Natural Language Engineering (2019), **25**, pp. 561–583
doi:10.1017/S135132491900038X

CAMBRIDGE UNIVERSITY PRESS

ARTICLE

Language model adaptation for language and dialect identification of text

T. Jauhiainen*, K. Lindén and H. Jauhiainen

Department of Digital Humanities, University of Helsinki, Helsinki 00014, Finland
*Corresponding author. Email: tommi.jauhiainen@helsinki.fi

(Received 5 November 2018; revised 12 April 2019; accepted 17 April 2019; first published online 31 July 2019)

Abstract

This article describes an unsupervised language model (LM) adaptation approach that can be used to enhance the performance of language identification methods. The approach is applied to a current version of the HeLI language identification method, which is now called HeLI 2.0. We describe the HeLI 2.0 method in detail. The resulting system is evaluated using the datasets from the German dialect identification and Indo-Aryan language identification shared tasks of the VarDial workshops 2017 and 2018. The new approach with LM adaptation provides considerably higher F1-scores than the basic HeLI or HeLI 2.0 methods or the other systems which participated in the shared tasks. The results indicate that unsupervised LM adaptation should be considered as an option in all language identification tasks, especially in those where encountering out-of-domain data is likely.

Keywords: Language identification; Language model adaptation

1. Introduction

Automatic language identification of text has been researched since the 1960s. It has been considered a subspecies of general text categorization and most of the methods used are similar to those used in categorizing text according to their topic. However, deep learning techniques have not proven to be as efficient in language identification as they have been in other categorization tasks (Medvedeva, Kroon, and Plank 2017; Ali 2018).

For the past 6 years, we have been developing a language identifying method, which we call HeLI, for the Finno-Ugric languages and the Internet project (Jauhiainen, Jauhiainen, and Lindén 2015a). The HeLI method is a supervised general-purpose language identification method relying on observations of word and character *n*-gram frequencies from a language labeled corpus. The method is similar to naive Bayes (NB) when using only relative frequencies of words as probabilities. Unlike NB, it uses a back-off scheme to approximate the probabilities of individual words if the words themselves are not found in the language models (LMs). As LMs, we use word unigrams and character level *n*-grams. The optimal combination of the LMs used with the back-off scheme depends on the situation and is determined empirically using a development set. The latest evolution of the HeLI method, HeLI 2.0, is described in this article.

One of the remaining difficult cases in language identification is the identification of language varieties or dialects. The task of language identification is less difficult if the set of possible languages does not include very similar languages. If we try to discriminate between very close languages or dialects, the task becomes increasingly more difficult (Tiedemann and Ljubešić 2012). The first ones to experiment with language identification for close languages were Sibun and

Reynar (1996) who had Croatian, Serbian, and Slovak as part of their language repertoire. The differences between definitions of dialects and languages are not usually clearly defined, at least not in terms which would be able to help us automatically decide whether we are dealing with languages or dialects. Furthermore, the methods used for dialect identification are most of the time exactly the same as for general language identification. During the last 5 years, the state-of-the-art language identification methods have been put to the test in a series of shared tasks as part of VarDial workshops (Zampieri *et al.* 2014; Zampieri *et al.* 2015; Malmasi *et al.* 2016; Zampieri *et al.* 2017; Zampieri *et al.* 2018). We have used the HeLI method and its variations in the shared tasks of the four latest VarDial workshops (Jauhiainen, Jauhiainen, and Lindén 2015b; Jauhiainen, Lindén, and Jauhiainen 2016; 2017a; Jauhiainen, Jauhiainen, and Lindén 2018a; 2018b; 2018c). The HeLI method has proven to be robust and it competes well with other state-of-the-art language identification methods.

Another remaining difficult case in language identification is the situation when the training data are not in the same domain as the data to be identified. Being out-of-domain can mean several things. For example, the training data can be from a different genre, different time period, and/or produced by different writers than the data to be identified. The identification accuracies are considerably lower on out-of-domain data (Li, Baldwin, and Cohn 2018) depending on the degree of out-of-domainness. The extreme example of in-domainness is when the training data and test data are from different parts of the same text, as it has been in several language identification experiments in the past (Vatanen, Väyrynen, and Virpioja 2010; Brown 2012; Brown 2013; Brown 2014). Classifiers can be more or less sensitive to the domain differences between the training and the testing data depending on the machine learning methods used (Blodgett, Wei, and O'Connor 2017). One way to diminish the effects of the phenomena is to create domain-general LMs using adversarial supervision which reduces the amount of domain-specific information in the LMs (Li *et al.* 2018). We suggest that another way to do this is to use active LM adaptation.

In LM adaptation, we use the unlabeled mystery text itself to enhance the LMs used by a language identifier. The LM adaptation scheme introduced in this article is not an off-line adaptation technique used to adapt an existing identifier to a particular domain; it is a general on-line adaptation technique that is used each time the language of a new text is to be identified. The language identification method used in combination with the LM adaptation approach presented in this article must be able to produce a confidence score of how well the identification has performed. As the LMs are updated regularly while the identification is ongoing, the approach also benefits from the language identification method being non-discriminative. If the method is non-discriminative, all the training materials do not have to be reprocessed when adding new information into the LMs. To the best of our knowledge, LM adaptation has not been used in language identification of digital text before the first versions of the method presented in this article were used in the shared tasks of the 2018 VarDial workshop (Jauhiainen *et al.* 2018a; 2018b; 2018c). Concurrently with our current work, Ionescu and Butnaru (2018) presented an adaptive version of the Kernel Ridge Classifier which they evaluated on the Arabic Dialect Identification (ADI) dataset from the 2017 VarDial workshop (Zampieri *et al.* 2017).

In this article, we first review the previous work relating to German dialect identification, Indo-Aryan language identification, and LM adaptation (Section 2). We then present the methods used in the article: the HeLI 2.0 method for language identification, three confidence estimation methods, and the algorithm for LM adaptation (Section 3). In Section 4, we introduce the datasets used for evaluating the methods, and in Section 5, we evaluate the methods and present the results of the experiments.

2. Related work

The first automatic language identifier for digital text was described by Mustonen (1965). Since this first article, hundreds of conference and journal articles describing language identification

https://doi.org/10.1017/S135132491900038X Published online by Cambridge University Press

experiments and methods have been published. For a recent survey on language identification and the methods used in the literature, see Jauhiainen *et al.* (2018d). The HeLI method was first published in 2010 as part of a master's thesis (Jauhiainen 2010) and has since been used, outside the VarDial workshops, for language set identification (Jauhiainen, Lindén, and Jauhiainen 2015*c*) as well as general language identification with a large number of languages (Jauhiainen, Lindén, and Jauhiainen 2017b). One of the main strengths of the HeLI method is that it uses the word-level models and is still able to graciously deal with any possible out of vocabulary words that it encounters.

