Representation: There are 3 slots, 5 possible areas, 91 cuisines, and 3 price ranges. Workers acting as customers were requested to deviate the conversation in the middle of the dialog to various slots and their possible values to make data robust, less intuitive, and unconditional and avoid unnecessary patterns in data.

The workers transcribed the conversations and labeled the utterances with different dialog states. For example, each utterance was labeled with its semantic intent representation (request[area], inform[area = north]) and the dialog turns were labeled with annotations such as constraints on the slots (cuisine = Italian), requested slots (requested = phone, address) and the method of search (by constraints, by alternatives). Such annotations are useful for domain-specific slot-filling based dialog systems (Banerjee et al., 2018). This whole process consists of three phases. Extracting unique utterances from DSTC2, i.e., dialogues with only change in the slot values rest same are filtered out. Creating code mixed translations using Amazon Mechanical Turk (AMT) tool for crowd-sourcing and by in-house workers. The evaluation involved taking random dialogues from the dataset for colloquialism (unforced translation), Intelligibility (easily understandable), and Coherence(irrespective of neighboring utterances knowledge).

The quantitative measure of code-mixing : As per the evaluation process. The authors analyzed the obtained code-mixed data for code-mixing measures. (Gambäck and Das, 2016) gave a metric to measure code-mixing in a sentence given as:

$$: if N(x) > 0$$

$$C_u(x) = 100.\frac{N(x) - \max_{L_i \in \mathcal{L}}\{t_{L_i}\} + P(x)}{2N(x)} \quad (1)$$

Here, the measure of code-mixing for sentence x, $C_u(x)$ is given by $N(x)$, number do foreign language tokens in sentence. Maximum number of tokens t in language L_i from the set of languages $\mathcal{L}$. In addition, number of language switch points given by $P(x)$. In the above equation, the language of the majority of words in the sentence serve as *Embedding* however, irrespective of the majority, we need to consider English as *Embedding* and Hindi as *Matrix* language. To over come this the authors proposed to replace the general $\max_{L_i \in \mathcal{L}}\{t_{L_i}\}$ with $native(x)$ given as:

$$native(x) = \begin{cases} \{t_{L_n}\} & : t_{L_n} > 0 \\ N(x) & : t_{L_n} = 0 \end{cases} \quad (2)$$

Here, t_{L_i} is the number of tokens in native language(Hindi). Also, the term $\delta(x_i)$ with values 0(if switch in Matrix language) or 1(purely English) to measure the extent of inter-utterance code-mixing and frequency. The authors also considered the fraction of code-mixed utterances $\frac{S:CMUtterances}{U:TotalUtterances}$. And the final equation is given as $C_u(x)$ for one utterance and $C_c(x)$ for complete corpus:

Property	Count
Total Utterances	49167
Unique utterances	6733
Utterances per dialogue	15
Words per utterance	8
Words per dialogue	120
KB triples per dialogue	38
Train dialogue	1168
Validation dialogue	500
Test dialogue	1117
Vocab Size	1229

Table 3: Raw Dataset Analysis.

Property	Count in Hinglish
Unique Utterances	6549
Code-Mixed Utterances	5750
Hindi Only Utterances	348
English Only Utterances	451
Utterances per dialogue	12
Words per utterance	8
Hindi Vocab Size	739
English Vocab Size	551
Code-Mixed Vocab Size	386

Table 4: Code-Mixed Dataset Analysis.

$$\begin{aligned} C_u(x) &= \left(1 - \frac{native(x) + P(x)}{N(x)} + \delta(x)\right) \\ C_c(x) &= \frac{100}{U}\left[\frac{1}{2}\sum_{i=1}^{U} C_u(x) + \frac{5}{6}S\right] \end{aligned} \quad (3)$$

4 Proposed Methodology

We propose a novel method of robust systems for low resource similar token spelling variations. The main reason for little spelling variations in the pronunciation and ambiguous phonetics of the word. Due lack of any formal conversion criterion, the writer introduces variations. The aim is to map the semantically similar romanized words such as *Kabhi, kabi, kbhi, etc* together so that they can be neutralized to a single term and have identical embedding. Then give a most common and close to

