PACMAN: PArallel CodeMixed dAta generatioN for POS tagging

Arindam Chatterjee[1,2]*, **Chhavi Sharma**[1]*, **Ayush Raj**[1], **Asif Ekbal**[2]
[1]Wipro AI Research Lab45, Bangalore, India
[2]Indian Institute of Technology, Patna, India
{arindam.chatterjee4, chhavi.sharma5, ayush.raj3}@wipro.com
asif@iitp.ac.in

Abstract

Code-mixing or Code-switching is the mixing of languages in the same context, predominantly observed in multilingual societies. The existing code-mixed datasets are small and primarily contain social media text that does not adhere to standard spelling and grammar. Computational models built on such data fail to generalise on unseen code-mixed data. To address the unavailability of quality code-mixed annotated datasets, we explore the combined task of *generating* annotated code-mixed data, and building *computational models* from this generated data, specifically for code-mixed Part-Of-Speech (POS) tagging. We introduce *PACMAN*[1] (*PA*rallel *C*ode*M*ixed d*A*ta generatio*N*) - a synthetically generated code-mixed POS tagged dataset, with above *50K* samples, which is the *largest annotated code-mixed dataset*. We build POS taggers using classical machine learning and deep learning based techniques on the generated data to report an F_1-score of 98% (8% above current State-of-the-art (SOTA)). To determine the efficacy of our data, we compare it against the existing benchmark in code-mixed POS tagging. PACMAN outperforms the benchmark, ratifying that our dataset and, subsequently, our POS tagging models are generalised and capable of handling even natural code-mixed and monolingual data.

1 Introduction

Code-Mixing or Code-Switching is primarily observed and archetypal in multilingual societies across the globe (Gumperz, 1964; Thompson, 2009; Auer, 2020; Schwab, 2021). Although linguists differentiate between code-mixing (Boggs, 1983) and code-switching (Myers-Scotton, 1993), we use CM to mean both. CM has recently garnered significant interest for researchers because of its gradual emergence as the *prima lingua* for social media posts, blogs, chats, and messages.

The first computational work on CM text was explored by Solorio and Liu (2008) for English-Spanish POS tagging. They used individual POS taggers for English and Spanish and heuristics, machine learning, and word-level language information (WLI) to find the optimal tag for each word. Subsequent work on POS Tagging for CM text like Jamatia et al. (2015); Vyas et al. (2014) use similar approaches, using language information and a combination tagger on *social media* CM text. They emphasize that WLI is a mandatory ingredient for CM POS tagging, necessitating the use of a combination tagger. Recent work by Singh et al. (2018) applies a set of hand-crafted features (including WLI) to POS tag CM text, with higher accuracy, and *without* using a combination tagger. This establishes that CM computational models can be *independently* built without using models from constituent languages.

Code-mixing is abundantly observed in our daily lives through personal chats, messages *etc.*, apart from social media. Since code-mixing is majorly used on social media, CM datasets are built primarily on social media data, which is *domain-specific*, *noisy*, and has *non-standard spellings* and *grammar*, especially for languages written in scripts other than Roman (Vyas et al., 2014). This makes *annotation* of such data a separate challenge in itself. Additionally, extracting genuine CM data from social media is a difficult task, accounting for: **(i)** inadequate code-mixed datasets (Jose et al., 2020), **(ii)** small sizes of available CM datasets and **(iii)** low Code-mixing Ratio (CMR) (discussed in Section 2) of such datasets (Jamatia et al., 2015; Singh et al., 2018). To address these existing bottlenecks of CM language research, through this work, we explore the following fundamental question:

Can we generate high-quality annotated CM data and build computational models with it, comparable to natural CM data and its subsequent computational models?

*denotes equal contribution
[1]PACMAN dataset to be released later

Proceedings of the 19th International Conference on Natural Language Processing (ICON), pages 234 - 244
December 15-18, 2022

To address the above question, we explore the dual task of annotated CM data *generation* and building *computational models* from this generated data, specifically for CM POS tagging. We introduce **PACMAN** (**PA**rallel **C**ode**M**ixed d**A**ta generatio**N**) for CM POS tagging. Through **PACMAN**, our motivation is to address the gaps in the existing CM datasets and computational models.

First, we generate annotated CM POS tagged data for *Hinglish*, through an *alignment*, *annotation* and *replacement* strategy from parallel *Hindi-English* corpus. The generated data has standard spelling and grammar, unlike social media datasets. Further, this alleviates the strenuous and difficult task of annotating social media CM data. CM datasets usually have sizes equal to or less than a meagre *1000* samples (Jose et al., 2020). We report a corpus of above ***50K*** annotated POS tagged CM samples, which are 100% code-mixed. Although our data is generated for *Hinglish*, we claim that our technique can be used to generate annotated data for any CM language pair, provided a parallel corpus and POS taggers are available for both languages. To our knowledge, our work is a *first-of-a-kind*, with respect to the generation of CM annotated data. We discuss our data generation pipeline in detail in Section 3.

Second, we build POS taggers using both Machine Learning (ML) and Deep Learning (DL) based techniques on PACMAN data. Singh et al. (2018) reports better accuracy with ML techniques than DL models. Through this work, we illustrate that this is predominantly due to the *inadequacy* of data, typically required to build better deep learning models. Our DL model outperforms our ML model by 1.5%. We would like to assert here that we build our models independently on CM data itself and *not* using a *combination tagging strategy*. We also analyse the contribution of WLI on the CM POS tagging task by infusing language information into the models. The results obtained reveal that with sizeable data, instances where language information is useful, dwindle to a small fraction, as we obtain similar accuracy without language information. We discuss our POS taggers in detail in Section 4.

