2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)

Language Identification on Massive Datasets of Short Messages using an Attention Mechanism CNN

Duy-Tin Vo
Chata Technologies Inc.
Calgary, Alberta T2P 2Z1
duytinvo@gmail.com

Richard Khoury
Université Laval
Quebec City, Quebec G1V 0A6
Richard.Khoury@ift.ulaval.ca

***Abstract*—Language Identification (LID) is a challenging task, especially when the input texts are short and noisy such as microblog posts on social media or chat logs on gaming forums. The task has been tackled by either designing a feature set for a traditional classifier (e.g. Naive Bayes) or applying a deep neural network classifier (e.g. Bi-directional GRU, Encoder-Decoder). These methods are usually trained and tested on a private corpus, then used as off-the-shelf packages by other researchers on their own datasets, and consequently the various results published are not directly comparable. In this paper, we first create a new massive labeled dataset based on one year of Twitter data. We use this dataset to test several existing LID systems, in order to obtain a set of coherent benchmarks, and we make our dataset publicly available so that others can add to this set of benchmarks. Finally, we propose a shallow but efficient neural LID system, which is a ngram-regional convolution neural network enhanced with an attention mechanism. Experimental results show that our architecture is able to predict tens of thousands of samples per second and surpasses all state-of-the-art systems in accuracy and F1 score, including outperforming the popular langid system by 5%.**

***Index Terms*—LID, NN, Data mining, corpus, AI**

I. Introduction

Language Identification (LID) is the Natural Language Processing (NLP) task of automatically recognizing the language that a document is written in. While this task was called "solved" by some authors over a decade ago, it has seen a resurgence in recent years thanks to the rise in popularity of social media [1], [2], and the corresponding daily creation of millions of new messages in dozens of different languages including rare ones that are not often included in LID systems. Moreover, these messages are typically very short (Twitter messages were until recently limited to 140 characters) and very noisy (including an abundance of spelling mistakes, non-word tokens like URLs, emoticons, or hashtags, as well as foreign-language words in messages of another language), whereas LID was solved using long and clean documents. Indeed, several studies have shown that LID systems trained to a high accuracy on traditional documents suffer significant drops in accuracy when applied to short social-media texts [3], [4].

Given its massive scale, multilingual nature, and popularity, Twitter has naturally attracted the attention of the LID research community. Several attempts have been made to construct LID datasets from that resource. However, a major challenge is to assign each tweet in the dataset to the correct language among the more than 70 languages used on the platform. The three commonly-used approaches are to rely on human labeling [5], [6], machine detection [6], [7], or user geolocation [4], [8], [9]. Human labeling is an expensive process in terms of workload, and it is thus infeasible to apply it to create a massive dataset and get the full benefit of Twitter's scale. Automated LID labeling of this data creates a noisy and imperfect dataset, which is to be expected since the purpose of these datasets is to create new and better LID algorithms. And user geolocation is based on the assumption that users in a geographic region use the language of that region; an assumption that is not always correct, which is why this technique is usually paired with one of the other two. Our first contribution in this paper is to propose a new approach to build and automatically label a Twitter LID dataset, and to show that it scales up well by building a dataset of over 18 million labeled tweets. Our hope is that our new Twitter dataset will become a benchmarking standard in the LID literature.

Traditional LID models start by designing a set of useful features, which is then passed to a traditional machine learning algorithms such as Naive Bayes (NB) or SVM [3], [4], [10]. The resulting systems are capable of labeling thousands of inputs per second with moderate accuracy. Meanwhile, neural network models [7], [11] start with a deep architecture like gated recurrent unit (GRU) or encoder-decoder net. They use the message text itself as input using a sequence of character embeddings, and automatically learn its hidden structure via a deep neural network. Consequently, they obtain better results in the task but with an efficiency trade-off. To alleviate these drawbacks, our second contribution in this paper is to propose a shallow but efficient neural LID algorithm. We follow previous neural LID [7], [11] in using character embeddings as input. However, instead of using a deep neural net, we propose a shallow ngram-regional convolution neural network (CNN) with an attention mechanism to learn input representation.

IEEE/ACM ASONAM 2020, December 7-10, 2020
978-1-7281-1056-1/20/$31.00 © 2020 IEEE

We experimentally show that this architecture is much more efficient than other deep neural networks, and that the attention structure focuses on the most important LID features in the text. Compared with other benchmarks on our Twitter datasets, our proposed model consistently achieves new state-of-the-art results with an improvement of 5% in accuracy and F1 score and a competitive inference time.

The rest of this paper is structured as follows. After a background review in the next section, we present our Twitter dataset in Section 3. Our novel LID algorithm is the topic of Section 4. We then present and analyze some experiments we conducted with our algorithm in Section 5, along with benchmarking tests of popular and literature LID systems, before drawing some concluding remarks in Section 6. Our Twitter dataset and LID algorithm's source code are publicly available[1].

II. Related Work

In this section, we consider recent advances on the specific challenge of LID on short text messages. Readers interested in a general overview of the area of LID, including older works and other challenges, are encouraged to read the thorough survey of [1].