2.1 German dialect identification

German dialect identification has earlier been considered by Scherrer and Rambow (2010), who used a lexicon of dialectal words. Hollenstein and Aepli (2015) experimented with a perplexity-based language identifier using character trigrams. They reached an average F-score of 0.66 on sentence level distinguishing between 5 German dialects.

The results of the first shared task on German dialect identification (*GDI*) are described by Zampieri *et al.* (2017). Ten teams submitted results on the task utilizing a variety of machine learning methods used for language identification. The team *MAZA* (Malmasi and Zampieri 2017) experimented with different types of support vector machine (SVM) ensembles: plurality voting, mean probability, and meta-classifier. The meta-classifier ensemble using the Random Forest algorithm for classification obtained the best results. The team *CECL* (Bestgen 2017) used SVMs as well, and their best results were obtained using an additional procedure to equalize the number of sentences assigned to each category. Team *CLUZH* experimented with NB, conditional random fields (CRF), as well as a majority voting ensemble consisting of NB, CRF, and SVM (Clematide and Makarov 2017). Their best results were reached using CRF. Team *qcri_mit* used an ensemble of two SVMs and a stochastic gradient classifier (SGD). Team *unibuckernel* experimented with different kernels using kernel ridge regression (KRR) and kernel discriminant analysis (KDA) (Ionescu and Butnaru 2017). They obtained their best results using KRR based on the sum of three kernels. Team *tubasfs* (Çöltekin and Rama 2017) used SVMs with features weighted using sublinear TF-IDF (product of term frequency and inverse document frequency) scaling. Team *ahaqst* used cross entropy (CE) with character and word *n*-grams (Hanani, Qaroush, and Taylor 2017). Team *Citius_Ixa_Imaxin* used perplexity with different features (Gamallo, Pichel, and Alegria 2017). Team *XAC_Bayesline* used NB (Barbaresi 2017) and team *deepCybErNet* Long Short-Term Memory (LSTM) neural networks. We report the F1-scores obtained by the teams in Table 8 together with the results presented in this article.

The second shared task on German dialect identification was organized as part of the 2018 VarDial workshop (Zampieri *et al.* 2018). We participated in the shared task with an earlier version of the method described in this article and our submission using the LM adaptation scheme reached a clear first place (Jauhiainen *et al.* 2018a). Seven other teams submitted results on the shared task. Teams *Twist Bytes* (Benites *et al.* 2018), *Tübingen-Oslo* (Çöltekin, Rama, and Blaschke 2018), and *GDI_classification* (Ciobanu, Malmasi, and Dinu 2018a) used SVMs. The team *safina* used convolutional neural networks (CNN) with direct one-hot encoded vectors, with an embedding layer, as well as with a Gated Recurrent Unit (GRU) layer (Ali 2018). The team *LaMa* used a voting ensemble of eight classifiers. The best results for the team *XAC* were achieved using NB, but they experimented with Ridge regression and SGD classifiers as well (Barbaresi 2018). The team *dkosmajac* used normalized Euclidean distance. After the shared task, the team *Twist Bytes* was able to slightly improve their F1-score by using a higher number of features (Benites *et al.* 2018). However, the exact number of included features was not determined using the development set, but it was the optimal number for the test set. Using the full set of features resulted again in a lower score. We report the F1-scores obtained by the teams in Table 11 together with the results obtained in this article.

https://doi.org/10.1017/S135132491900038X Published online by Cambridge University Press

2.2 Language identification for Devanagari script

Language identification research in distinguishing between languages using the Devanagari script is much more uncommon than for the Latin script. However, some research was done already before the Indo-Aryan Language Identification shared task (*ILI*) at VarDial 2018 (Zampieri *et al.* 2018). Kruengkrai *et al.* (2006) presented results from language identification experiments between ten Indian languages, including four languages written in Devanagari: Sanskrit, Marathi, Magahi, and Hindi. For the ten Indian languages they obtained over 90% accuracy with 70-byte long mystery text sequences. As language identification method, they used SVMs with string kernels. Murthy and Kumar (2006) compared the use of LMs based on bytes with models based on aksharas. Aksharas are the syllables or orthographic units of the Brahmi scripts (Vaid and Gupta 2002). After evaluating the language identification between different pairs of languages, they concluded that the akshara-based models perform better than byte-based. They used multiple linear regression as the classification method.

Sreejith, Indu, and Reghu Raj (2013) tested language identification with Markovian character and word *n*-grams from one to three with Hindi and Sanskrit. A character bigram-based language identifier fared the best and managed to gain an accuracy of 99.75% for sentence-sized mystery texts. Indhuja *et al.* (2014) continued the work of Sreejith *et al.* (2013) investigating the language identification between Hindi, Sanskrit, Marathi, Nepali, and Bhojpuri. In a similar fashion, they evaluated the use of Markovian character and word *n*-grams from one to three. For this set of languages, word unigrams performed the best, obtaining 88% accuracy with the sentence-sized mystery texts.

Bergsma *et al.* (2012) collected tweets in three languages written with the Devanagari script: Hindi, Marathi, and Nepali. They managed to identify the language of the tweets with 96.2% accuracy using a logistic regression (LR) classifier (Hosmer, Lemeshow, and Sturdivant 2013) with up to 4-grams of characters. Using an additional training corpus, they reached 97.9% accuracy with the A-variant of prediction by partial matching. Later, Pla and Hurtado (2017) experimented with the corpus of Bergsma *et al.* (2012). Their approach using words weighted with TF-IDF and SVMs reached 97.7% accuracy on the tweets when using only the provided tweet training corpus. Hasimu and Silamu (2018) included the same three languages in their test setting. They used a two-stage language identification system where the languages were first identified as a group using Unicode code ranges. In the second stage, the languages written with the Devanagari script were individually identified using SVMs with character bigrams. Their tests resulted in an F1-score of 0.993 within the group of languages using Devanagari with 700 best distinguishing bigrams. Indhuja *et al.* (2014) provided test results for several different combinations of the five languages, and for the set of languages used by Hasimu and Silamu (2018), they reached 96% accuracy with word unigrams.