Third, in order to gauge the quality of our data and models, we test them against the existing benchmark in CM POS tagging (Aguilar et al., 2020), which is based on social media text. Our models outperform the SOTA benchmark by 10%, despite having a *higher* CMR. Our results also show that our models built on PACMAN data are able to handle social media text as well as monolingual text. This is largely because of the size of our dataset, standard spelling and grammar integral to PACMAN, and uniform distribution of words and POS tags across Hindi and English. This makes our models more generalized and equipped to handle different types of CM data. We discuss qualitative analysis of our data in Section 5, error analysis in Section 6 and conclude in Section 7.

2 Code-mixing Terminology

In this section we define a few terms akin to code-mixing, that we use in later sections of the paper:

- **Matrix Language (MtxL) and Embedded Language (EL):** In a code-mixed sentence, containing two (2) languages, the *base* language, or the language from which the sentence is 'coming from' is called the *matrix language* and the other language is the *embedded language* (Joshi, 1982)
- **Switching Point (SP):** Switching Points are the junctions in a code-mixed text, where the language switches (Chatterjee et al., 2020).
- **Code-Mixing Index (CMI):** It is the measurement of the level of mixing between the constituent languages in a code-mixed context (Gambäck, 2014).
- **Code-Mixing Ratio (CMR):** CMR is the fraction of samples in a code-mixed corpus that are actually code-mixed.

3 Data Generation Pipeline

Code-mixed language research has recently received a lot of attention in the NLP community. Despite the recent interest, Jose et al. (2020) report that there is a scarcity of datasets in the domain, and existing datasets are very small (*1000* samples per dataset on average). Besides, as CM is prevalent on social media, the existing CM datasets contain only social media text, which have non-standard spelling and grammar and are difficult to annotate. In order to bridge this gap, in the present work, we endeavour to generate an *exhaustive corpus* of *annotated code-mixed data*, that is generic across domains and has standardised spelling and grammar, unlike social media data.

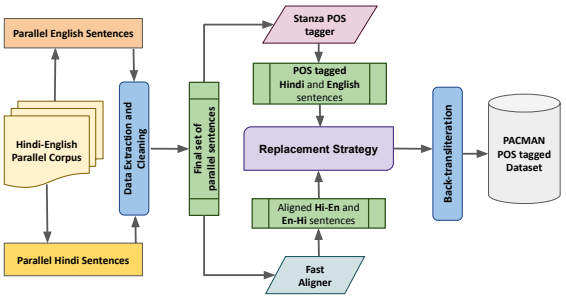

Figure 1: Data generation pipeline for PACMAN

We introduce **PACMAN** (**PA**rallel **C**ode**M**ixed d**A**ta generatio**N** a *one-of-a-kind* data generation framework for CM **POS** tagging. It is a synthetically generated data framework, curated via an *alignment*, *annotation* and *replacement* based strategy. Our approach is similar to the one applied by Srivastava and Singh (2021), but with the added objective of *uniform distribution of words* across constituent languages of the CM language pair. Currently, PACMAN is implemented for *Hinglish*, but using the same strategy will work for *any* CM language pair. The subsequent subsections present the details involved in creating the PACMAN dataset.

3.1 Resources and Tools

As already discussed, we generate the PACMAN dataset using an align, annotate, and replace strategy. In order to facilitate this, we use the following resources and tools:

- **Parallel Corpus**: We have used the *Hindi-English* parallel corpus proposed by Kunchukuttan et al. (2018) containing 1.6M English-Hindi parallel sentences. The Hindi counterpart of the corpus is available in *Devanāgari* script.

- **Fast Align**: It is a fast, simple and unsupervised aligner introduced by Dyer et al. (2013) that learns the word alignment between the parallel sentences using log-linear reparameterisation of IBM Model 2, based on a variation of the lexical translation models proposed by Brown et al. (1993).

- **Stanza**: An open-source python NLP toolkit proposed by Qi et al. (2020), that covers a wide range of text analytics tools, such as tokenization, lemmatization, POS tagging *etc.* in 66 languages.

- **Indic-trans**: It is used for transliteration framework proposed by Bhat et al. (2014) to convert the *Devanāgari* script to *Roman* and vice-versa, based on a structured perceptron model that uses letter alignments learned from GIZA++ (Och and Ney, 2003).

3.2 Methodology

For generating annotated POS tagged *Hinglish* data, we use a *Hindi-English parallel corpus* and combine parallel sentences following the *matrix language theory* introduced by Joshi (1982), using an elegant rule-based algorithm illustrated in Figure 1. A similar strategy was proposed by Srivastava and Singh (2021), but they only embed English words in Hindi sentences (Hindi as *matrix language*). We additionally embed Hindi words in English sentences (English as *matrix language*). This ensures a *uniform distribution* of words and subsequently POS tags across both the constituent languages *viz.*, Hindi and English. These word replacements account for the introduction of switching points and hence code-mixing of the generated sentences. Analysis of code-mixed sentences reveals that the words where the switch in languages happens, are primarily *nouns* and in some cases *adjectives* (Srivastava and Singh, 2021). We discuss each step of our data generation pipeline in detail below:

1. **Data Extraction:** From the parallel corpus, we filter out sentences of either language with less than *five* (5) words and instances where the Hindi parallel sentence contains English words. We pre-process and clean the resulting samples by removing unimportant tokens like URLs *etc.*

2. **Alignment:** Next, we train *fast aligner* on the pre-processed sample, which provides aligned indices of words for Hindi to English as well as English to Hindi, for each parallel sentence.