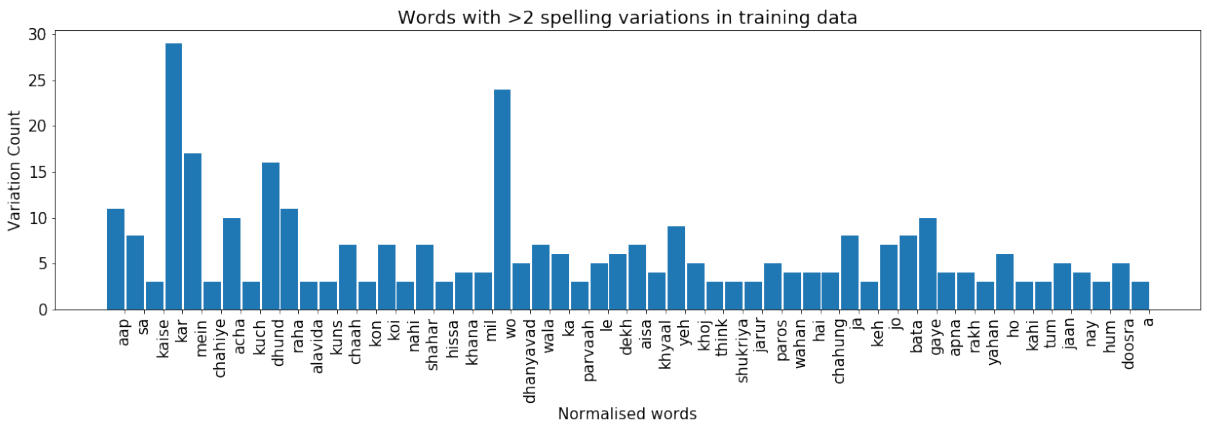

Figure 2: Spelling Variation Count Analysis

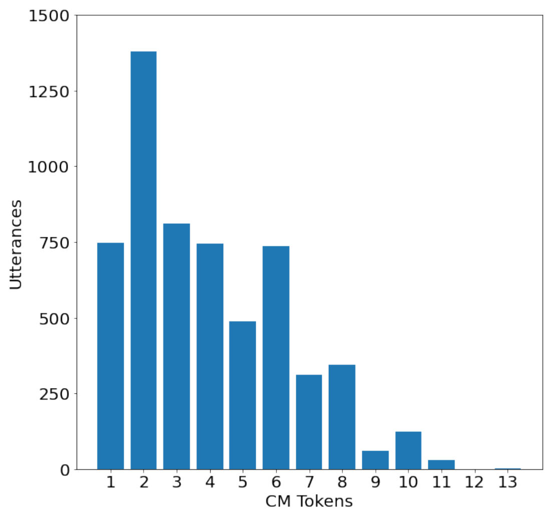

Figure 3: Utterance Count VS Hindi word count

correct representation of that word. We approach this statement in 5 phases as follows:

1. **Capture Code Mixing:** The point of focus is the tokens that belong to *Matrix* Language. For this, we markdown the predefined tokens (*here, intent labels, slot labels* and special tokens) according to the training data. For this purpose, we use English Dictionary for identifying the embedded language words. We leave the English words unchanged and sent the non-English words to the second phase.

2. **CM Elocution:** In this phase, we try to find a set of possible transliterations of each token in their native script *here, Devanagari*. We have an option to take the transliterated terms as to be syntactically and semantically incorrect in the *Matrix* Language since the whole idea is to capture the different pronunciation styles in the native language of the word. We use *indic-transliterator* (Bhat et al., 2015) for this job and stored top five closest transliterations for each romanised token. These transliterations sound similar to each other, with a minor change in terms of vowels and consonants.

