Language Identification from Very Short Strings

Vol. 1, Issue 14 • July 2019
by Input & Intelligence — Natural Language Processing Team

Many language-related tasks, such as entering text on your iPhone, discovering news articles you might enjoy, or finding out answers to questions you may have, are powered by language-specific natural language processing (NLP) models. To decide which model to invoke at a particular point in time, we must perform language identification (LID), often on the basis of limited evidence, namely a short character string. Performing reliable LID is more critical than ever as multi-lingual input is becoming more and more common across all Apple platforms. In most writing scripts — like Latin and Cyrillic, but also including Hanzi, Arabic, and others — strings composed of a few characters are often present in more than one language, making reliable identification challenging. In this article, we explore how we can improve LID accuracy by treating it as a sequence labeling problem at the character level, and using bi-directional long short-term memory (bi-LSTM) neural networks trained on short character sequences. We observed reductions in error rates varying from 15% to 60%, depending on the language, while achieving reductions in model size between 40% and 80% compared to previously shipping solutions. Thus the LSTM LID approach helped us identify language more correctly in features such as QuickType keyboards and Smart Responses, thereby leading to better auto-corrections, completions, and predictions, and ultimately a more satisfying user experience. It also made public APIs like the Natural Language framework more robust to multi-lingual environments.

Introduction

In many NLP applications, it is critical to be able to accurately identify the language in which the current input is written. For example, this capability is needed to load the right autocorrection lexicon and the right language model for predictive and multilingual typing. Additionally, LID is used to select the appropriate linguistic tagger and other advanced NLP analysis tools for possible downstream semantic processing. As an instance, Spotlight indexing mechanism uses LID to launch the correct tokenizer so that appropriately tokenized substrings are indexed for quick retrieval for any user search query.

The task of identifying the language in which a given document is written has been studied extensively over the past decades, and several classes of methods are available. A summary of their respective drawbacks is available in [1]. Briefly, lexically inspired solutions are not practical on embedded devices, syntactically inspired techniques are restricted to long documents where plenty of evidence is available, and generative statistical models based on character n-grams (as in [2][3]) suffer from the conditional independence assumptions implicit in the Markov strategy.

Today, a majority of the data being generated by our users contains only a relatively small number of characters (e.g., text messaging, social network posts). For such short strings of only 10-50

characters, past n-gram approaches based on cumulative frequency addition tend to fall short of accuracy requirements [3], so a more robust sequence classifier is required.

In this article, we describe a neural network-based LID system designed to improve classification when only a few characters have been observed. In the next section, we go over the bi-LSTM architecture we adopted. Then we describe the improvements we obtained on both Latin and Hanzi/Kana scripts. Finally, we discuss the insights that we gained and our perspectives on future development.

Neural LID Architecture

We model LID as a character level sequence labeling problem. Figure 1 describes the generic bi-directional recurrent neural network architecture we adopted, where each hidden node is modeled as an LSTM node [4][5]. Initial experiments showed that mixing scripts (e.g., Latin, Cyrillic, etc.) in a single classifier resulted in higher confusability. Since Unicode based script identification is cheap, each script thus gets its own network.

For each script, the architecture of Figure 1 maps character sub-sequences to languages based on global evidence gathered from a database of documents representative of all languages considered for that script. The recurrent neural network (RNN) paradigm naturally handles the problem of variable-length sequences.

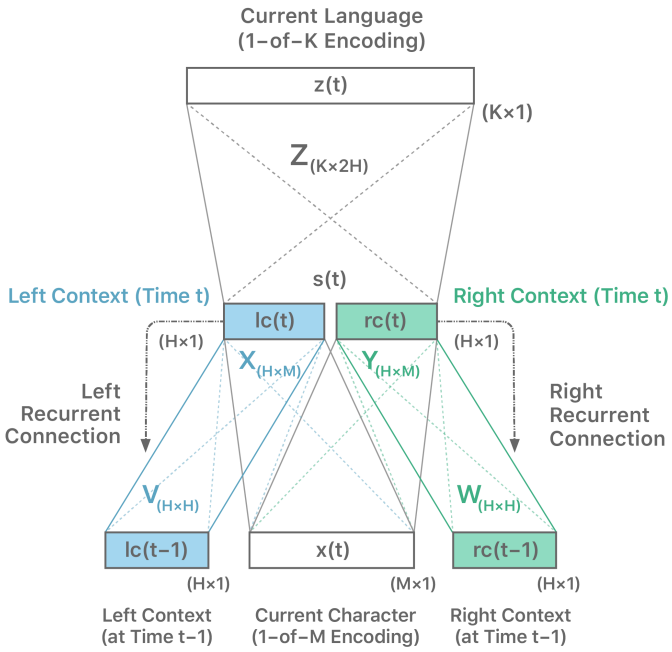


Figure 1. Bi-LSTM architecture for LID

Furthermore, by considering a bi-directional architecture, the model not only exploits left context (from the beginning of the string), but also right context (from the end of the string, as it becomes known).

