Code-Switched Language Identification is Harder Than You Think

Anonymous ACL submission

Abstract

Code switching (CS) is a very common phenomenon in written and spoken communication, but is handled poorly by many natural language processing (NLP) applications. Looking to the application of building CS corpora, we explore CS language identification (LID) for corpus building. We make the task more realistic by scaling it to more languages and considering models with simpler architectures for faster inference. We also reformulate the task as a sentence-level multi-label tagging problem to make it more tractable. Having defined the task, we investigate three reasonable architectures for this task and define metrics which better reflect desired performance. We present empirical evidence that no current approach is adequate, and finally provide recommendations for future work in this area.

1 Introduction

Code switching (CS), or the use of one or more languages within the same utterance (Sitaram et al., 2019), is a very common phenomenon in written and spoken communication (Doğruöz et al., 2021), but one which many natural language processing (NLP) applications currently struggle to deal with (Solorio et al., 2021; Winata et al., 2023). An obvious first step in building better systems for CS is gathering the data necessary for training effective models, which is currently lacking for CS (Mendels et al., 2018). A fundamental part of this process is identifying CS in the first place.

In this paper, we look at CS language identification (LID) for text and the challenges in getting CS LID systems to work at scale. Previous shared tasks on CS LID have produced systems which achieve impressive results (Solorio et al., 2014; Molina et al., 2016), albeit limited to two languages. We seek to extend CS LID systems to work in a realistic setting as part of a corpus building pipeline by scaling up both the number of languages covered and the speed of inference.

We therefore reformulate CS LID as a multi-label task where the aim is to assign a set of language labels to each sentence, rather than a word-level or document-level tagging task as in previous work (Section 3). We experiment with high-coverage LID systems (200+ languages) which are simple enough to scale easily, and investigate three different model architectures as reasonable baseline approaches to the task (Section 4). We test on wide range of CS and single-label LID test sets aiming to cover as wide a range of languages as possible (Section 5), and we measure performance with metrics chosen to better reflect true performance in our multi-label setting than those commonly used for LID (Section 6).

We find that even the best-performing models are still inadequate for identifying CS text at scale (Section 7), due to the inherent difficulty of defining CS and detecting the intended language(s) in realistic settings. We make recommendations for future work in this area based on our findings (Section 8) and make our code freely available to aid further research.[1]

2 Previous work

LID has been an active topic of research for a long time in NLP (Jauhiainen et al., 2019). Much of the most recent research on this topic has been towards covering more and more languages, with some models claiming to cover over a thousand languages (Brown, 2014; Dunn, 2020; Adebara et al., 2022; NLLB Team et al., 2022; Burchell et al., 2023). However, nearly all general-purpose LID systems assume that text is entirely monolingual (e.g. NLLB Team et al., 2022) or occasionally that any different languages present occur in discrete chunks (e.g. Ooms, 2023). This leads to pipelines where CS text is ignored or discarded.

[1] github.com/link-after-publication

Previous work on multiple-label LID specifically can be split into two main sub-tasks: multilingual LID, where the expected input is a document containing discrete monolingual chunks in different languages; and CS LID, where the expected input is a sentence or short text containing CS text. The former task has a longer history and its intended application is to segment web text (Baldwin and Lui, 2010; Lui et al., 2014; Jauhiainen et al., 2015; Kocmi and Bojar, 2017). The latter task has received more attention recently including several shared tasks on CS LID, where the aim was word-level tagging of CS text given a known pair of languages (Solorio et al., 2014; Molina et al., 2016). However, both tasks have a limited application to web-scale text, since they made the assumption that the input could only be in a small number of known languages and they relied on computationally-expensive, high-capacity models like transformers (Vaswani et al., 2017) or LLMs for classification. We argue that these are not realistic for filtering web crawls.

Finally, we note that despite the wide range of approaches towards monolingual LID (Jauhiainen et al., 2019), LID algorithms are still found to perform poorly in practice compared to test performance, particularly for low resource languages (Caswell et al., 2020; Kreutzer et al., 2022). This shows that even the simpler task of monolingual high-coverage LID remains a challenging problem.

3 Task definition

We define our task as follows: given a short input text (around sentence length), return a set of codes corresponding to the language(s) it contains. Following NLLB Team et al. (2022), we output modified ISO 639-3 language codes encoding both the language variety and the script: for example, `eng_Latn` means English written in Latin text.

This way of framing the task differs to most previous work on CS LID by assigning tags on the sentence-level rather than on the word level (e.g. Solorio et al., 2014; Molina et al., 2016). However, we felt there was too much ambiguity when labelling at the word level. Our model covers many more languages than the previous shared tasks (201 rather just two) in CS LID so the search space becomes much larger and less tractable at the word level. In addition, the shared tasks included extra tags aside from the two included languages, covering categories such as named entities, 'foreign words', and non-linguistic content like emojis. We wished to avoid this complication since it was not relevant to our aim of dataset building. Finally, labelling at the sentence level rather than word level speeds up inference and so is more practical for web-scale text.