3. **Candidate Selection (Devanagari Phoneme):** The set of recently formed *Devanagari* tokens are the closest possible phonetically similar terms that all sound the same but differ in the writing style and hence are close to each other. To reduce variation, we need to normalize all the possible variations with the most commonly used term. Now that we have to query for Hindi (*Devanagari*) text, there is plenty of corpora that we can take as a benchmark. It may or may not be semantically correct, but it will be the most used term by the majority of the population. We used the IIT Bombay parallel corpus (Kunchukuttan et al., 2017) to perform candidate selection. We use TF-IDF on the *Devanagari* translation of the dataset. The term with the max score is selected to be the best possible normalization for all the remaining terms. This newly elected normalized term which syntactically correct as per the *Matrix* Language. This way, we are able to pull the spelling variations together in multi-dimensional embedding space of the *Matrix* Language.

4. **Romanisation:** This step is similar to the 2^{nd} phase. For the CM dataset, the Devanagari terms are converted to their romanized

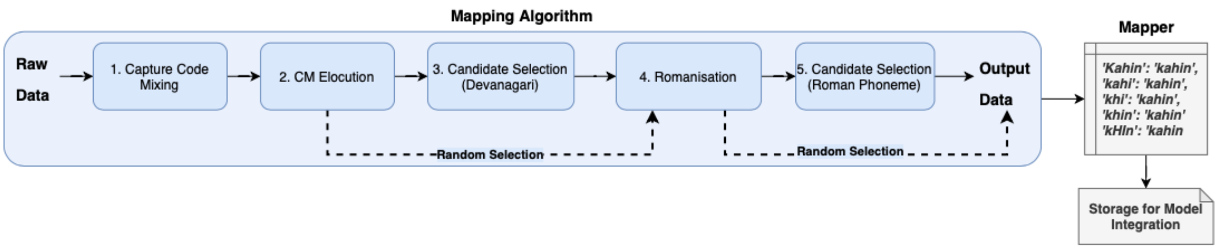

Figure 4: Proposed Data Pipeline.

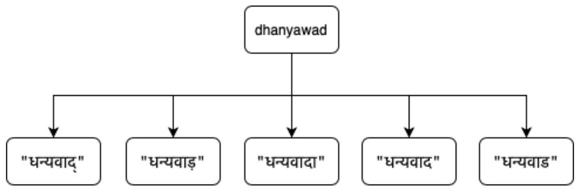

Figure 5: Devanagari Transliteration Generation

elocution by transliteration. Now, this is the crucial step; we do not have any order or protocol for elocution until now. However, we intentionally introduce minor spelling variations as part of code-mixing noise. Now when we have the map of code-mixed words, we can normalize it again based on candidate selection, and this way, we normalize the whole set of writer introduced code-mixing to abide by the code-mixing methodology.

Count	Language	Train	Test	Dev
Sentences		1,492,827	2,507	520
Tokens	English	20,667,259	57,803	10,656
	Hindi	22,171,543	63,853	10,174

Table 5: IIT Bombay Corpus Statistics

5. **Candidate Selection (Romanised Phoneme):** This step is similar to the earlier candidate selection, the only difference being the language to the terms. Here, we select one most used romanized *matrix* language term. TF-IDF scores catered to this selection from a rich bi-lingual corpus collected from various social media platforms to capture the latest trends of code-mixing generalized to various writers. We used Facebook, Twitter, WhatsApp chat dataset given by (Das, 2016) further explored in (Ramesh and Kumar, 2016). This dataset is enriched with a quality code-mixing that best caters to our need to figure out the most widely used representations of a *matrix* language term. With this, we get a single normalized term for all the syntactic variations of semantically similar hi-English terms. This output contains non-English terms without spelling variations for roman English (Hindi) words.

With this, we give novel methods and pre-determined mapping for most commonly used spelling variations with their preferred normalization. This mapping can be helpful in learning models for spelling correction for natural language understanding tasks.

