Improved Sentiment Detection via Label Transfer from Monolingual to Synthetic Code-Switched Text

Bidisha Samanta[1], Niloy Ganguly[1], and Soumen Chakrabarti[2]

[1]Indian Institute of Technology, Kharagpur, bidisha@iitkgp.ac.in, niloy@cse.iitkgp.ernet.in
[2]Indian Institute of Technology, Bombay, soumen@cse.iitb.ac.in

Abstract

Multilingual writers and speakers often alternate between two languages in a single discourse, a practice called "code-switching". Existing sentiment detection methods are usually trained on sentiment-labeled monolingual text. Manually labeled code-switched text, especially involving minority languages, is extremely rare. Consequently, the best monolingual methods perform relatively poorly on code-switched text. We present an effective technique for synthesizing labeled code-switched text from labeled monolingual text, which is more readily available. The idea is to replace carefully selected subtrees of constituency parses of sentences in the resource-rich language with suitable token spans selected from automatic translations to the resource-poor language. By augmenting scarce human-labeled code-switched text with plentiful synthetic code-switched text, we achieve significant improvements in sentiment labeling accuracy (1.5%, 5.11%, 7.20%) for three different language pairs (English-Hindi, English-Spanish and English-Bengali). We also get significant gains for hate speech detection: 4% improvement using only synthetic text and 6% if augmented with real text.

1 Introduction

Sentiment analysis on social media is critical for commerce and governance. Multilingual social media users often use code-switching, particularly to express emotion [28]. However, a basic requirement to train any sentiment analysis (SA) system is the availability of large sentiment-labeled corpora. These are extremely challenging to obtain [8, 39, 2], requiring volunteers fluent in multiple languages.

We present **CSGen**, a system which provides supervised SA algorithms with synthesized unlimited sentiment-tagged code-switched text, without involving human labelers of code-switched text, or any linguistic theory or grammar for code-switching. These texts can then train state-of-the-art SA algorithms which, until now, primarily worked with monolingual text.

A common scenario in code-switching is that a resource-rich *source* language is mixed with a resource-poor *target* language. Given a sentiment-labeled source corpus, we first create a parallel corpus by translating to the target language, using a standard translator. Although existing neural machine translators (NMTs) can translate a complete source sentence to a target sentence with good quality, it is difficult to translate only designated source segments in isolation because of missing context and lack of coherent semantics.

Among our key contributions is a suite of approaches to automatic segment conversion. Broadly, given a source segment selected for code-switching, we propose intuitive ways to select a corresponding segment from the target sentence, based on maximum similarity or minimum dissimilarity with the source segment, so that the segment blends naturally in the outer source context. Finally, the generated synthetic sentence is tagged with the same sentiment label as the source sentence. The source segment to replace is carefully chosen based on an observation that, apart from natural switching points dictated by syntax, there is a propensity to code-switch between highly opinionated segments.

Extensive experiments show that augmenting scarce natural labeled code-switched text with plentiful synthetic text associated with 'borrowed' source labels enriches the feature space, enhances its coverage, and improves sentiment detection accuracy, compared to using only natural text. On four natural corpora having gold sentiment tags, we

demonstrate that adding synthetic text can improve accuracy by 5.11% in English-Spanish, 7.20% in English-Bengali and (1.5%, 0.97%) in English-Hindi (Twitter, Facebook). The synthetic code-switch text, even when used by itself to train SA, performs almost as well as natural text in several cases. Hate speech is an extreme emotion expressed often on social media. On an English-Hindi gold-tagged hate speech benchmark, we achieve 6% absolute F1 improvement with data augmentation, partly because synthetic text mitigates label imbalance present in scarce real text.

2 Related Work

Recent SA systems are trained on labeled text [33, 38, 13]. For European and Indian code-switched sentiment analysis, several shared tasks have been initiated [2, 27, 24, 32, 34]. Some of these involve human annotations on code-switched text. [38] have annotated the data set released for POS tagging by [35]. [13] had Hindi-English code-switched Facebook text manually annotated and developed a deep model for supervised prediction.

In a different direction, synthetic monolingual text has been created by Generative Adversarial Networks (GAN) [14, 41, 42, 18], or Variational Auto Encoders (VAE) [6]. Some of these models can be used to generate sentiment-tagged synthetic text. However, most of them are not directly suitable for generating bilingual code-mixed text, due to the unavailability of sufficient volume of gold-tagged code-mixed text. [29] proposed a generative method using a handful of gold-tagged data; but they cannot produce sentence level tags. Recently, [25] used linguistic constraints arising from Equivalence Constraint Theory to design a code-switching grammar that guides text synthesis. Earlier, [4] presented similar ideas, but without empirical results. In contrast, CSGen uses a data-driven combination of word alignment weights, similarity of word embeddings between source and target, and attention [1].

3 Generation of code-switched text

CSGen takes a sentiment-labeled source sentence s and translates it into a target language sentence t. Then it generates text with language switches on particular constituent boundaries. This involves two sub-steps: select a segment in s (§3.1), and then select text from t that can replace it (§3.2–§3.3). This generation process is sketched in Algorithm 1.