4 Models

We compare the performance of three model architectures for CS LID: OpenLID, a pre-existing single-label LID model adapted to a multi-label setting (Burchell et al., 2023), MultiLID, a novel LID architecture, and Franc, a high-coverage LID package.[2] The first two models are trained on the same data (that provided by OpenLID) to help isolate the effect of the change in architecture. We employ Franc as a comparison point, since it allocates prediction scores in a different way and covers more languages than the other two. In this way, we aim to measure the performance of three reasonable approaches to CS LID, explore their limitations, and so guide further research.

4.1 Baseline: OpenLID

We use the OpenLID model provided by Burchell et al. (2023) as a baseline approach, where we adapt an existing single-label LID model to a multi-label problem. We choose this model because it covers a large number of languages with good performance, it scales well to large datasets, and its openly-available training data means we can compare two models trained on the same data and thus eliminate a potential confounding variable.

OpenLID is a *fastText* model (Joulin et al., 2017). The architecture consists of a input sentence vector obtained by averaging word and n-gram embeddings, which is then fed to a simple linear classifier. The output logits are transformed to a probability distribution over the output labels with a softmax activation function. It uses cross entropy loss to update the weights.

We use thresholding to obtain multi-label outputs, since this is a standard method to adapt softmax-based classifiers to a multi-label task. This means that rather than returning the label with the maximum probability, we instead return all labels with a predicted probability over some chosen threshold k. This means that the classifier may return no labels in the case where no language is predicted a probability over the threshold. It also

[2] `github.com/wooorm/franc`

limits the maximum number of labels to $\lfloor k^{-1} \rfloor$ because the predicted probabilities for all the classes must sum to one. We set $k = 0.3$ so that the classifier can return a maximum of three labels.

Softmax-based classifiers like OpenLID make the implicit assumption that each input should be assigned one and only one label. This is because their output is a probability distribution over mutually-exclusive classes. We therefore experiment with altering the basic architecture of OpenLID to relax this assumption.

4.2 MultiLID

We create MultiLID, a novel LID architecture which conceptualises LID as a multi-label rather than single-label problem. In this way, we aim to handle both monolingual and CS text. There are a range of approaches for multi-label problems (Zhang and Zhou, 2013), but inspired by Stahlberg and Kumar (2022), we explore using binary cross entropy (BCE) loss: rather than use a softmax activation followed by cross-entropy loss as in OpenLID, MultiLID uses a sigmoid activation plus cross-entropy loss. The effect is that the predicted scores are no longer normalised into a probability distribution so the model can predict multiple classes independently.

More formally, BCE is defined as follows. Let N be the number of languages covered by the classifier, $L = \{l_1, \ldots, l_k, \ldots, l_N\}^\top$ be the output vector of predicted scores for each language where $l_k \in [0, 1]$, and $l_k^* \in \{0, 1\}$ be the true label assigned to some input representation x_k. The BCE loss for some particular element l_k is thus:

$$\text{BCELoss}_{l_k} = l_k^* \cdot \log \sigma(x_k) + (1 - l_k^*) \cdot \log\left(1 - \sigma(x_k)\right)$$

We sum the loss for each element to generate the final loss since we have a sparse output vector.

When deciding which labels to return, we found that a fixed threshold was ineffective due to the unnormalised scores. Instead, we use the following heuristic to choose the labels to return. We note that the BCE loss function encourages most scores to be close to zero, and so the mean score is very close to zero. Only some of the scores are significantly above the mean, and these correspond to the labels we want to return. We therefore calculate the mean and standard deviation of the output scores for a particular example, and set a dynamic threshold of two standard deviations above the mean based on empirical results using the LinCE training sets. We choose the language label with the highest score to ensure we always return a label, and optionally return a second label provided its score exceeds the dynamic threshold.

We build our model architecture using Python and Pytorch, and we aim to keep it as close to *fastText* as possible by design. We first clean the data and remove emoji and hash symbols, then build the vocabulary from all words seen more than 1000 times, plus the 2- to 5-grams of these words. The input sentence representation vector is formed as a bag of vocabulary embeddings, which is then fed to a linear transformation layer. The output logits are converted to output scores using a sigmoid function.

We note that our model is trained on single-label data rather than CS data, even though it is designed to be able to return multiple labels if necessary. We made this decision due to the lack of CS training data for most languages so to be practical, a CS LID model would need to work without specifically CS data for every language pair. Future work could look at exploiting what CS data does exist.

4.3 Franc

The final LID architecture we use is Franc, a LID package covering 414 languages. We include Franc because it is alternative pre-existing architecture that covers an even larger number of languages than the other two models, and it returns scores which adapt easily to a multi-label setting. Franc is not trained on the same training data as the other two models, but rather we use the pre-trained Python model to predict.[3]

At inference time, Franc returns scores for all languages that use the same script as the input text in decreasing order of probability. These scores are calculated based on the distances of the trigram distributions in the input text and the language model, scaled such that the closest language will have a score of 1. Since we often have short strings in our test sets, we set the minimum valid string length to 1 so Franc always returns a prediction. To choose which language labels to return, we select the closest predicted language label plus the second-closest language label provided its predicted score is higher than 0.99 (since this is sufficiently close to still be a valid label). This selection heuristic is based empirical results on the LinCE training sets.

