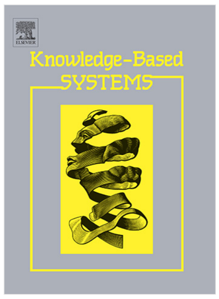


Multitask deep learning for native language identification

Vuk Habic, Alexander Semenov, Eduardo L. Pasiliao

PII: S0950-7051(20)30569-4
DOI: https://doi.org/10.1016/j.knosys.2020.106440
Reference: KNOSYS 106440

To appear in: *Knowledge-Based Systems*

Received date : 26 March 2020
Revised date : 7 September 2020
Accepted date : 8 September 2020

Multitask deep learning for native language identification

Vuk Habic

Department of Electrical Engineering, University of Florida. Email: vuk.habic.1@us.af.mil

Alexander Semenov

Faculty of Information Technology, University of Jyväskylä . Email: alexander.v.semenov@jyu.fi

Eduardo L. Pasiliao

Munitions Directorate, Air Force Research Laboratory. Email: eduardo.pasiliao@eglin.af.mil

Abstract

Identifying the native language of a person by their text written in English (L1 identification) plays an important role in such tasks as authorship profiling and identification. With the current proliferation of misinformation in social media, these methods are especially topical. Most studies in this field have focused on the development of supervised classification algorithms, that are trained on a single L1 dataset. Although multiple labeled datasets are available for L1 identification, they contain texts authored by speakers of different languages and do not completely overlap. Current approaches achieve high accuracy on available datasets, but this is attained by training an individual classifier for each dataset. Studies show that joint training of multiple classifiers on different datasets can result in sharing information between the classifiers, leading to an increase in the accuracy of both tasks. In this study, we develop a novel deep neural network (DNN) architecture for L1 classification; it is based on an adversarial multitask learning method that integrates shared knowledge from multiple L1 datasets. We propose several variants of the architecture and rigorously evaluate their

*Corresponding author

performance on multiple datasets. Our results indicate the proposed multitask architecture is more efficient in terms of classification accuracy than previously proposed methods.

Keywords: multitask learning; text classification; natural language processing; deep learning

1. Introduction

With the rapid development of the World Wide Web and social media during the early 21st century, the amount of textual data has grown exponentially. With the growing availability of data and computational resources, the past 20 years have seen increasingly rapid advances in the field of computational linguistics and natural language processing; typical tasks in these fields include text classification, document summarization, text understanding, named entity recognition, and others. The majority of textual data on the Internet consist of text written in English; it is estimated that English is used by 54% of websites[1], and it is the most common language used for communication on the Internet[2], including such social media platforms as Twitter[3]. In contrast, only about 5% of people use English as their first language. Indeed, there are approximately 6,500 languages in the world, each of them being unique in their own way, but only ten languages of these are adopted as a mother tongues by approximately 3.5 billion of people (about 50% of the world population), as estimated by Ethnologue[4]. The top three languages by number of speakers are Mandarin Chinese with 1.3 billion speakers, Spanish with 460 million speakers, and English with 379 million speakers. Many people speak more than one language; the most common second language is English, with about 753 million speakers. Further, we refer to the first language as L1 and the second language as L2.

Recently, there has been increased interest in the automatic determination

[1]https://w3techs.com/technologies/overview/content_language/all
[2]https://www.statista.com/statistics/262946/share-of-the-most-common-languages-on-the-internet/
[3]https://www.technologyreview.com/s/522376/the-many-tongues-of-twitter/
[4]https://www.ethnologue.com/statistics/size

of authors' L1 based on their text written in an L2. It is especially important to determine the L1 based on text written in English. In part, this is motivated by the proliferation of misinformation on social media, information that is often alleged to be authored by individuals whose L1 is not English. Knowing the L1 can also facilitate such tasks as authorship profiling, authorship attribution, and authorship identification. Indeed, writing in English is sometimes difficult for non-native speakers. Even after they study the grammar and mechanics, it does not have the same natural flow as their mother tongue. There are errors or certain ways that non-native speakers write in English that are often consistent among speakers of the same native tongue. Below, we provide examples of English texts written by native Arabic speakers, extracted from the Test of English as a Foreign Language 11 (TOEFL11) dataset.

Arabic: "*I agree that most advertisements make products seems to be much more better than they really are concerning the material used and satisfaction to consumers. Useing a below average materials in some products and advertise it the best*"

Chinese: "*Now advertisements have come to pervade every aspect of our lifes and, as a result, we can see advertisements every where, where in the school or in the store, where on the TV or on the newspaper. Some people think that most advertisements make products seem much better that they really are*".

A considerable amount of literature has been published on L1 identification. These studies usually approach the problem using supervised classification algorithms; for example, a report on a native language identification shared task [1] described the results of a competition between 19 teams. The goal of the competition was to develop a machine learning method that would be capable of classifying L1 with the highest accuracy for the given dataset. The teams used the TOEFL11 dataset containing essays authored by speakers of 11 languages who were taking the TOEFL exam. A predefined set of 11,000 entities were used for training, and 1,100 entities were used for evaluation; the test entities were chosen by the competition organizers. The top team achieved about 88% accuracy, with a baseline accuracy of 71%. The baseline was achieved using support vector machines (SVMs) trained on word unigrams. Data from several other studies also suggest that L1 classification can be performed with rather high accuracy.

Aside from L1 classification, there has been a growing recent trend toward the application of deep machine learning for various natural language processing (NLP) tasks; it is estimated [2] that about 70% of papers in top-level NLP conferences employ deep learning. Deep learning has been successfully applied for text classification, machine translation, text understanding, part-of-speech (PoS) tagging, and other tasks. Indeed, many teams from the competition reported in [1] used deep learning for L1 prediction. It has also been shown that very deep neural networks (DNNs) often outperform other architectures in text classification tasks [3], such as sentiment analysis or text topic prediction.

Current approaches used for L1 identification [1] demonstrate high accuracy; however, all existing studies develop these methods for a specific dataset at a time, either TOEFL11, International Corpus of Learner English (ICLE), or others; treating the problem as single-task learning.