Rani *et al.* (2018) described a language identification system which they used for discriminating between Hindi and Magahi. Their language identifier using lexicons and suffixes of three characters obtained an accuracy of 86.34%. Kumar *et al.* (2018) provided an overview of experiments on an earlier version of the dataset used in the ILI shared task including five closely related Indo-Aryan languages: Awadhi, Bhojpuri, Braj, Hindi, and Magahi. They managed to obtain an accuracy of 96.48% and a macro F1-score of 0.96 on the sentence level. For sentence level language identification, these results are quite good and as such they indicate that the languages, at least in their written form as evidenced by the corpus, are not as closely related as for example the Balkan languages: Croatian, Serbian, and Bosnian.

The results of the first shared task on Indo-Aryan language identification are described by Zampieri *et al.* (2018). Eight teams submitted results on the task. Like in the second edition of the GDI shared task, we participated with an earlier version of the method described in this article. Again, our submission using a LM adaptation scheme reached a clear first place (Jauhiainen *et al.* 2018c). Seven other teams submitted results on the shared task. The team with the second-best results, *Tübingen-Oslo*, submitted their best results using SVMs (Çöltekin *et al.* 2018). In addition to the SVMs, they experimented with Recurrent Neural Networks (RNN) with GRUs and LSTMs,

https://doi.org/10.1017/S135132491900038X Published online by Cambridge University Press

but their RNNs never achieved results comparable to the SVMs. The team *ILIdentification* used an SVM ensemble (Ciobanu *et al.* 2018b). The best results for the team *XAC* were achieved using Ridge regression (Barbaresi 2018) in addition to which they experimented with NB and SGD classifiers. The team *safina* used CNNs with direct one-hot encoded vectors, with an embedding layer, as well as with a GRU layer (Ali 2018b). The team *dkosmajac* used normalized Euclidean distance. The team *we_are_indian* used word-level LSTM RNNs in their best submission and statistical n-gram approach with mutual information in their second submission (Gupta *et al.* 2018). The team *LaMa* used NB. We report the F1-scores obtained by the teams in Table 14 together with the results presented in this article.

2.3 LM adaptation

Even though LM adaptation has not been used in language identification of text in the past, it has been used in other areas of natural language processing. Jelinek *et al.* (1991) used a dynamic LM and Bacchiani and Roark (2003) used self-adaptation on a test set in speech recognition. Bacchiani and Roark (2003) experimented with iterative adaptation on their LMs and noticed that one iteration made the results better but that subsequent iterations made them worse. Zlatkova *et al.* (2018) used a LR classifier in the Style Change Detection shared task (Kestemont *et al.* 2018). Their winning system fitted their TF-IDF features on the testing data in addition to the training data.

LM adaptation was used by Chen and Liu (2005) for identifying the language of speech. In the system built by them, the speech is first run through Hidden Markov Model-based phone recognizers (one for each language) which tokenize the speech into sequences of phones. The probabilities of those sequences are calculated using corresponding LMs and the most probable language is selected. An adaptation routine is then used so that each of the phonetic transcriptions of the individual speech utterances is used to calculate probabilities for words t, given a word n-gram history of h as in Equation (1).

$$P_a(t|h) = \lambda P_o(t|h) + (1 - \lambda)P_n(t|h) \tag{1}$$

where P_o is the original probability calculated from the training material, P_n the probability calculated from the data being identified, and P_a the new adapted probability. λ is the weight given to original probabilities. This adaptation method resulted in decreasing the error rate in three-way identification between Chinese, English, and Russian by 2.88% and 3.84% on an out-of-domain (different channels) data and by 0.44% on in-domain (same channel) data.

Later, also Zhong *et al.* (2007) used LM adaptation in language identification of speech. They evaluated three different confidence measures and the best faring measure C is defined as follows:

$$C(g_i, M) = \frac{1}{n}[\log(P(M|g_i)) - \log(P(M|g_j))] \tag{2}$$

where M is the sequence to be identified, n the number of frames in the utterance, g_i the best identified language, and g_j the second-best identified language. The two other evaluated confidence measures were clearly inferior. Although the $C(g_i, M)$ measure performed the best of the individual measures, a Bayesian classifier-based ensemble using all the three measures gave slightly higher results. Zhong *et al.* (2007) used the same language adaptation method as Chen and Liu (2005), using the confidence measures to set the λ for each utterance.

We used an earlier version of the LM adaptation technique presented in this article in three of the 2018 VarDial workshop shared tasks (Jauhiainen *et al.* 2018a; 2018b; 2018c).

The adaptive language identification method presented by Ionescu and Butnaru (2018) improved the accuracy from 76.27% to 78.35% on the ADI dataset. In their method, they retrain the LMs once by adding 1000 of the best identified (sorted by the confidence scores produced by their language identification method) unlabeled test samples to the training data.

https://doi.org/10.1017/S135132491900038X Published online by Cambridge University Press

3. The methods

In this section, we present the detailed descriptions of the methods used in the experiments. First, we describe HeLI 2.0, the language identification method used. Then we present the confidence measures we consider in this article. We conclude this section by describing the LM adaptation method used.

3.1 Language identification

We use the HeLI method (Jauhiainen *et al.* 2016) for language identification. The HeLI method has been rivalling SVMs already before the LM adaptation was added, reaching a shared first place in the 2016 Discriminating Similar Languages (DSL) shared task (Malmasi *et al.* 2016). The HeLI method is mostly non-discriminative,[a] and it is relatively quick to incorporate new material into the LMs of the language identifier. We have made a modification to the method where the original penalty value is replaced with a smoothing value that is calculated from the sizes of the LMs. This modification is needed especially for such cases where the LMs grow considerably because of LM adaptation, as the original penalty value was depending on the sizes of the training corpus during the development phase. The penalty modifier p_{mod} is introduced to penalize those languages where features encountered during the identification are absent. The p_{mod} parameter is optimized using the development corpus and in the experiments presented in this article, the optimal value varies between 1.09 and 1.16. The complete formula for the HeLI 2.0 method is presented here, and we provide the modified equations for the values used in the LMs in a similar notation as that used by Jauhiainen *et al.* (2016).

The method aims to determine the language $g \in G$ in which the mystery text M has been written, when all languages in the set G are known to the language identifier. Each language is represented by several different LMs only one of which is used for every word t found in the mystery text. The LMs for each language are one or more models[b] based on words and/or one or more models based on character n-grams from n_{min} to n_{max}. The mystery text is processed one word at a time. The word-based models are used first and if an unknown word is encountered in the mystery text, the method backs off to using the character n-grams of the size n_{max}. If it is not possible to apply the character n-grams of the size n_{max}, the method backs off to lower order character n-grams and, if needed, continues backing off until character n-grams of the size n_{min}.

