Hierarchical Character-Word Models for Language Identification

Aaron Jaech[1] **George Mulcaire**[2] **Shobhit Hathi**[2] **Mari Ostendorf**[1] **Noah A. Smith**[2]
[1]Electrical Engineering [2]Computer Science & Engineering
University of Washington, Seattle, WA 98195, USA
ajaech@uw.edu, gmulc@uw.edu, shathi@uw.edu
ostendor@uw.edu, nasmith@cs.washington.edu

Abstract

Social media messages' brevity and unconventional spelling pose a challenge to language identification. We introduce a hierarchical model that learns character and contextualized word-level representations for language identification. Our method performs well against strong baselines, and can also reveal code-switching.

1 Introduction

Language identification (language ID), despite being described as a solved problem more than ten years ago (McNamee, 2005), remains a difficult problem. Particularly when working with short texts, informal styles, or closely related language pairs, it is an active area of research (Gella et al., 2014; Wang et al., 2015; Baldwin and Lui, 2010). These difficult cases are often found in social media content. Progress on language ID is needed especially since downstream tasks, like translation and semantic parsing, depend on correct language ID.

This paper brings continuous representations for language data, which have produced new states of the art for language modeling (Mikolov et al., 2010), machine translation (Bahdanau et al., 2015), and other tasks, to language ID. We adapt a hierarchical character-word neural architecture from Kim et al. (2016), demonstrating that it works well for language ID. Our model, which we call C2V2L ("character to vector to language") is hierarchical in the sense that it explicitly builds a continuous representation for each word from its character sequence, capturing orthographic and morphology-related patterns, and then combines those word level representations in context, finally classifying the full word sequence. Our model does not require any special handling of casing or punctuation nor do we need to remove URLs, usernames, or hashtags, and it is trained end-to-end using standard procedures.

We demonstrate the model's state-of-the-art performance in experiments on two datasets consisting of tweets. This hierarchical technique works well compared to classifiers using character or word n-gram features as well as a similar neural model that treats an entire tweet as a single character sequence. We find further that the model can benefit from additional out-of-domain data, unlike much previous work, and with little modification can annotate word-level code-switching. We also confirm that smoothed character n-gram language models perform very well for language ID tasks.

2 Model

Our model has two main components trained together, end-to-end.[1] The first, "char2vec," applies a convolutional neural network (CNN) to a whitespace-delimited word's Unicode character sequence, providing a word vector.[2] The second is a bidirectional LSTM recurrent neural network (RNN) that maps a sequence of such word vectors to a language label.

2.1 Char2vec

The first layer of char2vec is an embedding learned for each Unicode code point that appears at least twice in the training data, including punctuation, emoji, and other symbols. If C is the set of characters then we let the size of the character embed-

[1] Code available here: http://github.com/ajaech/twitter_langid

[2] For languages without word segmentation, e.g., Chinese, the entire character sequence is treated as a single word. This still works well (see Section 3.2.)

Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media, pages 84–93,
Austin, TX, November 1, 2016. ©2016 Association for Computational Linguistics

ding layer be $d = \lceil \log_2 |C| \rceil$. (If each dimension of the character embedding vector holds one bit of information then d bits should be enough to uniquely encode each character.) The character embedding matrix is $\mathbf{Q} \in \mathbb{R}^{d \times |C|}$. Words are given to the model as a sequence of characters. When each character in a word of length l is replaced by its embedding vector we get a matrix $\mathbf{C} \in \mathbb{R}^{d \times (l+2)}$. There are $l + 2$ columns in C because padding characters are added to the left and right of each word.

The char2vec architecture uses two sets of filter banks. The first set is comprised of matrices $\mathbf{H}_{a_i} \in \mathbb{R}^{d \times 3}$ where i ranges from 1 to n_1. The matrix $\mathbf{C}$ is narrowly convolved with each $\mathbf{H}_{a_i}$, a bias term b_a is added and an ReLU non-linearity, $\text{ReLU}(x) = \max(0, x)$, is applied to produce an output $\mathbf{T}_1 = \text{ReLU}(\text{conv}(\mathbf{C}, \mathbf{H}_a) + \mathbf{b}_a)$. $\mathbf{T}_1$ is of size $n_1 \times l$ with one row for each of the filters and one column for each of the characters in the input word. Since each of the $\mathbf{H}_{a_i}$ is a filter with a width of three characters, the columns of $\mathbf{T}_1$ each hold a representation of a character trigram. During training, we apply dropout on $\mathbf{T}_1$ to regularize the model. The matrix $\mathbf{T}_1$ is then convolved with a second set of filters $\mathbf{H}_{b_i} \in \mathbb{R}^{n_1 \times w}$ where b_i ranges from 1 to $3n_2$ and n_2 controls the number of filters of each of the possible widths, $w = 3, 4$, or 5. Another convolution and ReLU non-linearity is applied to get $\mathbf{T}_2 = \text{ReLU}(\text{conv}(\mathbf{T}_1, \mathbf{H}_b) + \mathbf{b}_b)$. Max-pooling across time is used to create a fix-sized vector $\mathbf{y}$ from $\mathbf{T}_2$. The dimension of $\mathbf{y}$ is $3n_2$, corresponding to the number of filters used.