The existing body of research on multitask learning [4] suggests that learning efficiency can increase if multiple learners of different but related tasks would learn simultaneously, provided that they share the learned information between themselves. This way, the learners would exploit the commonalities between the tasks, allowing them to mutually increase their performance. In practice, different learning tasks are represented as different but related datasets, and the goal of learning is to improve performance on all the datasets at the same time. Multitask learning architectures may be implemented as a special class of neural networks with shared parameters, at the same time having separate inputs and outputs for different datasets. Multitask learning has been previously applied to NLP; for example, adversarial multitask learning has been successfully applied for the segmentation of Chinese texts based on multiple heterogeneous datasets [5]. However, to the best of our knowledge, multitask learning has not yet been applied to the L1 identification problem.

The purpose of this paper is to develop novel L1 multitask learning architectures, based on DNNs and to perform a rigorous evaluation of them on multiple datasets. Our results show the effectiveness of multitask learning, with increased the classification accuracy compared to previously proposed models and the baselines.

To summarize, this paper makes the following contributions:

- We develop a multitask learning method that improves L1 classification accuracy on both TOEFL11 and ICLE datasets.
- We conduct extensive experiments and demonstrate the performance of our method.
- We propose multiple architectures with shared layers and experiment with adversarial strategies to force shared layers to be dataset-invariant.

The remainder of the paper is structured as follows. Section 2 discusses related literature. Section 3 discusses the objectives and introduces the problem. Section 4 describes our suggested architectures, and Section 5 describes and analyzes the experimental results. Section 6 concludes the paper.

2. Literature overview

A number of studies have proposed different architectures of DNNs for text classification tasks [3]. Representative examples include convolutional neural networks (CNNs) for sentence classification [6], and character-level CNNs [7]. These architectures are conceptually similar to CNNs used in image classification, but they use one-dimensional convolutions. In addition deep CNNs have been proposed for the categorization of texts in other papers, such as [8], [9]; a recent article [10] proposes a graph CNN for text classification, and [7] describes architecture combining a CNN and a recurrent neural network. A related area of text classification is language recognition. In [11], the authors propose sparse representation of letters and words; their design shows accuracy of 95.4%. Language recognition aims at identifying the language of a text written using that language. It is fundamentally different from L1 classification, the goal of which is to identify the native language of the person using text written in English as an input. In recent years, there has been an increasing amount of literature on L1 classification. All of it deals with classification using a single dataset, and as stated, none of it deals with multitask learning. A related area, the application of transfer learning for NLP, is described in a survey [12].

There is a general consensus on what elements to use as features for text processing, and they are character N-grams, word N-grams, and PoS N-grams. In one paper [13], the authors evaluated several architectures while considering several different versions of the N-gram features. They achieved accuracy of 0.83 using an ensemble residual neural network with word uni-grams and character 5-grams and 6-grams as input features. This was the most accurate model in the competition. Authors of [13] also outlined the different options for features and architectures that can be utilized. Bjerva et al. [13] discussed PoS tagged sentences, continuous bag of words features, and spelling features. One important note to take from [13] is that the performance of any model is lower when using external sources, such as those features mentioned above. Another takeaway is that DNNs are able to perform this classification, although traditional methods such as SVMs still appear to be better. L1 classification has also been done in [14] using SVM on N-grams, achieving 83% accuracy. In fact, most works in this field sport relatively simple model architectures and basic character and word N-grams.

In another paper [15], the authors illustrate the different features used for native language identification, including character, word, and PoS N-grams. They achieved a maximum accuracy of 81.17% .

In [16], the authors reference several techniques to achieve natural language identification. They look at unique character string identification, frequent word recognition, and bigraph/trigraph-based recognition. They found that the trigraph approach was the best. They define trigraphs the same way we define character 3-grams.

[2] discusses natural language identification using spectrogram- and cochleagram-based features for very short speech utterances. That paper differs from our

approach in that they use speech as opposed to text. Nevertheless, it was advantageous for us to look at their experiments. For example, they use SDC features that include the number of cepstral coefficients in each frame, time advance and delay for delta computation, and time shift between between consecutive frames. They use a bidirectional long short-term memory neural network however, they only achieved about 75% accuracy.

In [17], the authors create a model in order to classify the TOEFL11 dataset. They achieve about 83% classification accuracy. The features they used include words, characters, and external features such as PoS tags. The word category includes lexemes, lemmas, and PoS tags. Character features included character N-grams from 1–9 grams. Their trial method contains several models, including SVM and logistic regression.

Information from the sources cited above suggests that character N-grams and words are the most important features and that shallow neural network architectures and methods such as SVM perform well for L1 identification. None of the sources go further than the standard text prepossessing, and external sources seem to be ineffective.

3. Objectives

In this study, we address the following research questions:

- What is the architecture of a DNN, leveraging multitask learning for a joint training machine learning model on multiple L1 datasets?
- Is multitask learning beneficial for L1 identification in terms of classification accuracy?
- How does the resulting classifier compare to the state-of-the art methods?

4. Approach

Consider a dataset $\mathcal{D}_t = \{(x_i^t, y_i^t)\}_{i=1}^{n_t}$, where $x_i^t \in \mathcal{X}$ is the ith observed variable, $y_i^t \in \mathcal{Y}$ is the corresponding ith label, and t denotes a learning task,

$t \in 1, ..., T$. Our goal is to learn a function $\hat{y}_j^t = f(x_j^t, \boldsymbol{\theta})$, where $\hat{y}_j^t$ is the predicted class, and $\boldsymbol{\theta}$ is a vector of parameters. In order to find the vector of optimal parameters $\boldsymbol{\theta}$, we need to minimize the loss function $\mathcal{L}(\boldsymbol{y}, \hat{y})$ between the actual and predicted classes. In the case when $t = 1$ we have a standard supervised machine learning problem the goal of which is to predict labels for items from one dataset. However, when t is greater than 1, goal of optimization becomes finding a shared vector of parameters θ that will predict labels for all learning tasks $t \in 1, ..., T$. In this case, learning of vector $\boldsymbol{\theta}$ is conducted in all learning tasks simultaneously. In contrast to multitask learning, this problem could have been solved by training multiple vectors θ_t separately for each learning task. In our case, $\mathcal{X}$ contains essays written by non-native English speakers, and $\mathcal{Y}$ contains their native languages. In what follows, we propose multiple DNNs to find optimal $\boldsymbol{\theta}$. To begin, we describe the general architecture of the model and then show its multiple variations.