Creating the LMs: The training data can be preprocessed in different ways to produce different types of LMs. The most usual way is to lowercase the text and tokenize it into words using non-alphabetic and non-ideographic characters as delimiters. It is possible to generate several LMs for words using different preprocessing schemes, and then use the development material to determine which models and in which back-off order are usable for the current task.

The relative frequencies of the words are calculated. Also, the relative frequencies of character n-grams from 1 to n_{max} are calculated inside the words, so that the preceding and the following space-characters are included.[c] The character n-grams are overlapping, so that for example a word with three characters includes three character trigrams. Word n-grams were not used in the experiments of this article, so all subsequent references to n-grams in this article refer to n-grams of characters. After calculating the relative frequencies, we transform those relative frequencies into scores using 10-based logarithms.

The corpus containing only the word tokens in the LMs is called C. A corpus C in language g is denoted by C_g. $dom(O(C))$ is the set of all words found in the models of any of the languages

[a]Setting any of the parameters using a development corpus is dependent on the other languages present, and thus the system learns to discriminate between them.

[b]There can be several models for words, depending on the preprocessing scheme.

[c]A space character is added to the beginning and the end of each word even if it was not there originally.

https://doi.org/10.1017/S135132491900038X Published online by Cambridge University Press

$g \in G$. For each word $t \in dom(O(C))$, the values $v_{C_g}(t)$ for each language g are calculated, as in Equation (3).

$$v_{C_g}(t) = \begin{cases} -\log_{10}\left(\frac{c(C_g,t)}{l_{C_g}}\right), & \text{if } c(C_g,t) > 0 \\ -\log_{10}\left(\frac{1}{l_{C_g}}\right)p_{mod}, & \text{if } c(C_g,t) = 0 \end{cases} \tag{3}$$

where $c(C_g, t)$ is the number of words t and l_{C_g} is the total number of all words in language g. The parameter p_{mod} is the penalty modifier which is determined empirically using the development set.

The corpus containing the n-grams of the size n in the LMs is called C^n. The domain $dom(O(C^n))$ is the set of all character n-grams of length n found in the models of any of the languages $g \in G$. The values $v_{C_g^n}(u)$ are calculated in the same way for all n-grams $u \in dom(O(C^n))$ for each language g, as shown in Equation (4).

$$v_{C_g^n}(u) = \begin{cases} -\log_{10}\left(\frac{c(C_g^n,u)}{l_{C_g^n}}\right), & \text{if } c(C_g^n,u) > 0 \\ -\log_{10}\left(\frac{1}{l_{C_g^n}}\right)p_{mod}, & \text{if } c(C_g^n,u) = 0 \end{cases} \tag{4}$$

where $c(C_g^n, u)$ is the number of n-grams u found in the corpus of the language g and $l_{C_g^n}$ is the total number of the n-grams of length n in the corpus of language g. These values are used when scoring the words while identifying the language of a text.

Scoring the text: The mystery text M is tokenized into words using the same tokenization scheme as when creating the LMs. The words are lowercased when lowercased models are being used. After this, a score $v_g(t)$ is calculated for each word t in the mystery text for each language g. If the word t is found in the set of words $dom(O(C_g))$, the corresponding value $v_{C_g}(t)$ for each language g is assigned as the score $v_g(t)$, as shown in Equation 5.

$$v_g(t) = \begin{cases} v_{C_g}(t), & \text{if } t \in dom(O(C_g)) \\ v_g(t, min(n_{max}, l_t + 2)), & \text{if } t \notin dom(O(C_g)) \end{cases} \tag{5}$$

If a word t is not found in the set of words $dom(O(C_g))$ and the length of the word l_t is at least $n_{max} - 2$, the language identifier backs off to using character n-grams of the length n_{max}. In case the word t is shorter than $n_{max} - 2$ characters, $n = l_t + 2$.

When using n-grams, the word t is split into overlapping n-grams of characters u_i^n, where $i = 1, ..., l_t - n$, of the length n. Each of the n-grams u_i^n is then scored separately for each language g in the same way as the words.

If the n-gram u_i^n is found in $dom(O(C_g^n))$, the values in the models are used. If the n-gram u_i^n is not found in any of the models, it is simply discarded. We define the function $d_g(t, n)$ for counting n-grams in t found in a model in Equation 6.

$$d_g(t,n) = \sum_{i=1}^{l_t-n} \begin{cases} 1, & \text{if } u_i^n \in dom(O(C^n)) \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

When all the n-grams of the size n in the word t have been processed, the word gets the value of the average of the scored n-grams u_i^n for each language, as in Equation (7).

$$v_g(t,n) = \begin{cases} \frac{1}{d_g(t,n)} \sum_{i=1}^{l_t-n} v_{C_g^n}(u_i^n), & \text{if } d_g(t,n) > 0 \\ v_g(t, n-1), & \text{otherwise,} \end{cases} \tag{7}$$

https://doi.org/10.1017/S135132491900038X Published online by Cambridge University Press

where $d_g(t, n)$ is the number of n-grams u_i^n found in the domain $dom(O(C_g^n))$. If all of the n-grams of the size n were discarded, $d_g(t, n) = 0$, the language identifier backs off to using n-grams of the size $n - 1$.

The whole mystery text M gets the score $R_g(M)$ equal to the average of the scores of the words $v_g(t)$ for each language g, as in Equation (8).

$$R_g(M) = \frac{\sum_{i=1}^{l_{T(M)}} v_g(t_i)}{l_{T(M)}} \tag{8}$$

where $T(M)$ is the sequence of words and $l_{T(M)}$ is the number of words in the mystery text M. Since we are using negative logarithms of probabilities, the language having the lowest score is returned as the language with the maximum probability for the mystery text.

3.2 Confidence estimation

In order to be able to select the best candidate for LM adaptation, the language identifier needs to provide a confidence score for the identified language. We evaluated three different confidence measures that seemed applicable to the HeLI 2.0 method.

In the first measure, we estimate the confidence of the identification as the difference between the scores $R(M)$ of the best and the second best identified language. Zhong *et al.* (2007) call this confidence score CM_{BS}, and in our case it is calculated using the following equation:

$$CM_{BS}(M) = R_h(M) - R_g(M) \tag{9}$$

where g is the best scoring language and h the second best scoring language.