3.1 Sentiment-oriented source segment selection

In this step, our goal is to select a contiguous segment from the source sentence that could potentially be replaced by some segment in the target sentence. (Allowing non-contiguous target segments usually led to unnatural sentences.) Code switching tends to occur at constituent boundaries [30], an observation that holds even for social media texts [3]. Therefore, we apply a constituency parser to the source sentence. Specifically, we use the Stanford CoreNLP shift-reduce parser [43] to generate a parse tree[1]. Then we select segments under non-terminals, i.e., subtrees, having certain properties, chosen using heuristics informed by patterns observed in real code-switched text.

NP and VP: We allow as candidates all subtrees rooted at NP (noun phrase) and VP (verb phrase) nonterminals, which may cover multiple words. Translating single-word spans is more likely to result in ungrammatical output [30].

SBAR: Bilingual writers often use a clause to provide a sentiment-neutral part and then switch to another language in another sentence-piece to express an opinion or vice-versa. An example is "Ramdhanu ended with tears *kintu sesh ta besh onho rokom etar*" (Ramdhanu ended with tears but the ending was quite different). Here the constituent "*but the ending was quite different*" comes under the subtree of SBAR.

Highly opinionated segments: We also include segments which have a strong opinion polarity, as detected by a (monolingual) sentiment analyzer [12]. E.g., the tweet "*asimit khusi prasangsakako ke beech* ... as India won the world cup after 28 years" translates to "Unlimited happiness among fans ... as India won the world cup after 28 years".

[1] `http://stanfordnlp.github.io/CoreNLP/`

Algorithm 1 CSGen overview.

Input: Sentiment-labeled source sentences $S = \{(s_n, y_n)\}$
Output: Synthetic code-switched sentences $C = \{(c, y)\}$
$t_n \leftarrow$ Translate(s_n) $\forall s_n \in S$ `/* Make parallel corpus */`
$C \leftarrow \varnothing$
for each parallel sentence pair s, t **do**
  `/*Collect word alignment signals*/`
  $a \leftarrow$ AttentionScore(s, t), $g \leftarrow$ GizaScore(s, t)
  `/* Source segment selection */`
  $P \leftarrow$ SentimentOrientedSegmentSelection(s)
  for each segment $p_s \in P$ to replace **do**
    `/* Target segment selection */`
    $\hat{q}^1, \hat{q}^2 \leftarrow$ MaxSimTargetSeg(s, p_s, t, a, g)
    $\hat{q}^3, \hat{q}^4 \leftarrow$ MinDissimTargetSeg$(s, p_s, t, 1-a, 1-g)$
    `/*Code-switched text generation*/`
    $C_k \leftarrow$ Project$(s, t, p_s, \hat{q}^k)$ where $k \in \{1, \ldots, 4\}$
    $C \leftarrow C \cup$ SelectBest$(\{C_k : k \in \{1, \ldots, 4\}\})$
  end for
end for
$C \leftarrow$ Threshold(C) `/* Retain only best replacements */`

An example sentence, its parse tree, and its candidate replacement segments are shown in Figure 1. In Algorithm 1, $p_s \in P$ denotes the set of candidate replacement subtrees, which correspond to segments. For each candidate segment, we generate a code-switched version of the source sentence, as described next.

3.2 Target segment selection

Given a source sentence s, corresponding target t, and one (contiguous) source segment $p_s = \{w_s^i \cdots w_s^{i+x}\}$, the goal is to identify the best possible a contiguous target segment $q_t = \{w_t^j \cdots w_t^{j+y}\}$ that could be used to replace p_s to create a realistic code-switched sentence. We adopt two approaches to achieve this goal: (a) selecting a target segment that has maximum similarity with p_s, and (b) selecting a target segment having minimum dissimilarity with p_s, for various definitions of similarity and dissimilarity. Below, we describe methods that achieve this goal after describing several alignment scores which will be used in these methods. Overall, these lead to target segments $\hat{q}_t^1, \hat{q}_t^2, \ldots$ shown in Algorithm 1, with t removed for clarity.

3.2.1 Word alignment signals

Signals based on word alignment methods are part of the recipe in choosing the best possible q_t given the sentence pair and p_s.

GIZA score: The standard machine translation word alignment tool Giza++ [22] uses IBM statistical word alignment models 1–5 [10, 31, 7, 26]. This tool incorporates principled probabilistic formulations of the IBM models and gives a correspondence score $G[w_t^i, w_s^j]$ between target and source words for a given sentence pair. This word-pair score is used as a signal to find the best $\hat{q}_t$.

NMT attention score: Given an attention-guided trained sequence-to-sequence neural machine translation (NMT) model [1, 17] and sentence pair s, t, we use the attention score matrix $A[w_t^i, w_s^j]$ as an alignment signal.