4.1. General architecture of the model

We represent each x_i^t, ith input essay from dataset t as a set of sequences of tokens, where tokens are defined by the method of document tokenization $j \in J$ and may be either words or character N-grams $\boldsymbol{W_i^t}_j^J = \{(w_1^j w_2^j ... w_{m^{i,j,t}}^j)_i^t\}_j^J$, where $m^{i,j,t}$ is the length of the resulting sequence for method j for document i in dataset t; and j indexes the method of tokenization, $j \in 1, ..., J$. Without loss of generality, these methods may denote either words or character N-grams for varying N. In the experimental section, we will demonstrate and compare results for different tokenizations j and their combinations.

Each representation is padded with zeros and loaded into the separate input layer of the neural network. This is followed by the embedding layer that corresponds to each input representation j. The embeddings are created by learning weights of lookup tables. Each lookup table maps each token w_k^j, where $k \in 1, ..., m^{i,j,t}$, of the sequence $\boldsymbol{W_i^t}_j$ for particular j to the vector $\boldsymbol{e}_k^j \in \boldsymbol{R}^{d_j}$, where d_j denotes the dimensionality of the resulting embedding vectors. In all the models described below, embeddings for each tokenization method j

are shared among the T datasets. The size of each jth lookup table is thus $(|\bigcup_{t=1}^{T}\bigcup_{i=1}^{n_t}\bigcup_{k=1}^{m^{i,j,t}} w_k^j| \times d_j)$.

The next layer is *global average pooling* connected to individual embedding of tokenization j. The input for this layer is a sequence of embeddings $\{(\boldsymbol{e}_1, ..., \boldsymbol{e}_m^{i,j,t})_i^t\}$ corresponding to input sequence $\boldsymbol{W_i^t}$ with length $m^{i,j,t}$ constructed from training item x_i^t. The output of this layer is a vector of average of embeddings' features $f_{(avg)}{}_j^{i,t} = \boldsymbol{a}_j^{i,t}{}_{d_e}$, $\boldsymbol{a}_j^{i,t}{}_{d_e} \in \boldsymbol{R}^{d_e^j}$. The dimensionality of the resulting vector is equal to d_e, that is, the dimensionality of input embedding. The following layers differ depending on the chosen architecture.

We use multiple softmax outputs corresponding to specific datasets. Cross-entropy loss function for each output, $\mathcal{L}_{CE}(y_i, \hat{y}_i) = -\sum_{k=1}^{n_k} y_i^k log(f^k(x_i, \boldsymbol{\theta}))$, where n_k is the number of classes, y_i^k is the ground truth label, and $f^k(x_i, \boldsymbol{\theta})$ is the probability that item i belongs to class k returned by the neural network. The resulting loss function is the sum of loss functions for all output layers, $\mathcal{L}_{CE_{total}}(\cdot,\cdot) = -\sum_{t=1}^{T}\sum_{k=1}^{n_k} y_{i,t}^k log(f_t^k(x_{i,t}, \boldsymbol{\theta}))$. Below, we discuss three variants of the general architecture described above and the adversarial loss function.

4.1.1. Architecture 1

In this variant of the general architecture, depicted in figure 1, embedding layers specific to the inputs are connected to the corresponding average pooling layers, and outputs of all global average pooling layers are concatenated $O_c = \bigoplus_t^T \boldsymbol{a}_j^{i,t}{}_{d_e}$ and sent to the dense layer. The dense layer is followed by softmax layers specific to each learning task t, the output of which is $f_t(\cdot) = \sigma_t(\boldsymbol{W}^T O_c + \boldsymbol{b})$.

4.1.2. Architecture 2

In this architecture (depicted in Figure 2) the outputs of the global average pooling layers corresponding to particular dataset t are individually concatenated $O_{c,t}^T = \bigoplus_{i,t}^{n_t,T} \boldsymbol{a}_j^{i,t}{}_{d_e}$ in addition to the concatenation of all their outputs, as in Architecture 1. The result of the concatenation of all average pooling

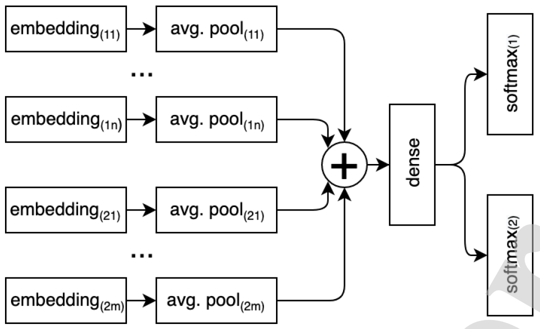


Figure 1: Architecture 1; outputs of each of the embeddings (denoted as $\mathbf{embedding_{(tj)}}$, where t indexes the dataset, and j indexes the method of tokenization) are connected to global average pooling layers (denoted as $\mathbf{avg.\ pool_{(tj)}}$). Their outputs are concatenated, and connected to Dense layer and multiple $\mathbf{softmax_{(t)}}$ outputs.

layers is connected to the dense layer as in Architecture 1, but softmax outputs receive output from the shared dense layer concatenated with $O^T_{c,t}$ that is, $f_t(\cdot) = \sigma_t(\boldsymbol{W}^T[O_{c,t} \oplus O_c] + \boldsymbol{b})$.

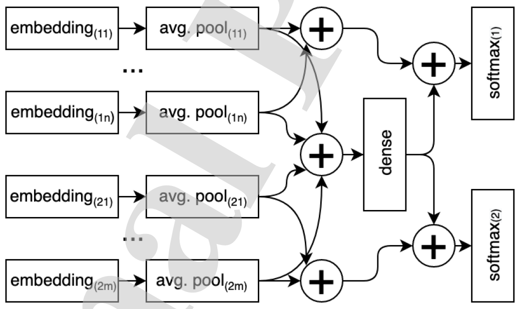


Figure 2: Architecture 2; the outputs of all average pooling layers (denoted as $\mathbf{avg.\ pool_{(tj)}}$ for the method of split j and dataset t) are concatenated and connected to the dense layer. Its output is concatenated with the concatenation of the outputs of average poolings for dataset t.