[3] github.com/cyb3rk0tik/pyfranc

The language labels returned by Franc differ somewhat to those assigned to the test sets. We normalise these to some extent using the `langcodes` Python package,[4] so if the language code is not among those covered by FLORES-200*, we find an equivalent tag.[5] If a match exists, we replace the predicted tag with this match; otherwise, we simply return the original prediction. When calculating the metrics, we count all languages outside of the languages not covered by FLORES-200* as empty tags for ease of computation.

5 Test sets

Our aim when choosing test sets was to cover as many CS language pairs as possible, despite the limited number of easily-accessible CS test sets. We were further hampered by the fact that the OpenLID training data does not include Indian languages written using Roman characters which are some of the most common languages to include in CS test sets (Aguilar et al., 2020; Khanuja et al., 2020; Winata et al., 2023). Nonetheless, we source six CS test sets which include eight languages, plus a high-coverage monolingual test set.

We describe all test datasets below and include fuller instructions on how to obtain them in Appendix A. Most of the datasets we use are annotated with language tags at the token level. To fit with our task, we convert these to sentence-level tags by relabelling the sentence as CS if two language labels are present, monolingual if only one is present, and discarding the sentence if it has no language labels (e.g. the sentence only contains named entities or emojis).

Turkish–English dataset Yirmibeşoğlu and Eryiğit (2018) created a CS Turkish–English dataset as part of their work on detecting CS for this language pair. The data is sourced from Twitter and the Ekşi Sözlük online forum, then labelled at the token level as either Turkish or English. After recombining sentences, the dataset consists of 376 lines of data and 98.9% of the sentences are labelled as CS.

Indonesian–English dataset Barik et al. (2019) created a CS Indonesian–English dataset from Twitter data, where each token in each tweet is annotated with a language tag. After pre-processing, the dataset consists of 825 lines of data and 93.5% of the sentences are labelled as CS.

BaSCo Basque–Spanish corpus This corpus contains Spanish and Basque sentences sourced from a collection of text samples used in training bilingual chatbots (Aguirre et al., 2022). These sentences were shown to volunteers who were asked to provide a realistic alternative text with the same meaning in Euskañol (Basque–Spanish CS). The created sentences were checked for validity by a team of annotators. We process this corpus into our test set by extracting all Spanish, Basque, and Euskañol utterances present in the final corpus and labelling them using the provided utterance-level language labels. After processing, the dataset consists of 2304 lines of data, of which 59.8% are labelled as CS.

LinCE Spanish–English and Modern Standard Arabic–Egyptian Arabic Aguilar et al. (2020) provide a benchmark for linguistic CS evaluation, used in previous shared tasks on CS LID (Solorio et al., 2014; Molina et al., 2016). We test on two of its suite of language pairs and tasks, Spanish–English LID and Modern Standard Arabic (MSA)–Egyptian Arabic LID,[6] using the dev sets since the test sets are private. These datasets are both sourced from Twitter and are annotated at the word level. After relabelling at the sentence level and filtering, there are 3247 lines of Spanish–English data, of which 35.2% are marked as CS, and 1107 lines of MSA–Egyptian Arabic data, of which only 14.5% are marked as CS.

ASCEND Mandarin Chinese–English Lovenia et al. (2022) created a corpus of conversational Mandarin Chinese–English CS speech which is transliterated and labelled by language at the utterance level. We extract the transliterated sentences from the training split of this dataset. After processing, there are 9869 lines of data of which 27.8% are labelled as containing CS.

FLORES-200* We assess single-label LID performance using this evaluation benchmark, which consists of professional translations from 842 distinct web articles (Guzmán et al., 2019; Goyal et al., 2022). It includes 3001 sentences for each one of 204 language varieties. Following Burchell et al. (2023), we test on 201 of these taken from the devtest split, which we refer to as FLORES-200*. We test on this dataset to assess the monolingual performance of our classifier. FLORES-200* consists

[4] `github.com/rspeer/langcodes`

[5] Specifically, we filter on `tag_distance` < 10.

[6] The other two include transliterated Hindi and transliterated Nepali, neither of which are covered by our LID models.

of 203,412 lines of data.

6 Measuring performance

The most common metrics for single-label, multi-class problems are precision and recall (defined in Appendix B). However, whilst these metrics give some insight into the functioning of our models, we found them too easy to misinterpret in a multi-label setting. The first reason for this is that precision and recall are undefined when there are no true positive examples of a predicted class in the dataset. This was very common given our high-coverage models, but precision and recall could not detect this key performance issue. Secondly, neither precision nor recall account for true negatives which are key for our application to building web corpora, since avoiding spurious labels is part of preventing noisy datasets.