Similar to Kim et al. (2016) who use a highway network after the max-pooling layer, we apply a residual network layer. Both highway and residual network layers allow values from the previous layer to pass through unchanged but the residual layer is preferred in our case because it uses half as many parameters (He et al., 2015). The residual network uses a matrix $\mathbf{W} \in \mathbb{R}^{3n_2 \times 3n_2}$ and bias vector $\mathbf{b}_3$ to create the vector $\mathbf{z} = \mathbf{y} + f_R(\mathbf{y})$ where $f_R(\mathbf{y}) = \text{ReLU}(\mathbf{W}\mathbf{y} + \mathbf{b}_3)$. The resulting vector $\mathbf{z}$ is used as a word embedding vector in the word-level LSTM portion of the model.

There are three differences between our version of the model and the one described by Kim et al. (2016). First, we use two layers of convolution instead of just one, inspired by Ling et al. (2015a) who used a 2-layer LSTM for character modeling. Second, we use the ReLU function as a nonlinearity as opposed to the tanh function. ReLU has been highly successful in computer vision applications in conjunction with convolutional layers (Jarrett et al., 2009). Finally, we use a residual network layer instead of a highway network layer after the max-pooling step, to reduce the model size.

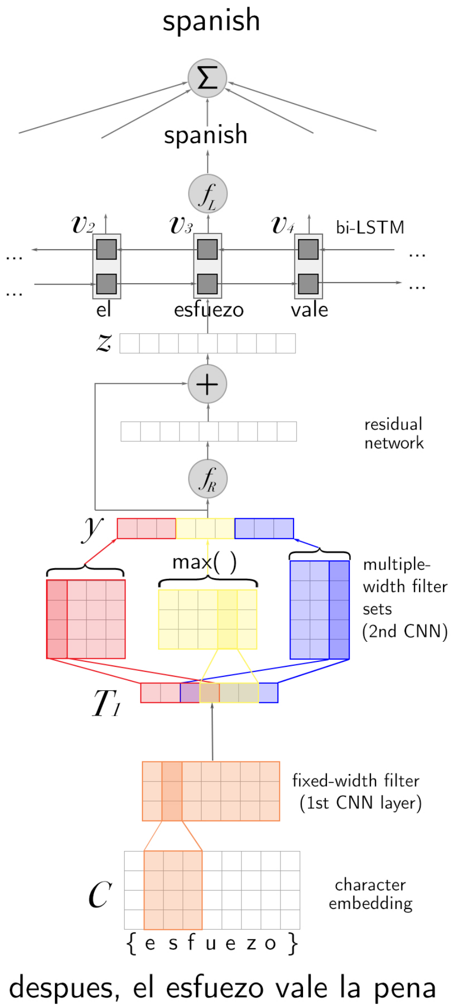

Figure 1: C2V2L model architecture. The model takes the (misspelled) word "esfuezo," and produces a word vector via the two CNN layers and the residual layer. The word vectors are then combined via the LSTM, and the words' predictions averaged for a tweet prediction.

It is possible to use bi-LSTMs instead of convolutional layers in char2vec as done by Ling et al. (2015a). We did explore this option in preliminary experiments but found that using convolutional layers has several advantages, including a large improvement in speed for both the forward and backward pass, many fewer parameters, and improved language ID accuracy.

2.2 Sentence-level Language ID

The sequence of word embedding vectors is processed by a bi-LSTM, which outputs a sequence of vectors, $[\mathbf{v}_1, \ldots \mathbf{v}_T]$ where T is the number of words in the tweet. All LSTM gates are used as defined by Sak et al. (2014). Dropout is used as a regularizer on the inputs to the LSTM, as in Pham et al. (2014). The output vectors $\mathbf{v}_i$ are transformed into probability distributions over the set of languages by applying an affine transformation followed by a softmax:

$$\mathbf{p}_i = f_L(\mathbf{v}_i) = \frac{\exp(\mathbf{A}\mathbf{v}_i + b)}{\sum_{t=1}^{T} \exp(\mathbf{A}\mathbf{v}_t + b)}$$

(These word-level predictions, we will see in §5.4, are useful for annotating code-switching.) The sentence-level prediction $\mathbf{p}_S$ is then given by averaging the word-level language predictions.