The second confidence measure, CM_{AVG}, was presented by Chen and Liu (2005). In CM_{AVG}, we calculate the difference between the score for the best identified language and the average of the scores of the rest of the languages. CM_{AVG} adapted to our situation is calculated as follows:

$$CM_{AVG}(M) = \frac{1}{l_G - 1} \sum_{j=1, j \neq g}^{l_G} R_j(M) - R_g(M) \tag{10}$$

The third measure, CM_{POST}, presented by Zhong *et al.* (2007), is based on the posterior probability. We calculated it using the following equation:

$$CM_{POST}(M) = \log \sum_{j=1}^{l_G} e^{R_j(M)} - R_g(M) \tag{11}$$

3.3 LM adaptation algorithm

In the first step of our adaptation algorithm, all the mystery texts M in the mystery text collection MC (e.g., a test set) are preliminarily identified using the HeLI 2.0 method. They are subsequently ranked by their confidence scores CM and the preliminarily identified collection is split into $k - q$ parts $MC_{1\ldots k}$. k is a number between 1 and the total number of mystery texts, l_{MC}, depending on in how many parts we want to split the mystery text collection.[d] The higher k is, the longer the identification of the whole collection will take. The number of finally identified parts is q, which in the beginning is 0. After ranking, the part MC_1 includes the most confidently identified texts and MC_{k-q} the least confidently identified texts.

[d]The only difference between the language adaptation method presented here and the earlier version of the method we used at the shared tasks is that in the shared tasks, the k was always equal to l_{MC}.

https://doi.org/10.1017/S135132491900038X Published online by Cambridge University Press

Table 1. List of the Swiss German varieties used in the datasets distributed for the 2017 GDI shared task. The sizes of the training and the test sets are in words

Variety (code)	Training	Test
Bern (BE)	28,558	7025
Basel (BS)	28,680	7064
Lucerne (LU)	28,653	7509
Zurich (ZH)	28,715	7949

Words and character n-grams up to the length n_{max} are extracted from each mystery text in MC_1 and added to the respective LMs. Then, all the mystery texts in the part MC_1 are set as finally identified and q is increased by 1.

Then for as long as $q < k$, the process is repeated using the newly adapted LMs to perform a new preliminary identification for those texts that are not yet finally identified. In the end, all features from all of the mystery texts are included in the LM. This constitutes one epoch of adaptation.

In iterative LM adaptation, the previous algorithm is repeated from the beginning several times.

4. Test setting

We evaluate the methods presented in the previous section using three standard datasets. The first two datasets are from the GDI shared tasks held at VarDials 2017 and 2018. The third dataset is from the ILI shared task held at VarDial 2018.

4.1 GDI 2017 dataset

The dataset used in the GDI 2017 shared task consists of manual transcriptions of speech utterances by speakers from different areas in Switzerland: Bern, Basel, Lucerne, and Zurich. The variety of German spoken in Switzerland is considered to be a separate language (Swiss German, *gsw*) by the ISO-639-3 standard (Lewis, Simons, and Fennig 2013), and these four areas correspond to separate varieties of it. The transcriptions in the dataset are written entirely in lowercased letters. Samardžić *et al.* (2016) describe the ArchiMob corpus, which is the source for the shared task dataset. Zampieri *et al.* (2017) describe how the training and test sets were extracted from the ArchiMob corpus for the 2017 shared task. The sizes of the training and test sets can be seen in Table 1. The shared task was a four-way language identification task between the four German dialects present in the training set.

4.2 GDI 2018 dataset

The dataset used in the GDI 2018 shared task was similar to the one used in GDI 2017. The sizes of the training, the development, and the test sets can be seen in Table 2. The first track of the shared task was a standard four-way language identification between the four German dialects present in the training set. The GDI 2018 shared task included an additional second track dedicated to unknown dialect detection. The unknown dialect was included neither in the training nor in the development sets, but it was present in the test set. The test set was identical for both tracks, but the lines containing an unknown dialect were ignored when calculating the results for the first track.

4.3 ILI 2018 dataset

The dataset used for the ILI 2018 shared task included text in five languages: Bhojpuri, Hindi, Awadhi, Magahi, and Braj. The texts were mainly literature published over the web as well as

https://doi.org/10.1017/S135132491900038X Published online by Cambridge University Press

Table 2. List of the Swiss German varieties used in the datasets distributed for the 2018 GDI shared task. The sizes of the training, the development, and the test sets are in words

Variety (code)	Training	Development	Test
Bern (BE)	28,558	7404	12,013
Basel (BS)	27,421	9544	9802
Lucerne (LU)	29,441	8887	11,372
Zurich (ZH)	28,820	8099	9610
Unknown dialect (XY)			8938

Table 3. List of the Indo-Aryan languages used in the datasets distributed for the 2018 ILI shared task. The sizes of the training, the development, and the test sets are in words

Language (code)	Training	Development	Test
Bhojpuri (BHO)	258,501	56,070	50,206
Hindi (HIN)	325,458	44,215	35,863
Awadhi (AWA)	123,737	19,616	22,984
Magahi (MAG)	234,649	37,809	35,358
Braj (BRA)	249,243	40,023	31,934

in print. As can be seen in Table 3, there was considerably less training material for the Awadhi language than the other languages. The training corpus for Awadhi had only slightly over 9000 lines, whereas the other languages had around 15,000 lines of text for training. An earlier version of the dataset, as well as its creation, was described by Kumar *et al.* (2018). The ILI 2018 shared task was an open one, allowing the use of any additional data or means. However, we have not used any external data, and our results would be exactly the same on a closed version of the task.

5. Experiments and results

In our experiments, we evaluate the HeLI 2.0 method, the HeLI 2.0 method using LM adaptation, as well as the iterative version of the adaptation. We test all three methods with all of the datasets described in the previous section. First we evaluate the confidence measures using the GDI 2017 dataset and afterwards we use the best performing confidence measure in all further experiments.