4.1.3. Architecture 3

This architecture is depicted in Figure 3; it extends Architecture 2. In addition to shared embeddings, it contains one additional shared average pooling layer per corresponding type of tokenization j. These shared average poolings are concatenated and connected to the dense layer; the output of the dense layer is concatenated with private average pooling layers and sent to the softmax outputs.

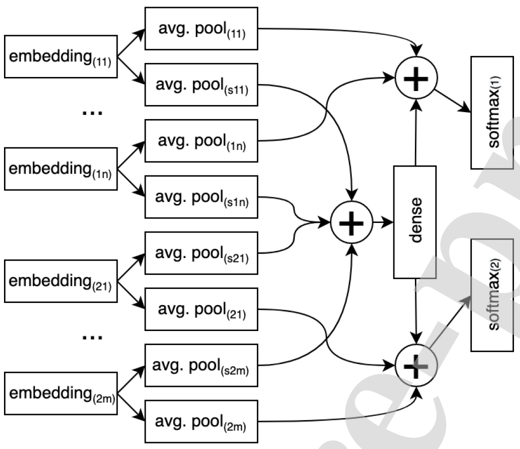


Figure 3: architecture 3; each $\mathbf{embedding_{tj}}$ is connected to two average pooling layers, one of which ($\mathbf{avg.\ pool_{(stj)}}$) is shared for particular j across the datasets $1, ..., T$. Subsequent connections are similar to that of figure 2.

4.2. Adversarial loss

As follows from the architecture descriptions above, classification in architectures 2 and 3 is based on the inputs from the *shared* dense layer with task-invariant features and *private* layers with features specific to corresponding learning tasks. In this case dense layers may contain not only task-invariant shared features, but also features private to the tasks $1, ..., T$. It could be beneficial to ensure that shared layers contain only features that are shared between all tasks, and that private layers contain only task-specific features [5]. This can be achieved by guaranteeing that it would not be possible to perform correct classification of input data based solely on the shared layer. In order to achieve this, we introduce adversarial loss into architectures 2 and 3. Adversarial loss is represented as a softmax layer connected to a shared dense layer, followed by negative cross-entropy loss. This loss function is optimized together with the main loss functions. Thus, total loss becomes:

$$\mathcal{L}_{CE_{adv\_total}}(\cdot,\cdot) = -\sum_{t=1}^{T}\sum_{k=1}^{n_k} y_{i,t}^{k} log(f_{\sigma,t}^{k}(x_{i,t},\boldsymbol{\theta})) + \sum_{t=1}^{T}\sum_{k=1}^{n_k} y_{i,t}^{k} log(f_{adv,t}^{k}(x_{i,t},\boldsymbol{\theta})),$$

where $f_{\sigma,t}$ is the output of the sigmoid function used for classification of the dataset t, and $f_{adv,t}$ is the adversarial output corresponding to t. The loss is jointly optimized using a stochastic gradient descent algorithm.

5. Results

In this section, we describe our datasets and report the experimental results.

5.1. Datasets

We use two large and very specific data sets. They are, the TOEFL11 and the ICLE. TOEFL11 was developed by the Educational Testing Service and is comprised of 12,100 English essays written by speakers of 11 non-English native languages as part of an international test of academic English proficiency (TOEFL). ICLE is the result of collaboration with a wide range of partner universities internationally, with argumentative essays written by intermediate to advanced learners of English who have different mother tongues. TOEFL includes 11 different mother tongues, while ICLE includes 16. TOEFL11 includes Arabic, Chinese, French, German, Hindi, Italian, Japanese, Korean, Spanish, Telugu, and Turkish authors. ICLE includes Bulgarian, Chinese, Czech, Dutch (from Netherlands and from Belguim), Finnish, French, German, Italian, Japanese, Norwegian, Polish, Russian, Spanish, Swedish, Turkish and Tswana. Most of these languages have a relatively large speaker base, and it would not be difficult to obtain samples of texts from authors with these mother tongues. Table 1 provides a summary of the datasets' characteristics. We can see that TOEFL11 has almost twice as many essays as ICLE, but the ICLE essays are generally longer. Table 1 also displays the number of character 3- and 4- grams in the datasets; in our architectures, we use all available data and do not trim the N-grams and words.

Property	ICLE	TOEFL11
number of essays	6,085	12,100
number of distinct labels	17	11
number of distinct words	63,358	73,807
number of distinct 3-grams	22,102	20,511
number of distinct 4-grams	106,131	99,991
average number of characters in essay	3547.9	1766.4
average number of words in essay	618.0	314.9

Table 1: Summary of the descriptive characteristics of the TOEFL11 and ICLE datasets

5.2. Experimental design

We implemented the architectures proposed above and developed two variants of each, one having only words and character 4-grams as inputs, and the other having character 3-grams as additional input representation. Next, we implemented the same models but excluded the final dense layer. To integrate the adversarial loss function, we included in our experiment architectures 2 and 3 with adversarial loss layers attached to their dense layers. These modifications resulted in 16 different variants of the neural networks, listed in Table 2. We used multiple baseline methods. The first one, referred to as "*Shallow ANN*" in Table 2 is a shallow neural network from the competition [1] as described in [18] with and without the final dense layer. We also implemented shallow ANN with one-dimensional convolutions attached to input embeddings (model 21). Further, we implemented a random forest classifier with 1000 estimators, with 5,000 input features, processed by TF-IDF vectorization and linear SVM with 200 features. The implemented models are listed in Table 2.

We implemented our architectures using Keras with TensorFlow backend. Two datasets, TOEFL11 and ICLE, were used as input data. Before tokenization, we converted all texts to lowercase. Stopwords were not removed. We used $d_t = 25$ for all the embeddings and the dense layer with 64 neurons and a ReLU activation function as hyperparameters. Further, each softmax output had a dropout layer with p = 0.5 before it. We trained the models for 40 epochs and used *adam* as the optimization algorithm. The code was executed on a machine with four Tesla V100-SXM2-16GB GPUs and 32 CPU cores. We used k-fold validation with k = 5, that is, a traditional 80/20 split. Different from our setup, competition results listed in [1] were evaluated using about 9% of the TOEFL11 data; the rest was used for training.