One of the first systems for microblog LID is the graph-based method of [6]. Their graph is composed of vertices, or character n-grams (n = 3) observed in messages in all languages, and of edges, or connections between successive n-grams weighted by the observed frequency of that connection in each language. Identifying the language of a new message is then done by identifying the most probable path in the graph that generates that message. Their method achieves an accuracy of 0.975 on their own Twitter corpus.

[4] computed the prior probability of the message being in a given language independently of the content of the message itself, in five different ways: by identifying the language of external content linked to by the message, the language of previous messages by the same user, the language used by other users mentioned in the message, the language of previous messages in the on-going conversation, and the language of other messages with the same hashtags. They achieve an accuracy of 0.972 when combining these priors with a linear interpolation.

One of the most popular LID packages is the langid.py library [3], thanks to the fact it is an open-source, ready-to-use library written in Python. It is a multinomial Naïve Bayes classifier trained on character n-grams ($1 \leq n \leq 4$) from 97 different languages. The training data comes from longer document sources, both formal ones (government publications, software documentation, newswire) and informal ones (online encyclopedia articles, websites). In their experiments, they outperform the standard linear-kernel SVM LID benchmarks. While their system is not specialized for short messages, the authors claim their algorithm can generalize across domains off-the-shelf, and they conducted experiments using the Twitter datasets [4], [6] that achieved accuracies of 0.941 and 0.886, which is weaker than the specialized short-message LID systems [4], [6].

Starting from the basic observation that each language has a small number of words that occur very frequently, [10] created a dictionary-based algorithm using ranked dictionaries of the 1,000 most popular words of each language it is trained on. Given a new message, recognized words are given a weight based on their rank in each language, and the identified language is the one with the highest sum of word weights. They achieve an F1-score of 0.733 on the TweetLID corpus [12].

[2] built a hierarchical system of two neural networks. The first level is a Convolutional Neural Network (CNN) that converts white-space-delimited words into a word vector. The second level is a Long-Short-Term Memory (LSTM) network (a type of recurrent neural network (RNN)) that takes in sequences of word vectors from the first level and maps them to language labels. They trained and tested their network on Twitter's official Twitter70 dataset, and achieved an F-score of 0.912, compared to langid.py's performance of 0.879 on the same dataset. They also trained and tested their system using the TweetLID corpus and achieved an F1-score of 0.762, above the system of [10] and above the top system of the TweetLID competition, the SVM LID system of [13] which achieved an F1-score of 0.752.

[11] also used a RNN system, but found that the Gated Recurrent Unit (GRU) architecture performed slightly better than the LSTM in their experiments. Their system breaks the text into non-overlapping 200-character segments, and feeds character n-grams (n = 8) into the GRU network to classify each letter into a probable language. The segment's language is the most probable one over all letters, and the text's language is the most probable one over all segments. The authors tested their system on short messages by dividing their data into 200-character segments. On that corpus, they achieve an accuracy of 0.955, while langid.py achieves 0.912.

[7] also created a character-level LID network using a GRU architecture, in the form of a three-layer encoder-decoder RNN. They trained and tested their system using their own Twitter dataset, and achieved an F1-score of 0.982, while langid.py achieved 0.960 on the same dataset.

III. Our Twitter LID Datasets

A. Source Data and Language Labeling

Unlike other authors who built Twitter datasets, we chose not to mine tweets from Twitter directly, but instead use tweets that have been archived on the Internet Archive[2]. This has two important benefits: this site makes its content freely available for research purposes, and the tweets are backed-up permanently while tweets on Twitter may be deleted at any time and become unavailable for future research or replication of past studies. The Internet Archive has made available a set

[1] https://github.com/duytinvo/LID_NN

[2] https://archive.org/details/twitterstream

Lang	P(%)	Lang	P(%)	Lang	P(%)
EN	36.458	EL	0.086	LV, BG,	
JA	23.750	SV	0.046	UR, TA	10^{-3}
ES	9.964	FA	0.027	MR, BN,	
AR	7.627	VI	0.021	MR, BN,	
PT	6.839	FI	0.020	IN, KN,	
KO	5.559	CS	0.015	ET, SL,	
TH	2.965	UK	0.015	GU, CY,	
FR	2.180	HI	0.013	ZH, CKB,	
TR	2.152	DA	0.007	IS, LT,	
RU	0.948	HU	0.006	ML, SI,	
IT	0.490	NO	0.005	IW, NE,	
DE	0.356	RO	0.003	KM, MY,	
PL	0.251	SR	0.003	TL, KA,	
NL	0.187	EU	0.002	BO	$< 10^{-3}$

Table I: Twitter corpus distribution by language label.

of 1.7 billion tweets collected over the year of 2017, including all tweet metadata attributes. Five of these attributes are of particular importance to us. They are $tweet.id$, $tweet.user.id$, $tweet.text$, $tweet.lang$, and $tweet.user.lang$, corresponding to the tweet ID number, the user ID number, the text content of the tweet, the tweet's language as determined by Twitter's LID software, and the user's self-declared language.