3. **Annotation:** Once the alignment is done, we POS tag both the Hindi and English sentences using *Stanza* POS tagger. We use the *universal* POS tagset (discussed in Section 4.1), which is available as part of Stanza to annotate the parallel sentences.

4. **Replacement:** This step is the most critical step in our strategy. Once the Hindi and English parallel sentences are aligned and POS tagged, we look for *one-one* word mappings in

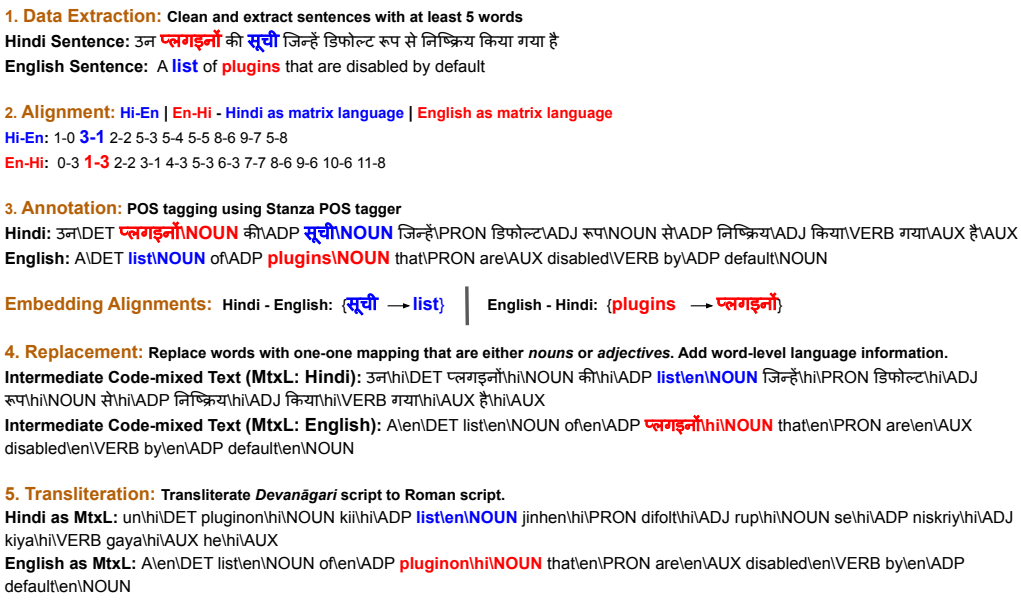

Figure 2: A sample execution of our data generation pipeline on example parallel sentences. A sample token in PACMAN is of the form: `<word/LID/POS>`. We have used the *universal* POS tagset. For the language identifier (LID), each token is tagged as ***hi*** (Hindi), ***en*** (English) or ***rest*** (others). The final generated text for PACMAN data both for *Hindi* and *English* as *matrix languages* can be observed.

the alignment that are either a *noun* or an *adjective*. If such a mapping exists, from either Hindi to English or English to Hindi alignments, we call them *embedding alignments*. Essentially, we locate junctions in the parallel sentences where code-mixing can happen. For the Hindi parallel sentence, we replace the words in Hindi with the words in English that constitute the *embedding alignments*. This is the case where Hindi is the *matrix language*, and English is the *embedded language*. The same is actuated for English as the MtxL and Hindi as the EL. Using both Hindi and English as matrix languages ensures that the vocabulary as well as the POS tag distribution is uniform across both languages. We add the word-level language information (*hi* for Hindi, *en* for English, *rest* for others) in the generated sentences, as the information is known based on the *matrix* and *embedded* language.

5. **Transliteration:** The Hindi words in the generated data, are in *Devanāgari* script, as they occur in the same form in the parallel corpus. We use *Indic-trans* to transliterate the words in *Devanāgari* to Roman script in order to generate the final PACMAN code-mixed annotated data, which is *completely* in Roman script.

The generated PACMAN dataset is Roman in form and each word (token) is annotated with a language identifier (LID) (*hi* for Hindi, *en* for English, *rest* for others) and POS label from Stanza. A sample token of PACMAN is of the form: `<word/LID/POS>`. We have demonstrated the execution of our data generation pipeline for a sample set of parallel sentences in Figure 2. An example of generated PACMAN data can also be visualized in this figure.

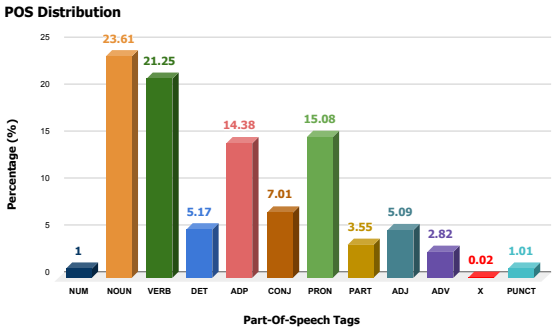

Figure 3: Percentage-wise POS distribution of PACMAN over 12 universal POS tags

3.3 Corpus Statistics

We generate the PACMAN dataset through an *alignment*, *annotation* and *replacement* based strategy discussed in Section 3.2. The generated dataset

contains **51118** unique pure CM *Hinglish* POS tagged sentences, with an average CMI of **14.61** and **100%** CMR. The average sample length in the dataset is **28** as shown in Table 2. The POS distribution over 12 universal POS tags is shown in Figure 3.

4 Part-Of-Speech Tagging

In this section, we discuss the tagset used for POS annotation, details and performance of the sequence labeling models applied on the PACMAN dataset for word-level POS prediction. In addition, we also analyse the contribution of WLI for the CM POS labeling task.