Inverse document frequency (rarity): The inverse document frequency (IDF) of a word in a corpus signifies its importance in the sentence [36]. We define $\mathcal{I}(w) = \sigma(a \text{ IDF}(w) - b)$ as a shifted, squashed IDF that normalizes the

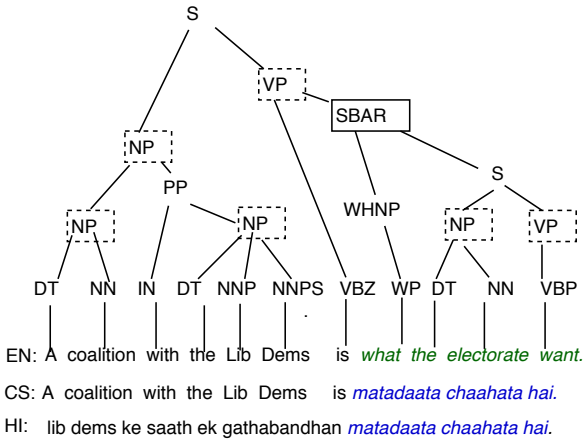

Figure 1: A phrase-structure tree for a sample synthesis. Dotted-boxes around constituents indicate that they are candidates for replacement on the source side (§3.1). EN: English source sentence, HI: Hindi target sentence, CS: code-switched sentence. The *italicized segment* is the target segment to replace the *source segment* under the non-terminal SBAR.

raw corpus-level score. Here σ is the sigmoid function and parameters a and b are empirically tuned. This IDF-based signal is optionally incorporated while choosing $\hat{q}_t$.

3.2.2 Target segment with maximum similarity

Given word-pair scores derived from either Giza++ or NMT attention described in §3.2.1, we formulate two methods for identifying target segments. First, we identify the best target segment given Giza++ scores, $G[\cdot,\cdot]$, as follows:

$$\hat{q}_t^1 \leftarrow \underset{q_t}{\text{argmax}} \prod_{w_t \in q_t} \sum_{w_s \in p_s} G[w_t, w_s] \tag{1}$$

For each word in q_t, we compute the total attention score concentrated in p_s and then multiply them as if they are independent.

Second, we use the attention score learned by the NMT system of [16] (a bidirectional LSTM model with attention). Essentially, given the attention score $A[\cdot,\cdot]$ between target and source words, we intend to select the target segment q_t whose maximum attention is concentrated in the given p_s.

Initial exploration of the above method revealed that the attention of a target word may spread out over several related but less appropriate source words, and accrue better overall similarity than a single more appropriate word. Here IDF can come to the rescue, the intuition being that words w_t^i and w_s^j with very different IDFs are less likely to align, because (barring polysemy and synonymy) rare (common) words in one language tend to translate to rare (common) words in another. This intuition is embodied in the improved formulation:

$$\hat{q}_t^2 \leftarrow \underset{q_t}{\text{argmax}} \prod_{w_t \in q_t} \mathcal{I}(w_t) \sum_{w_s \in p_s} \mathcal{I}(w_s) A[w_t, w_s] \tag{2}$$

Informally, if a source segment contains many rare words, the target segment should also have a similar number of rare words from the target domain, and vice-versa.

3.2.3 Target segment with minimum dissimilarity

We examine an alternative method for identifying target segments that leverage the *Earth Mover's Distance (EMD)* [37]. [15] extended EMD to the *Word Mover Distance* to measure the dissimilarity between documents by 'transporting' word vectors from one document to the vectors of the other. In the same spirit, we define a dissimilarity measure between p_s and candidate target segments using EMD. We present here EMD as a minimization over fractional trans-

portation matrix $\boldsymbol{F} \in \mathbb{R}^{|q_t| \times |p_s|}$ as below:

$$\text{EMD}(q_t, p_s) = \min_{\boldsymbol{F}} \sum_{i=1}^{|q_t|} \sum_{j=1}^{|p_s|} F_{i,j} d_{i,j} \tag{3}$$

where $\sum_i F_{i,j} = \frac{1}{|q_t|}$ and $\sum_j F_{i,j} = \frac{1}{|p_s|}$ and $d_{i,j}$ is a distance metric between a target and a source word pair, given suitable representations. Finally, we choose the target segment which is least dissimilar to a given source segment defined by the EMD. We compute $d_{i,j}$ in two ways, described below.

Attention-based distance: Here the distance between the embeddings is defined as:

$$d^A_{i,j} = 1 - A[w^i_t, w^j_s] \tag{4}$$

Giza-based distance: Similarly we can compute the distance using Giza score as:

$$d^G_{i,j} = 1 - G[w^i_t, w^j_s] \tag{5}$$

Given the two types of distances in Eq. (4)–(5) and the definition of EMD in Eq. (3) we can formulate two methods for identifying target segments:

$$\hat{q}^k_t \leftarrow \operatorname*{argmin}_{q_t} \min_{\boldsymbol{F}} \sum_{i=1}^{|q_t|} \sum_{j=1}^{|p_s|} F_{i,j} d^k_{i,j} \tag{6}$$

where $k \in \{3, 4\}$ and $d^3_{i,j} \equiv d^A_{i,j}$ and $d^4_{i,j} \equiv d^G_{i,j}$.

