Byte-based Language Identification with Deep Convolutional Networks

Johannes Bjerva
University of Groningen
The Netherlands
j.bjerva@rug.nl

Abstract

We report on our system for the shared task on discrimination of similar languages (DSL 2016). The system uses only byte representations in a deep residual network (ResNet). The system, named ResIdent, is trained only on the data released with the task (closed training). We obtain 84.88% accuracy on subtask A, 68.80% accuracy on subtask B1, and 69.80% accuracy on subtask B2. A large difference in accuracy on development data can be observed with relatively minor changes in our network's architecture and hyperparameters. We therefore expect fine-tuning of these parameters to yield higher accuracies.

1 Introduction

Language identification is an unsolved problem, certainly in the context of discriminating between very similar languages (Baldwin and Lui, 2010). This problem is tackled in the Discriminating between Similar Languages (DSL) series of shared tasks (Zampieri et al., 2014; Zampieri et al., 2015). Most successful approaches to the DSL shared task in previous years have relied on settings containing ensembles of classifiers (Goutte et al., 2016). These classifiers often use various combinations of features, mostly based on word, character, and/or byte n-grams (see, e.g., Cavnar et al. (1994), Lui and Baldwin (2012)).

We are interested in exploring a single methodological aspect in the current edition of this shared task (Malmasi et al., 2016). We aim to investigate whether reasonable results for this task could be obtained by applying recently emerged neural network architectures, coupled with sub-token input representations. To address this question, we explore convolutional neural networks (CNNs) and recurrent neural networks (RNNs). Deep residual networks (ResNets) are a recent building block for CNNs which have yielded promising results in, e.g., image classification tasks (He et al., 2015; He et al., 2016). ResNets are constructed by stacking so-called residual units. These units can be viewed as a series of convolutional layers with a 'shortcut' which facilitates signal propagation in the neural network. This, in turn, allows for training deeper networks more easily (He et al., 2016). In Natural Language Processing (NLP), ResNets have shown state-of-the-art performance for Semantic and Part-of-Speech tagging (Bjerva et al., 2016). However, no previous work has attempted to apply ResNets to language identification.

2 Method

Several previous approaches in the DSL shared tasks have formulated the task as a two-step classification, first identifying the language group, and then the specific language (Zampieri et al., 2015). Instead of taking this approach, we formulate the task as a multi-class classification problem, with each language / dialect representing a separate class. Our system is a deep neural network consisting of a bidirectional Gated Recurrent Unit (GRU) network at the upper level, and a Deep Residual Network (ResNet) at the lower level (Figure 1). The inputs of our system are byte-level representations of each input sentence, with byte embeddings which are learnt during training. Using byte-level representations differs from character-level representations in that UTF-8 encodes non-ascii symbols with more than one byte, which

Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects,
pages 119–125, Osaka, Japan, December 12 2016.

potentially allows for more disambiguating power. A concrete example can be found when considering the relatively similar languages Norwegian and Swedish. Here, there are two pairs of letters which are interchangeable: where Swedish uses 'ä' (C3 A4) and 'ö' (C3 B6), Norwegian uses 'æ' (C3 A6) and 'ø' (C3 B8). Hence, using the lower-level byte representation, we allow the model to take advantage of the first shared byte between these characters. The architecture used in this work is based on the sequence-to-sequence labelling architecture used in Bjerva et al. (2016), modified for the task of language identification. Our system is implemented in Keras using the Tensorflow backend (Chollet, 2015; Abadi et al., 2016).

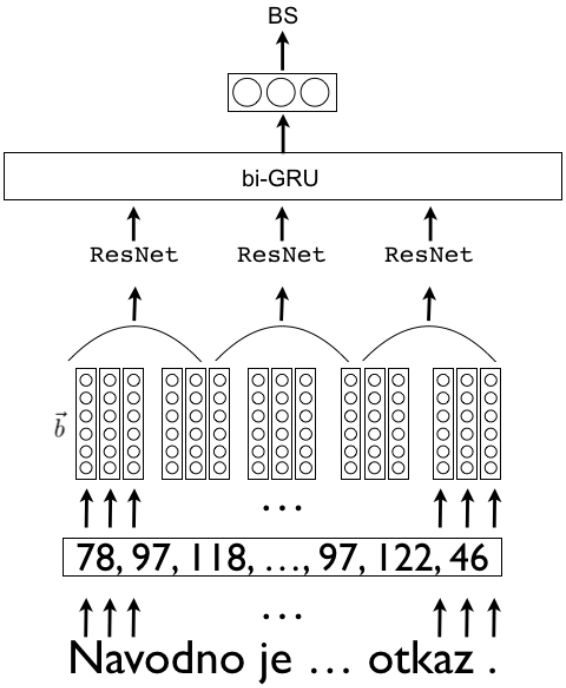

Figure 1: Model architecture: ResNet with byte representations ($\vec{b}$), with a bi-GRU at the upper level. The input example sequence is converted to a sequence of byte identifiers (one integer per byte, rather than one integer per character), which are converted to a byte embedding representation. This input is treated by the ResNet, followed by the bi-GRU, finally yielding the language id *BS* (Bosnian).