We begin by filtering the corpus to keep only tweets where the user's self-declared language and the tweet's detected language correspond; that language becomes the tweet's correct language label. This is in line with automated LID labeling techniques found in the literature [4], [9], [12], and while it will favor tweets written in each user's first language the nature of Twitter communications means those tweets will show a huge variation of quality, from well-written professional messages to slang chit-chat. This operation leaves us with a corpus of about 900 million tweets in 54 languages distributed according to Table I. It is a very imbalanced distribution, with English and Japanese accounting for 60% of all tweets. This is consistent with other studies and statistics of language use on Twitter[3] and with other massive imbalanced Twitter LID datasets [7]. It does however make it very difficult to use this corpus to train a LID system for other languages, especially for one of the dozens of seldom-used languages. This is our motivation for creating a balanced Twitter dataset.

Table I also shows that only major languages are identified in Twitter, while regional dialects are not. For example, there is an entry for Portuguese (PT), but it does not distinguish between Portugal and Brazil's dialects. This means that closely-related LID, such as distinguishing different variations of Portuguese [14], cannot be studied from Twitter data without enriching the data by manually annotating the different dialects [15].

B. Our Balanced Datasets

When creating a balanced Twitter LID dataset, we face a design question: should our dataset seek to maximize the number of languages present, to make it more interesting and challenging for the task of LID, but at the cost of having fewer tweets per language to include seldom-used languages. Or should we maximize the number of tweets per language to make the dataset more useful for training deep neural networks, but at the cost of having fewer languages present and eliminating the seldom-used languages. To circumvent this issue, we propose to build three datasets: a small-scale one with more languages but fewer tweets, a large-scale one with more tweets but fewer languages, and a medium-scale one that is a compromise between the two. Moreover, for our datasets to become a standard benchmarking tool, we have subdivided the tweets of each language into training, validation, and testing sets.

[3]https://www.statista.com/statistics/267129/most-used-languages-on-twitter/

Lang	C1	C2	C3	C4
EN	99	0	0	1
FR	98	2	0	0
VI	84	16	0	0
DE	88	8	0	4
ES	97	1	0	3
AR	98	0	0	2

Table II: Validation of language labels.

Our small-scale dataset is composed of 28 languages with 13,000 tweets per language, subdivided into 7,000 training set tweets, 3,000 validation set tweets, and 3,000 testing set tweets, for a total of 364,000 tweets in the corpus. Referring to Table I, this dataset includes every language that represents 0.002% or more of Twitter. It is possible to create a smaller dataset with all 54 languages and much fewer tweets per language, but we feel that this is the lower limit to be useful for training LID deep neural systems. The medium scale dataset keeps 22 of the 28 languages of the small-scale one, but has 10 times as many tweets per language. Each language has 70,000 training tweets, 30,000 validation tweets, and 30,000 testing tweets, for a total of 2,860,000 tweets. For the large-scale dataset we again increased tenfold the number of tweets per language, and kept the 14 languages that had sufficient tweets in our initial 900 million tweet corpus. Each language thus has 700,000 tweets in its training set, 300,000 tweets in its validation set, and 300,000 tweets in its testing set, for a total 18,200,000 tweets. Referring to Table I, this dataset includes every language that represents 0.1% or more of the Twitter corpus.

One noteworthy consequence of creating a balanced dataset, as opposed to maintaining the language distribution of Table I, is that the priors learned from our dataset will be different form those observed on Twitter. That is deliberate: the aim of our dataset is to be used to train and test microblog LID systems independently of the source (Facebook, chat rooms, etc.), and an imbalanced dataset would bias the trained systems to expect the language distribution of Twitter and hinder their performances on other sources that have different language distributions. The neutrality of a balanced dataset is preferable in these circumstances.

To verify the quality of our automated language labeling, we randomly selected 100 tweets labeled in each of the languages the co-authors are familiar with, namely English, French, Viet-

namese, German, Spanish, and Arabic. We manually read and classified these 600 tweets into one of four categories: (C1) tweets written in the labeled language; (C2) tweets mixing the labeled language and another language; (C3) mislabeled tweets written in another language; and (C4) tweets that are gibberish. The results are found in Table II. We find that, in each case, between 84 and 99 tweets are indeed written in the labeled language, and the others are either mixed languages or gibberish (most frequently long lists of usernames). Most notably, we found no case of mislabeled tweets in our random sample.

C. Comparison to Existing Twitter Datasets

To be sure, ours are not the first Twitter datasets created for the task of LID.

Few balanced LID datasets exist. There is the dataset of [6], which covers German, English, Spanish, French, Italian, and Dutch, and contains about 1,500 tweets per language. It was built by selecting six monolingual users per language, and manually validating the content. The dataset of [4] is likewise balanced at 1,000 tweets per language for Dutch, English, French, German, and Spanish. It was built by selecting users geolocalized in regions where each language was dominant, and manually validating their tweets. Finally, one of the datasets of [5] is a balanced dataset of 65 languages, assembled by randomly sampling Twitter users over one month, and identifying the language by majority vote of five different LID software.