We can also use Euclidean distance as $d_{i,j}$. However, this method requires multilingual word embeddings for every word to calculate the distance. The volume of labeled source text we can use is usually smaller than the vocabulary size, making it difficult to learn reliable word embeddings. Also, if these corpora contain informal social media text like the ones described in §4.1, then publicly available pretrained word embeddings exclude a significant percentage of them.

3.3 Projecting target segments

Given a source sentence s with designated segment p_s to replace, and target sentence t, we have by now identified four possible target segments $\hat{q}^k_t$ where $k \in \{1, \ldots, 4\}$ as described in §3.2.2–§3.2.3. We now *project* the target segment to the source sentence, meaning, (a) replace the source segment with the target segment and (b) transliterate the replacement using the Google Transliteration API to the source script[2]. This creates four possible synthetic code-switched sentences for each instance of (s, p_s, t). Finally, we transfer the labels of the original monolingual corpora to the generated synthetic text corpora.

3.4 Best candidate via reverse translation

From these four code-switched sentences $c_1, \ldots, c_4$, we wish to retain the one that retains most of the syntactic structures of the source sentence. Each code-switched sentence c_k has an associated score as defined in §3.2. We use two empirically tuned thresholds: a lower cut-off for the similarity score of c_1, c_2 and an upper cut-off for the dissimilarity score of c_3, c_4, to improve the quality of candidates retained. These scores are not normalized and cannot be compared across different methods. Therefore, we perform a reverse translation of each candidate back to the source language using the Google translation API to obtain $\tilde{s}$. We retain the candidate whose retranslated version $\tilde{s}$ has the highest BLEU score [23] wrt s. In case of a tie, we select the candidate with maximum word overlap with s.

3.5 Thresholding and stratified sampling

In addition to retaining only the best among code-switched candidates $c_{1,\ldots,4}$, we discard the winner if its BLEU score is below a tuned threshold. Further, we sample source sentences such that the surviving populations of sentiment labels of the code-switched sentences match the populations in the low-resource evaluation corpus. Another tuned system parameter is the amount of synthetic text to generate to supplement the gold text.

[2] `http://www.google.com/transliterate?langpair=hi|en&text=<text>`

We do not depend on any domain coherence between the source corpus used to synthesize text and the gold 'payload' corpus — this is the more realistic situation. Our expectation, therefore, is that adding some amount of synthetic text should improve sentiment prediction, but excessive amounts of off-domain synthetic text may hurt it. In our experiments we grid search the synthetic:gold ratio between 1/4 and 2 using 3-fold cross validation.

4 Experiments

We demonstrate the effectiveness of augmenting gold code-switched text with synthetic code-switched text. We also measure the usefulness of synthetic text without gold text. In this section, we will first describe the data sets used to generate the synthetic text and then the resource-poor labeled code-switched text used for evaluation. Next, we will present the method used for sentiment detection, baseline performance, and finally our performance, along with a detailed comparative analysis.

4.1 Source corpora for text synthesis

We use publicly available monolingual sentiment-tagged (positive, negative or neutral) gold corpora in the source language.

ACL: [9] released about 6000 manually labeled English tweets.

Election: [40] published about 5000 human-labeled English tweets.

Mukherjee: This data set contains about 8000 human-labeled English tweets [19, 20].

Semeval shared task: This provides about 10000 human-labeled English tweets [27].

Union: This is the union of above mentioned different data sets.

Hatespeech: We collected 15K tagged English tweets from [11] which consists of 4.7K *abusive*, 1.7K *hateful* and 4K *normal* tweets.

We picked Spanish, which is homologous to English, and Hindi and Bengali, which are comparatively dissimilar to English, for our experiments. We translated these monolingual tweets to Spanish, Hindi and Bengali using Google Translation API[3] and used as parallel corpus to train attention-based NMT models and statistical MT model (GIZA) to learn the word alignment signals as described in §3.2.1.

4.2 Preliminary qualitative analysis

Analysis of texts synthesized by various mechanisms proposed in §3.2 shows that similarity based methods contribute 82–85% of the best candidates and the rest come from dissimilarity based methods. Similarity-based methods using NMT attention and Giza perform well because the segments selected for replacement often constitute nouns and entity mentions, which have a very strong alignment in the corresponding target segment. NMT attention and Giza-EMD perform well when segments contract or expand in translation.

4.3 Low-resource evaluation corpora

To evaluate the usefulness of the generated synthetic tagged sentences as a training set for sentiment analysis, we have used three different code-switched language pair data sets. Each data set below was divided into 70% training, 10% validation and 20% testing folds. The training fold was (or was not) augmented with synthetic labeled text to train sentiment classifiers, which were then applied on the test fold judge the quality of synthesis.

