A Neural Model for Language Identification in Code-Switched Tweets

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

Language identification systems suffer when working with short texts or in domains with unconventional spelling, such as Twitter or other social media. These challenges are explored in a shared task for Language Identification in Code-Switched Data (LICS 2016). We apply a hierarchical neural model to this task, learning character and contextualized word-level representations to make word-level language predictions. This approach performs well on both the 2014 and 2016 versions of the shared task.

1 Introduction

Language identification (language ID) remains a difficult problem, particulary in social media text where informal styles, closely related language pairs, and code-switching are common. Progress on language ID is needed especially since downstream tasks, like translation or semantic parsing, depend on it.

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, can be useful for language ID. For the Language Identification in Code-Switched Data shared task (LICS 2016), we submitted a hierarchical character-word model closely following Jaech et al. (2016b), focusing on word-level language ID. Our discussion of the model closely follows that paper.

This 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 to use context from the full word sequence before making predictions for each word. The use of character representations is well motivated for code-switching tasks, since the presence of multiple languages means that one is more likely to encounter a previously unseen word.

Our model does not require special handling of casing or punctuation, nor do we need to remove the URLs, usernames, or hashtags, and it is trained end-to-end using standard procedures.

2 Model

Our model has two main components, though they are trained together, end-to-end. The first, "char2vec," applies a convolutional neural network (CNN) to a whitespace-delimited word's Unicode character sequence, providing a word vector. 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 embeds characters. An embedding is 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 embedding layer be $d = \lceil \log_2 |C| \rceil$. (If each dimension of the character embedding vector holds just 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 be-

Proceedings of the Second Workshop on Computational Approaches to Code Switching, pages 60–64,
Austin, TX, November 1, 2016.

cause 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 $\mathbf{H}_{a_i}$ is a filter with a width of three characters, the columns of $\mathbf{T}_1$ each hold a representation of a character tri-gram. 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. 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) which uses 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 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.

2.2 Sentence-level Context

The sequence of word embedding vectors is processed by a bi-LSTM, which outputs a sequence of

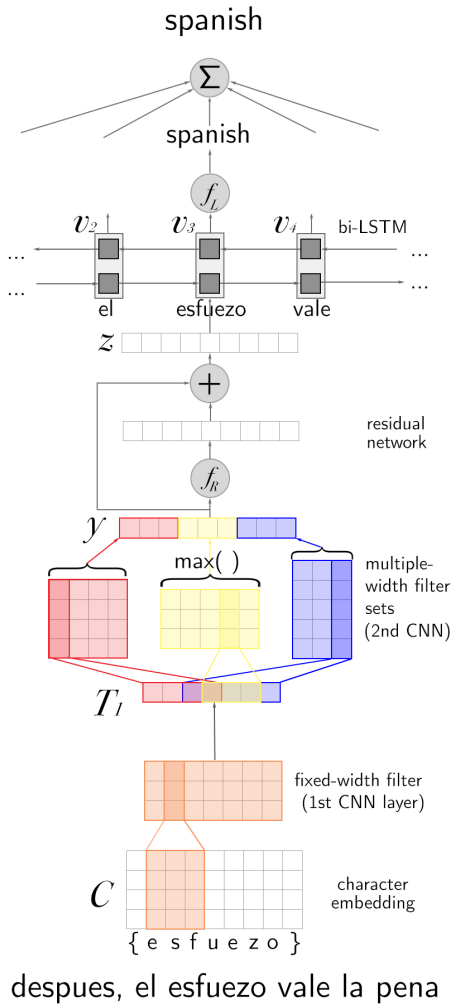

Figure 1: C2V2L model architecture. The model takes the word "esfuezo," a misspelling of the Spanish word "esfuerzo," and maps it to a word vector via the two CNN layers and the residual layer. The word vector is then combined with others via the LSTM, and a prediction made for each word.

vectors, $[\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \ldots \mathbf{v}_T]$ 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 (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:

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

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 token. The loss is the cross-entropy between each word's prediction and the corresponding gold label for that word.

3 Implementation Details

3.1 Preprocessing

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 usernames or hashtags without spaces between them. These create extra long "words" that cause our implementation to use more memory and do extra computation during training. We therefore 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).

3.2 Training and Tuning

Training is done using minibatches of size 25 and a learning rate of 0.001 using Adam (Kingma and Ba, 2015), and continued for 50,000 minibatches.

There are only four hyperparameters to tune for each model: the number of filters in the first convolutional layer, the number of filters in the second convolutional layer, the size of the word-level LSTM vector, and the dropout rate.

To tune hyperparameters, we trained 10 models with random parameter settings on 80% of the data from the Spanish-English 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 (respectively, 59, 108, 23, and 25%). Our final model architecture has roughly 177K parameters.