The largest imbalanced LID dataset available is the one of [7], at almost 98 million tweets in 53 languages. It is built by sampling Twitter users over two months, and automatically labeling the language using LID software. Other imbalanced datasets do use manual labeling to validate the language labels. This is notably the case of [8], [9], and [12]. However their datasets are necessarily samller, due to the increased labeling workload; they feature 10,000 tweets in three dialects, 18,000 tweets in nine languages, and 35,000 tweets in 6 languages, respectively. Finally, Twitter released their own Twitter70 LID dataset[4]. The dataset is composed of 120,575 tweets in 70 languages sampled during one month and manually labeled. The dataset is imbalanced and reflects the distribution of languages on Twitter, with English and Japanese accounting for more than half of tweets while a dozen languages have only one tweet each. Moreover, Twitter only made the tweet IDs available, and many tweets have become unavailable since the corpus was assembled in 2014: [2] report being only able to download 82% of the tweets in 2016, while [8] could only retrieve 62% of the tweets the following year.

It thus appears that our balanced datasets have a considerably larger number of tweets than all of the other datasets save [7]. Moreover, our sampling period of one year is much longer than the one month typically used (only [8] has a longer sampling period) and we do not use select users on geolocalization. This is important for guaranteeing a variety in the content of the messages; a corpus sampled over a short period or a small geolocalized may be dominated by messages on a single locally-popular topic. Finally, our datasets are balanced, which is rare in language identification datasets, and unique for a dataset of this size.

[4]https://blog.twitter.com/engineering/en_us/a/2015/evaluating-language-identification-performance.html

IV. Proposed Model

Since many languages have unclear word boundaries, character n-grams, rather than words, have become widely used as input in LID systems [3], [6], [7], [11]. The LID problem can be stated as: given a tweet tw consisting of n ordered characters ($tw = [ch_1, ch_2, ..., ch_n]$) selected within the vocabulary set $char$ of V unique characters ($char = \{ch_1, ch_2, ..., ch_V\}$) and a set l of L languages ($l = \{l_1, l_2, ..., l_L\}$) , predict the language $\hat{l}$ present in tw using a classifier:

$$\hat{l} = argmax_{l_i \in l} Score(l_i|tw), \tag{1}$$

where $Score(l_i|tw)$ quantifies how likely it is that l_i was used given the observed message tw.

Most statistical LID systems follow the model of [3]. They start off by representing each character ch_i as a one-hot vector $\mathbf{x}_i^{oh} \in \mathbb{Z}_2^V$ according to the index of this character in $char$. This transforms tw into a matrix $\mathbf{X}^{oh}$:

$$\mathbf{X}^{oh} = [\mathbf{x}_1^{oh}, \mathbf{x}_2^{oh}, ..., \mathbf{x}_n^{oh}] \in \mathbb{Z}_2^{V \times n}$$
$$\text{where } \mathbf{x}_i{}^{oh}[j] = \begin{cases} 1, & \text{if } ch_i = char_j \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The vector $\mathbf{X}^{oh}$ is passed to a feature extraction function, for example row-wise sum or tf-idf weighting, to obtain a feature vector $\mathbf{h}$, which is fed to a classifier model for either discriminative scoring (e.g. Support Vector Machine) or generative scoring (e.g. Naïve Bayes).

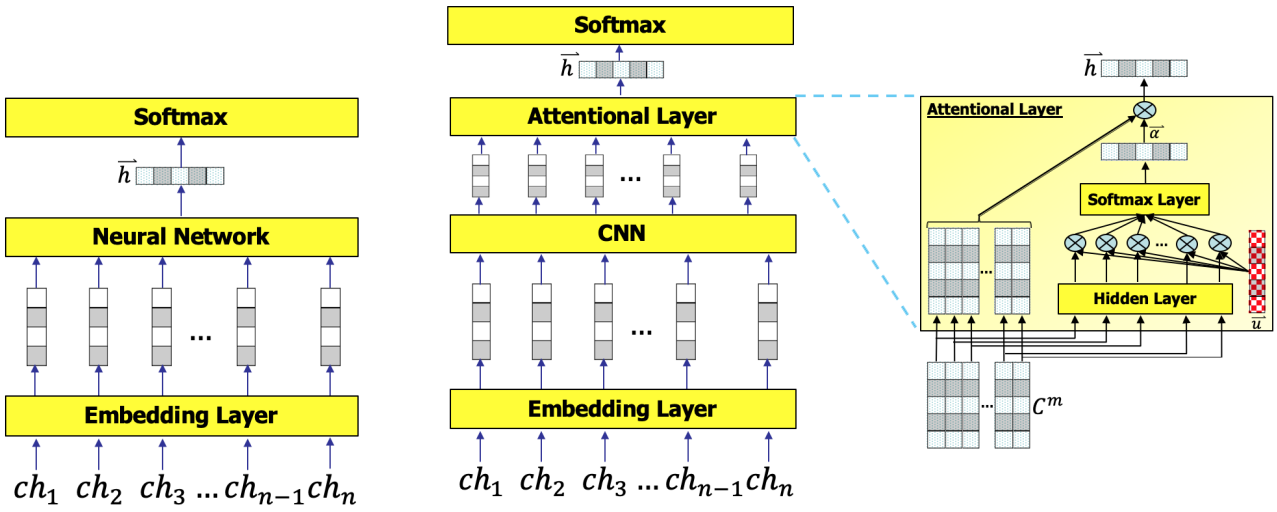
A typical neural network LID system, as illustrated in Figure 1a, first passes the input through an embedding layer to map each character $ch_i \in tw$ to a low dimensional and dense vector $\mathbf{x}_i \in \mathbb{R}^d$, where d denotes the dimension of character embedding. This gives an embedded matrix:

$$\mathbf{X} = [\ \mathbf{x}_1\ \mathbf{x}_2\ ...\mathbf{x}_n\ ] \in \mathbb{R}^{d \times n}, \tag{3}$$