We are measuring language identification performance using the macro and the weighted F1-scores. These are the same performance measures that were used in the GDI 2017, GDI 2018, and ILI 2018 shared tasks (Zampieri *et al.* 2017; 2018). F1-score is calculated using the precision and the recall as in Equation (12).

$$\text{F1-score} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{12}$$

The macro F1-score is the average of the individual F1-scores for the languages and the weighted F1-score is similar but weighted by the number of instances for each language.

https://doi.org/10.1017/S135132491900038X Published online by Cambridge University Press

Table 4. Average accuracies within the 10% portions when the results are sorted by the confidence scores *CM*

% most confident (%)	$\mathbf{CM_{BS}}$ accuracy (%)	$\mathbf{CM_{AVG}}$ accuracy (%)	$\mathbf{CM_{POST}}$ accuracy (%)
0–10	98.5	95.6	88.3
10–20	98.3	96.3	91.4
20–30	98.2	96.2	92.6
30–40	98.2	95.1	92.1
40–50	97.8	94.6	92.0
50–60	96.9	94.9	91.7
60–70	96.0	94.0	91.5
70–80	94.4	93.2	91.0
80–90	92.4	91.7	89.9
90–100	89.0	89.0	89.0

5.1 Evaluating the confidence measures

We evaluated the three confidence measures presented in Section 3.2 using the GDI 2017 training data. The results of the evaluation are presented in Table 4. The underlying data for the table consists of pairs of confidence values and the corresponding Boolean values indicating whether the identification results were correct or not. The data have been ordered according to their confidence score for each of the three measures. The first column in the table tells the percentage of examined top scores. The other columns give the average accuracy in that examined portion of identification results for each confidence measure.

The first row tells us that in the 10% of the highest confidence identification results according to the CM_{BS}-measure, 98.5% of the performed identifications were correct. The two other confidence measures on the other hand fail to arrange the identification results so that the most confident 10% would be the most accurate 10%. As a whole, this experiment tells us that the CM_{BS}-measure is stable and performs well when compared with the other two.

In addition to evaluating each individual confidence measure, Zhong *et al.* (2007) evaluated an ensemble combining all of the three measures, gaining somewhat better results than with the otherwise best performing CM_{BS}-measure. However, in their experiments the two other measures were much more stable than in ours. We decided to use only the simple and well-performing CM_{BS}-measure with our LM adaptation algorithm in the following experiments.

5.2 Experiments on the GDI 2017 dataset

5.2.1 Baseline results and parameter estimation

As there was no separate development set provided for the GDI 2017 shared task, we divided the training set into training and development partitions. The last 500 lines from the original training data for each language was used for development. The development partition was then used to find the best parameters for the HeLI 2.0 method using the macro F1-score as the performance measure. The macro F1-score is equal to the weighted F1-score, which was used as a ranking measure in the shared task, when the number of tested instances in each class are equal. On the development set, the best macro F1-score of 0.890 was reached with the language identifier where $n_{max} = 5$ words being used and $p_{mod} = 1.16$. We then used the whole training set to train the LMs. On the test set, the language identifier using the same parameters reached the macro F1-score of 0.659 and the weighted F1-score of 0.639.

https://doi.org/10.1017/S135132491900038X Published online by Cambridge University Press

Table 5. The weighted F1-scores obtained by the identifier using LM adaptation with different values of *k* when tested on the development partition of the GDI 2017 dataset

k	weighted F1-score	*k*	weighted F1-score
1 or 2	0.890	30, 35, or 40	0.899
3	0.893	45	0.900
4	0.894	50	0.899
5	0.896	60	0.900
10 or 15	0.899	70, 80, 90,	
20 or 25	0.898	100, 150, or 200	0.899

Table 6. Weighted F1-scores with confidence threshold for LM adaptation on the development set

Conf. threshold	Macro F1-score	Conf. threshold	Macro F1-score
0.00–0.01	0.900	0.10	0.898
0.02	0.899	0.16	0.896
0.04	0.898	0.32	0.889
0.06 or 0.08	0.899		

5.2.2 Experiments with LM adaptation

First, we determined the best value for the number of splits k using the development partition. Table 5 shows the increment of the weighted F1-score with different values of k on the development partition using the same parameters with the HeLI 2.0 method as for the baseline. The results with $k = 1$ are always equal to the baseline. If k is very high, the identification becomes computationally costly as the number of identifications grows exponentially in proportion to k. The absolute increase of the F1-score on the development partition was 0.01 when using $k = 45$.

5.2.3 Experiments with thresholding

We experimented with setting a confidence threshold for the inclusion of new data into the LMs. Table 6 shows the results on the development partition. The results show that there is no confidence score that could be used for thresholding, at least not with the development partition of GDI 2017.

5.2.4 Results of the LM adaptation on the test data

Based on the evaluations using the development partition, we decided to use $k = 45$ for the test run. All the training data were used for the initial LM creation. The language identifier using LM adaptation reached the macro F1-score of 0.689 and the weighted F1-score of 0.687 on the test set. The weighted F1-score was 0.048 higher than the one obtained by the nonadaptive version and clearly higher than the other results obtained using the GDI 2017 dataset.

5.2.5 Iterative adaptation

We tested repeating the LM adaptation algorithm for several epochs, and the results of those trials on the development partition can be seen in Table 7. The improvement of 0.003 on the original macro F1-score using 13–956 epochs was still considerable. The results seem to indicate that the LMs become very stable with repeated adaptation.

https://doi.org/10.1017/S135132491900038X Published online by Cambridge University Press

Table 7. Macro F1-scores with iterative LM adaptation on the development partition

Number of epochs	Macro F1-score	Number of epochs	Macro F1-score
1	0.900	13–956	0.903
2–4	0.901	957–999	0.902
5–12	0.902		

We decided to try iterative LM adaptation using 485 epochs with the test set. The tests resulted in a weighted F1-score of 0.700, which was a further 0.013 increase on top of the score obtained without additional iterations. We report the weighted F1-scores from the GDI 2017 shared task together with our own results in Table 8. The methods used are listed in the first column, used features in the second column, and the best reached weighted F1-score in the third column. The results from this paper are bolded. The results using other methods (team names are in parentheses) are collected from the shared task report (Zampieri *et al.* 2017) as well as from the individual system description articles. The 0.013 point increase obtained with the iterative LM adaptation over the non-iterative version might seem small when compared with the overall increase over the scores of the HeLI 2.0 method, but the increase is still more than the difference between the first and third best submitted methods on the original shared task.

5.3 Experiments on the GDI 2018 dataset

5.3.1 Baseline results and parameter estimation

The GDI 2018 dataset included a separate development set (Table 2). We used the development set to find the best parameters for the HeLI 2.0 method using the macro F1-score as the performance measure. The macro F1-score of 0.659 was obtained by the HeLI 2.0 method using just character n-grams of the size 4 with $p_{mod} = 1.15$. The corresponding recall 66.17% was slightly higher than the 66.10% obtained with the HeLI method used in the GDI 2018 shared task. We then used the combined training and the development sets to train the LMs. On the test set, the language identifier using these parameters obtained a macro F1-score of 0.650. The HeLI 2.0 method reached 0.011 higher macro F1-score than the HeLI method we used in the shared task. Even without the LM adaptation, the HeLI 2.0 method beats all the other reported methods.