As a consequence of these findings, we decided that precision and recall were not suitable for use as main metrics. Instead, we chose three alternative metrics as a better reflection of the desired downstream performance: exact match ratio, Hamming loss, and false positive rate (FPR). These metrics allow direct comparison between our different datasets and are easy to interpret correctly even in a multi-label set up with many classes such as ours. We define and discuss each metric below.

Exact match ratio This metric is simply that for each sentence i in our dataset of length N, we count a correct match if all the predicted labels $(\hat{y})$ match the gold labels (y):

$$\text{Exact match ratio} = \frac{1}{N}\sum_{i=0}^{N-1} \mathbb{I}(\hat{y}_i = y_i)$$

The higher the metric, the better. The exact match ratio has the advantage of being easy to understand, but it is a strict measure of success and does not reward partial matches.

Hamming loss We therefore also report Hamming loss which allows us to both give credit for partial matches and to penalise predicting too many labels. It can be understood as the fraction of wrong labels to the total number of labels, and the smaller the value of the loss the better. More precisely, let L be the number of classes (languages), $Y_{i,l}$ $(\hat{Y}_{i,l})$ signify the Boolean that the i^{th} example (prediction) is assigned the l^{th} language label, and $\oplus$ denote exclusive-or:

$$\text{Hamming loss} = \frac{1}{LN}\sum_{i=0}^{N-1}\sum_{l=0}^{L-1} Y_{i,l} \oplus \hat{Y}_{i,l}$$

False positive rate Finally, we report the macro-average of false positive rate (FPR) with respect to each language class, or the ratio of number of examples incorrectly identified as a particular language (false positives, FP) to the total number of ground truth negatives (true negatives plus false positives, $TN + FP$).

$$\text{False positive rate} = \frac{FP}{TN + FP}$$

The smaller the FPR, the better. Measuring non-relevant predictions is particularly important given our intended application of building web corpora. This is because the internet mostly consists of non-CS data, so using a classifier with a high FPR on the web will result in a final dataset where most of the content is not relevant (Caswell et al., 2020).

7 Results

	MultiLID	OpenLID	Franc
Exact match ↑	0.861	**0.926**	0.672
Hamming loss ↓	0.00121	**0.000694**	0.00279
FPR ↓	0.000885	**0.000395**	0.00123
Precision ↑	0.879	**0.942**	0.666
Recall ↑	0.933	**0.939**	0.706
Mean # preds.	1.11	1.02	1.08

Table 1: Results on FLORES-200* test set. We include results using the OpenLID model returning all labels with predicted probability > 0.3 and the top two predictions from Franc with score > 0.99.

FLORES-200 results We first consider the results on the single-label LID test set FLORES-200* in order to provide a point of comparison with later results on CS datasets. Table 1 shows that the OpenLID classifier achieves the best results for each assessed metric, which is unsurprising given that it is designed as a single-label classifier which covers the languages of FLORES-200*. MultiLID still shows reasonable performance, though Hamming loss and FPR are markedly higher. This is likely because MultiLID is more likely to predict multiple labels as shown at the bottom of Table 1. The performance for Franc is markedly lower across all metrics, though it should be noted that this model is disadvantaged here by covering far more languages than the other two.

		Exact match ↑			Hamming loss ↓			False positive rate ↓		
	% c/s	MultiLID	OpenLID	Franc	MultiLID	OpenLID	Franc	MultiLID	OpenLID	Franc
tur–eng	98.9	**0.0665**	0.0213	0.00532	0.00732	**0.00531**	0.00903	0.00206	**0.000291**	0.00119
ind–eng	93.5	**0.184**	0.0448	0.0182	**0.00617**	0.00680	0.00995	0.00199	**0.00153**	0.00164
eus–spa	59.8	0.317	**0.360**	0.201	0.00576	**0.00383**	0.00746	0.00213	**0.000620**	0.00169
spa–eng	35.2	0.379	**0.417**	0.146	0.00613	**0.00451**	0.00721	0.00314	**0.00126**	0.00168
zho–eng	27.8	**0.508**	0.507	0.301	0.00399	**0.00386**	0.00447	0.00197	0.00130	**0.000332**
arb–arz	14.5	0.345	0.625	**0.691**	0.00631	0.00281	**0.00242**	0.00500	**0.00174**	0.00481

Table 2: Main metrics calculated for predictions on the CS datasets, plus the percentage of CS in each dataset.

CS test sets: main metrics Moving on to the results for the CS test sets, Table 2 gives the exact match ratio, Hamming loss, and FPR for the three assessed models. As mentioned in Section 5, there is a wide variation between how many sentences labelled as CS are present in each test set, from 98.8% in the Turkish–English dataset to just 14.5% in the MSA–Egyptian Arabic dataset.