HI-EN, FB (Hindi-English, Facebook): [13] released around 4000 labeled code-switched sentences from the Facebook timeline of Narendra Modi (Indian Prime Minister) and Salman Khan (Bollywood actor).

[3] `https://translation.googleapis.com`

HI-EN, TW (Hindi-English, Twitter): This is a shared task from ICON 2017 [24] with 15575 instances.

ES-EN (English-Spanish): We collected 2883 labeled tweets specified by [38].

BN-EN (Bengali-English): This is another shared task from ICON 2017 [24] with 2499 instances.

HI-EN, Hatespeech: [5] published 4000 manually-labeled code-switched Hindi-English tweets: 1500 exhibiting *hate speech* and 2500 *normal*. We also found a significant number of abusive tweets marked *hate speech*. For uniformity, we merged hate speech tweets and abusive tweets.

4.4 Sentiment classifier

We adopt the sub-word-LSTM system of [13]. We prefer this over feature-based methods because (a) feature extraction for code-switched text is very difficult, and varies widely across language pairs, and (b) the vocabulary is large and informal, with many tokens outside standard (full-) word embedding vocabularies and (c) sub-word-LSTM captures semantic features via convolution and pooling.

Loss functions: If the sentiment labels $\{-1, 0, +1\}$ are regarded as categorical, cross-entropy loss is standard. However, prediction errors between the extreme polarities $\{-1, +1\}$ need to be penalized more than errors between {-1,0} or {0,+1}. Hence, we use ordinal cross-entropy loss [21], introducing a weight factor proportional to the order of intended penalty multiplied with the cross entropy loss. On the test fold, we report 0/1 accuracy and per class micro-averaged F1 score.

Baseline and prior art: Our baseline scenario is a self-contained train-dev-test split of the gold corpus. The primary prior art is the work of [25].

Feature space coherence: Our source corpora are quite unrelated to the gold corpora. Table 1 shows that the average Euclidean distance between feature space of gold training and testing texts is much lower than that between gold and synthetic texts. While this may be inescapable in a low-resource situation, the gold baseline does not pay for such decoherence, which can lead to misleading conclusions.

	HI_EN,TW	HI_EN,FB	ES_EN	BN_EN
ACL	2.21 (2.13)	3.72 (2.24)	2.09 (1.73)	4.11 (2.67)
Election	2.40 (2.12)	6.27 (2.67)	1.58 (1.49)	5.23 (2.43)
Mukherjee	2.47 (2.33)	3.82 (2.26)	1.64 (1.64)	5.18 (2.50)
Semeval	2.23 (2.11)	4.04 (2.26)	1.69 (1.67)	3.63 (2.59)
Union	2.55 (2.15)	3.80(2.65)	1.65 (1.53)	5.48 (2.56)
Gold	2.05	1.87	1.64	1.83

Table 1: Average pairwise Euclidean distance between training data and test data features. Rows correspond to standalone (respectively, augmented) text for training. Gray: reference distance of gold test from gold train. Red: largest distance observed.

Training regimes: Absence of coherent tagged gold text may lead to substantial performance loss. Hence, along with demonstrating the usefulness of augmenting natural with synthetic text, we also measure the efficacy of synthetic text on its own. We train the SA classifier with three labeled corpora: (a) limited gold code-switched text, (b) gold code-switched text augmented with synthetic text and (c) only synthetic text. Then we evaluate the resulting models on labeled gold code-switched test fold.

4.5 Sentiment detection accuracy

Table 2 shows the benefits of augmenting natural with synthetic text. Test accuracy increases further (shown in brackets) if thresholding and stratified sampling are used. Gains for HI_EN,TW, HI_EN,FB, ES_EN and BN_EN are 1.5% (2.43%), 0.23% (1.43%), 4.76% (6.24%), and 2.8% (4.8%) respectively. Categorical cross-entropy loss was

Train \ Test	HI_EN (TW)	HI_EN (FB)	ES_EN	BN_EN	HI_EN (TW)	HI_EN (FB)	ES_EN	BN_EN
	Categorical cross entropy training loss				Ordinal cross entropy training loss			
ACL	51.80 (52.59)	62.59 (65.33)	44.80 (48.84)	52.59 (59.81)	52.68 (53.76)	62.72 (64.22)	45.69 (50.31)	50.08 (51.60)
Election	52.84 (54.59)	65.59 (66.80)	43.07 (43.07)	57.99 (59.00)	52.89 (54.64)	64.88 (65.26)	45.21 (45.80)	53.40 (55.60)
Mukherjee	53.76 (54.69)	64.82 (65.85)	47.06 (46.36)	49.79 (57.40)	53.79 (53.53)	59.66 (64.57)	44.85 (43.07)	51.00 (49.40)
Semeval	52.99 (54.19)	65.33 (65.46)	47.36 (44.69)	53.40 (59.99)	52.99 (53.83)	63.26 (64.66)	47.36 (44.64)	53.14 (57.59)
Union	53.28 (54.64)	65.50 (65.99)	44.24 (45.32)	57.23 (59.89)	53.65 (53.69)	64.30 (67.65)	44.04 (46.00)	53.40 (57.40)
MSR	54.50	65.58	48.14	59.79	53.69	62.80	47.50	52.8
Gold	52.26	65.37	42.6	55.19	52.34	64.29	45.20	50.39