The final affine transformation can be interpreted as a language embedding, where each language is represented by a vector of the same dimensionality as the LSTM outputs. The goal of the LSTM then is (roughly) to maximize the dot product of each word's representation with the language embedding(s) for that sentence. The only supervision in the model comes from computing the loss of sentence-level predictions.

3 Tasks and Datasets

We consider two datasets: TweetLID and Twitter70. Summary statistics for each of the datasets are provided in Table 1.

3.1 TweetLID

The TweetLID dataset (Zubiaga et al., 2014) comes from a language ID shared task that focused on six commonly spoken languages of the Iberian peninsula: Spanish, Portuguese, Catalan, Galician, English, and Basque. There are approximately 15,000 tweets in the training data and 25,000 in the test set. The data is unbalanced, with the majority of examples being in the Spanish language. The "undetermined" label ('und'), comprising 1.4% of the training data, is used for tweets that use only non-linguistic tokens or belong to an outside language. Additionally, some tweets are ambiguous ('amb') among a set of languages (2.3%), or code-switch between languages (2.4%). The evaluation criteria take into account all of these factors, requiring prediction of at least one acceptable language for an ambiguous tweet or all languages present for a code-switched tweet. The fact that hundreds of tweets were labeled ambiguous or undetermined by annotators who were native speakers of these languages reveals the difficulty of this task.

For tweets labeled as ambiguous or containing multiple languages, the training objective distributes the "true" probability mass evenly across each of the languages, e.g., 50% Spanish and 50% Catalan.

The TweetLID shared task had two tracks: one that restricted participants to only use the official training data and another that was unconstrained, allowing the use of any external data. There were 12 submissions in the constrained track and 9 in the unconstrained track. Perhaps surprisingly, most participants performed worse on the unconstrained track than they did on the constrained one.

As supplementary data for our unconstrained-track experiments, we collected data from Wikipedia for each of the six languages in the TweetLID corpus. Participants in the TweetLID shared task also used Wikipedia as a data source for the unconstrained track. We split the text into 25,000 sentence fragments per language, with each fragment of length comparable to that of a tweet. The Wikipedia sentence fragments are easily distinguished from tweets. Wikipedia fragments are more formal and are more likely to use complex words; for example, one fragment reads "ring homomorphisms are identical to monomorphisms in the category of rings." In contrast, tweets tend to use variable spelling and more simple words, as in "Haaaaallelujaaaaah http://t.co/axwzUNXk06" and "@justinbieber: Love you mommy http://t.co/xEGAxBl6Cc http://t.co/749s6XKkgK awe ♡". Previous work confirms that language ID is more challenging on social media text than sentence fragments taken from more formal text, like Wikipedia (Carter, 2012). Despite the domain mismatch, we find in §5.2 that additional text in training helps our model.

The TweetLID training data is too small to divide into training and validation sets. We created a tuning set by adding samples taken from Twitter70

	TweetLID	Twitter70
Tweets	14,991	58,182
Character vocab.	956	5,796
Languages	6	70
Code-switching?	Yes	Not Labeled
Balanced?	No	Roughly

Table 1: Dataset characteristics.

and from the 2014 Workshop on Computational Approaches to Code Switching (Solorio et al., 2014) to the official TweetLID training data. We used this augmented dataset with a 4:1 train/development split for hyperparameter tuning.[3]

3.2 Twitter70

The Twitter70 dataset was published by the Twitter Language Engineering Team in November 2015.[4] The languages come from the Afroasiatic, Dravidian, Indo-European, Sino-Tibetan, and Tai-Kadai families. Each person who wants to use the data must redownload the tweets using the Twitter API. In between the time when the data was published and when it is downloaded, some of the tweets can be lost due to account deletion or changes in privacy settings. At the time when the data was published there were approximately 1,500 tweets for each language. We were able to download 82% of the tweets but the amount we could access varied by language with as many as 1,569 examples for Sindhi and as few as 371 and 39 examples for Uyghur and Oriya, respectively. The median number of tweets per language was 1,083. To our knowledge, there are no published benchmarks on this dataset.

Unlike TweetLID, the Twitter70 data has no unknown or ambiguous labels. Some tweets do contain code-switching but it is not labeled as such; a single language is assigned. There is no predefined test set so we used the last digit of the identification number to partition them. Identifiers ending in zero (15%) were used for the test set and those ending in one (5%) were used for tuning.