In a typical set-up, we design such an LID system for K=20 Latin script languages. The character inventory consists of around M=250 characters. To enhance discriminative power, we use a two-layer bi-directional variant of the LSTM architecture. The LSTM cells are followed by a softmax layer. A max pooling style majority voting decides the dominant language of the string.

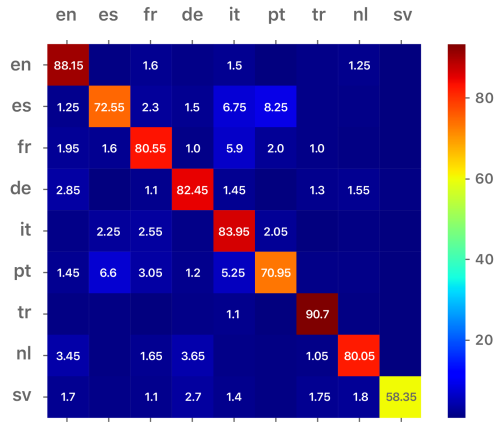
Additionally, we constrain every training sequence so it begins at the beginning of a word, since user-generated short strings rarely start at an arbitrary location. We didn't gain any additional accuracy beyond the first 50 characters of a sequence. This upper bound also benefits time-

performance in resource-constrained mobile devices.

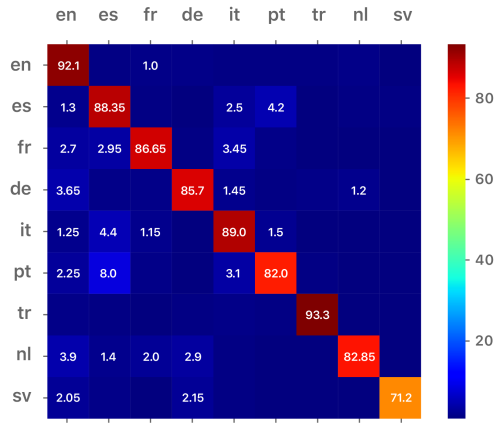
Experimental Results

Results are reported as confusion matrices for two scripts: Latin (cf. Figure 2) and Hanzi (cf. Figure 3). The training and disjoint test sets are a mixture of newswire and short conversational texts. In the case of the Latin script, we restricted the input string length to be 10 characters, corresponding to 1 to 2 words on average. In the case of the Hanzi script, we restricted the input string length to be 5 characters, also corresponding to 1 to 2 words on average. The diagonal represents accuracy, while off-diagonal cells show the confusability with other languages.

For readability, the Latin script matrix is truncated to the top nine Latin languages: English (en), Spanish (es), French (fr), German (de), Italian (it), Portuguese (pt), Turkish (tr), Dutch (nl), and Swedish (sv). The Hanzi/Kana matrix is for Simplified Chinese (zh-hans), Traditional Chinese (zh-hant) and Japanese (ja). Comparative confusion matrices are also shown for the n-gram-based cumulative frequency approach of [3] on the same data set.