4.1 Part-Of-Speech tagset

We have already discussed earlier that during our data generation phase, we use the *Universal POS tagset* introduced by Petrov et al. (2012), who established that a coarse-grained POS tagset of **12** tags is sufficient for POS tagging and performs well for the task, compared to a fine-grained tagset as in Marcus et al. (1993). The *Stanza* POS tagger has provision for the universal POS tagset (called *upos*), but has *17* unique POS tags. In order to maintain the 12 POS tags proposed by Petrov et al. (2012), we have post-processed our data by replacing `AUX` to `VERB`, `PROPN` to `NOUN`, `SCONJ` and `CCONJ` to `CONJ`. Further, from the POS distribution statistics shared in Figure 3, it can be seen that tag `X` contributes only **0.02%** of total tag counts. Hence, we remove the samples containing `X` tag from our dataset, as it is statistically insignificant. The final set of **11** POS tags are: `ADJ`, `ADP`, `ADV`, `VERB`, `CONJ`, `DET`, `NOUN`, `NUM`, `PART`, `PRON`, `PUNCT`.

4.2 Sequence Labeling Models

To establish the efficacy of the generated PACMAN dataset, we built sequence labeling models on it. We experimented with an ML model and a DL model *viz.*, Conditional Random Field (CRF) and Bidirectional LSTM (BiLSTM). Previous research has validated the use of CRFs (Toutanova et al., 2003; Choi et al., 2005; Peng and McCallum, 2006) and LSTMs / BiLSTMs (Ghosh et al., 2016; Wang et al., 2015) for POS tagging and other sequence labeling NLP tasks. We experimented with the transformer-based models as in Aguilar et al. (2020), but do not report them, as they were outperformed by CRF and BiLSTM.

We have used the CRF model proposed by Lafferty et al. (2001), using the faster "*L-BFGS*" (Liu and Nocedal, 1989) optimization. We used the following set of hand-crafted linguistic features for the CRF classifier: **(i)** The current token W, **(ii)** index of the W, **(iii)** affixes of length 1 to 3, **(iv)** a binary feature indicating whether all characters in W are uppercase or lowercase, **(v)** a binary feature indicating whether W has any upper case character, **(vi)** a binary feature indicating whether there is any digit character in W, **(vii)** previous and next word of W, **(viii)** a binary feature indicating whether W has a hyphen (-). To prevent over-fitting, we use L_1 and L_2 regularization. We used grid search to extract the optimal hyper-parameters for the CRF model. We call this model PACMAN$_{CRF}$.

We do not use any hand-crafted features for our BiLSTM model. Instead, we train a set of *word embeddings* as part of the neural network designed for the word level POS prediction task. This ensures that word embeddings are tuned for the POS tagging task. We have kept the dimension of the word embeddings as *128*. These embeddings are passed on to the BiLSTM layer (output dimension *512*), followed by a set of *feed forward* layers (dimensions *512* and *256*), and finally a *softmax layer* for the POS prediction. To prevent over-fitting, we add a dropout (*0.25*) layer and L_1, L_2 regularizations. We experimented with different sets of hyper-parameters, layer sequences, and dimensions, but this configuration yielded the best performance. We call this model PACMAN$_{BiLSTM}$.

For both the sequence labeling tasks, we take a **75:5:20** split for *training*, *validation* and *testing* sets for our models. The *Precision*, *Recall* and *F_1-score* for PACMAN$_{CRF}$ and PACMAN$_{BiLSTM}$ are reported in the first half of Table 1. It can be observed that the PACMAN$_{CRF}$ model achieves an overall F_1-score of **0.965**, whereas the PACMAN$_{BiLSTM}$ model outperforms the PACMAN$_{CRF}$ model with an overall F_1-score of **0.979**. Singh et al. (2018) reports that ML-based techniques work better than DL-based techniques for CM POS labeling. Our results show that this is predominantly due to the *inadequacy of data*, typically required to build better deep learning models. Since, our data is almost *50* times that of Singh et al. (2018), our DL model *viz.*, PACMAN$_{BiLSTM}$ with an F_1-score of 98% performs better than our ML model PACMAN$_{CRF}$ by 1.5%.

POS	Without WLI						With WLI					
	PACMAN$_{CRF}$			PACMAN$_{BiLSTM}$			PACMAN$^{L}_{CRF}$			PACMAN$^{L}_{BiLSTM}$		
	P	R	F	P	R	F	P	R	F	P	R	F
ADJ	0.939	0.92	0.929	0.955	0.946	0.95	0.943	0.912	0.927	0.95	0.948	0.949
ADP	0.983	0.987	0.985	0.988	0.99	0.989	0.985	0.989	0.987	0.988	0.991	0.99
ADV	0.914	0.908	0.911	0.947	0.956	0.952	0.915	0.902	0.909	0.951	0.956	0.954
CONJ	0.959	0.968	0.963	0.978	0.976	0.977	0.959	0.97	0.964	0.979	0.975	0.977
DET	0.972	0.956	0.964	0.988	0.984	0.986	0.978	0.962	0.97	0.99	0.983	0.987
NOUN	0.974	0.975	0.974	0.983	0.981	0.982	0.97	0.977	0.974	0.981	0.983	0.982
NUM	0.963	0.978	0.97	0.974	0.984	0.979	0.993	0.988	0.991	0.99	0.987	0.988
PART	0.987	0.989	0.988	0.994	0.993	0.993	0.99	0.99	0.99	0.993	0.992	0.993
PRON	0.975	0.978	0.976	0.99	0.991	0.99	0.978	0.98	0.979	0.99	0.993	0.991
VERB	0.971	0.973	0.972	0.988	0.99	0.989	0.975	0.976	0.975	0.99	0.988	0.989
PUNCT	0.983	0.983	0.983	0.992	0.981	0.986	0.985	0.983	0.984	0.99	0.983	0.986
Avg	0.965	0.965	**0.965**	0.98	0.979	**0.979**	0.97	0.966	**0.968**	0.981	0.98	**0.981**

Table 1: Precision (P), Recall (R) and F_1-score (F) for CRF and BiLSTM sequence labeling models, on PACMAN data, with and without WLI. It can be observed that the PACMAN$_{BiLSTM}$ performs better than the PACMAN$_{CRF}$. Also, infusing WLI parameter *does not* boost the F_1-scores for both models.