In terms of exact label match, MultiLID performs better on the most code-mixed datasets, though the absolute numbers are still much lower compared to single-label performance: compare 0.93 for top-1 OpenLID on FLORES-200 to just 0.06 for MultiLID on the Turkish–English dataset. Similarly, the Hamming loss for all models differs by an order of magnitude compared to OpenLID single-label performance in Table 1, showing that they struggle to label CS text correctly.

Franc's algorithm means that it is at a particular disadvantage when dealing with CS text, since it bases its prediction partially on the script. In the case of mixed scripts (as in the Chinese–English CS data), it often did not return a label at all. This lead to the high FPR but low exact match on this dataset. Additionally, Franc does not cover Arabic dialects including Egyptian Arabic, so it labelled nearly all sentences in the MSA–Egyptian Arabic dataset as MSA. This gave it a high exact match score and low Hamming loss compared to the other models since it could not confuse similar Arabic dialects and most of the dataset was actually single-label. However, the fact remains that it does not cover Egyptian Arabic at all, and the higher results here show the limitations of the testing regime.

Notably, the FPR of the OpenLID model is lower for every test set compared to the other two models (apart from for Chinese–English as discussed above), sometimes by as much as an order of magnitude. This is likely because the OpenLID model is much more conservative when assigning labels. Table 3 gives the percentage of empty predictions by the OpenLID classifier. It shows that null predictions are often a significant proportion of the results, and the proportion of null predictions is nearly always larger for the CS sentences in particular. OpenLID's conservatism lowers the FPR but increases the probability of missing CS sentences. This may not be desirable when building a corpus since these sentences are a small proportion of the total data to begin with.

	% empty	% c/s empty
FLORES-200	0.092	-
Turkish–English	0.798	0.806
Indonesian–English	9.46	9.21
Basque–Spanish	0.608	0.726
Spanish–English	10.6	12.0
Chinese–English	1.91	4.42
MSA–Egyptian Arabic	0.632	1.24

Table 3: Percentage of empty predictions returned by the OpenLID classifier. The left column gives results over the entire dataset, the right only the CS sentences.

Performance on CS sentences Table 4 gives the the main metrics solely on the CS sentences in each dataset. MultiLID shows higher performance on exact match for every test sets, but the absolute numbers are still low and there is a notable reduction in performance for the datasets with the least amount of CS. This shows that the better numbers in Table 2 were mostly driven by good results on the single-label sentences. Hamming loss is more mixed but the FPR for OpenLID is now an order of magnitude lower across the board. This suggests again that the OpenLID classifier is much more conservative in assigning labels, so the FPR is low without getting more labels correct. Franc achieves zero in exact match for nearly every test set, suggesting that the algorithm is not suited to CS text. The low FPR for the last two test sets is due to a lack of prediction rather than desired performance, showing the necessity of suite of metrics.

Precision and recall We return to the entire CS tests sets to calculate precision and recall with respect to each language present. Precision was

	Exact match ↑			Hamming loss ↓			False positive rate ↓		
	MultiLID	OpenLID	Franc	MultiLID	OpenLID	Franc	MultiLID	OpenLID	Franc
tur–eng	**0.0618**	0.0134	0	0.00737	**0.00535**	0.00907	0.00206	**0.000281**	0.00117
ind–eng	**0.153**	0.00649	0.0013	**0.00634**	0.00704	0.0103	0.00175	**0.000968**	0.00148
eus–spa	**0.0247**	0.0189	0	0.00789	**0.00563**	0.00979	0.00226	**0.000408**	0.00171
spa–eng	**0.0184**	0.00613	0	0.00844	**0.00729**	0.0105	0.00259	**0.000985**	0.0019
zho–eng	**0.0365**	0.0164	0	**0.00618**	0.00637	0.00777	0.00107	0.000703	**0.000620**
arb–arz	**0.0994**	0.0373	0	0.00766	0.00584	**0.00535**	0.00294	0.000587	**0.000127**

Table 4: Results over CS sentences only (3 significant figures)

nearly always very close to one, showing that the predictions that the model did make were very likely to be correct. The only exception to this was Egyptian Arabic, where precision was 0.645 for the OpenLID model, 0.485 for the MultiLID model, and 0 for Franc. This was due to former two models struggling to distinguish between Arabic dialects and a lack of coverage for the latter.

Recall for each model and language label was much more varied, as can be seen in Table 5. For the datasets with the highest amount of CS (Turkish–English and Indonesian–English), there is a large difference between the recall of the OpenLID model. This suggests that its predictions only contain one of the classes and it is failing to detect the other. The difference is less pronounced for MultiLID, suggesting that it is more likely to detect the presence of the other language. For the other datasets, MultiLID does slightly better in recall overall compared to OpenLID, likely because it returns multiple labels more often. Franc nearly always has lower recall compared to the other two models (apart from the degenerate results for MSA) though it is important to note that it is disadvantaged by covering more labels.