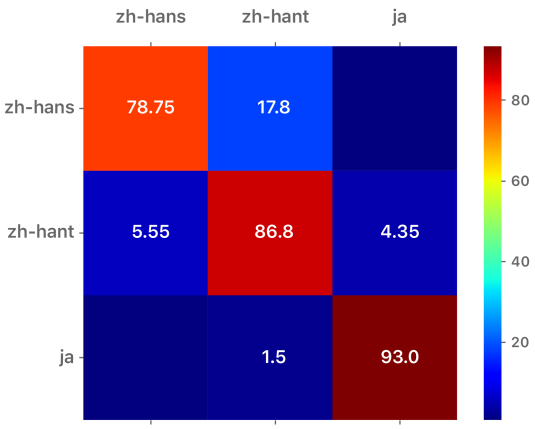


(a) n-gram confusion matrix

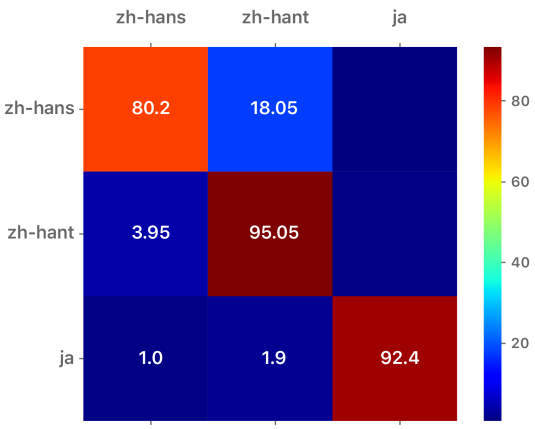


(b) LSTM confusion matrix

Figure 2. Confusion matrices for LID at 10 characters for Latin script



(a) n-gram confusion matrix



(b) LSTM confusion matrix

Figure 3. Confusion matrices for LID at 5 characters for Hanzi/Kana script

For both scripts, the darker red diagonals and darker blue off-diagonal cells associated with the LSTM approach indicate a more accurate LID system. We conclude that the proposed approach is well-suited for robust LID on short strings.

With respect to disk footprints (Figure 4), the combined model size for bi-LSTM LID (4 MB) represents a 43% reduction over previous n-gram model on iOS (7 MB) and an 84% reduction over macOS n-gram model (25 MB).

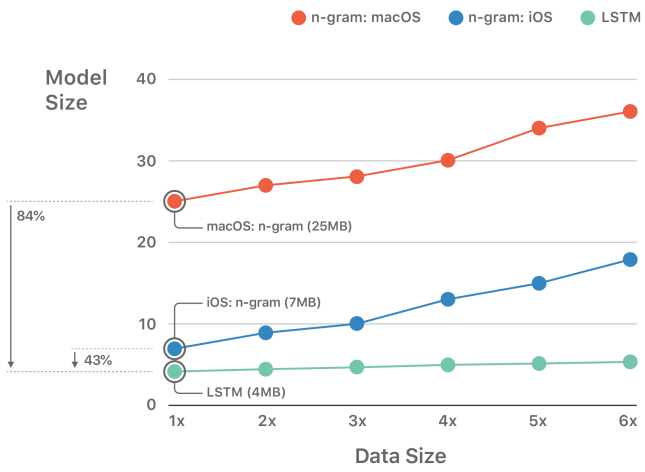


Figure 4. Model sizes comparison and growth as training data size is increased

Another key point is to examine the growth of model size as we increase the amount of training data. Figure 4 shows that with additional training data the n-gram model sizes grow linearly, while for LSTM the model size is almost constant. This correlation is because the more the data, the more the n-grams, and hence greater the model size. LSTM model sizes, on the other hand, are simply a function of the network parameters. So no matter how much data we feed, as long as we have the same number of network parameters or a few more hidden nodes with which to better model the increasing data, the model size is going to remain almost constant, while the overal quality of the model improves with more data.

Conclusion

We set out to assess the potential benefits of switching to bi-LSTMs to perform LID from limited evidence. Robust LID from very short strings is highly desirable in multiple NLP pipelines, including in usage of autocorrection lexicons and predictive and multilingual typing [1], for part-of-speech and named entity tagging, when performing document classification [2], and as a component of TTS engines [6]. At Apple, bi-LSTM LID is now used for most tasks which require language identification, like text tagging and other public APIs part of the Natural Language framework. It also provides the correct expected language in features like QuickType keyboards and Smart Responses.

We observed that a bi-LSTM LID architecture can indeed lead to improved accuracy across multiple scripts. Such results confirm the viability of the approach for a fast and accurate language identifier operating on limited evidence. Performance requirements regarding responsiveness and resources led us to limit our experiments to relatively shallow LSTM networks rather than newer but computationally more complex architectures such as transformers. Regardless, the LSTM LID system has smaller memory footprint compared to n-gram and is scalable with addition of larger amounts of data without becoming prohibitively large.

References

[1] B. Barman and J.R. Bellegarda. **Language Identification from Short Strings**. *U.S. Patent No. 10,127,220*, November 2018.

[2] W. B. Canvar and J. M. **Trenkle. N-gram based Text Categorization**. *Proceedings of*

Symposium on Document Analysis and Information Retrieval, University of Nevada, Las Vegas, pp. 161-176, 1994.

[3] B. Ahmed, S.-H. Cha, and C. Tappert. **Language Identification from Text Using N-gram Based Cumulative Frequency Addition**. *Proceedings of Student/Faculty Research Day*, CSIS, Pace University, 2004.

[4] S. Hochreiter and J. Schmidhuber. **Long Short-Term Memory**. *Neural Computation* 9, 8 pp. 1735-1780, November 1997.

[5] M. Sundermeyer, R. Schlüter, and H. Ney. **LSTM Neural Networks for Language Modeling**. *InterSpeech 2012*, pp. 194–197, September 2012.

[6] R. Sproat. **Multilingual Text Analysis for Text-To-Speech Synthesis**. *Proceedings of ICSLP'96*, Philadelphia, October 1996.