5.3. Experimental results and discussion

#	**Name**	**inputs**
	No adversarial loss	
1	Architecture 1 with Dense	words, 4-grams
2	Architecture 1 with Dense	words, 3-grams, 4-grams
3	Architecture 2 with Dense	words, 4-grams
4	Architecture 2 with Dense	words, 3-grams, 4-grams
5	Architecture 3 with Dense	words, 4-grams
6	Architecture 3 with Dense	words, 3-grams, 4-grams
7	Architecture 1 w/o Dense	words, 4-grams
8	Architecture 1 w/o Dense	words, 3-grams, 4-grams
9	Architecture 2 w/o Dense	words, 4-grams
10	Architecture 2 w/o Dense	words, 3-grams, 4-grams
11	Architecture 3 w/o Dense	words, 4-grams
12	Architecture 3 w/o Dense	words, 3-grams, 4-grams
	With adversarial loss	
13	Architecture 2 with Dense	words, 4-grams
14	Architecture 3 with Dense	words, 4-grams
15	Architecture 2 with Dense	words, 3-grams, 4-grams
16	Architecture 3 with Dense	words, 3-grams, 4-grams
	Baselines	
17	Shallow ANN (w/o Dense)	words, 4-grams
18	Shallow ANN (with Dense)	words, 4-grams
19	Shallow ANN(w/o Dense)	words, 3-grams, 4-grams
20	Shallow ANN (with Dense)	words, 3-grams,4-grams
21	Shallow ANN (with Convolutional layer)	words, 3-grams,4-grams
22	Random Forest	3-grams, 4-grams
23	Random Forest	words
24	SVM	3-grams, 4-grams

Table 2: Description of the implemented neural network architectures, and baselines; and input data types

#	Accuracy,TOEFL	F1, TOEFL	Accuracy, ICLE	F1, ICLE
No adversarial loss				
1	0.753 ± 0.00015	0.752 ± 0.00017	0.751 ± 0.00015	0.702 ± 0.00037
2	0.753 ± 0.00015	0.753 ± 0.00014	0.746 ± 0.00011	0.700 ± 0.00019
3	0.806 ± 0.00016	0.805 ± 0.00019	0.846 ± 0.00038	0.815 ± 0.00047
4	0.812 ± 0.00018	0.810 ± 0.00022	0.834 ± 0.00015	0.803 ± 0.00022
5	0.850 ± 0.00006	0.849 ± 0.00006	0.960 ± 0.00005	0.952 ± 0.00008
6	0.803 ± 0.00017	0.803 ± 0.00016	0.819 ± 0.00093	0.783 ± 0.00107
7	0.851 ± 0.00003	0.851 ± 0.00004	0.898 ± 0.00011	0.875 ± 0.00019
8	$\mathbf{0.851 \pm 0.00002}$	$\mathbf{0.851 \pm 0.00002}$	0.905 ± 0.00007	0.885 ± 0.00015
9	0.839 ± 0.00007	0.839 ± 0.00008	$\mathbf{0.973 \pm 0.00001}$	$\mathbf{0.968 \pm 0.00002}$
10	0.847 ± 0.00003	0.847 ± 0.00003	0.958 ± 0.00001	0.949 ± 0.00002
11	0.848 ± 0.00003	0.848 ± 0.00004	0.956 ± 0.00005	0.946 ± 0.00008
12	0.847 ± 0.00006	0.846 ± 0.00006	0.961 ± 0.00004	0.954 ± 0.00008
With adversarial loss				
13	0.830 ± 0.00003	0.830 ± 0.00004	0.864 ± 0.00035	0.837 ± 0.00056
14	0.827 ± 0.00010	0.827 ± 0.00009	0.851 ± 0.00013	0.820 ± 0.00015
15	0.830 ± 0.00006	0.830 ± 0.00007	0.871 ± 0.00013	0.841 ± 0.00027
16	0.837 ± 0.00006	0.836 ± 0.00009	0.878 ± 0.00032	0.853 ± 0.00048
Baselines				
17	0.790 ± 0.00004	0.790 ± 0.00006	0.950 ± 0.00001	0.938 ± 0.00002
18	0.602 ± 0.00166	0.605 ± 0.00165	0.933 ± 0.00163	0.915 ± 0.00340
19	0.789 ± 0.00001	0.789 ± 0.00002	0.962 ± 0.00003	0.953 ± 0.00003
20	0.631 ± 0.00052	0.634 ± 0.00043	0.954 ± 0.00003	0.943 ± 0.00005
21	0.657 ± 0.00008	0.656 ± 0.00010	0.701 ± 0.00086	0.654 ± 0.00071
22	0.500 ± 0.00020	0.493 ± 0.00021	0.668 ± 0.00006	0.599 ± 0.00001
23	0.570 ± 0.00038	0.566 ± 0.00034	0.716 ± 0.00014	0.659 ± 0.00024
24	0.339 ± 0.00011	0.336 ± 0.00010	0.300 ± 0.00025	0.147 ± 0.00006

Table 3: Experimental results: accuracy and F1 score ± standard deviation for the TOEFL11 and ICLE datasets (k-fold validation with $k = 5$) for proposed models and baselines, listed in table 2. Maximal values of corresponding metrics are highlighted in bold.

The experimental results are presented in Table 3. The table lists the accuracy for TOEFL11 averaged over 5 folds, the F1 score for TOEFL11, after macro averaging over 5 folds, and corresponding values for the ICLE dataset with standard deviations. We can make the following observations:

1. The best results in terms of accuracy vis-à-vis the TOEFL11 dataset are achieved by Architecture 1 without a dense layer with words, 3-grams, and 4-grams as inputs (model number 8); its accuracy is 0.851. The best performing model for ICLE is Architecture 2, also without the final dense

layer, with words and 3-grams as inputs (model number 9). Its accuracy is 0.973. The accuracy of model 8 vis-à-vis TOEFL11 dataset is 7.7% higher than the best-performing baseline (model number 17, 0.79 accuracy). The accuracy of model 9 for the ICLE dataset is higher than the best-performing ICLE baseline (model number 19) by 1.11%. Both multitask architectures significantly outperform the baselines, with the $p-$value of the $t-$test for comparing TOEFL11 model 8 accuracy with baseline 17 being to 1.58×10^{-7} and the corresponding $p-$value of ICLE accuracy for model 9 and that of baseline 19 being 0.0047. Our experiment shows that neural network-based multitask architectures outperform single-task architectures.