4.3 Contribution of Word-Level Language Information

Solorio and Liu (2008); Vyas et al. (2014) emphasize that word-level language information (WLI) is a requisite for POS tagging CM text. Singh et al. (2018) have shown a slight increase in the overall F_1-score (2%) when language information is considered. In Section 4.2, we observed the accuracy of our sequence labeling models on CM POS tagging *without* language information for each word. To gauge the effect of language information on CM POS tagging, we model the POS sequence labeling task with the *annotated* WLI, which is captured during data generation. We *do not* build a *language identifier* (LID) model. The WLI tags are added to the train, validation, and test data, split in **75:5:20**, as discussed in Section 4.2. For the PACMAN$_{CRF}$ model, we add a *language feature* for word W, and for PACMAN$_{BiLSTM}$, we model the language information for each word, as part of the *word embeddings*. We name the models PACMAN$^{L}_{CRF}$ and PACMAN$^{L}_{BiLSTM}$ respectively.

The *Precision*, *Recall* and *F_1-score* for PACMAN$^{L}_{CRF}$ and PACMAN$^{L}_{BiLSTM}$ are reported in the second half of Table 1. It is evident from the results that post the infusion of WLI, there is almost no increase in performance (0.3% for CRF and 0.2% for BiLSTM) of the CM POS label prediction, even though *ground-truth* WLI labels are provided in test data and **not** *probabilistic* labels from a LID model. We also found that the predicted POS labels for PACMAN$_{BiLSTM}$ and PACMAN$^{L}_{BiLSTM}$ differ by only 1.13%. As our findings are contrary to previous research in CM POS tagging, we investigated the statistical significance of our findings. We obtained a *p-value* of $9.6e^{-9}$ ($<<$ 0.05 threshold), validating that our observations are *statistically significant*.

This establishes that for a significantly *large dataset*, that is *uniformly distributed* across constituent languages of a CM setting, the models learn the sequential data better and are able to assign classes to each word based on the context, *inherently* capturing the language information for each word. Thus, the requirement of WLI is *nullified* for the CM POS labeling task.

It can be observed in Table 1 that despite similar overall F_1-scores, there is a dissimilarity in Precision, Recall, and F_1-scores of individual POS tags. To understand this, we did further analysis on the impact of WLI on our sequential models. We found that WLI aids in the correct POS category identification of words when they have the *exact same spelling* in both English and transliterated Hindi. Words like `the` (was), `he` (is), `main` (me) which are also English words, are some examples of such cases (example shown in Appendix A through Figure 5). These cases are just a *handful* though, and hence ***do not*** affect the overall accuracy of the sequence labeling models.

5 PACMAN: Qualitative Evaluation

In this section, we gauge the quality of our dataset and models against the existing *benchmark* for CM POS tagging *i.e.*, *Linguistic Code-switching Evaluation* (LinCE), reported by Aguilar et al. (2020). Since PACMAN is synthetically generated, it is crucial that its efficacy is tested against a benchmark social media dataset (LinCE), which is considered natural CM data. To this end, we devise a set of

intricately designed experimental scenarios for this comparative investigation.

5.1 Dataset Statistics Comparison

We first compare the statistical parameters across the proposed PACMAN dataset and the benchmark dataset LinCE, exhibited in Table 2. The comparison of POS distributions between PACMAN and LinCE are also shown in Appendix A through Figure 4.

Parameters	PACMAN	LinCE
# Code-mixed samples	**51118**	**1077**
# English samples	0	343
# Hindi samples	0	69
CMR (%)	**100**	**72.33**
Average CMI	**14.12**	**14.16**
Avg. Sample Length	28.16	15.11
Total # tags	1477765	26416

Table 2: Comparison of statistics between PACMAN and LinCE code-mixed datasets. The key parameters to note here are the avg CMI, CMR and avg sample length.

In terms of the number of samples and tags, PACMAN is almost *50* times that of LinCE. Jamatia et al. (2015) stated that in order to compare, two CM datasets, it is imperative that their complexities are similar *i.e.*, their average CMIs are close to each other. The average CMIs of both PACMAN and LinCE are around **14** and thus comparable. As for the number of actual CM samples, PACMAN and LinCE have CMR values of 100% and 72%, respectively. The average sample length in LinCE is *15* and that of PACMAN is *28*.

5.2 Experimental Setup

Comparing code-mixed datasets is tough, due to differences in source and level of mixing observed across such datasets. The benchmark LinCE dataset is based on *tweets*, extracted over a handful of *topics*, making it domain-specific and noisy in nature. Whereas PACMAN is domain agnostic and has standard spelling and grammar. We pre-processed both PACMAN and LinCE, for homogeneity, and built a set of customised scenarios to compare them.

5.2.1 Data Pre-processing

For the LinCE dataset, we observed that the authors had followed a customised annotation scheme, with *3* extra tags *viz.*, `PART_NEG`, `PRON_WH` and `PROPN`. To map LinCE to the standard universal POS tagset, we converted `PART_NEG` to `PART`, `PRON_WH` to `PRON` and `PROPN` to `NOUN`. Further, the LinCE dataset does not contain the `PUNCT` tag and PACMAN does not contain `X` tag. We remove all occurrences of `PUNCT` from PACMAN and `X` from LinCE, without affecting the context/ meaning of the samples, resulting in a total of 10 POS tags across both datasets. We name these pre-processed datasets PACMAN$_{PCD}$ and LinCE$_{PCD}$.