We draw attention to the (sometimes) relatively high scores for recall and the low scores in Tables 2 and 4. In particular, we note that considering precision and recall in isolation might lead to the conclusion that using one of these LID models in a pipeline would create an adequate CS dataset. However, the low exact match scores show just how few of the labels are actually correct, especially for CS sentences. This demonstrates the importance here of careful metric selection.

Number of unique languages predicted We see from Table 6 that the predictions for all classifiers contain a large number of languages despite there being only two language labels in each test set. This suggests that all three are struggling to form a consistent representation of each language based on the input feature vectors. This may be due to the 'confusion' of CS, or possibly because of a change of domain from training to test: the training data (at least for OpenLID and MultiLID) is mostly formal text whereas the test data is primarily social media. The predictions for MultiLID contain far more unique languages than those for OpenLID. This is likely because the lack of normalisation in its architecture results in a less strong prior over languages, so it is more likely to predict rarer languages. Franc's predictions nearly always contain far more again, which is probably an artifact of the large number of languages it includes.

	Recall ↑		
Label	MultiLID	OpenLID	Franc
tur	0.731	**0.952**	0.435
eng	**0.206**	0.032	0.027
ind	0.723	**0.727**	0.227
eng	**0.372**	0.066	0.063
eus	0.706	**0.858**	0.459
spa	**0.377**	0.312	0.128
spa	0.467	**0.469**	0.193
eng	**0.642**	0.560	0.211
zho	**0.792**	0.695	0.467
eng	**0.517**	0.451	0.222
arb	0.540	0.734	**0.995**
arz	**0.891**	0.721	0.000

Table 5: Recall with respect to each pair of languages in each CS test dataset. Precision is nearly always ≈ 1.

	MultiLID	OpenLID	Franc
tur–eng	54	11	97
ind–eng	79	27	118
eus–spa	94	50	193
spa–eng	126	86	234
zho–eng	134	85	225
arb–arz	18	10	8

Table 6: Number of unique languages in the predictions by each model for each CS test set.

8 Analysis

Considering the results as a whole, it is clear that none of the models are adequate for the task of

detecting the language(s) of CS text. The OpenLID architecture is not designed to return multiple labels and so misses many examples of CS sentences, preferring to label them with a single label or not return a label at all. The MultiLID architecture has the advantage of being designed to return multiple labels, but the lack of normalisation in the scores means that it is more likely to return spurious labels, as shown in its high FPR and larger number of unique languages in the predictions. Franc's algorithm is not suitable for CS text since it assumes a single script and is designed for longer pieces of text. In all cases, the low exact match ratios show that if we were building a corpus from this data, we would miss most of the CS sentences.

The performance in general is hampered by one of the inherent problems in CS LID: the boundaries of CS are not defined clearly, even at a linguistic level. In her book on the subject, Gardner states that CS "is not an entity which exists out there in the objective world, but a construct which linguists have developed to help them describe their data" (Gardner-Chloros, 2009, p.10). However, both linguists and language users disagree on what should count as CS, meaning assigning language labels to text can be an ambiguous task in itself.

We illustrate our point with two contrasting examples. Firstly, this tweet is a fairly straightforward example of a separate English fragment followed by a Spanish fragment:

> `@USER delete that tweet. . . ya lo hize.`

This makes it easy (for a human annotator) to assign language labels to it. However, there are many more cases of potential CS which are much more ambiguous and harder to label. The most common of these is a single-word switch in a sentence (Gardner-Chloros, 2009, p.30), for example:

> `hoy me siento bien senior. . . .`

These short switches complicate labelling for two main reasons. Firstly, there is no clear line between a 'borrowed' word, CS, and a loan word which is now an accepted part of the language (indeed, loan words start out as CS) (Gardner-Chloros, 2009, p.30). Secondly, short fragments of CS can make it difficult to work out which language was intended by the author. This leads to disagreement even amongst expert annotators and consequent 'noisy' labels. We also note that the non-standard orthography of social media and informal text can also hamper n-gram based approaches to LID.

8.1 Recommendations

In light of our results and analysis, we have the following suggestions for improving CS LID over the baseline approaches explored in this paper.

Firstly, we recommend that researchers consider which metrics they use carefully and think about how they relate to the downstream performance: for example, the metrics we use in this paper aim to reflect how useful the LID model will be for corpus building. We have shown that using metrics common in multi-class tasks for multi-label tasks is easily misleading and that a suite of metrics is necessary to capture performance fully.

Secondly, any approach should embrace the ambiguity inherent in the task, and aim for a common sense rather than prescriptive definition of what counts as a language (Gardner-Chloros, 2009, pp.165-7). With respect to NLP, this means considering the task of language labelling in light of the downstream application, rather than assuming that labels are fixed and exclusive. CS is too heterogeneous a concept for a 'one size fits all' definition to be useful for improving NLP tooling for multilingual users.

Finally, we believe that the performance of CS LID depends heavily on the input representation. All of the models we study in the paper rely on n-gram representations, and the poor results across the board suggest that these are not adequate for representing CS in actual use. Further work should move beyond n-gram based embeddings so that the input representation could more easily pick up short switches.