Matrix $\mathbf{X}$ is then fed through a neural network, which transforms it into an output vector $\mathbf{h} = f(\mathbf{X})$ of length L that represents the likelihood of each language in l, and which is passed through a $Softmax$ function. This updates Equation 1 as:

$$\hat{l} = argmax_{l_i \in l} Softmax_i(\mathbf{h}) \tag{4}$$

Tweets are noisy messages that can contain a mix of multiple languages. To deal with this challenge, most previous neural network LID systems use deep sequence layers, such as an encoder-decoder [7] or a GRU [11], to extract global representations at a high computational cost. By contrast, we employ a shallow (single-layer) CNN to locally learn region-based features along with an attention mechanism to proportionally merge together these local features for an entire tweet tw. The attention mechanism will efficiently capture



(a) Neural network LID baseline.

(b) Our attentional-based CNN model.

Figure 1: Neural network classifier architectures.

which local features of a particular language are the dominant features of the tweet. There are two major advantages to our architecture: the CNN has very few parameters, which simplifies the model and decreases the inference latency, while the attention mechanism makes it possible to model the mix of languages without affecting performance.

A. *ngam-regional CNN Model*

To begin, we present a traditional CNN with an ngam-regional constraint. The convolution operation of a filter with a region size m is parameterized by a weight matrix $\mathbf{W}_{cnn} \in \mathbb{R}^{d_{cnn} \times md}$ and a bias vector $\mathbf{b}_{cnn} \in \mathbb{R}^{d_{cnn}}$, where d_{cnn} is the dimension of the CNN. The inputs are a sequence of m consecutive input columns in $\mathbf{X}$, represented by a concatenated vector $\mathbf{X}[i:i+m-1] \in \mathbb{R}^{md}$. The region-based feature vector $\mathbf{c}_i$ is computed as:

$$\begin{aligned} \mathbf{X}[i:i+m-1] &= \mathbf{x}_i \oplus ... \oplus \mathbf{x}_{i+m-1}, \\ \mathbf{c}_i &= g(\mathbf{W}_{cnn} \cdot \mathbf{X}[i:i+m-1] + \mathbf{b}_{cnn}), \end{aligned} \tag{5}$$

where $\oplus$ denotes a concatenation operation and g is a non-linear function. The region filter is slid from the beginning to the end of $\mathbf{X}$ to obtain a convolution matrix $\mathbf{C}$:

$$\mathbf{C} = [\mathbf{c}_1, ..., \mathbf{c}_{n-m+1}] \in \mathbb{R}^{d_{cnn} \times (n-m+1)}. \tag{6}$$

We add a zero-padding constrain at both sides of $\mathbf{X}$ to ensure that the number of columns in $\mathbf{C}$ is equal to the number of columns in $\mathbf{X}$. Consequently, each $\mathbf{c}_i$ feature vector corresponds to an $\mathbf{x}_i$ input vector at the same index position i, and is learned from concatenating the surrounding m-gram embeddings. Particularly:

$$\begin{aligned} 2p + (n - m + 1) &= n, \\ p &= \frac{m-1}{2}, \end{aligned} \tag{7}$$

where p is the number of zero-padding columns. Finally, in a normal CNN, a row-wise max-pooling function is applied on $\mathbf{C}$ to extract the d_{cnn} most salient features, as shown in Equation 8. However, one weakness of this approach is that it extracts the most salient features out of sequence.

$$\begin{aligned} \mathbf{h} &= f_{CNN}(\mathbf{X}) \\ &= pooling_{max}(\mathbf{C}) \in \mathbb{R}^{d_{cnn}} \end{aligned} \tag{8}$$

B. *Attention Mechanism*

Instead of the traditional pooling functions of Equation 8, our CNN uses an attention mechanism to model the interaction between region-based features from the beginning to the end of an input, as shown in Figure 1b. The sequence of regional feature vectors $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_n]$ computed in Equation 6 passes through a fully-connected hidden layer to learn a sequence of regional hidden vectors $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n] \in \mathbb{R}^{d_{hd} \times n}$ using:

$$\mathbf{h}_i = g_2(\mathbf{W}_{hd} \cdot \mathbf{c}_i + \mathbf{b}_{hd}) \in \mathbb{R}^{d_{hd}}, \tag{9}$$

where g_2 is a non-linear activation function, $\mathbf{W}_{hd}$ and $\mathbf{b}_{hd}$ denote model parameters, and d_{hd} is the dimension of the hidden layer. Inspired by attention structures [16]–[18] which are widely applied on sequence models such as LSTM and Encoder-Decoder, we employ a regional context vector $\mathbf{u} \in \mathbb{R}^{d_{hd}}$ to measure the importance of each window-based hidden vector as:

$$\mathbf{t} = \mathbf{u} \times \mathbf{H} \in \mathbb{R}^n. \tag{10}$$

The importance factors are then fed to a *Softmax* layer to obtain the normalized weights:

$$\alpha_i = \frac{\mathbf{t}_i}{\sum_{i'} \mathbf{t}_{i'}}. \tag{11}$$

Parameter	our CNN	our att CNN
d	50	50
g	$relu$	$relu$
g_2	$n.a.$	$relu$
d_{cnn}	100	100
m	5	5
p	2	2
d_{hd}	$n.a.$	100
lr	0.001	0.001
$decay\_rate$	0.05	0.05
max_epochs	512	512
$patience$	64	64
$clip\_rate$	5	5