4 Experiments

Because C2V2L produces language predictions for every word, the architecture is well suited to analysis of code-switched text, in which different words may belong to different languages. We used the Spanish-English dataset from the EMNLP 2014 shared task on Language Identification in Code-Switched Data (Solorio et al., 2014), and the Spanish-English and Arabic-MSA datasets from the EMNLP 2016 version of the shared task. Each dataset is a collection of monolingual and code-switched tweets in two main languages: English and Spanish, or Modern Standard Arabic (MSA) and Arabic dialects.

4.1 LICS 2014

The LICS 2014 dataset (Zubiaga et al., 2014) comes from a language ID shared task that focused on tweets that code-switch between two languages. While several language pairs were included in the shared task, we evaluated only on Spanish-English in these experiments. There are approximately 110,000 labeled examples in the training data and 34,000 in the test set. The data is unbalanced, with over twice as many examples in English (74,000) as in Spanish (35,000). A further 30,000 are labeled "other" for punctuation, emoji, and unintelligible words. There are also a small number of examples labeled "NE" for named entities (2.16%), "mixed" for words that include both English and Spanish (0.03%), and "ambiguous" for words that could be interpreted as either language in context (0.22%). "NE" and "other" are predicted as if they were separate languages, but the 'ambiguous' label is ignored.

C2V2L performed well at this task, scoring 95.1 F_1 for English (which would have achieved second place in the 2014 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).[1] While our code-switching results are not quite state-of-the-art, they show that our model learns accurate word-level predictions.

[1]Full results for the 2014 shared task can be found at `http://emnlp2014.org/workshops/CodeSwitch/results.php`.

Lang. pair (L1-L2)	L1	L2	NE	other
Spanish-English (ours)	0.931	0.977	0.454	0.910
Spanish-English (best)	0.931	0.977	0.537	0.994
Arabic-MSA (ours)	0.603	0.603	0.468	0.712
Arabic-MSA (best)	0.854	0.904	0.828	0.988

Table 1: F1 scores for the LICS 2016 shared task. The best result from any system in each category is provided for comparison.

4.2 LICS 2016

The LICS 2016 shared task uses a similar format. Here, the training and development sets for each language pair correspond to the training and test sets from LICS 2014, and new data was added to create a new test set. However, labels were updated to correct errors and add two new categories: "fw" (foreign word) for examples that belong to a language other than the main two, and "unk" for examples that cannot be classified. We ignore "unk" and "fw."

We submitted labels for both available language pairs: Spanish-English and Arabic-MSA (distinguishing Modern Standard Arabic from Arabic dialects). Partial results,[2] showing only the four largest categories, are given in Table 1.

Our results for Spanish-English are competitive with the best submitted systems. We ranked first or tied for first in F_1 for the primary categories, English and Spanish, out of nine submitted systems. On Arabic-MSA, we came in last among five systems. This is likely due in part to the fact that we tuned only on Spanish-English data and did not make any adjustments when training the Arabic-MSA model. A model that is tuned to the specific language pair, and perhaps handled the 'other' category with regular expressions in preprocessing, would likely perform better.

5 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). Chang and Lin (2014) outperformed the top results for English-Spanish and English-Nepali in the EMNLP 2014 Language Identitication in Code-Switched Data (Solorio et al., 2014), using an RNN with skip-gram 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.

Several other studies have investigated the use of character sequence models in language processing. These techniques were first applied 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 (POS) 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). A more extensive discussion of related work on language ID and character sequence models can be found in Jaech et al. (2016b).

6 Conclusion

We present C2V2L, a hierarchical neural model for language ID that preforms competitively on challenging word-level language ID tasks. Without feature engineering, we achieved the best performance in two common categories and good results in two others. Future work could include adapting C2V2L for other sequence labeling tasks, having shown that the current architecture already performs well.

Acknowledgments

We thank the anonymous reviewers for helpful feedback. Part of this material is based upon work supported by a subcontract with Raytheon BBN Technologies Corp. under DARPA Prime Contract No. HR0011-15-C-0013. This work was also supported 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 of the U.S. government.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly

[2]Full results can be found at `http://care4lang1.seas.gwu.edu/cs2/results.html`

learning to align and translate. In *Proc. Int. Conf. Learning Representations (ICLR)*.

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)*.

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*.

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

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

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, 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. Hierarchical character-word models for language identification. In *Proc. Int. Workshop on Natural Language Processing for Social Media (SocialNLP)*.

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*. IEEE.

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

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)*.

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. In *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*.

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)*.

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)*.

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 (ICFHR)*.

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)*.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Ghoneim, 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)*.

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*.