2. For the TOEFL11 dataset all, but two of the proposed multitask architectures outperform all baselines. Table 1 shows that model numbers 1 and 2 are outperformed by baselines 17 and 19. Models 1 and 2 implement architecture 1 with a dense layer at the end of the neural network. We can see from Table 3, architectures 1–3 without a dense layer demonstrate higher accuracy than the baselines for TOEFL11; however, the baseline demonstrates higher accuracy than the proposed models for the ICLE dataset using 3- and 4- gram inputs. Otherwise proposed models perform better. We can also see that the shallow neural network with convolutional layers (baseline 21) performs worse than the baselines without a dense layer (baselines 17 and 19).
3. Most of the models having a dense layer at the end perform worse than the same models without this layer. However, adversarial loss greatly increases the accuracy of these models, but unexpectedly, architectures with no dense layer are generally better.
4. For the TOEFL11 dataset, Architecture 1 without a final dense layer is the best, followed by Architecture 3 without a final dense layer, followed by Architecture 2, without a final dense layer. For the ICLE dataset, the second best accuracy is shown by baseline 19, and the third best performing model is architecture 3 without a final dense layer. However,

the difference between accuracy values for the latter two models is not statistically significant, based on a t-test.

5. In general, the number of parameters of one multitask architecture equals to combined number of parameters of both corresponding single-task TOEFL11 and ICLE baselines. This is also reflected in the time of training, which is longer for multitask models. We have observed that time of training of multitask models is in average twice longer.
6. Improvement in F1 score goes at par with the accuracy.

To study the best-performing model further, we trained the model implementing Architecture 1 without the dense layer, accepting words and character 4-grams as input for 80 epochs on the 90.9% of training data, and tested it on the remaining 9.1%. This roughly corresponds to the setup of the task of L1 classification competition described in [1]. The maximum accuracy demonstrated by the developed model for the TOEFL11 dataset is 88.75 %, while the accuracy of the top-performing model from [1] was 88.18 %, the model is described in [19]. Figure 4 depicts the dependence of accuracy on the training epoch for training and validation datasets. In summary, these results show that multitask learning is beneficial for L1 identification tasks and that it is capable of outperforming state-of-the-art methods.

In order to analyze the results, we constructed confusion matrices from the validation dataset containing 9.1% of the data. Figure 5 shows the confusion matrix for the TOEFL11 dataset, Figure 6 shows confusion matrix for the ICLE dataset. Matrices are visualized as heatmaps. We can see that for TOEFL11 the most often confusing for the model is Japanese versus Korean, and Japanese vs Chinese. Italian speakers are sometimes confused with Spanish speakers. Although Korean and Japanese are not related [20], they belong to the Japonic language family. Therefore, similar mistakes in English could be typical for speakers of these languages; this may explain the observation that texts written by speakers of these languages are confused by the model. Italian and Spanish are both Romance languages with similarities in their vocabulary; this may

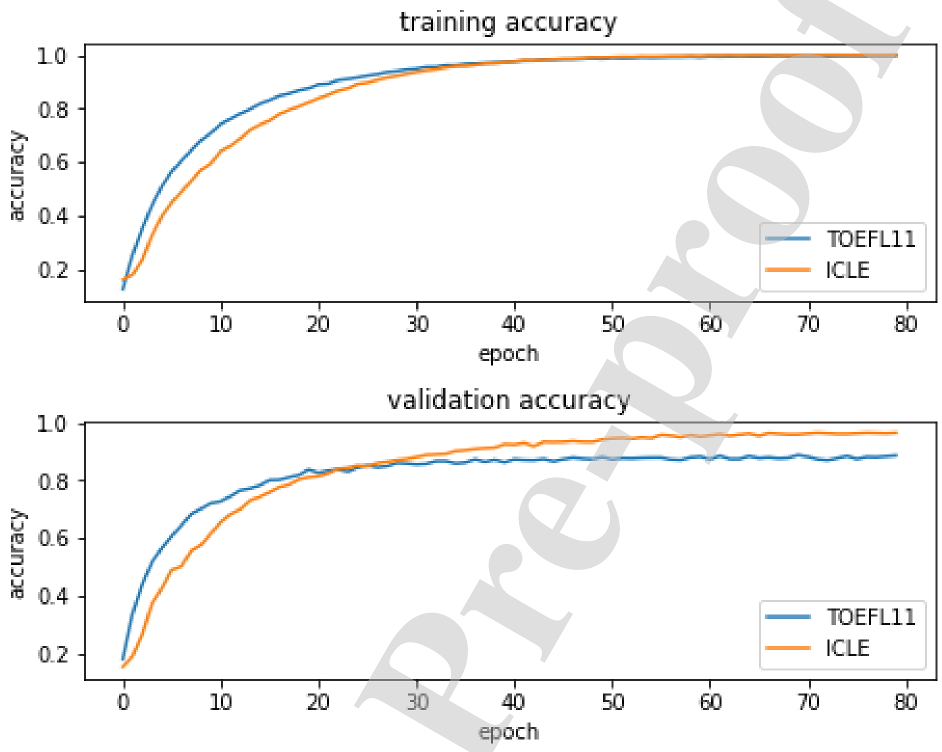


Figure 4: Accuracy of the model implementing Architecture 2 (without a dense layer) with words and 4-grams as input, trained for 80 epochs. The top figure shows training accuracy, and the figure on the bottom shows validation accuracy. The blue line represents accuracy for the TOEFL11 dataset, and the orange one represents accuracy for the ICLE dataset. We can see that training accuracy tends to be 100% at around the 50th training epoch.

explain the confusion of the model. As opposed to TOEFL11, there is no clear indication of specific confusion of the model for the ICLE dataset (see Figure 6). This may be explained by the fact that accuracy vis-à-vis ICLE is much higher than vis-à-vis TOEFL11, and even simple baselines achieve high accuracy (see Table 3) on this dataset. Interestingly, there is no confusion between related languages, such as Bulgarian, Russian, Polish, and Czech; also different from TOEFL11, there is no confusion between Chinese and Japanese.