Table III: Parameter settings

The final representation of an input is computed by a weighted sum of its regional feature vectors:

$$\mathbf{h} = \sum_i \alpha_i \mathbf{c}_i. \tag{12}$$

While other NLP systems have used attention models and architectures similar to ours [16]–[18], it is worth noting that our work is different from the literature on three points. First, we take multilingual character embeddings as inputs while other models' inputs are word and sentence embeddings [16]. Second, while other approaches rely on max pooling to extract features [17], [18], we use the output of the attention layer directly. Finally, other authors tackled other sub-problems in NLP, such as question-answering [17] and sentiment analysis [18], while ours is the first to use this architecture for LID.

V. Experimental Results

A. Benchmarks

We selected five LID benchmark systems. We picked the langid.py library which is frequently used to compare systems in the literature. We selected two neural network systems from the literature, specifically the encoder-decoder EquiLID system of [7] and the GRU neural network LanideNN system of [11]. In addition, we included CLD2[5] and CLD3[6], which are two implementations of the Naïve Bayes LID and the feedforward neural network LID, respectively. Both software packages are deployed in the Chrome web browser [1], [5], [9] and sometimes used as a benchmark in the LID literature [3], [7]–[9], [11]. We obtained publicly-available implementations of each of these algorithms, and used their pre-trained and optimized versions and parameters, which are supposed to give optimal LID results for each system out of the box. We tested them on our three datasets. In Table IV, we report each algorithm's accuracy and F1 score, the two metrics usually reported in the LID literature. We also include precision and recall values for more details. And we include the speed in number of messages handled per second. This metric is not often discussed in the LID literature, but is of particular importance when dealing with a massive dataset such as ours or a massive streaming source such as Twitter.

[5]https://github.com/CLD2Owners/cld2

[6]https://github.com/google/cld3

We compare these benchmarks to our two models: the basic CNN and the CNN with an attention mechanism. They are labeled *CNN* and *Attention CNN* in Table IV. In both models, we filter out characters that appear less than 5 times and apply a dropout rate of 0.5. ADAM optimization and early stopping are employed during training. The parameters are listed in Table III. This configuration was randomly selected; it takes four days to train our system, making a parameter optimization step impractical.

B. Analysis

The first thing that appears from these results is the speed difference between algorithms. CLD3 and langid.py can process several thousands of messages per second and CLD2 is an order of magnitude better, but the two neural network software have considerably worse performances, at less than a dozen messages per second. This is the efficiency trade-off of neural-network LID systems we mentioned in Section 1; although we should note that those two systems are research prototypes and may not have been optimized.

In terms of accuracy and F1 score, langid.py, LanideNN, and EquiLID have very similar performances. All three consistently score above 0.90, and each achieves the best accuracy or F1 score at some point. By contrast, CLD2 and CLD3 have weaker performances; significantly so in the case of CLD3. Using our small, medium, or large-scale test set does not significantly affect these results.

The last two lines of Table IV report the results of our basic CNN and our attention CNN. It can be seen that both of them outperform the benchmark systems in accuracy, precision, recall, and F1 score in all experiments. Moreover, the attention CNN outperforms the basic CNN in every metric, and while the gains may be small (less than 1% at times), combined with the other benefits of the attention mechanism which we will explore in the next subsection they make it the preferred version of our system. In terms of processing speed, only the CLD2 system surpasses ours, but it does so at the cost of a 10% drop in accuracy and F1 score. Looking at the choice of datasets, it can be seen that training our systems with either the small, medium, or large-scale dataset yields roughly the same performance.

C. Impact of Attention Mechanism

We can illustrate the impact of our attention mechanism by displaying the importance factor α_i of equation 11 corresponding to each character ch_i in selected tweets. Table V shows a set of tweets that were correctly identified by the attention CNN but misclassified by the regular CNN in three different languages: English, French, and Vietnamese. The color intensity of a letter's cell is proportional to the attention mechanism's normalized weight α_i, or the focus the network puts on that character. In order words, the attention CNN puts more importance on the features that have the darkest color.