When processing the input at the character level, the vocabulary for each data source is defined as the set of Unicode code-points that occur at least twice in the training data: 956 and 5,796 characters for TweetLID and Twitter70, respectively. A small number of languages, e.g. Mandarin, are responsible for most characters in the Twitter70 vocabulary.

Gillick et al. (2016) processed the input one byte at a time instead of by character. In early experiments, we found that when using bytes the model would often make mistakes that should have been obvious from the orthography alone. We do not recommend using the byte sequence for language ID.

4 Implementation Details

4.1 Preprocessing

An advantage of the hybrid character-word model is that only limited preprocessing is required. The runtime of training char2vec is proportional to the longest word in a minibatch. The data contains many long and repetitive character sequences such as "hahahaha..." or "arghhhhh...". To deal with these, we restricted any sequence of repeating characters to at most five repetitions where the repeating pattern can be from one to four characters. There are many tweets that string together large numbers of Twitter usernames or hashtags without spaces between them. These create extra long "words" that cause our implementation to need more memory and computation during training. To solve this we enforce the constraint that there must be a space before any URL, username, or hashtag. To deal with the few remaining extra-long character sequences, we force word breaks in non-space character sequences every 40 bytes. This primarily affects languages that are not space-delimited like Chinese. We do not perform any special handling of casing or punctuation nor do we need to remove the URLs, usernames, or hashtags as has been done in previous work (Zubiaga et al., 2014). The same preprocessing is used when training the n-gram models.

4.2 Training and Tuning

Training is done using minibatches of size 25 and a learning rate of 0.001 using the Adam method for

[3] We used this augmented data to tune hyperparameters for both constrained and unconstrained models. However, after setting hyperparameters, we trained our constrained model using only the official training data, and the unconstrained model using only the training data + Wikipedia. Thus, no extra data was used to learn actual model parameters for the constrained case.

[4] For clarity, we refer to this data as "Twitter70" but it can be found in the Twitter blog post under the name "recall oriented." See `http://t.co/EOVqA0t79j`

Parameter	TweetLID	Twitter70
1st Conv. Layer (n_1)	50	59
2nd Conv. Layer (n_2)	93	108
LSTM	23	38
Dropout	25%	30%
Total Params.	193K	346K

Table 2: Hyperparameter settings for selected models.

optimization (Kingma and Ba, 2015). For the Twitter70 dataset we used 5% held out data for tuning and 15% for evaluation. To tune, we trained 15 models with random hyperparameters and selected the one that performed the best on the development set. Training is done for 80,000 and 100,000 minibatches for TweetLID and Twitter70 respectively.

The only hyperparameters to tune are the number of filters in each of the two convolutional layers, the size of the word-level LSTM vector, and the dropout rate. The selected values are listed in Table 2.

5 Experiments

For all the studies below on language identification, we compare to two baselines: i) `langid.py`, a popular open-source language ID package, and ii) a classifier using n-gram character language models. For the TweetLID dataset, additional comparisons are included as described next. In addition, we test our model's word-level performance on a code-switching dataset.

The first baseline, based on the `langid.py` package, uses a naïve Bayes classifier over *byte* n-gram features (Lui and Baldwin, 2012). The pretrained model distributed with the package is designed to perform well on a wide range of domains, and achieved high performance on "microblog messages" (tweets) in the original paper. `langid.py` uses feature selection for domain adaptation and to reduce the model size; thus, retraining it on in-domain data as we do in this paper does not provide an entirely fair comparison. However, we include it for its popularity and importance.

The second baseline is built from character n-gram language models. It assigns each tweet according to language $\ell^* = \arg\max_\ell p(\text{tweet} \mid \ell)$, i.e., applying Bayes' rule with a uniform class prior (Dunning, 1994). For TweetLID, the rare 'und' was handled with a rejection model. Specifically, after ℓ^* is chosen, a log likelihood ratio test is applied to decide whether to reject the decision in favor of the 'und' class, using the language models for ℓ^* and 'und' with a threshold chosen to optimize F_1 on the development set. The models were trained using Witten-Bell smoothing (Bell et al., 1989), but otherwise the default parameters of the SRILM toolkit (Stolcke, 2002) were used.[5] N-gram model training ignores tweets labeled as ambiguous or containing multiple languages, and the unconstrained models use a simple interpolation of TweetLID and Wikipedia component models. The n-gram order was chosen to minimize perplexity with 5-fold cross validation, yielding n=5 for TweetLID and Twitter70, and n=6 for Wikipedia.

Note that both of these baselines are generative, learning separate models for each language. In contrast, the neural network models explored here are trained on all languages, so parameters may be shared across languages. In particular, a character sequence corresponding to a word in more than one language (e.g., "no" in English and Portuguese) has a language-independent word embedding.