5.2.2 Comparison Scenarios

We gauge the quality of the PACMAN dataset against the LinCE benchmark, without language information, as we have already established in Section 4.3 that WLI does not affect the accuracy of the sequence labeling models. To compare the pre-processed datasets (PACMAN$_{PCD}$ and LinCE$_{PCD}$), we devised a set of experimental scenarios. For actuating the best performance framework, we utilise the highest performing models for LinCE (**CRF** which is the SOTA, as reported by Singh et al. (2018)), and PACMAN (**BiLSTM**) as discussed in Section 4) on the following carefully devised experimental scenarios:

1. **S**$_1$: Trained and tested on LinCE data.
   Train: LinCE$_{PCD}$ | **Test:** LinCE$_{PCD}$
   Purpose: Estimate benchmark accuracy

2. **S**$_2$: Trained and tested on PACMAN data.
   Train: PACMAN$_{PCD}$ | **Test:** PACMAN$_{PCD}$
   Purpose: Measure PACMAN accuracy

3. **S**$_3$: Trained on LinCE, tested on PACMAN.
   Train: LinCE$_{PCD}$ | **Test:** PACMAN$_{PCD}$
   Purpose: Evaluate how well LinCE generalises on PACMAN data (standard spelling and grammar)

4. **S**$_4$: Trained on PACMAN, tested on LinCE.
   Train: PACMAN$_{PCD}$ | **Test:** LinCE$_{PCD}$
   Purpose: Gauge how well PACMAN generalises on LinCE data (social media text)

5.3 Results and Discussion

The results of the experiments proposed in Section 5.2.2 are shown in Table 3. For scenarios S$_1$ and S$_2$, it can be seen that PACMAN outperforms the benchmark LinCE, by 9% and 14% for CRF and BiLSTM, respectively. Essentially, PACMAN$_{BiLSTM}$ (S$_2$: BiLSTM, with an f$_1$-score of 98%) surpasses the SOTA benchmark (S$_2$: CRF, with an f$_1$-score of 88%) by 10%.

Comparing the scenarios S$_3$ and S$_4$, it can be seen that the models that are trained on PACMAN

data perform better than the ones trained on LinCE data (7% and 10% for CRF and BiLSTM respectively). Also, CRF outperforms BiLSTM with LinCE data, as LinCE dataset is small and not adequate for training/testing the BiLSTM model (as discussed in Section 4.2).

POS	CRF				BiLSTM			
	S_1	S_2	S_3	S_4	S_1	S_2	S_3	S_4
ADJ	0.67	0.93	0.34	0.57	0.65	0.95	0.27	0.50
ADP	0.94	0.98	0.85	0.79	0.93	0.99	0.86	0.83
ADV	0.82	0.91	0.49	0.62	0.80	0.98	0.48	0.76
CONJ	0.91	0.96	0.79	0.58	0.92	0.99	0.79	0.75
DET	0.88	0.96	0.64	0.80	0.87	0.99	0.68	0.78
NOUN	0.88	0.97	0.72	0.79	0.81	0.98	0.41	0.72
NUM	0.96	0.97	0.76	0.86	0.92	0.94	0.70	0.59
PART	0.84	0.99	0.54	0.75	0.86	0.99	0.64	0.74
PRON	0.86	0.98	0.67	0.72	0.83	0.98	0.58	0.66
VERB	0.90	0.98	0.72	0.83	0.82	0.98	0.55	0.62
Average	0.88	**0.97**	0.70	**0.77**	0.84	**0.98**	0.60	**0.70**

Table 3: F_1-scores obtained for CRF and BiLSTM on scenarios S_{1-4}. S_1 and S_2 show that PACMAN outperforms the benchmark by 9%. S_3 and S_4 show that PACMAN generalises better on LinCE data, but LinCE is not able to generalise equally well on PACMAN data.

This essentially emphasizes that: **(i)** Sequence labeling models trained on PACMAN data are able to *generalise better* than LinCE, even on social media data, as PACMAN data follows standard spelling and grammar. **(ii)** Models trained on LinCE (social media data) do not generalise as well on standardised data. **(iii)** LinCE has 28% monolingual data, but PACMAN is able to handle monolingual data as well, due to the *exhaustive* and *uniform* word and POS tag distribution across *Hindi* and *English*. We can conclusively establish that not only is PACMAN *larger* and *standardised*, but also computationally *superior*, and is capable of *generalising* on natural CM data as well as monolingual data.

6 PACMAN: Error Analysis

With the motivation of bridging gaps in existing CM datasets and computational models we generate PACMAN, using an *alignment*, *annotation* and *replacement* strategy from parallel *Hindi-English* corpus. Although the generated data is *clean*, *formalised*, and yields *impressive* results, our analysis shows that the usage of *probabilistic* tools and resources adversely affects the quality of the generated dataset, although such cases are *scarce*. We highlight these inaccuracies with examples, for every *stage* of our data generation pipeline.

Inaccuracies in the Parallel Corpus: We observed a few errors in the Hindi parallel sentences in the corpus like:

1. Multiple *Devanāgari* forms of same Hindi words
   e.g., है/ हे - *hai* (is); नहि/ नहिं - *nahi* (no)

2. Hindi translations of English words being merely their *Devanāgari* forms
   e.g., डिफोल्ट (default) , डेटाबेस (database)

This results in *incorrect non-standard* words in the dataset, from the transliterations (*he* (hai), *nahin* (nahi), *difolt* (default), *databes* (database)).