5.3.2 Experiments with LM adaptation

Table 9 shows the increment of the macro F1-score with different values of k, the number of parts the examined mystery text collection is split into, on the development set using the same parameters with the HeLI 2.0 method as for the baseline. On the development set, $k = 57$ gave the best F1-score, with the absolute increase of 0.116 over the baseline. The corresponding recall was 77.74%, which was somewhat lower than the 77.99% obtained at the shared task.

5.3.3 Results of the LM adaptation on the test set

Based on the evaluations using the development set, we decided to use $k = 57$ for the test run. All the training and the development data were used for the initial LM creation. The method using the LM adaptation algorithm reached the macro F1-score of 0.707. This macro F1-score is 0.057 higher than the one obtained by the nonadaptive version and 0.021 higher than the results we obtained using LM adaptation in the GDI 2018 shared task. The performance improvement with LM adaptation is partly due to the ability of the HeLI 2.0 to better handle the adapted LMs and the optimization of the k-value.

https://doi.org/10.1017/S135132491900038X Published online by Cambridge University Press

Table 8. The weighted F1-scores using different methods on the 2017 GDI test set. The results from the experiments presented in this article are bolded. The system description papers of each team, if existing, are listed in Section 2.1

Method (Team)	Features used	wgh. F1
HeLI 2.0 + iterative LM-adapt.	ch. *n*-grams 1-5 and words	**0.700**
HeLI 2.0 + LM-adapt.	ch. *n*-grams 1-5 and words	**0.687**
SVM meta-classifier ensemble (MAZA)	ch. *n*-grams 1-6 and words	0.662
SVM, cat. equal. 2 (CECL)	BM25 ch. *n*-grams 1-5	0.661
CRF (CLUZH)	ch. *n*-grams, affixes...	0.653
NB, CRF, and SVM ensemble (CLUZH)	ch. *n*-grams, affixes...	0.653
SVM probability ensemble (MAZA)	ch. *n*-grams 1-6 and words	0.647
SVM + SGD ensemble (qcri_mit)	*n*-grams 1-8	0.639
HeLI 2.0	ch. *n*-grams 1-5 and words	**0.639**
SVM, cat. equal. 1 (CECL)	BM25 ch. *n*-grams 1-5	0.638
KRR, sum of 3 kernels (unibuckernel)	*n*-grams 3-6	0.637
KDA, sum of 2 kernels (unibuckernel)	*n*-grams 3-6	0.635
KDA, sum of 3 kernels (unibuckernel)	*n*-grams 3-6	0.634
SVM voting ensemble (MAZA)	ch. *n*-grams 1-6 and words	0.627
Linear SVM (tubasfs)	TF-IDF ch. *n*-grams and words	0.626
SVM (CECL)	BM25 ch. *n*-grams 1-5	0.625
NB (CLUZH)	ch. *n*-grams 2-6	0.616
Cross Entropy (ahaqst)	ch. *n*-grams up to 25 bytes	0.614
Perplexity (Citius_Ixa_Imaxin)	words	0.612
Perplexity (Citius_Ixa_Imaxin)	ch. 5-7 and word 1-3 *n*-grams	0.611
Naive Bayes (XAC_Bayesline)	TF-IDF	0.605
Perplexity (Citius_Ixa_Imaxin)	ch. 7-grams	0.577
Cross Entropy (ahaqst)	word *n*-grams 1-3	0.548
LSTM NN (deepCybErNet)	characters or words	0.263

Table 9. The macro F1-scores gained with different values of *k* when evaluated on the GDI 2018 development set

k	Macro F1-score	*k*	Macro F1-score
1	0.659	52, 54, or 55	0.774
2	0.719	56 or 57	0.776
4	0.755	58	0.774
8	0.769	60 or 64	0.775
16	0.773	96 or 128	0.774
32, 40, 44, or 46	0.774	256 or 512	0.775
48	0.775	1024, 2048, or 4658	0.774

https://doi.org/10.1017/S135132491900038X Published online by Cambridge University Press

Table 10. Macro F1-scores with iterative LM adaptation on the GDI 2018 development set

Number of epochs	Macro F1-score	Number of epochs	Macro F1-score
1	0.776	17–19	0.813
2	0.787	20	0.814
3	0.792	21	0.813
4	0.797	22–33	0.814
5	0.800	34–54	0.815
6	0.801	55–82	0.816
7	0.804	83–88	0.815
8	0.806	89–94	0.816
9	0.807	95–111	0.815
10	0.808	112–122	0.816
11	0.809	123–129	0.815
12–13	0.810	130–476	0.816
14	0.811	477–999	0.817
15–16	0.812		

5.3.4 Iterative adaptation

We tested repeating the LM adaptation algorithm for several epochs, and the results of those trials on the GDI 2018 development set can be seen in Table 10. There was a clear improvement of 0.041, at 477–999 epochs, on the original macro F1-score. It would again seem that the LMs become very stable with repeated adaptation, at least when there is no unknown language present in the data which is the case with the development set. Good scores were obtained already at 20 iterations, after which the results started to fluctuate up and down.

Based on the results on the development set, we decided to try two different counts of iterations: 738, which is the number of epochs in the middle of the best scores, and 20, after which the results started to fluctuate. The tests resulted in a macro F1-score of 0.696 with 738 epochs and 0.704 with 20 epochs. As an additional experiment, we evaluated the iterative adaptation on a test set, from which the unknown dialects had been removed and obtained an F1-score of 0.729 with 738 epochs. From the results, it is clear that the presence of the unknown language is detrimental to repeated LM adaptation. In Table 11, we report the macro F1-scores obtained by the teams participating in the GDI 2018 shared task, as well as our own. The methods used are listed in the first column, used features in the second column, and the best reached macro F1-score in the third column.