Table 2: Accuracy (%) on 20% test data after training with augmented and only gold text. Rows correspond to sources of augmentation. In most cells we show (A) no thresholding or stratification and (B) with thresholding and stratification (within brackets). Gray: reference accuracy with only gold training. Blue: A or B or MSR outperforms gold. Green: B performs best. Row 'MSR' uses text synthesized by [25].

used here. Similar improvements in accuracy of 1.45% (1.5%), 0.59% (0.97%), 2.16% (5.11%), 3% (7.20%) are observed after training with ordinal loss function. Our conclusion is that *careful augmentation with synthetic data can lead to useful gain in accuracy*. Moreover, by selecting synthetic text which is syntactically more natural, even larger gains can be achieved. Notably, the distance between training and test features (Table 1) is negatively correlated with accuracy gain (Pearson correlation coefficient of -0.48).

Comparison with [25]: They depend on finding correspondences between constituency parses of the source and target sentences. However, the common case is that a constituency parser is unavailable or ineffective for the target language, particularly for informal social media. They are thus restricted to synthesizing text from only a subset of monolingual data. Training SA with natural text augmented with their synthesized text leads to poorer accuracy, albeit by a small amount, than using CSGen. The performance is worse for target languages that are more resource-poor.

Ordinal vs. categorical loss: Table 2 shows that ordinal loss helps when the neutral label dominates. However, neither is a clear winner and the gains are small. Therefore, we use categorical loss henceforth.

Choice of monolingual corpus: Across all monolingual corpora, *Election* performs consistently well. Best test performance on HI_EN,TW was obtained by synthesizing from the *Mukherjee* corpus. Text synthesized from *Election* provides the best results for HI_EN,FB for both setups. The performance of *Union* is also good but not the best. This is because although a larger and diverse amount of data is available which ensures its quality, the Euclidean distance between test data and some individual corpora is still large.

	Categorical Cross Entropy training			Ordinal Cross Entropy training		
	Pos	Neu	Neg	Pos	Neu	Neg
	HI_EN,TW					
CSGen	0.52	0.62	0.38	0.55	0.63	0.34
Gold	0.48	0.63	0.24	0.50	0.62	0.35
	HI_EN,FB					
CSGen	0.59	0.73	0.56	0.62	0.71	0.55
Gold	0.60	0.74	0.54	0.60	0.71	0.44
	ES_EN					
CSGen	0.38	0.53	0.37	0.48	0.50	0.42
Gold	0.47	0.44	0.41	0.40	0.53	0.43
	BN_EN					
CSGen	0.63	0.49	0.58	0.55	0.47	0.59
Gold	0.55	0.51	0.65	0.37	0.49	0.61

Table 3: F1 score for each class prediction. Blue: CSGen is better than Gold.

4.6 Sentiment detection F1 score

Beyond 0/1 accuracy, Table 3 shows F1 score gains. Election yields consistently good results. We have reported the F1 score gain for different sentiment classes only for Election in Table 3 for brevity. Augmenting synthetic data with gold data yields better F1 score than training only with gold tagged data. Also, it is interesting to observe that there is a sharp drop of F1 score for HI_EN,FB and BN_EN data sets for Gold data while training with ordinal cross entropy function across all the sentiment labels. As described in §4.5, this is due to non-discriminative features. However, mixing them with synthetic data helps in achieving better results.

Train \ Test	HI_EN (TW)	HI_EN (FB)	ES_EN	BN_EN
ACL	40.33	49.96	38.40	47.81
Election	47.22	48.78	31.20	42.44
Mukherjee	46.22	48.98	39.76	44.42
Semeval	45.80	48.38	39.18	45.99
Union	43.50	49.80	41.90	41.20
Gold	52.26	65.37	42.60	55.19

Table 4: Percent accuracy on 20% test data after training on only synthetic and only gold text. Each row corresponds to a source. Grey: Accuracy achieved with only gold training. Blue: The closest accuracy achieved to best.

Category of failure	Example sentence	Gold	Predicted
Keywords with different polarity	manana voy conquistar la will forever be an amazing song not because me la dedicaron but because my momma always jams to it "tomorrow I will conquer the will" forever be an amazing song not because they dedicated it to me but because my momma always jams to it.	Positive	Negative
	twin brothers lost in fair reunited in adulthood amidst dramatic circumstances ei themer movie akhon ar viewers der attract kore na twin brothers lost in fair reunited in adulthood amidst dramatic circumstances, this theme does not yet attract viewers.	Neutral	Positive
Ambiguous overall meaning	elizaibq ellen quiere entrevistar julianna margulies clooney says she is tough cookie she is hard one to crack elizaibq ellen wants to interview julianna margulies clooney says she is tough cookie she is hard one to crack	Negative	Neutral
	hum kam se kam fight ker haaray lekin tum loog zillat ki maut maaray gaye We lost at least after a fight, but you died a terrible death.	Positive	Negative

Table 5: Examples cases of failure in prediction. Red: Negative Polarity words. Green: Positive polarity words. Blue represents the English translation of the code-switched sentence.