5 Experiments

We perform numerous trials and runs to prove the importance of spelling normalization. We took a downstream task of intent classification and slot prediction. We use the BIO-Tagging format to transform the data for performing joint learning. We then compare the performance of state-of-the-art algorithms on our normalized data and old data with variations.

1. **CS-ELMO** (Aguilar and Solorio, 2019) used a state-of-the-art monolingual model for finding the sentence embedding for dialogue systems and used transfer learning to develop a model for code-switched bilingual text. Their model transfer English knowledge from a pre-trained ELMo model to code-switched language (Hi-English) using the task of language identification (Matrix language and Embedding Language). They used character convolutions for capturing character positions of unknown words. Their model outperformed the multilingual BERT (Devlin et al., 2018), and another code-switching ignorant monolingual model like ELMO (Peters et al., 2018).

2. **Stack Propagation** (Qin et al., 2019) considered the strong correlation of intent classification and slot filling (Zhang and Wang, 2016;

Hakkani-Tür et al., 2016; Liu and Lane, 2016). Qin's model used the intent information directly into the slot filling stage. Once the utterance's intent is classified, we concatenate the label to the slot representation for predicting the slots for each of the tokens simultaneously. They performed token level intent classification for further alleviating error propagation and finally passed the representation to the BERT layer for further performance gain. The semantic knowledge in the form of intents of each utterance act as a differential link between the two tasks. They also demonstrated the use of gated architecture proposed in (Goo et al., 2018).

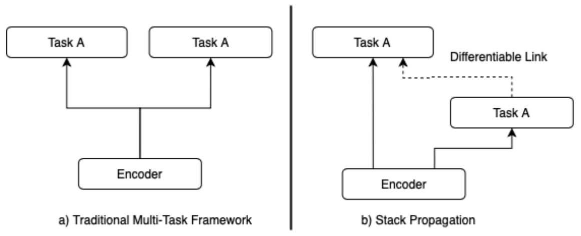

Figure 6: Multi-Task Vs Stack Propagation

3. **DCA-Net** Recently, (Qin et al., 2021) demonstrated an excellent work of propagating semantic knowledge from one task to another in the form of joint learning. In (Qin et al., 2019) there was a unidirectional flow of the information to overcome this Qui et al. further extended their work and proposed a co-interactive transformer module by establishing a bidirectional cross-impact between the two tasks in a unified architecture. Both intent classification and slot filling can take advantage of mutual information. Their model achieved state-of-the-art performance. The major drawback of their model is that it fails to incorporate code-switching and fails on our Hi-English data. We can further tune this method for extending the scope for multilingual data.

4. **HIT** A robust representation learning method was recently proposed by (Sengupta et al., 2021). They computed the weighted sum of two attention modules, multi-headed self-attention, and an outer product attention module, to obtain the final attention weights. An outer product attention (OPA) (Le et al., 2020) to extract higher-order character-level similarities and multi-headed self attention (MSA) (Vaswani et al., 2017), a standard transformer module that computes a scaled query-key vector pair dot product. HIT was able to encode syntactic and semantic features in embedding space by learning sub-word level representations with their fused mechanism called *FAME*. FAME extends the MSA module by including OSA and calculates their weighted sum. This model did not perform well due to incompetencies of efficiently reducing the distance between semantically same but syntactically varied code-switched terms.

6 Implementation

We report BLEU-4 (Papineni et al., 2002) and ROUGE-1, ROUGE-2 and ROUGE-L scores (Lin, 2004) for natural language generation machine translation task to compare the results with the dataset baselines given by (Banerjee et al., 2018; Banerjee and Khapra, 2019). Further, we compute Precision, Recall, and F1 scores for intent classification and Slot Filling evaluation. We calculate weighted and macro average scores and then chose macro overweighted because this is a class imbalance problem. The weighted average will give significant weightage to the most frequent class whose performance may lead to 100%. To encounter this issue, we report macro averaged scores, i.e., an average of independent scores for each class, treating all classes equally.