The case studies of Table V show the noise-tolerance that comes from the attention mechanism. It can be seen that the system puts virtually no weight on URL links (tw_{en_1},

Model	Small-scale dataset					Medium-scale dataset					Large-scale dataset				
	Acc	P	R	F1	Speed	Acc	P	R	F1	Speed	Acc	P	R	F1	Speed
langid.py	0.9229	0.9290	0.9229	0.9240	3710.96	0.9449	0.9475	0.9449	0.9454	3797.34	0.9483	0.9502	0.9483	0.9486	4630.59
CLD2	0.8670	0.9624	0.8670	0.8997	**43308.31**	0.8784	0.9638	0.8784	0.9067	**40287.01**	0.8711	0.9527	0.8711	0.8952	**42297.95**
CLD3	0.7284	0.8686	0.7284	0.7456	5911.94	0.7131	0.8792	0.7131	0.7333	6265.21	0.6976	0.9133	0.6976	0.7326	6139.86
LanideNN	0.9304	0.9052	0.8984	0.9003	11.47	0.9414	0.9064	0.9005	0.9021	9.38	0.9370	0.8811	0.8745	0.8763	7.08
EquiLID	0.9244	0.9516	0.9244	0.9325	7.53	0.9430	0.9616	0.9430	0.9484	7.64	0.9489	0.9648	0.9489	0.9532	7.05
CNN	0.9675	0.9677	0.9675	0.9675	39562.69	0.9832	0.9834	0.9832	0.9832	38427.42	0.9866	0.9867	0.9866	0.9866	31647.77
Attention CNN	**0.9712**	**0.9716**	**0.9712**	**0.9713**	31782.44	**0.9841**	**0.9842**	**0.9841**	**0.9842**	24828.28	**0.9915**	**0.9915**	**0.9915**	**0.9915**	35266.82

Table IV: Benchmarking results.

IDs	Tweets with attention values	att. CNN	CNN
tw_{en_1}	f l o b o t s l i v e **i** n D e n v e r ! h t t p s : / / t . c o / Z B s S K t 8 V i b	en	de
tw_{en_2}	@ M a r i o n B u r r 5 @ S t a r s e e d 5 2 @ A m y _ S i s k l n d D o n a l d T r u m p I m p e a c h m e **n** t .	en	de
tw_{en_3}	R T @ R e a l K o b e H a t c h : C o u s **i** n C e c e # b e s t t a l e n t i n o u r h o u s e # m o t h e r o f n o **n** e # m a k e k i d s g e t t h e j o b d o **n** e h t t p s : / / t . c o / F t l i N x n 8 N X	en	fr
tw_{fr_1}	@ F l o B o u z i n a c A u s e r v **l** c e d **e** s a u t r e s . . .	fr	de
tw_{fr_2}	**l** e r o y a l b l o g . . . d u s i e **c** l **e** . . . i n t e r a c t i f h t t p s : / / t . c o / W K T O Z c o o a u	fr	ro
tw_{fr_3}	R T @ v i c k y _ s a l l a : **L** e r é a l M a d r i d **e** s t **l** e p l u s g r a n d c l u b a u m o n **d** e . H a l a m a d r i d **y** n a d a m a s .	fr	es
tw_{vi_1}	@ h o a n g t h u y s e o h a y q **u** á	vi	es
tw_{vi_2}	D o n ' t c r y J o n n y (S h e e t b **à** i h á t) h t t p s : / / t . c o / o Q d C W M c L w g	vi	hu
tw_{vi_3}	[V I D C U T] C h ú c m **ừ** n g s i n h n h **ậ** t K i m T a e h y u n g ! ! ! & l t ; 3 # 생일축하해 # 태형생일 H a v e a n i c e d a y ! J u s t e n j o y y o u r l i f e i n y o u r . . . h t t p s : / / t . c o / o B g g E c L Y 1 1	vi	ko

Table V: Tweets misclassified by the CNN but recognized by the Attention CNN

tw_{fr_2}, tw_{vi_2}), on hashtags (tw_{en_3}), or on usernames (tw_{en_2}, tw_{fr_1}, tw_{vi_1}). We should emphasize that our system does not implement any text preprocessing steps to filter these elements out; the input tweets are kept as-is. Despite that, the network learned to distinguish between words and non-words, and to focus mainly on the former. In fact, when the network does put attention on non-word elements, it is when they appear to use real words ("star" and "seed" in the username of tw_{en_2}, "mother" and "none" in the hashtag of tw_{en_3}). This also illustrates that the attention mechanism can pick out fine-grained features within noisy text, such as the real-word components of longer non-word strings.

The examples of Table V also show that the attention CNN learns to focus on common words to recognize languages. Some of the highest-weighted characters in the example tweets are found in common determiners, adverbs, and verbs of each language. These include "**i**n" (tw_{en_1}), "d**e**s" (tw_{fr_1}), "**l**e" (tw_{fr_2}), "**e**st" (tw_{fr_3}), "q**u**á" (tw_{vi_2}), and "nh**ậ**t" (tw_{vi_3}).

Finally, when multiple languages are found within a tweet, the attention network spots all of them. For example, tw_{fr_3} switches from French to Spanish and tw_{vi_2} mixes both English and Vietnamese. In both cases, the network identifies features of both languages; it focuses strongly on "**e**st" and "**y**" in tw_{fr_3}, and on "Don**'**t" and "b**à**i" in tw_{vi_2}. The message of tw_{vi_3} mixes Vietnamese, English, and Korean, and the network focuses on all three parts, by picking out "nh**ậ**t" and "m**ừ**ng" in Vietnamese, "#생일**축**하해" and "#태형생**일**" in Korean, and "**h***ave*" in English. Since our system is setup to classify each tweet into a single language, the strongest feature of each tweet wins out and the message is classified in the corresponding language. But since features of all languages are picked out, a future version of our system could identify multilingual messages and decompose them into segments of each language.