4.7 Performance of standalone synthetic data

The accuracy of using only synthetic data as training is reported in Table 4. We can see that for EN_HI,TW and EN_ES the synthetic data is very close to the gold data performance (lagging by 5.04% and 2.84%). However, it performed poorly for HI_EN,FB and BN_EN dataset. This is because there is heavy mismatch between the synthetic text set generated and the test data distribution (Table 1) in these two datasets. The Pearson rank correlation coefficient between the distance (between test and training set) measures and relative accuracy gain is highly negative, -0.66.

To further establish the importance of domain coherence, we report on an experiment performed with HI_EN,FB gold dataset. This dataset has texts corresponding to two different entities namely *Narendra Modi* and *Salman Khan*. Training SA with natural text corresponding to one entity and testing on the rest leads to a steep accuracy drop from 65.37% to 52.32%.

4.8 Error analysis

We found two dominant error modes where synthetic augmentation confuses the system. Table 5 shows a few examples. The first error mode can be triggered by the presence of words of different polarities, one polarity more common than the other, and the gold label being the minority polarity. The second error mode is prevalent when the emotion

is weak or mixed. Either there is no strong opinion, or there are two agents, one regarded positively and the other negatively.

4.9 Hate speech detection results

Table 6 shows hate speech detection results. Training with only synthetic text after thresholding and stratified sampling outperforms training with only gold-tagged text by 4% F1, and using both gold and synthetic text gives a F1 boost of 6% beyond using gold alone. Remarkably, synthetic text alone outperforms gold text, because gold text has high class imbalance, leading to poorer prediction. Because we can create arbitrary amounts of synthetic text, we can balance the labels to achieve better prediction.

	Prec	Recall	F-score
Only synthetic	0.58 (0.63)	0.60 (0.63)	0.51 (0.52)
Synthetic +Gold	0.59 (0.60)	0.63 (0.63)	0.53 (0.54)
Gold	0.40	0.62	0.48

Table 6: Hate speech results (3-fold cross val.). In most cells we show performance without thresholding and stratification (within bracket with thresholding and stratification). Green: Best performance in each column.

5 Conclusion

Code-mixing is an important and rapidly evolving mechanism of expression among multilingual populations on social media. Monolingual sentiment analysis techniques perform poorly on code-mixed text, partly because code-mixed text often involves resource-poor languages. Starting from sentiment-labeled text in resource-rich source languages, we propose an effective method to synthesize labeled code-mixed text without designing switching grammars. Augmenting scarce natural text with synthetic text improves sentiment detection accuracy.

Acknowledgments

Bidisha Samanta was supported by Google India Ph.D. Fellowship. We would like to thank Dipanjan Das and Dan Garrette for their valuable inputs. Soumen Chakrabarti was partly supported by IBM.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*, 2015.

[2] U. Barman, A. Das, J. Wagner, and J. Foster. Code mixing: A challenge for language identification in the language of social media. In *Proceedings of the first workshop on computational approaches to code switching*, pages 13–23, 2014.

[3] R. Begum, K. Bali, M. Choudhury, K. Rudra, and N. Ganguly. Functions of code-switching in Tweets: An annotation scheme and some initial experiments. *Proceedings of LREC*, 2016.

[4] G. Bhat, M. Choudhury, and K. Bali. Grammatical constraints on intra-sentential code-switching: From theories to working models. *arXiv preprint arXiv:1612.04538*, 2016.

[5] A. Bohra, D. Vijay, V. Singh, S. S. Akhtar, and M. Shrivastava. A dataset of hindi-english code-mixed social media text for hate speech detection. In *Proceedings of the Second Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media*, pages 36–41, 2018.

[6] S. R. Bowman, L. Vilnis, O. Vinyals, and Dai. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[7] P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

[8] G. Chittaranjan, Y. Vyas, K. Bali, and M. Choudhury. Word-level language identification using CRF: Code-switching shared task report of MSR India system. In *Proceedings of The First Workshop on Computational Approaches to Code Switching*, pages 73–79, 2014.

[9] L. Dong, F. Wei, C. Tan, D. Tang, M. Zhou, and K. Xu. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 49–54, 2014.

[10] P. M. Fernández. *Improving Word-to-word Alignments Using Morphological Information*. PhD thesis, San Diego State University, 2008.

[11] A.-M. Founta, C. Djouvas, D. Chatzakou, I. Leontiadis, J. Blackburn, G. Stringhini, A. Vakali, M. Sirivianos, and N. Kourtellis. Large scale crowdsourcing and characterization of twitter abusive behavior. *arXiv preprint arXiv:1802.00393*, 2018.