Further, We play with different combinations of the state-of-the-art algorithms and compared the performance on both syntactically normalized *(Spell)* and un-normalized *Inc_spell* versions of the dataset. We represent the data as each line containing a token and utterances separated by an empty line. Each token corresponds to a BIO tagging label, where B refers to the label's beginning, I refers to the intermediate words of the label, and O refers to other classes (no label of interest).

7 Results and Comparison

The table 7 shows the effect of normalizing spelling variations in code-mixed data and clearly shows that our novel architecture helps further to improve the quality of natural language generation tasks. The baseline algorithms Seq2Seq and Hred (Banerjee et al., 2018) along with graph convolutions network with sequential attention (Banerjee and

			Intent Macro Average				Slot Macro Average			Improvement
Model	Data	Intent Acc	Pre	Rec	F1	Slot Acc	Pre	Rec	F1	
CS-ELMo	Original	86.7	69.89	60.3	64.75	**99.55**	72.89	70.06	71.44	0.5464
CS-ELMo	Normalized	86.03	69.12	58.66	63.46	99.47	72.04	68.78	70.37	
Stack Prop	Original	93.56	86.51	78.89	81.98	98.85	76	76	77	0.3942
Stack Prop	Normalized	91.17	85.04	77.38	81.03	98.65	76	77	77	
cselmo+ stackProp	Original	**94.17**	**87.22**	**79.71**	**83.29**	99.23	**78.77**	78.38	**79.81**	0.4364
cselmo+ stackProp	Normalized	93.23	86.02	78.92	82.45	98.97	78.02	78.19	78.67	
DCA-Net	Original	85.73	75.3	62.7	66.86	98.92	69.21	63.64	64.08	0.2792
DCA-Net	Normalized	84.6	74.43	63.53	66.25	98.76	68.47	62.81	63.68	
cselmo+DCA-Net	Original	85.77	62.01	61.82	60.57	98.52	66.85	67.69	64.28	0.4771
cselmo+DCA-Net	Normalized	84.32	61.17	61.02	59.94	98.12	65.96	66.81	63.49	
HIT	Original	86.33	76.75	63.59	67.68	98.28	61.35	**84.39**	52.78	0.37217
HIT	Normalized	85.36	75.83	64.52	66.55	97.44	60.72	83.98	51.54	

Table 6: Comparing different combinations of state-of-the-art models on Hi-English Data.

Model	Data	Bleu	Rouge-1	Rouge-2	Rouge-L
Seq2Seq	Normalized	55.1	62.9	52.5	61
Seq2Seq	Original	55.9	63.55	53.1	62.09
Hred	Normalized	55.3	63.4	52.7	61.15
Hred	Original	55.61	63.92	53.25	61.91
GCN-SeA	Normalized	57.1	66.4	56.8	64.4
GCN-SeA	Original	57.4	66.78	56.4	65.98

Table 7: Effect of Spelling Normalization

Khapra, 2019) both results in higher performance increased by *0.5 - 1.5 units*, when normalised data is used. Normalization forces the vector representations for semantically identical but syntactically close terms to overlap each other in multidimensional embedding space and forces the model to treat all possible variations as the same only.

From table 6 we can infer, for all the state-of-the-art approaches, the proposed modification has lead to a significant performance gain by a factor of 0.5%-1.5%. The maximum improvement of 1.23 in slot F1 and 1.35 in intent F1 score can be seen in CS-ELMO, considering it explicitly focuses on the various types of character level embedding to capture context. The combinations of CS-ELMO with DCA-Net (0.8 for slot F1 and 0.67 in intent F1) and CS-ELMO with Stack Propagation (1.14 in slot F1 and 0.84 in intent F1) algorithms also shows significant performance improvement. Hence, the importance of tackling spelling variation in code mixed data is evident from the above results. Integrating different modules results in overcoming the drawbacks of each module in one way or another. This helps in improving the over performance.