5.1 TweetLID: Constrained Track

In the constrained track of the 2014 shared task, Hurtado et al. (2014) attained the highest performance (75.2 macroaveraged F_1). They used a set of one-vs-all SVM classifiers with character n-gram features, and returned all languages for which the classification confidence was above a fixed threshold. This provides our third, strongest baseline.

In the unconstrained track, the winning team was Gamallo et al. (2014), using a naïve Bayes classifier on word unigrams. They incorporated Wikipedia text to train their model, and were the only team in the competition whose unconstrained model outperformed their constrained one. We compare to their constrained-track result here.

We also consider a version of our model, "C2L," which uses only the char2vec component of C2V2L, treating the entire tweet as a single word. This tests the value of the intermediate word representations in C2V2L; C2L has no explicit word representations. Hyperparameter tuning was carried out separately for C2L.

[5] Witten-Bell works well with small character vocabularies.

Results The first column of Table 3 shows the aggregate results across all labels. Our model achieves the state of the art on this task, surpassing the shared task winner, Hurtado et al. (2014). As expected, C2L fails to match the performance of C2V2L, demonstrating that there is value in the hierarchical representations. The performance of the n-gram LM baseline is notably strong, beating eleven out of the twelve submissions to the TweetLID shared task. We also report category-specific performance for our models and baselines in Table 3. Note that performance on underrepresented categories such as 'glg' and 'und' is much lower than the other categories. The category breakdown is not available for previously published results.

One important advantage of our model is its ability to handle special categories of tokens that would otherwise require special treatment as out-of-vocabulary symbols, such as URLs, hashtags, emojis, usernames, etc. Anecdotally, we observe that the input gates of the word-level LSTM are less likely to open for these special classes of tokens. This is consistent with the hypothesis that the model has learned to ignore tokens that are non-informative with respect to language ID.

5.2 TweetLID: Unconstrained Track

We augmented C2V2L's training data with 25,000 fragments of Wikipedia text, weighting the TweetLID training examples ten times more strongly. After training on the combined data, we "fine-tune" the model on the TweetLID data for 2,000 minibatches, which helped to correct for bias away from the undetermined language category, not covered in the Wikipedia data. The same hyperparameters were used as in the constrained experiment.

For the n-gram baseline, we interpolate the models trained on TweetLID and Wikipedia for each language. Interpolation weights given to the Wikipedia language models, set by cross-validation, ranged from 16% for Spanish to 39% for Galician, the most and least common labels respectively.

We also compare to unconstrained-track results of Hurtado et al. (2014) and Gamallo et al. (2014).

Results The results for these experiments are given in Table 4. Like Gamallo et al. (2014), we see a benefit from the use of out-of-domain data, giving a new state of the art on this task as well. Overall, the n-gram language model does not benefit from Wikipedia, but we observe that if the undetermined category, which is not found in the Wikipedia data, is ignored, then there is a net performance gain.

In Table 5, we show the top seven neighbors to selected input words based on cosine similarity. In the left column we see that words with similar features, such as the presence of the "n't" contraction, can be grouped together by char2vec. In the middle column, an out-of-vocabulary username is supplied and similar usernames are retrieved. When working with n-gram features, removing usernames is common, but some previous work demonstrates that they still carry useful information for predicting the language of the tweet (Jaech and Ostendorf, 2015). The third example,"noite" (Portuguese for "night"), shows that the word embeddings are largely invariant to changes in punctuation and capitalization.

5.3 Twitter70

We compare C2V2L to `langid.py` and the 5-gram language model on the Twitter70 dataset; see Table 6. Although the 5-gram model achieves the best performance, the results are virtually identical to those for C2V2L except for the closely-related Bosnian-Croatian language pair.

The lowest performance for all the models is on closely related language pairs. For example, using the C2V2L model, the F_1 score for Danish is only 62.7 due to confusion with the mutually intelligble Norwegian (Van Bezooijen et al., 2008). Distinguishing Bosnian and Croatian, two varieties of a single language, is also difficult. Languages that have unique orthographies such as Greek and Korean are identified with near perfect accuracy.

A potential advantage of the C2V2L model over the n-gram models is the ability to share information between related languages. In Figure 2 we show a T-SNE plot of the language embedding vectors taken from the softmax layer of our model trained with a rank constraint of 10 on the softmax layer.[6] Many languages appear close to related languages, although a few are far from their *phonetic* neighbors due to *orthographic* dissimilarity.

[6] The rank constraint was added for visualization; without it, the model makes all language embeddings roughly orthogonal to each other, making T-SNE visualization difficult.