[12] C. H. E. Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14). Available at (20/04/16) http://comp. social. gatech. edu/papers/icwsm14. vader. hutto. pdf*, 2014.

[13] A. Joshi, A. Prabhu, M. Shrivastava, and V. Varma. Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2482–2491, 2016.

[14] A. Kannan and O. Vinyals. Adversarial evaluation of dialogue models. *arXiv preprint arXiv:1701.08198*, 2017.

[15] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966, 2015.

[16] M. Luong, E. Brevdo, and R. Zhao. Neural machine translation (seq2seq) tutorial. *https://github.com/tensorflow/nmt*, 2017.

[17] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*, 2015.

[18] U. Maqsud. Synthetic text generation for sentiment analysis. In *Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, 2015.

[19] S. Mukherjee and P. Bhattacharyya. Sentiment analysis in twitter with lightweight discourse analysis. *Proceedings of COLING 2012*, pages 1847–1864, 2012.

[20] S. Mukherjee, A. Malu, B. AR, and P. Bhattacharyya. Twisent: a multistage system for analyzing sentiment in twitter. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2531–2534. ACM, 2012.

[21] Z. Niu, M. Zhou, L. Wang, X. Gao, and G. Hua. Ordinal regression with multiple output cnn for age estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4920–4928, 2016.

[22] F. J. Och and H. Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

[23] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, 2002.

[24] B. G. Patra, D. Das, and A. Das. Sentiment analysis of code-mixed indian languages: An overview of sail_code-mixed shared task@ icon-2017. *arXiv preprint arXiv:1803.06745*, 2018.

[25] A. Pratapa, G. Bhat, M. Choudhury, S. Sitaram, S. Dandapat, and K. Bali. Language modeling for code-mixing: The role of linguistic theory based synthetic data. In *Proceedings of ACL*, 2018.

[26] D. Riley and D. Gildea. Improving the IBM alignment models using variational Bayes. In *Proceedings of ACL*, 2012.

[27] S. Rosenthal, N. Farra, and P. Nakov. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval '17, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[28] K. Rudra, S. Rijhwani, R. Begum, K. Bali, M. Choudhury, and N. Ganguly. Understanding language preference for expression of opinion and sentiment: What do hindi-english speakers do on twitter? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1131–1141, 2016.

[29] B. Samanta, S. Reddi, H. Jagirdar, N. Ganguly, and S. Chakrabarti. A deep generative model for code-switched text. In *Proceedings of IJCAI*, 2019.

[30] D. Sankoff and S. Poplack. A formal grammar for code-switching. *Research on Language & Social Interaction*, 14(1):3–45, 1981.

[31] T. Schoenemann. Computing optimal alignments for the IBM-3 translation model. In *Proceedings of CoNLL*, 2010.

[32] R. Sequiera, M. Choudhury, and K. Bali. POS tagging of Hindi-English code mixed text from social media: Some machine learning experiments. In *Proceedings of International Conference on NLP*, 2015.

[33] S. Sharma, P. Srinivas, and R. C. Balabantaray. Text normalization of code mix and sentiment analysis. In *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on*, pages 1468–1473. IEEE, 2015.

[34] T. Solorio, E. Blair, S. Maharjan, S. Bethard, M. Diab, M. Ghoneim, A. Hawwari, F. AlGhamdi, J. Hirschberg, A. Chang, et al. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 62–72, 2014.

[35] T. Solorio and Y. Liu. Part-of-speech tagging for English-Spanish code-switched text. In *Proceedings of EMNLP*, 2008.

[36] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979. Online at `http://www.dcs.gla.ac.uk/Keith/Preface.html`.

[37] L. N. Vaserštein. Markov processes over denumerable products of spaces describing large systems of automata. *Problems of Information Transmission*, 5(3):47–52, 1969.

[38] D. Vilares, M. A. Alonso, and C. Gómez-Rodríguez. Sentiment analysis on monolingual, multilingual and code-switching twitter corpora. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 2–8, 2015.

[39] Y. Vyas, S. Gella, J. Sharma, K. Bali, and M. Choudhury. POS tagging of English-Hindi code-mixed social media content. In *Proceedings of EMNLP*, 2014.

[40] B. Wang, M. Liakata, A. Zubiaga, and R. Procter. Tdparse: Multi-target-specific sentiment recognition on twitter. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 483–493, 2017.

[41] Y. Zhang, Z. Gan, and L. Carin. Generating text via adversarial training. In *NIPS workshop on Adversarial Training*, 2016.

[42] Y. Zhang, Z. Gan, K. Fan, Z. Chen, R. Henao, D. Shen, and L. Carin. Adversarial feature matching for text generation. *arXiv preprint arXiv:1706.03850*, 2017.

[43] M. Zhu, Y. Zhang, W. Chen, M. Zhang, and J. Zhu. Fast and accurate shift-reduce constituent parsing. In *Proceedings of ACL*, 2013.