8 Conclusion

With this emerging trend of code-mixing over social media platforms and daily communication in almost every region, the necessity to develop efficient natural language understanding models has increased. We communicate in a specific language (say, English) because the language has set standards, semantics, syntactic, and phonetics. Without any standard for low-resource languages, it is tough to communicate. Different people have different ways of pronouncing and depicting the alphabet, phonetics, and accent of a language close to their mother tongue. This action may lead to spelling variations while writing text as a medium for communication. We introduce a novel, computationally inexpensive, fully robust, and efficient method to normalize these spelling variations that work as an auto-correct to counter this problem. This mechanism helps in performance gain not only machine translation but also generic natural language tasks and any downstream task involving code-mixing problems. We give a universal mapping for Hi-English code-mixing that can be used directly by making a query in $O(1)$ time to normalize non-English words. It is evident from our work that integrating several modules together helps in performance improvement. We can further improve this solution by learning a model to further normalize out of the vocabulary terms.

Acknowledgment

Authors acknowledge the support of Infosys Center of Artificial Intelligence (CAI), IIIT Delhi.

References

Gustavo Aguilar and Thamar Solorio. 2019. From english to code-switching: Transfer learning with strong morphological clues. *arXiv preprint arXiv:1909.05158.*

Suman Banerjee and Mitesh M Khapra. 2019. Graph convolutional network with sequential attention for goal-oriented dialogue systems. *Transactions of the*

Association for Computational Linguistics, 7:485–500.

Suman Banerjee, Nikita Moghe, Siddhartha Arora, and Mitesh M Khapra. 2018. A dataset for building code-mixed goal oriented conversation systems. *arXiv preprint arXiv:1806.05997*.

Irshad Ahmad Bhat, Vandan Mujadia, Aniruddha Tammewar, Riyaz Ahmad Bhat, and Manish Shrivastava. 2015. Iiit-h system submission for fire2014 shared task on transliterated search. In *Proceedings of the Forum for Information Retrieval Evaluation*, FIRE '14, pages 48–53, New York, NY, USA. ACM.

Marcel Bollmann, Stefanie Dipper, Julia Krasselt, and Florian Petran. 2012. Manual and semi-automatic normalization of historical spelling-case studies from early new high german. In *KONVENS*, pages 342–350.

Marcel Bollmann and Anders Søgaard. 2016. Improving historical spelling normalization with bi-directional lstms and multi-task learning. *arXiv preprint arXiv:1610.07844*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. Multiwoz–a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Amitava Das. 2016. Tool contest on pos tagging for code-mixed indian social media (facebook, twitter, and whatsapp) text. In *Proceedings of the 13th International Conference on Natural Language Processing*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Miguel Domingo and Francisco Casacuberta. 2018. Spelling normalization of historical documents by using a machine translation approach. *Congresos - EAMT2018 - Proceedings*.

Brody Downs, Oghenemaro Anuyah, Aprajita Shukla, Jerry Alan Fails, Maria Soledad Pera, Katherine Wright, and Casey Kennington. 2020. Kidspell: A child-oriented, rule-based, phonetic spellchecker. *LREC 2020, Twelfth International Conference on Language Resources and Evaluation, 6937-6946*.

L. Durán. 1994. Toward a better understanding of code-switching and interlanguage in bilinguality: Implications for bilingual instruction. In *The journal of educational issues of language minority students*.

Björn Gambäck and Amitava Das. 2016. Comparing the level of code-switching in corpora. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1850–1855.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757.

John J Gumperz. 1982. *Discourse strategies*. 1. Cambridge University Press.

M. Gysels. 1992. French in urban lubumbashi swahili: Codeswitching, borrowing, or both? *Journal of Multilingual and Multicultural Development*, 13:41–55.

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech*, pages 715–719.

Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014a. The second dialog state tracking challenge. In *Proceedings of the 15th annual meeting of the special interest group on discourse and dialogue (SIGDIAL)*, pages 263–272.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014b. The third dialog state tracking challenge. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 324–329. IEEE.

Daniel Hládek, Ján Staš, and Matúš Pleva. 2020. Survey of automatic spelling correction. *Electronics*, 9(10).