Inaccurate Alignment: Due to inherent constraints in Fast Align, sometimes the alignment results rendered are inaccurate. This leads to incorrect word replacements and *redundant* words, resulting in erroneous data. In the example below it can be observed that the word *dhvani* (sound) is redundant and incorrect, due to the inaccurate alignment of the words ध्वनि(dhvani) and *event*.

English Sentence: Whether or not to play event sounds

Hindi Sentence: घटनाओं हेतु ध्वनि बजाएँ या नहीं

Alignment: En-Hi: 0-0 1-0 2-6 3-6 4-1 5-2

Final Generated text:

Whether\en\SCONJ or\en\CCONJ not\en\PART to\en\PART play\en\VERB dhvani\hi\NOUN sounds\en\NOUN

Inaccurate POS tagging: Due to a more flexible grammatical structure in Hindi, Stanza is occasionally unable to correctly adjudicate the relative positions of *adjectives* and *adverbs w.r.t nouns* and *verbs*. For *e.g.*, ADJ: बुरी / *buri* (bad) can come in one of the following orders:

1. *Before* a NOUN: बुरी चीज / *buri cheej* (bad thing)

2. *After* a NOUN: आदत बुरी / *aadat buri* (bad habit)

Stanza annotates the latter as: *aadat*\NOUN *buri*\NOUN, due to the confusion infused by Hindi grammar, resulting in an error in the dataset.

Stanza is capable of handling multi-word expressions (MWE), which leads to poor tagging resolutions in some cases. For *e.g.*, for the word sequence *ambient light* is detected as a MWE, resulting in *ambient* being annotated as NOUN instead of ADJ. This results in the incorrect entry '*ambient*\en\ADJ *prakaash*\hi\NOUN' in the dataset.

Inaccurate Transliteration: In some cases, Indic-trans renders erroneous transliteration. For *e.g.*, 'ऐ' (hey) in 'ऐ राम इधर आओ' (Hey Ram come here), is transliterated to 'I', which leads to Stanza *incorrectly* tagging 'ऐ' as `PRON`, instead of `PART`, resulting in inefficient data.

7 Conclusion and Future-work

In this paper, we address some of the existing limitations in the datasets and computational models for code-mixed languages, specifically for the CM POS tagging task.

We propose a *first-of-its-kind* work in generating CM annotated data. We introduce the PACMAN dataset, *generated* using an *alignment*, *annotation* and *replacement* strategy from *Hindi-English* parallel corpus. We claim that PACMAN is the largest CM annotated dataset (around *50K* samples). Although the generated dataset is for *Hinglish*, the strategy can be transferred to any CM language pair, having available *parallel corpus* and *POS taggers*.

The PACMAN data adheres to standard spelling and grammar, unlike social media data, primarily used in CM research work. The use of both *Hindi* and *English* as *matrix languages*, ensures uniform distribution of words in both languages, and the potential to understand monolingual contexts.

To establish the effectiveness of the dataset, we build both ML (*CRF*) and DL (*BiLSTM*) based sequential labeling models on PACMAN data. Unlike previous work, our DL model outperforms the ML model by 1.5%.

We analyse the effect of *word-level language information* on the CM POS tagging task, which reveals that, with a larger dataset, cases where WLI is crucial, are minuscule, thus elevating the need for WLI in CM POS labeling.

We validate our dataset against the existing benchmark dataset for CM POS tagging. Our best model outperforms the SOTA benchmark by 10% and in all computational scenarios, signifying that our dataset is more generalised and capable of handling a wide spectrum of CM data as well as monolingual data.

Scrutiny of errors observed in our dataset manifests that inaccuracies in probabilistic tools and resources used as a part of the data generation pipeline, adversely affect the quality of the dataset, although such cases are scarce.

In future, we endeavour to generate more data in order to build *transformer-based* models on PACMAN, and use *matrix language* information to prevent errors actuated by structural differences between *Hindi* and *English*.

References

Gustavo Aguilar, Sudipta Kar, and Thamar Solorio. 2020. LinCE: A centralized benchmark for linguistic code-switching evaluation. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1803–1813, Marseille, France. European Language Resources Association.

Peter Auer. 2020. *The pragmatics of code-switching: a sequential approach*, pages 123–138.

Irshad Ahmad Bhat, Vandan Mujadia, Aniruddha Tammewar, Riyaz Ahmad Bhat, and Manish Shrivastava. 2014. Iiit-h system submission for fire2014 shared task on transliterated search. In *Proceedings of the Forum for Information Retrieval Evaluation*, FIRE '14, page 48–53, New York, NY, USA. Association for Computing Machinery.

Stephen Boggs. 1983. Discourse strategies . john j. gumperz. *American Anthropologist*, 85.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Arindam Chatterjee, Vineeth Guptha, Parul Chopra, and Amitava Das. 2020. Minority positive sampling for switching points - an anecdote for the code-mixing language modeling. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6228–6236, Marseille, France. European Language Resources Association.

Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Identifying sources of opinions with conditional random fields and extraction patterns. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 355–362, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *NAACL*.

Björn Gambäck. 2014. On measuring the complexity of code-mixing.

Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry P. Heck. 2016. Contextual LSTM (CLSTM) models for large scale NLP tasks. *CoRR*, abs/1602.06291.

J.J. Gumperz. 1964. Hindi-punjabi code-switching in delhi. *Proceedings of the Ninth International Congress of Linguists*, pages 1115–1124.