2.1 Gated Recurrent Unit Networks

GRUs (Cho et al., 2014) are a recently introduced variant of RNNs, and are designed to prevent vanishing gradients, thus being able to cope with longer input sequences than vanilla RNNs. GRUs are similar to the more commonly-used Long Short-Term Memory networks (LSTMs), both in purpose and implementation (Chung et al., 2014). A bi-directional GRU makes both forward and backward passes over sequences, and can therefore use both preceding and succeeding contexts to predict a tag (Graves and Schmidhuber, 2005; Goldberg, 2015). Bi-directional GRUs and LSTMs have been shown to yield high performance on several NLP tasks, such as POS and semantic tagging, named entity tagging, and chunking (Wang et al., 2015; Yang et al., 2016; Plank et al., 2016; Bjerva et al., 2016).

2.2 Deep Residual Networks

Deep Residual Networks (ResNets) are built up by stacking residual units. A residual unit can be expressed as:

$$\begin{aligned} y_l &= h(x_l) + \mathcal{F}(x_l, \mathcal{W}_l), \\ x_{l+1} &= f(y_l), \end{aligned} \tag{1}$$

where x_l and x_{l+1} are the input and output of the l-th layer, $\mathcal{W}_l$ is the weights for the l-th layer, and $\mathcal{F}$ is a residual function (He et al., 2016), e.g., the identity function (He et al., 2015), which we also

use in our experiments. ResNets can be intuitively understood by thinking of residual functions as paths through which information can propagate easily. This means that, in every layer, a ResNet learns more complex feature combinations, which it combines with the shallower representation from the previous layer. This architecture allows for the construction of much deeper networks. ResNets have recently been found to yield impressive performance in both image recognition and NLP tasks (He et al., 2015; He et al., 2016; Östling, 2016; Conneau et al., 2016), and are an interesting and effective alternative to simply stacking layers. In this paper we use the *assymetric* variant of ResNets as described in Equation 9 in He et al. (2016):

$$x_{l+1} = x_l + \mathcal{F}(\hat{f}(x_l), \mathcal{W}_l). \tag{2}$$

Our residual block, using dropout and batch normalization (Srivastava et al., 2014; Ioffe and Szegedy, 2015), is defined in Table 1. In the table, *merge* indicates the concatenation of the input of the residual block, with the output of the final convolutional layer.

type	**patch/pool size**
Batch normalization + ReLu + Dropout ($p = 0.5$)	
Convolution	8
Batch normalization + ReLu + Dropout ($p = 0.5$)	
Convolution	4
Merge	
Maximum pooling	2

Table 1: Residual block overview.

2.3 System Description

We represent each sentence using a byte-based representation (S_b). This representation is a 2-dimensional matrix $S_b \in \mathbb{R}^{s \times d_b}$, where s is the zero-padded sentence length and d_b is the dimensionality of the byte embeddings. Byte embeddings are first passed through a ResNet in order to obtain a representation which captures something akin to byte n-gram features.[1] The size of n is determined by the convolutional window size used. We use a convolutional window size with length 8, meaning that for each byte in the input, the ResNet can learn a suitable representation incorporating up to 8 bytes of context information. These overlapping byte-based n-gram features are then passed through to the bi-GRU, which yields a *sentence level* representation. The softmax layer applied to the bi-GRU output is then used in order to obtain the network's predicted class per input.

2.3.1 Hyperparameters

The hyperparameters used by the system were tuned on an altogether different task (semantic tagging), and adapted for the current task. The dimensionality of our byte embeddings, d_b, is set to 64. Our residual block is defined in Section 2.2. We use rectified linear units (ReLUs) for all activation functions (Nair and Hinton, 2010), and apply dropout with $p = 0.1$ to both input weights and recurrent weights in the bi-GRU. All GRU layers have 100 hidden units.