Karen Kukich. 1992. Techniques for automatically correcting words in text. *ACM Comput. Surv.*, 24(4):377–439.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2017. The iit bombay english-hindi parallel corpus. *arXiv preprint arXiv:1710.02855*.

Hung Le, Truyen Tran, and Svetha Venkatesh. 2020. Self-attentive associative memory. In *International Conference on Machine Learning*, pages 5682–5691. PMLR.

Anuruth Lertpiya, Tawunrat Chalothorn, and Ekapol Chuangsuwanich. 2020. Thai spelling correction and word normalization on social text using a two-stage pipeline with neural contextual attention. *IEEE Access*, 8:133403–133419.

V. Levenshtein. 1965. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady*, 10:707–710.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*.

Melissa G. Moyer. 2002. Pieter muysken, bilingual speech: A typology of code-mixing. cambridge: Cambridge university press, 2000. pp. xvi, 306. hb $ 59.95. *Language in Society*, 31:621 – 624.

Dong Nguyen and Jack Grieve. 2020. Do word embeddings capture spelling variation? In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 870–881.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Libo Qin, Wanxiang Che, Yangming Li, Haoyang Wen, and Ting Liu. 2019. A stack-propagation framework with token-level intent detection for spoken language understanding. *arXiv preprint arXiv:1909.02188*.

Libo Qin, Tailu Liu, Wanxiang Che, Bingbing Kang, Sendong Zhao, and Ting Liu. 2021. A co-interactive transformer for joint slot filling and intent detection. In *ICASSP*.

Sree Harsha Ramesh and Raveena R Kumar. 2016. A pos tagger for code mixed indian social media text-icon-2016 nlp tools contest entry from surukam. *arXiv preprint arXiv:1701.00066*.

Martin Reynaert, Iris Hendrickx, and Rita Marquilhas. 2012. Historical spelling normalization. a comparison of two statistical methods: Ticcl and vard2. *Proceedings of ACRH-2*, pages 87–98.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

Mark Sebba. 2007. *Spelling and society: The culture and politics of orthography around the world*. Cambridge University Press.

Ayan Sengupta, Sourabh Kumar Bhattacharjee, Tanmoy Chakraborty, and Md Shad Akhtar. 2021. Hit: A hierarchically fused deep attention network for robust code-mixed language representation. *arXiv preprint arXiv:2105.14600*.

Kushagra Singh, Indira Sen, and Ponnurangam Kumaraguru. 2018. Language identification and named entity recognition in hinglish code mixed tweets. In *Proceedings of ACL 2018, Student Research Workshop*, pages 52–58.

Kristina Toutanova and Robert C Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 144–151.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Rafael Viana-Cámara, Mario Campos-Soberanis, and Diego Campos-Sobrino. 2021. Hybrid phonetic-neural model for correction in speech recognition systems. *arXiv preprint arXiv:2102.06744*.

HENRY Weld, Xiaoqi Huang, SIQU Long, Josiah Poon, and SOYEON CAREN Han. 2021. A survey of joint intent detection and slot-filling models in natural language understanding. *arXiv preprint arXiv:2101.08091*.

Wikipedia contributors. 2021. List of languages by number of native speakers — Wikipedia, the free encyclopedia.

Jason Williams, Antoine Raux, and Matthew Henderson. 2016. The dialog state tracking challenge series: A review. *Dialogue & Discourse*, 7(3):4–33.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413, Metz, France. Association for Computational Linguistics.

Jason D Williams, Matthew Henderson, Antoine Raux, Blaise Thomson, Alan Black, and Deepak Ramachandran. 2014. The dialog state tracking challenge series. *AI Magazine*, 35(4):121–124.

Steve J Young. 2000. Probabilistic methods in spoken–dialogue systems. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 358(1769):1389–1402.

Li Yujian and Liu Bo. 2007. A normalized levenshtein distance metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1091–1095.

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 2993–2999. AAAI Press.