9 Conclusion

We explored the task of scaleable CS LID such that it could to work as part of a corpus-building pipeline. We found that three reasonable approaches to the task fell short of the performance required to build useful corpora, demonstrating that the task of realistic CS LID at scale is far from solved. We recommend that future work choose metrics with care to reflect true performance, understand the ambiguity inherent in CS, and fit their definition of CS to the intended task rather than enforce a prescriptive definition of the phenomenon.

Limitations

The CS test sets we use only cover a small fraction of the potential language sets which could be used in multi-lingual communication, and additionally the languages we cover are mostly high-resource (particularly English). Creating more high-quality CS datasets for more of the world's languages would be incredibly useful further work.

Ethics Statement

Using social media data to build corpora needs to be done with care so as not to violate users' rights to privacy. The CS test sets based on social media in this work have been anonymised and we provide links to the data for further research rather than hosting the files ourselves; this is to help control distribution of the data. We hope that by creating more CS datasets, NLP technologies become accessible for more people in their preferred language and register of communication.

References

Ife Adebara, AbdelRahim Elmadany, Muhammad Abdul-Mageed, and Alcides Inciarte. 2022. AfroLID: A neural language identification tool for African languages. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1958–1981, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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

Maia Aguirre, Laura García-Sardiña, Manex Serras, Ariane Méndez, and Jacobo López. 2022. BaSCo: An annotated Basque-Spanish code-switching corpus for natural language understanding. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 3158–3163, Marseille, France. European Language Resources Association.

Timothy Baldwin and Marco Lui. 2010. Multilingual language identification: ALTW 2010 shared task data. In *Proceedings of the Australasian Language Technology Association Workshop 2010*, pages 4–7, Melbourne, Australia.

Anab Maulana Barik, Rahmad Mahendra, and Mirna Adriani. 2019. Normalization of Indonesian-English code-mixed Twitter data. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 417–424, Hong Kong, China. Association for Computational Linguistics.

Ralf Brown. 2014. Non-linear mapping for improved identification of 1300+ languages. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 627–632, Doha, Qatar. Association for Computational Linguistics.

Laurie Burchell, Alexandra Birch, Nikolay Bogoychev, and Kenneth Heafield. 2023. An open dataset and model for language identification. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 865–879, Toronto, Canada. Association for Computational Linguistics.

Isaac Caswell, Theresa Breiner, Daan van Esch, and Ankur Bapna. 2020. Language ID in the wild: Unexpected challenges on the path to a thousand-language web text corpus. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6588–6608, Barcelona, Spain (Online). International Committee on Computational Linguistics.

A. Seza Doğruöz, Sunayana Sitaram, Barbara E. Bullock, and Almeida Jacqueline Toribio. 2021. A survey of code-switching: Linguistic and social perspectives for language technologies. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1654–1666, Online. Association for Computational Linguistics.

Jonathan Dunn. 2020. Mapping languages: The corpus of global language use. *Language Resources and Evaluation*, 54(4):999–1018.

Penelope Gardner-Chloros. 2009. *Code-switching*. Cambridge University Press.

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzmán, and Angela Fan. 2022. The Flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics*, 10:522–538.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. The FLORES evaluation datasets for low-resource machine translation: Nepali–English and Sinhala–English. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6098–6111, Hong Kong, China. Association for Computational Linguistics.

Tommi Jauhiainen, Krister Lindén, and Heidi Jauhiainen. 2015. Language set identification in noisy synthetic multilingual documents. In *Computational Linguistics and Intelligent Text Processing: 16th International Conference, CICLing 2015, Cairo, Egypt,*

April 14-20, 2015, Proceedings, Part I 16, pages 633–643. Springer.

Tommi Jauhiainen, Marco Lui, Marcos Zampieri, Timothy Baldwin, and Krister Lindén. 2019. Automatic language identification in texts: A survey. *Journal of Artificial Intelligence Research*, 65:675–782.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

Simran Khanuja, Sandipan Dandapat, Anirudh Srinivasan, Sunayana Sitaram, and Monojit Choudhury. 2020. GLUECoS: An evaluation benchmark for code-switched NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3575–3585, Online. Association for Computational Linguistics.

Tom Kocmi and Ondřej Bojar. 2017. LanideNN: Multilingual language identification on character window. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 927–936, Valencia, Spain. Association for Computational Linguistics.