All experiments were run with early stopping monitoring validation set loss, using a maximum of 50 epochs, and a batch size of 100. Optimisation is done using the ADAM algorithm (Kingma and Ba, 2015), with the categorical cross-entropy loss function as training objective.

For the B tasks, we train the model in the same way as for the A tasks. Only a handful of instances ($n \approx 5$) per B run are classified as belonging to a language which the B group does not contain. These cases are automatically converted to being in the class *hr*. For the B tasks, we also perform a simple clean-up of the data. We first remove all hyperlinks, hashtags and usernames from the text with a simple regex-based script. We then remove all tweets classified as English. We submitted three runs for each subtask. The system used for runs 1, 2 and 3 contain five, four and three residual blocks respectively.

[1] Note that bytes are passed through the ResNet *one by one*, yielding one representation per byte, rather than as a whole sequence, which would yield a single representation per sentence.

3 Results

Test Set	Track	Run	Accuracy	F1 (micro)	F1 (macro)	F1 (weighted)
A	closed	Baseline	0.083			
A	closed	run1	0.8462	0.8462	0.8415	0.8415
A	closed	run2	0.8324	0.8324	0.8272	0.8272
A	closed	run3	**0.8488**	0.8488	0.8467	0.8467
B1	closed	Baseline	0.020			
B1	closed	run1	0.682	0.682	0.6802	0.6802
B1	closed	run2	0.676	0.676	0.6708	0.6708
B1	closed	run3	**0.688**	0.688	0.6868	0.6868
B2	closed	Baseline	0.020			
B2	closed	run1	0.684	0.684	0.6788	0.6788
B2	closed	run2	**0.698**	0.698	0.6942	0.6942
B2	closed	run3	0.664	0.664	0.6524	0.6524

Table 2: Results for all runs in subtasks A, B1 and B2 (closed training).

	es-ar	es-es	es-mx	fr-ca	fr-fr	id	my	pt-br	pt-pt	hr	bs	sr
es-ar	824	77	94	0	1	1	0	2	1	0	0	0
es-es	90	778	127	0	1	0	0	1	2	0	1	0
es-mx	210	269	520	0	0	0	0	1	0	0	0	0
fr-ca	0	0	0	956	44	0	0	0	0	0	0	0
fr-fr	0	0	0	93	905	0	0	1	0	1	0	0
id	0	0	0	0	0	951	48	0	0	0	0	1
my	0	0	0	0	0	30	970	0	0	0	0	0
pt-br	0	0	1	0	0	0	0	891	107	1	0	0
pt-pt	0	1	0	0	0	0	0	78	920	0	1	0
hr	0	0	0	0	0	0	0	0	0	823	150	27
bs	0	0	0	0	1	0	0	1	0	143	730	125
sr	0	0	0	0	1	0	0	0	0	15	67	917

Table 3: Confusion matrix, closed run 3, on test set A. The x-axis indicates predicted labels, and the y-axis indicates true labels.

We evaluate our system in subtasks A, B1 and B2. Subtask A contains data for five language groups, with two to three languages in each group (Tan et al., 2014). Subtask B1 and B2 contain data for a subset of the languages in subtask A, compiled from Twitter. Subtask B1 contains the amount of tweets necessary for a human annotator to make reliable judgements, whereas B2 contains the maximum amount of data available per tweet.

For subtasks A and B1, run 3 results in the best accuracy on test, whereas run 2 results in the best accuracy on B2. The results are shown in Table 2. Table 3 and Table 4 contain confusion matrices for the results in subtask A and B respectively.

4 Discussion

Judging from the confusion matrices in Section 3, our system has very low confusion between language groups. However, confusion can be observed within all groups. Although the system achieves reasonable

	B1					B2				
	pt-br	pt-pt	hr	bs	sr	pt-br	pt-pt	hr	bs	sr
pt-br	74	24	1	0	1	54	40	3	2	1
pt-pt	31	67	1	0	1	15	80	5	0	0
bs	0	0	60	31	9	0	0	75	20	5
hr	1	0	20	62	17	0	0	31	56	13
sr	4	0	5	10	81	2	0	8	6	84

Table 4: Confusion matrix, closed run 3 on test set B1 (left) and closed run 2 on test set B2 (right). The x-axis indicates predicted labels, and the y-axis indicates true labels.

performance, there is a large gap between our system and the best performing systems (e.g. Çöltekin and Rama (2016), who obtain 89.38% accuracy on task A, 86.2% on B1, and 82.2% on B2). This can to some extent be explained by limitations caused by our implementation.