Model	Avg. F_1	eng	spa	cat	eus	por	glg	und	amb
n-gram LM	75.0	74.8	94.2	82.7	74.8	**93.4**	49.5	**38.9**	87.0
`langid.py`	68.9	65.9	92.0	72.9	70.6	89.8	52.7	18.8	83.8
C2L	72.7	73.0	93.8	82.6	75.7	89.4	57.0	18.0	92.1
C2V2L	**76.2**	**75.6**	**94.7**	**85.3**	**82.7**	91.0	**58.5**	27.2	**94.5**

Table 3: F_1 scores on the TweetLID language ID task (constrained track), averaged and per language category (including undetermined and ambiguous). The scores for Hurtado et al. (2014) and Gamallo et al. (2014) are 75.2 and 75.6 respectively, as reported in Zubiaga et al. (2014); per-language scores are not available.

Model	F_1	Δ
Hurtado et al. (2014)	69.7	–4.5
Gamallo et al. (2014)	75.3	+2.7
n-gram LM	74.7	–0.3
C2V2L	**77.1**	+0.9

Table 4: F_1 scores for the unconstrained data track of the TweetLID language ID task. Δ measures change in absolute F_1 score from the constrained condition.

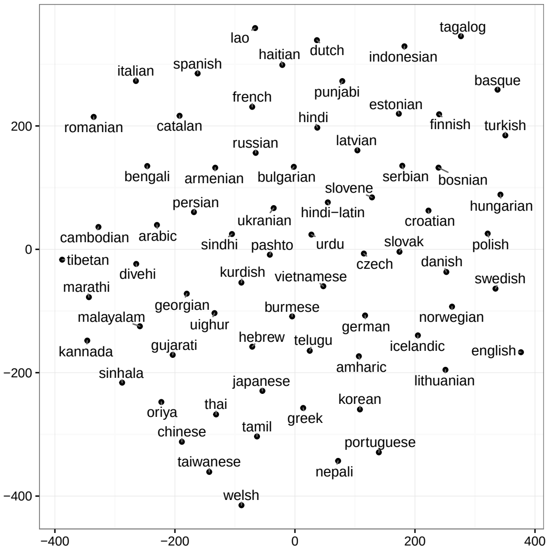

Figure 2: T-SNE plot of language embeddings.

5.4 Code-Switching

Because C2V2L produces language predictions for every word before making the tweet-level prediction, the same architecture can be used in word-level analysis of code-switched text, switching between multiple languages. Training a model with token level code-switching predictions requires a dataset that has token level labels. We used the Spanish-English dataset from the EMNLP 2014 shared task on Language Identification in Code-Switched Data (Solorio et al., 2014): a collection of monolingual and code-switched tweets in English and Spanish.

To train and predict at the word level, we remove the final average over the word predictions, and calculate the loss as the sum of the cross-entropy between each word's prediction and the corresponding gold label. Both the char2vec and word LSTM components of the model are unaffected, other than retraining their parameters.[7] To tune hyperparameters, we trained 10 models with random parameter settings on 80% of the data from the training set, and chose the settings from the model that performed best on the remaining 20%. We then retrained on the full training set with these settings.

C2V2L performed well at this task, scoring 95.1 F_1 for English (which would have achieved second place in the shared task, out of eight entries), 94.1 for Spanish (second place), 36.2 for named entities (fourth place) and 94.2 for Other (third place).[8] While our code-switching results are not quite state-of-the-art, they show that our model learns to make accurate word-level predictions. For other results on code-switched data, see Jaech et al. (2016b).

6 Related Work

Language ID has a long history both in the speech domain (House and Neuburg, 1977) and for text (Cavnar and Trenkle, 1994). Previous work on the text domain mostly uses word or character n-gram features combined with linear classifiers (Hurtado et al., 2014; Gamallo et al., 2014).

Recently published work by Radford and Gallé (2016) showed that combining an n-gram language model classifier (similar to our n-gram baseline)

[7] Both sentence and word-level supervision could be used to train the same model, but we leave that for future work.

[8] Full results for the 2014 shared task are omitted for space but can be found at `http://emnlp2014.org/workshops/CodeSwitch/results.php`.

couldn't		**@maria_sanchez**		**noite**	
can't	0.84	@Ainhooa_Sanchez	0.85	Noite	0.99
'don't	0.80	@Ronal2Sanchez:	0.71	noite.	0.98
ain't	0.80	@maria_lsantos	0.68	noite?	0.98
don't	0.79	@jordi_sanchez	0.66	noite..	0.96
didn't	0.79	@marialouca?	0.66	noite,	0.95
Can't	0.78	@mariona_g9	0.65	noitee	0.92
first	0.77	@mario_casas_	0.65	noiteee	0.90

Table 5: Top seven most similar words from the training data and their cosine similarities for inputs "couldn't", "@maria_sanchez", and "noite".