Anupam Jamatia, Björn Gambäck, and Amitava Das. 2015. Part-of-speech tagging for code-mixed english-hindi twitter and facebook chat messages. In *RANLP*.

Navya Jose, Bharathi Raja Chakravarthi, Shardul Suryawanshi, Elizabeth Sherly, and John P. McCrae. 2020. A survey of current datasets for code-switching research. *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 136–141.

Aravind K. Joshi. 1982. Processing of sentences with intra-sentential code-switching. In *Proceedings of the 9th Conference on Computational Linguistics - Volume 1*, COLING '82, page 145–150, CZE. Academia Praha.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2018. The IIT Bombay English-Hindi parallel corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *MATHEMATICAL PROGRAMMING*, 45:503–528.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Carol Myers-Scotton. 1993. Duelling languages: grammatical structure in code-switching.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Fuchun Peng and Andrew McCallum. 2006. Information extraction from research papers using conditional random fields. *Information Processing Management*, 42(4):963–979.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2089–2096, Istanbul, Turkey. European Language Resources Association (ELRA).

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. *CoRR*, abs/2003.07082.

Vincenz Schwab. 2021. *Codemixing und Codeswitching: Volkssprachige Inserte in Rechtsquellen des Frühmittelalters*, pages 75–90.

Kushagra Singh, Indira Sen, and Ponnurangam Kumaraguru. 2018. A twitter corpus for hindi-english code mixed pos tagging. pages 12–17.

Thamar Solorio and Yang Liu. 2008. Part-of-speech tagging for english-spanish code-switched text. pages 1051–1060.

Vivek Srivastava and Mayank Singh. 2021. Hinge: A dataset for generation and evaluation of code-mixed hinglish text. *CoRR*, abs/2107.03760.

Mark Thompson. 2009. Brenda danet susan c. herring (eds.), the multilingual internet: Language, culture and communication online. *Language in Society - LANG SOC*, 38.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, page 173–180, USA. Association for Computational Linguistics.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. Pos tagging of english-hindi code-mixed social media content. pages 974–979.

Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. 2015. A unified tagging solution: Bidirectional LSTM recurrent neural network with word embedding. *CoRR*, abs/1511.00215.

A Appendix

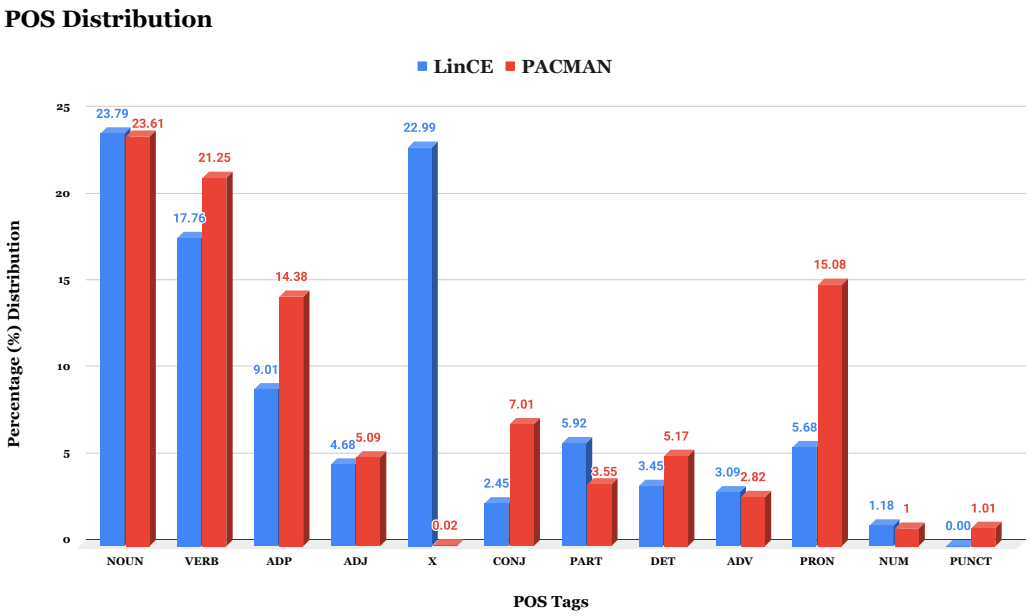

Figure 4: Percentage-wise comparison of POS distribution between PACMAN and LinCE over 12 universal POS tags. The high percentage of the 'X' POS tag indicates that the LinCE data is social media based and hence noisy.

Sample sentence: `use truth bolke burden kam kar diya`

Translation: `I lessened his burden by telling him the truth`

PACMAN$_{\text{BiLSTM}}$: `use\VERB truth\NOUN bolke\VERB burden\NOUN kam\ADJ kar\VERB diya\VERB`

Sentence with WLI: `use`$_{hi}$ `truth`$_{en}$ `bolke`$_{hi}$ `burden`$_{en}$ `kam`$_{hi}$ `kar`$_{hi}$ `diya`$_{hi}$

PACMAN$^{L}_{\text{BiLSTM}}$: `use`$_{hi}$`\PRON truth`$_{en}$`\NOUN bolke`$_{hi}$`\VERB burden`$_{en}$`\NOUN kam`$_{hi}$`\ADJ kar`$_{hi}$`\VERB diya`$_{hi}$`\VERB`

Explanation: '***use***' (his) is used in Hindi (hi) as a pronoun (PRON).
This is confused with 'use' is English (en), which is a verb (VERB).

Figure 5: Example showing where word-level language information (WLI) helps in correct POS identification. It can be seen that the word '*use*' (Hindi for *his*) has different meanings but the same spelling in English and transliterated Hindi. With WLI, this confusion is resolved, and the correct POS tag is predicted.