The largest limiting factor can be found in the fact that we only allowed our system to use the first ca. 384 bytes of each training/testing instance. For the training and development set, and subtask A, this was no major limitation, as this allowed us to use more than 90% of the available data. However, for subtasks B1 and B2, this may have seriously affected the system's performance. Additionally, we restricted our system to using only byte embeddings as input. Adding word-level representations into the mix, would likely increase system performance.

We also observed considerable differences in development accuracy when changing hyperparameters of our network in relatively minor ways. For instance, altering the patch sizes used in our CNNs had a noticeable impact on validation loss. However, altering the amount of residual blocks used, did not have a large effect on results. The neural network architecture, as well as most of the hyperparameters, were tuned on an altogether different task (semantic tagging), and adapted for the current task. Further fine tuning of the network architecture and hyperparameters for this task would therefore likely lead to narrowing the performance gap.

5 Conclusions

We implemented a language identification system using deep residual networks (ResNets) coupled with a bidirectional Gated Recurrent Unit network (bi-GRU), using only byte-level representations. In the DSL 2016 shared task, we achieved reasonable performance, with 84.88% accuracy on subtask A, 68.80% accuracy on subtask B1, and 69.80% accuracy on subtask B2. Although acceptable performance was achieved, further fine tuning of input representations and system architecture would likely improve performance.

Acknowledgements

We would like to thank the Center for Information Technology of the University of Groningen for their support and for providing access to the Peregrine high performance computing cluster, as well as the anonymous reviewers for their valuable feedback.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens,

Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Timothy Baldwin and Marco Lui. 2010. Language identification: The long and the short of the matter. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 229–237. Association for Computational Linguistics.

Johannes Bjerva, Barbara Plank, and Johan Bos. 2016. Semantic Tagging with Deep Residual Networks. In *Proceedings of COLING 2016*, Osaka, Japan, December.

William B Cavnar, John M Trenkle, et al. 1994. N-gram-based text categorization. *Ann Arbor MI*, 48113(2):161–175.

Çağrı Çöltekin and Taraka Rama. 2016. Discriminating similar languages: experiments with linear SVMs and neural networks. In *Proceedings of the 3rd Workshop on Language Technology for Closely Related Languages, Varieties and Dialects (VarDial)*, Osaka, Japan.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of EMNLP 2014*, Doha, Qatar.

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very Deep Convolutional Networks for Natural Language Processing. *arXiv preprint arXiv:1606.01781*.

Yoav Goldberg. 2015. A primer on neural network models for natural language processing. *arXiv preprint arXiv:1510.00726*.

Cyril Goutte, Serge Léger, Shervin Malmasi, and Marcos Zampieri. 2016. Discriminating Similar Languages: Evaluations and Explorations. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*.

Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 448–456.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR 2015*, San Diego, USA.

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

Shervin Malmasi, Marcos Zampieri, Nikola Ljubešić, Preslav Nakov, Ahmed Ali, and Jörg Tiedemann. 2016. Discriminating between Similar Languages and Arabic Dialect Identification: A Report on the Third DSL Shared Task. In *Proceedings of the 3rd Workshop on Language Technology for Closely Related Languages, Varieties and Dialects (VarDial)*, Osaka, Japan.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814.

Robert Östling. 2016. Morphological reinflection with convolutional neural networks. In *Proceedings of the 2016 Meeting of SIGMORPHON*, Berlin, Germany. Association for Computational Linguistics.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *Proceedings of ACL 2016*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Liling Tan, Marcos Zampieri, Nikola Ljubešic, and Jörg Tiedemann. 2014. Merging Comparable Data Sources for the Discrimination of Similar Languages: The DSL Corpus Collection. In *Proceedings of the 7th Workshop on Building and Using Comparable Corpora (BUCC)*, pages 11–15, Reykjavik, Iceland.

Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. 2015. A Unified Tagging Solution: Bidirectional LSTM Recurrent Neural Network with Word Embedding. *arXiv preprint arXiv:1511.00215*.

Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. 2016. Multi-Task Cross-Lingual Sequence Tagging from Scratch. *arXiv preprint arXiv:1603.06270*.

Marcos Zampieri, Liling Tan, Nikola Ljubešić, and Jörg Tiedemann. 2014. A Report on the DSL Shared Task 2014. In *Proceedings of the First Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects (VarDial)*, pages 58–67, Dublin, Ireland.

Marcos Zampieri, Liling Tan, Nikola Ljubešić, Jörg Tiedemann, and Preslav Nakov. 2015. Overview of the DSL Shared Task 2015. In *Proceedings of the Joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects (LT4VarDial)*, pages 1–9, Hissar, Bulgaria.