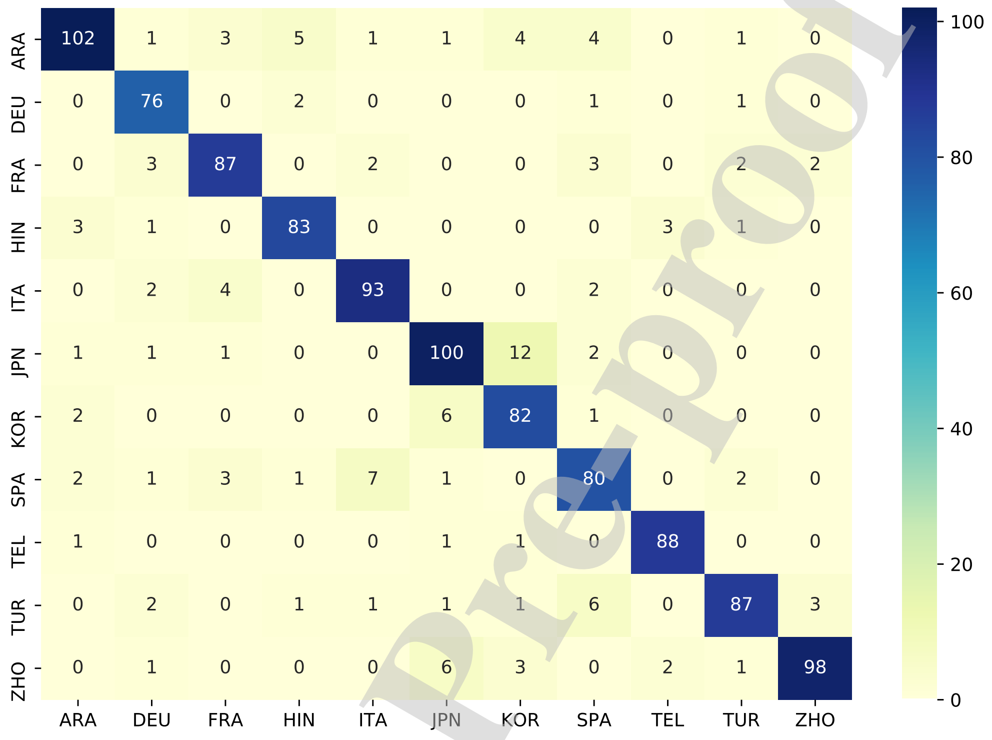
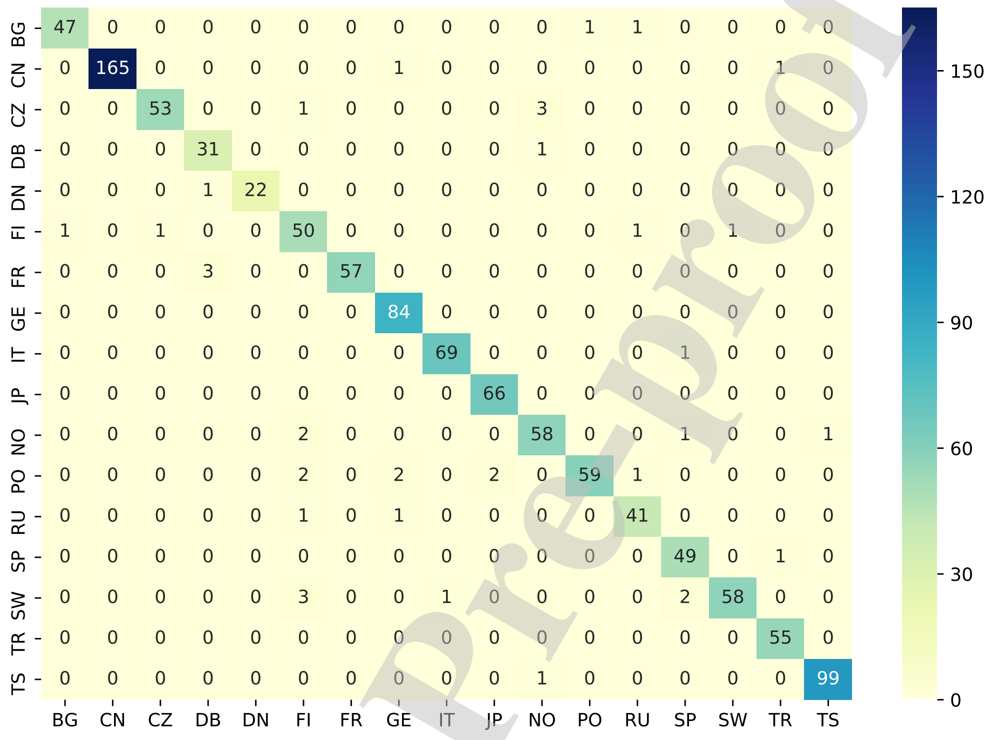


Figure 5: Confusion matrix of the model on TOEFL part of validation dataset (9.1% of TOEFL11).

6. Conclusions

This paper presents novel multitask learning methods for L1 language classification. We proposed three neural network architectures that are capable of sharing the knowledge between multiple heterogeneous input datasets. We evaluate our architectures using TOEFL11 and ICLE datasets; based on the results, we conclude that our approach shows the benefit of multitask learning is more accurate than state-of-the-art methods.

In future work, we would like to investigate further tuning of the models' hyper parameters to increase the classification accuracy. We would also like to experiment with additional input features, such as N-grams with different values of N.