VI. Conclusion

In this paper, we first demonstrated how to build balanced, automatically-labeled, and massive LID datasets. These

datasets are taken from Twitter, and are thus composed of real-world and noisy messages. We applied our technique to build three datasets, ranging from hundreds of thousands to tens of millions of messages in between 14 and 28 languages. We have made these datasets available online for other researchers, and our next step will be to expand the manual validation of Table II to more messages and languages using crowdsourced labelers. Next, we proposed our new neural LID system, a CNN-based network with an attention mechanism, which overcomes the speed limitation of other neural LID systems while maintaining a state-of-the-art accuracy. The results obtained by our system surpass five benchmark LID systems by 5% to 10%. Moreover, our analysis of the attention mechanism shed some light on the inner workings of the typically-black-box neural network, and demonstrated how it picks out the most important linguistic features of messages while ignoring noise. All of our datasets and source code are publicly available at https://github.com/duytinvo/LID_NN.

ACKNOWLEDGMENT

This research was made possible by the financial, material, and technical support of Two Hat Security Research Corp., and the financial support of the Canadian research funding agency MITACS.

REFERENCES

[1] T. Jauhiainen, M. Lui, M. Zampieri, T. Baldwin, and K. Lindén, "Automatic language identification in texts: A survey," *arXiv preprint arXiv:1804.08186*, 2018.

[2] A. Jaech, G. Mulcaire, S. Hathi, M. Ostendorf, and N. A. Smith, "Hierarchical character-word models for language identification," *arXiv preprint arXiv:1608.03030*, 2016.

[3] M. Lui and T. Baldwin, "langid. py: An off-the-shelf language identification tool," in *Proceedings of the ACL 2012 system demonstrations*. Association for Computational Linguistics, 2012, pp. 25–30.

[4] S. Carter, W. Weerkamp, and M. Tsagkias, "Microblog language identification: Overcoming the limitations of short, unedited and idiomatic text," *Language Resources and Evaluation*, vol. 47, no. 1, pp. 195–215, 2013.

[5] M. Lui and T. Baldwin, "Accurate language identification of twitter messages," in *Proceedings of the 5th workshop on language analysis for social media (LASM)*, 2014, pp. 17–25.

[6] E. Tromp and M. Pechenizkiy, "Graph-based n-gram language identification on short texts," in *Proc. 20th Machine Learning conference of Belgium and The Netherlands*, 2011, pp. 27–34.

[7] D. Jurgens, Y. Tsvetkov, and D. Jurafsky, "Incorporating dialectal variability for socially equitable language identification," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, vol. 2, 2017, pp. 51–57.

[8] S. L. Blodgett, J. Wei, and B. O'Connor, "A dataset and classifier for recognizing social media english," in *Proceedings of the 3rd Workshop on Noisy User-generated Text*, 2017, pp. 56–61.

[9] S. Bergsma, P. McNamee, M. Bagdouri, C. Fink, and T. Wilson, "Language identification for creating language-specific twitter collections," in *Proceedings of the second workshop on language in social media*. Association for Computational Linguistics, 2012, pp. 65–74.

[10] P. Gamallo, M. Garcia, S. Sotelo, and J. R. P. Campos, "Comparing ranking-based and naive bayes approaches to language detection on tweets." in *TweetLID@ SEPLN*, 2014, pp. 12–16.

[11] T. Kocmi and O. Bojar, "Lanidenn: Multilingual language identification on character window," *arXiv preprint arXiv:1701.03338*, 2017.

[12] A. Zubiaga, I. San Vicente, P. Gamallo, J. R. P. Campos, I. A. Loinaz, N. Aranberri, A. Ezeiza, and V. Fresno-Fernández, "Overview of tweetlid: Tweet language identification at sepln 2014." in *TweetLID@ SEPLN*, 2014, pp. 1–11.

[13] L. F. Hurtado, F. Pla, M. Giménez, and E. S. Arnal, "Elirf-upv en tweetlid: Identificación del idioma en twitter." in *TweetLID@ SEPLN*, 2014, pp. 35–38.

[14] M. Zampieri and B. G. Gebre, "Automatic identification of language varieties: The case of portuguese," in *KONVENS2012-The 11th Conference on Natural Language Processing*. Österreichischen Gesellschaft für Artificial Intelligende (ÖGAI), 2012, pp. 233–237.

[15] S. Malmasi, M. Zampieri, N. Ljubešić, P. Nakov, A. Ali, and J. Tiedemann, "Discriminating between similar languages and arabic dialect identification: A report on the third dsl shared task," in *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, 2016, pp. 1–14.

[16] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2016, pp. 1480–1489. [Online]. Available: http://aclweb.org/anthology/N16-1174

[17] W. Yin, M. Yu, B. Xiang, B. Zhou, and H. Schütze, "Simple question answering by attentive convolutional neural network," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, Dec. 2016, pp. 1746–1756. [Online]. Available: https://www.aclweb.org/anthology/C16-1164

[18] B. Shin, T. Lee, and J. D. Choi, "Lexicon integrated CNN models with attention for sentiment analysis," in *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 149–158. [Online]. Available: https://www.aclweb.org/anthology/W17-5220