Model	F_1
`langid.py`	87.9
5-gram LM	93.8
C2V2L (ours)	91.2

Table 6: F_1 scores on the Twitter70 dataset.

with information from the Twitter social graph improves language ID on TweetLID from 74.7 to 76.6 F_1, only slightly better than our result of 76.2.

Bergsma et al. (2012) created their own multilingual Twitter dataset and tested both a discriminative model based on n-grams plus hand-crafted features and a compression-based classifier. Since the Twitter API requires researchers to re-download tweets based on their identifiers, published datasets quickly go out of date when the tweets in question are no longer available online, making it difficult to compare against prior work.

Several other studies have investigated the use of character sequence models in language processing. These techniques were first used only to create word embeddings (dos Santos and Zadrozny, 2015; dos Santos and Guimaraes, 2015) and then later extended to have the word embeddings feed directly into a word-level RNN. Applications include part-of-speech tagging (Ling et al., 2015b), language modeling (Ling et al., 2015a), dependency parsing (Ballesteros et al., 2015), translation (Ling et al., 2015b), and slot filling text analysis (Jaech et al., 2016a). The work is divided in terms of whether the character sequence is modeled with an LSTM or CNN, though virtually all now leverage the resulting word vectors in a word-level RNN. We are not aware of prior results comparing LSTMs and CNNs on a specific task, but the reduction in model size compared to word-only systems is reported to be much higher for LSTM architectures. All analyses report that the greatest improvements in performance from character sequence models are for infrequent and previously unseen words, as expected.

Chang and Lin (2014) outperformed the top results for English-Spanish and English-Nepali in the EMNLP 2014 Language Identification in Code-Switched Data (Solorio et al., 2014), using an RNN with skipgram word embeddings and character n-gram features. Word-level language ID has also been studied by Mandal et al. (2015) in the context of question answering and by King and Abney (2013). Both used primarily character n-gram features, which are well motivated for code-switching tasks since the presence of multiple languages increases the odds of encountering a previously unseen word.

7 Conclusion

We present C2V2L, a hierarchical neural model for language ID that outperforms previous work on the challenging TweetLID task. We also find that smoothed character n-gram language models can work well as classifiers for language ID for short texts. Without feature engineering, our n-gram baseline beat eleven out of the twelve submissions in the TweetLID shared task, and gives the best performance on the Twitter70 dataset, where training data for some languages is quite small. In future work, we plan to further adapt C2V2L to analyze code-switching, having shown that the current architecture already performs well.

Acknowledgments

This work is supported in part by a subcontract with Raytheon BBN Technologies Corp. under DARPA Prime Contract No. HR0011-15-C-0013 and by a gift to the University of Washington by Google. Views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. government.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. Int. Conf. Learning Representations (ICLR)*.

Timothy Baldwin and Marco Lui. 2010. Language identification: The long and the short of the matter. In *Proc. Conf. North American Chapter Assoc. for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 229–237. Association for Computational Linguistics.

Miguel Ballesteros, Chris Dyer, and Noah Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 349–359.

Timothy Bell, Ian H Witten, and John G Cleary. 1989. Modeling for text compression. *ACM Computing Surveys (CSUR)*, 21(4):557–591.

Shane Bergsma, Paul McNamee, Mossaab Bagdouri, Clayton Fink, and Theresa Wilson. 2012. Language identification for creating language-specific Twitter collections. In *Proc. Workshop on Language in Social Media (LSM)*, pages 65–74. Association for Computational Linguistics.

Simon Christopher Carter. 2012. *Exploration and exploitation of multilingual data for statistical machine translation*. Ph.D. thesis, University of Amsterdam.

William B. Cavnar and John M. Trenkle. 1994. N-gram-based text categorization. In *In Proc. of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175.

Joseph Chee Chang and Chu-Cheng Lin. 2014. Recurrent-neural-network for language detection on Twitter code-switching corpus. *CoRR*, abs/1412.4314.

Cicero dos Santos and Victor Guimaraes. 2015. Boosting named entity recognition with neural character embeddings. In *Proc. ACL Named Entities Workshop*, pages 25–33.

Cicero dos Santos and Bianca Zadrozny. 2015. Learning character-level representations for part-of-speech tagging. In *Proc. Int. Conf. Machine Learning (ICML)*.

Ted Dunning. 1994. Statistical identification of language. Technical report, Computing Research Laboratory, New Mexico State University, March.