Julia Kreutzer, Isaac Caswell, Lisa Wang, Ahsan Wahab, Daan van Esch, Nasanbayar Ulzii-Orshikh, Allahsera Tapo, Nishant Subramani, Artem Sokolov, Claytone Sikasote, Monang Setyawan, Supheakmungkol Sarin, Sokhar Samb, Benoît Sagot, Clara Rivera, Annette Rios, Isabel Papadimitriou, Salomey Osei, Pedro Ortiz Suarez, Iroro Orife, Kelechi Ogueji, Andre Niyongabo Rubungo, Toan Q. Nguyen, Mathias Müller, André Müller, Shamsuddeen Hassan Muhammad, Nanda Muhammad, Ayanda Mnyakeni, Jamshidbek Mirzakhalov, Tapiwanashe Matangira, Colin Leong, Nze Lawson, Sneha Kudugunta, Yacine Jernite, Mathias Jenny, Orhan Firat, Bonaventure F. P. Dossou, Sakhile Dlamini, Nisansa de Silva, Sakine Çabuk Ballı, Stella Biderman, Alessia Battisti, Ahmed Baruwa, Ankur Bapna, Pallavi Baljekar, Israel Abebe Azime, Ayodele Awokoya, Duygu Ataman, Orevaoghene Ahia, Oghenefego Ahia, Sweta Agrawal, and Mofetoluwa Adeyemi. 2022. Quality at a glance: An audit of web-crawled multilingual datasets. *Transactions of the Association for Computational Linguistics*, 10:50–72.

Holy Lovenia, Samuel Cahyawijaya, Genta Winata, Peng Xu, Yan Xu, Zihan Liu, Rita Frieske, Tiezheng Yu, Wenliang Dai, Elham J. Barezi, Qifeng Chen, Xiaojuan Ma, Bertram Shi, and Pascale Fung. 2022. ASCEND: A spontaneous Chinese-English dataset for code-switching in multi-turn conversation. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 7259–7268, Marseille, France. European Language Resources Association.

Marco Lui, Jey Han Lau, and Timothy Baldwin. 2014. Automatic detection and language identification of multilingual documents. *Transactions of the Association for Computational Linguistics*, 2:27–40.

Gideon Mendels, Victor Soto, Aaron Jaech, and Julia Hirschberg. 2018. Collecting code-switched data from social media. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Giovanni Molina, Fahad AlGhamdi, Mahmoud Ghoneim, Abdelati Hawwari, Nicolas Rey-Villamizar, Mona Diab, and Thamar Solorio. 2016. Overview for the second shared task on language identification in code-switched data. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 40–49, Austin, Texas. Association for Computational Linguistics.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No Language Left Behind: Scaling Human-Centered Machine Translation.

Jeroen Ooms. 2023. *cld2: Google's Compact Language Detector 2*. Https://docs.ropensci.org/cld2/ (docs) https://github.com/ropensci/cld2 (devel) https://github.com/cld2owners/cld2 (upstream).

Sunayana Sitaram, Khyathi Raghavi Chandu, Sai Krishna Rallabandi, and Alan W Black. 2019. A survey of code-switched speech and language processing. *arXiv preprint arXiv:1904.00784*.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Ghoneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, and Pascale Fung. 2014. 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, Doha, Qatar. Association for Computational Linguistics.

Thamar Solorio, Shuguang Chen, Alan W. Black, Mona Diab, Sunayana Sitaram, Victor Soto, Emre Yilmaz, and Anirudh Srinivasan, editors. 2021. *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*. Association for Computational Linguistics, Online.

Felix Stahlberg and Shankar Kumar. 2022. Jam or cream first? modeling ambiguity in neural machine

translation with SCONES. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4950–4961, Seattle, United States. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Genta Winata, Alham Fikri Aji, Zheng Xin Yong, and Thamar Solorio. 2023. The decades progress on code-switching research in NLP: A systematic survey on trends and challenges. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2936–2978, Toronto, Canada. Association for Computational Linguistics.

Zeynep Yirmibeşoğlu and Gülşen Eryiğit. 2018. Detecting code-switching between Turkish-English language pair. In *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pages 110–115, Brussels, Belgium. Association for Computational Linguistics.

Min-Ling Zhang and Zhi-Hua Zhou. 2013. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837.

A Data sourcing

We provide instructions on how we obtained all datasets used in this paper to aid future work. These are correct at the time of writing; we cannot guarantee that datasets will be available in the future.

- OpenLID training dataset: downloaded from `https://github.com/laurieburchell/open-lid-dataset`.
- FLORES-200 benchmark: downloaded from `https://github.com/facebookresearch/flores/blob/main/flores200`.
- Turkish–English dataset: fill out and email requisition form at `http://tools.nlp.itu.edu.tr/Datasets`.
- Indonesian–English dataset: emailing lead author (see Barik et al., 2019, for contact details).
- BaSCo Basque–Spanish dataset: `valid_utterances.json` downloaded from `https://github.com/Vicomtech/BaSCo-Corpus`.
- LinCE LID benchmark: validation data sourced from `https://huggingface.co/datasets/lince`.
- ASCEND Chinese–English dataset: training data sourced from `https://huggingface.co/datasets/CAiRE/ASCEND`.

B Precision and recall

Let TP be the count of true positives, FP be the count of false positives, and FN be the count of false negatives. Then

$$\text{precision} = \frac{TP}{TP + FP},$$

$$\text{recall} = \frac{TP}{TP + FN}$$