	BG	CN	CZ	DB	DN	FI	FR	GE	IT	JP	NO	PO	RU	SP	SW	TR	TS
BG	47	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
CN	0	165	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
CZ	0	0	53	0	0	1	0	0	0	0	3	0	0	0	0	0	0
DB	0	0	0	31	0	0	0	0	0	0	1	0	0	0	0	0	0
DN	0	0	0	1	22	0	0	0	0	0	0	0	0	0	0	0	0
FI	1	0	1	0	0	50	0	0	0	0	0	0	1	0	1	0	0
FR	0	0	0	3	0	0	57	0	0	0	0	0	0	0	0	0	0
GE	0	0	0	0	0	0	0	84	0	0	0	0	0	0	0	0	0
IT	0	0	0	0	0	0	0	0	69	0	0	0	0	1	0	0	0
JP	0	0	0	0	0	0	0	0	0	66	0	0	0	0	0	0	0
NO	0	0	0	0	0	2	0	0	0	0	58	0	0	1	0	0	1
PO	0	0	0	0	0	2	0	2	0	2	0	59	1	0	0	0	0
RU	0	0	0	0	0	1	0	1	0	0	0	0	41	0	0	0	0
SP	0	0	0	0	0	0	0	0	0	0	0	0	0	49	0	1	0
SW	0	0	0	0	0	3	0	0	1	0	0	0	0	2	58	0	0
TR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	55	0
TS	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	99

Figure 6: Confusion matrix of the model on ICLE part of validation dataset (9.1% of ICLE)

7. Acknowledgements

This work was funded in part by the US Air Force Research Laboratory (AFRL) European Office of Aerospace Research and Development (grant no. FA9550-17-1-0030), and the AFRL Mathematical Modeling and Optimization Institute.

References

[1] S. Malmasi, K. Evanini, A. Cahill, J. Tetreault, R. Pugh, C. Hamill, D. Napolitano, Y. Qian, A report on the 2017 native language identification shared task, in: Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications, Association

for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 62–75. doi:10.18653/v1/W17-5007.
URL https://www.aclweb.org/anthology/W17-5007

[2] T. Young, D. Hazarika, S. Poria, E. Cambria, Recent trends in deep learning based natural language processing [review article], IEEE Computational Intelligence Magazine 13 (3) (2018) 55–75. doi:10.1109/MCI.2018.2840738.

[3] A. Conneau, H. Schwenk, L. Barrault, Y. Lecun, Very deep convolutional networks for text classification, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, Association for Computational Linguistics, Valencia, Spain, 2017, pp. 1107–1116.
URL https://www.aclweb.org/anthology/E17-1104

[4] R. Caruana, Multitask learning, Machine Learning 28 (1) (1997) 41–75. doi:10.1023/A:1007379606734.
URL https://doi.org/10.1023/A:1007379606734

[5] X. Chen, Z. Shi, X. Qiu, X. Huang, Adversarial multi-criteria learning for Chinese word segmentation, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 1193–1203. doi:10.18653/v1/P17-1110.
URL https://www.aclweb.org/anthology/P17-1110

[6] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1746–1751. doi:10.3115/v1/D14-1181.
URL https://www.aclweb.org/anthology/D14-1181

[7] X. Zhang, J. Zhao, Y. LeCun, Character-level convolutional networks for text classification, in: Proceedings of the 28th International Conference on

Neural Information Processing Systems - Volume 1, NIPS'15, MIT Press, Cambridge, MA, USA, 2015, p. 649–657.

[8] C. dos Santos, M. Gatti, Deep convolutional neural networks for sentiment analysis of short texts, in: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, Dublin City University and Association for Computational Linguistics, Dublin, Ireland, 2014, pp. 69–78.
URL `https://www.aclweb.org/anthology/C14-1008`

[9] P. Wang, J. Xu, B. Xu, C. Liu, H. Zhang, F. Wang, H. Hao, Semantic clustering and convolutional neural network for short text categorization, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), Association for Computational Linguistics, Beijing, China, 2015, pp. 352–357. `doi:10.3115/v1/P15-2058`.
URL `https://www.aclweb.org/anthology/P15-2058`

[10] L. Yao, C. Mao, Y. Luo, Graph convolutional networks for text classification, CoRR abs/1809.05679. `arXiv:1809.05679`.
URL `http://arxiv.org/abs/1809.05679`

[11] M. Imani, J. Hwang, T. Rosing, A. Rahimi, J. M. Rabaey, Low-power sparse hyperdimensional encoder for language recognition, IEEE Design Test 34 (6) (2017) 94–101.

[12] L. Qi, Literature survey: domain adaptation algorithms for natural language processing, Department of Computer Science The Graduate Center, The City University of New York, 2012.

[13] J. Bjerva, G. Grigonyte, R. Östling, B. Plank, Neural networks and spelling features for native language identification, in: EMNLP, Association for Computational Linguistics, 2017, pp. 235–239.

[14] A. Kulmizev, B. Blankers, J. Bjerva, M. Nissim, G. van Noord, B. Plank, M. Wieling, The power of character n-grams in native language identification, in: Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 382–389. doi:10.18653/v1/W17-5043.
URL https://www.aclweb.org/anthology/W17-5043

[15] T. Mizumoto, Y. Hayashibe, K. Sakaguchi, M. Komachi, Y. Matsumoto, Naist at the nli 2013 shared task, in: Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications, 2013, pp. 134–139.

[16] C. Souter et al., Natural language identification using corpus-based models, HERMES - Journal of Language and Communication in Business 7 (13) (2017) 183–203. doi:10.7146/hjlcb.v7i13.25083.
URL https://tidsskrift.dk/her/article/view/25083

[17] S. Jarvis, Y. Bestgen, S. Pepper, Maximizing classification accuracy in native language identification, in: Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications, Association for Computational Linguistics, Atlanta, Georgia, 2013, pp. 111–118.
URL https://www.aclweb.org/anthology/W13-1714

[18] Y. Sari, M. Rifqi Fatchurrahman, M. Dwiastuti, A shallow neural network for native language identification with character n-grams, in: Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 249–254. doi:10.18653/v1/W17-5027.
URL https://www.aclweb.org/anthology/W17-5027

[19] A. Cimino, F. Dell'Orletta, Stacked sentence-document classifier approach for improving native language identification, in: BEA@EMNLP, 2017.

[20] A. Vovin, From koguryŏ to t'amna: Slowly riding to the south with speakers of proto-korean, Korean Linguistics 15 (2) (2013) 222–240. doi:https://doi.org/10.1075/kl.15.2.03vov.
URL https://www.jbe-platform.com/content/journals/10.1075/kl.15.2.03vov

All authors contributed to writing paper, designing and analyzing the experiments. A. Semenov executed final version of the experiments. All authors reviewed the paper.