Pablo Gamallo, Marcos Garcia, and Susana Sotelo. 2014. Comparing ranking-based and naive Bayes approaches to language detection on tweets. In *TweetLID@ SEPLN*.

Spandana Gella, Kalika Bali, and Monojit Choudhury. 2014. ”ye word kis lang ka hai bhai?”: Testing the limits of word level language identification. In *Proc. Int. Conf. Natural Language Processing (ICON)*.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *Proc. Conf. North American Chapter Assoc. for Computational Linguistics (NAACL)*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

Arthur S House and Edward P Neuburg. 1977. Toward automatic identification of the language of an utterance. *The Journal of the Acoustical Society of America*, 62(3):708–713.

Lluís F Hurtado, Ferran Pla, and Mayte Giménez. 2014. ELiRF-UPV en TweetLID: Identificación del idioma en Twitter. In *TweetLID@ SEPLN*.

Aaron Jaech and Mari Ostendorf. 2015. What your username says about you. *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*.

Aaron Jaech, Larry Heck, and Mari Ostendorf. 2016a. Domain adaptation of recurrent neural networks for natural language understanding. In *Proc. Conf. Int. Speech Communication Assoc. (INTERSPEECH)*.

Aaron Jaech, George Mulcaire, Shobhit Hathi, Mari Ostendorf, and Noah A. Smith. 2016b. A neural model for language identification in code-switched Tweets. In *Proc. Int. Workshop on Computational Approaches to Linguistic Code Switching (CALCS)*.

Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann Lecun. 2009. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision*, pages 2146–2153. IEEE.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander Rush. 2016. Character-aware neural language models. In *Proc. AAAI*, pages 2741–2749.

Ben King and Steven Abney. 2013. Labeling the languages of words in mixed-language documents using weakly supervised methods. In *Proc. Conf. North American Chapter Assoc. for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1110–1119.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. Int. Conf. Learning Representations (ICLR)*.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015a. Finding function in form: Compositional character models for open vocabulary word representation. *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan Black. 2015b. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586v1*.

Marco Lui and Timothy Baldwin. 2012. langid. py: An off-the-shelf language identification tool. In *Proc. of the ACL 2012 system demonstrations*, pages 25–30. Association for Computational Linguistics.

Soumik Mandal, Somnath Banerjee, Sudip Kumar Naskar, Paolo Rosso, and Sivaji Bandyopadhyay. 2015. Adaptive voting in multiple classifier systems for word level language identification. In *the Working Notes in Forum for Information Retrieval Evaluation (FIRE)*.

Paul McNamee. 2005. Language identification: A solved problem suitable for undergraduate instruction. *J. Comput. Sci. Coll.*, 20(3):94–101, February.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proc. Conf. Int. Speech Communication Assoc. (INTERSPEECH)*, volume 2, page 3.

V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *Proc. Int. Conf. on Frontiers in Handwriting Recognition*, pages 285–290.

Will Radford and Matthias Gallé. 2016. Discriminating between similar languages in Twitter using label propagation. *arXiv preprint arxiv:1607.05408*.

Hasim Sak, Andrew W Senior, and Françoise Beaufays. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Proc. Conf. Int. Speech Communication Assoc. (INTERSPEECH)*, pages 338–342.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Gohneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, et al. 2014. Overview for the first shared task on language identification in code-switched data. In *Proc. Int. Workshop on Computational Approaches to Linguistic Code Switching (CALCS)*, pages 62–72.

Andreas Stolcke. 2002. SRILM-An extensible language modeling toolkit. In *Proc. Conf. Int. Speech Communication Assoc. (INTERSPEECH)*, volume 2002, page 2002.

Renée Van Bezooijen, Charlotte Gooskens, Sebastian Kürschner, and Anja Schüppert. 2008. Linguistic factors of mutual intelligibility in closely related languages. In *Article presented at the Symposium on Receptive Multilingualism, part II (organized by JD ten Thije), AILA 2008 conference'Multilingualism: Challenges and Opportunities', Essen*, pages 24–29.

Pidong Wang, Nikhil Bojja, and Shivasankari Kannan. 2015. A language detection system for short chats in mobile games. In *Proc. Int. Workshop on Natural Language Processing for Social Media (SocialNLP)*, pages 20–28, Denver, Colorado, June. Association for Computational Linguistics.

Arkaitz Zubiaga, Inaki San Vicente, Pablo Gamallo, José Ramom Pichel Campos, Iñaki Alegría Loinaz, Nora Aranberri, Aitzol Ezeiza, and Víctor Fresno-Fernández. 2014. Overview of TweetLID: Tweet language identification at SEPLN 2014. In *TweetLID@ SEPLN*, pages 1–11.