5.4 Experiments on the ILI 2018 dataset

5.4.1 Baseline results and parameter estimation

We used the development set to find the best parameters for the HeLI 2.0 method using the macro F1-score as the measure. Using both original and lowercased character n-grams from one to six with $p_{mod} = 1.09$, the method obtained the macro F1-score of 0.954. The corresponding recall was 95.26%, which was exactly the same we obtained with the HeLI method used in the ILI 2018 shared task. We then used the combined training and the development sets to train the LMs. On the test set, the language identifier using the above parameters obtained a macro F1-score of 0.880, which was clearly lower than the score we obtained using the HeLI method in the shared task.

https://doi.org/10.1017/S135132491900038X Published online by Cambridge University Press

Table 11. The macro F1-scores using different methods on the 2018 GDI test set. The results from the experiments presented in this article are bolded. The system description papers of each team, if existing, are listed in Section 2.1

Method (team)	Features used	F1
HeLI 2.0 with LM adapt.	ch. 4-grams	**0.707**
HeLI 2.0 with iter. (20) LM adapt.	ch. 4-grams	**0.704**
HeLI 2.0 with iter. (738) LM adapt.	ch. 4-grams	**0.696**
HeLI with LM adapt. (SUKI)	ch. 4-grams	0.686
HeLI 2.0	ch. 4-grams	**0.650**
SVM ensemble (Twist Bytes)	ch. and word *n*-grams 1-7	0.646
CNN with GRU (safina)	characters	0.645
SVMs (Tübingen-Oslo)	ch. *n*-grams 1-6, word *n*-grams 1-3	0.640
HeLI (SUKI)	ch. 4-grams	0.639
Voting ensemble (LaMa)	ch. *n*-grams 1-8, word *n*-grams 1-6	0.637
Naive Bayes (XAC)	TF-IDF ch. *n*-grams 1-6	0.634
Ridge regression (XAC)	TF-IDF ch. *n*-grams 1-6	0.630
SGD (XAC)	TF-IDF ch. *n*-grams 1-6	0.630
CNN (safina)	characters	0.645
SVM ensemble (GDI_classification)	ch. *n*-grams 2-5	0.620
CNN with embedding (safina)	characters	0.645
RNN with LSTM (Tübingen-Oslo)		0.616
Euclidean distance (dkosmajac)	ch. *n*-grams	0.591

Table 12. The macro F1-scores gained with different values of *k* when tested on the ILI 2018 development set

k	Macro F1-score	*k*	Macro F1-score
1	0.954	32 or 48	0.964
2	0.958	58	0.963
4	0.960	60 or 62	0.964
8 or 16	0.963		

5.4.2 Experiments with LM adaptation

Table 12 shows the increment of the macro F1-score with different values of k on the development set using the same parameters with the HeLI 2.0 method as for the baseline. On the development set, $k = 64$ gave the best F1-score, 0.964, which is an absolute increase of 0.010 on the original F1-score. The corresponding recall was 96.29%, which was a bit better than the 96.22% obtained in the shared task.

5.4.3 Results of the LM adaptation on the test data

Based on the evaluations using the development data, we decided to use $k = 64$ as the number of splits for the actual test run. All the training and the development data were used for the initial

https://doi.org/10.1017/S135132491900038X Published online by Cambridge University Press

Table 13. Macro F1-scores with iterative LM adaptation on the ILI 2018 development set

Number of epochs	Macro F1-score	Number of epochs	Macro F1-score
1	0.964	2-999	0.965

Table 14. The macro F1-scores using different methods on the 2018 ILI test set. The results presented for the first time are in bold. The system description papers of each team, if existing, are listed in Section 2.2

Method (team)	Features used	F1
HeLI 2.0 with iter. LM adapt.	ch. *n*-grams 1-6	**0.958**
HeLI with iter. LM adapt. (SUKI)	ch. *n*-grams 1-6	0.958
HeLI with LM adapt. (SUKI)	ch. *n*-grams 1-6	0.955
HeLI 2.0 with LM adapt.	ch. *n*-grams 1-6	**0.955**
SVM (Tübingen-Oslo)	ch. *n*-grams 1-6, word *n*-grams 1-3	0.902
Ridge regression (XAC)	ch. *n*-grams 2-6	0.898
SVM ensemble (ILIdentification)	ch. *n*-grams 2-4	0.889
HeLI (SUKI)	ch. *n*-grams 1-6	0.887
SGD (XAC)	ch. *n*-grams 2-6	0.883
HeLI 2.0	ch. *n*-grams 1-6	**0.880**
CNN (safina)	characters	0.863
NB (XAC)	ch. *n*-grams 2-6	0.854
Euclidean distance (dkosmajac)		0.847
CNN with embedding (safina)	characters	0.844
LSTM RNN (we_are_indian)	words	0.836
CNN with GRU (safina)	characters	0.826
NB (LaMa)		0.819
RNN with GRU (Tübingen-Oslo)		0.753
Mutual information (we_are_indian)		0.744

LM creation. The identifier using the LM adaptation algorithm obtained a macro F1-score of 0.955. This macro F1-score is basically the same we obtained with LM adaptation in the ILI 2018 shared task, only some small fractions lower.

5.4.4 Iterative adaptation

We experimented repeating the LM adaptation algorithm for several epochs, and the results of those trials on the development set can be seen in Table 13. There was a very small improvement of 0.001 on the original macro F1-score. The best absolute F-scores were reached at epochs 17 and 18. It would again seem that the LMs become very stable with repeated adaptation.

Based on the results on the development set, we decided to use LM adaptation with 18 iterations on the test set. The test resulted in a macro F1-score of 0.958, which is again almost the same as in the shared task, though this time some small fractions higher. We report the F1-scores obtained by the different teams participating in the ILI 2018 shared task in Table 14, with the results from

https://doi.org/10.1017/S135132491900038X Published online by Cambridge University Press

this article in bold. The methods used are listed in the first column, used features in the second column, and the macro F1-scores in the third column.

6. Discussion

The LM adaptation scheme proved to be of great importance with all three datasets. The F1-scores improved from 5 to 7 absolute points from the results gained by the same methods without LM adaptation. The fundamental component for the performance improvement is the ability to learn new information from the test set itself. As of this writing, the results from the shared tasks of VarDial 2019 (Zampieri *et al.* 2019) are being prepared for publication, and several participating teams had incorporated some sort of an LM adaptation algorithm into their systems. We used the same LM adaptation scheme as presented in this paper with the HeLI 2.0 method as well as with a custom NB implementation (Jauhiainen, Jauhiainen, and Lindén 2019), two teams used such a scheme with SVMs (Benites, von Däniken, and Cieliebak 2019; Wu *et al.* 2019) and one learned new information from the test set with deep neural networks (Bernier-Colborne, Goutte, and Léger 2019). All three shared tasks[e] concentrating on language, dialect, or variety identification were won using one of these systems.

The results using the HeLI 2.0 method and the improved LM adaptation are clearly better with the GDI 2018 dataset than the ones with the original HeLI method. However, with the ILI 2018 dataset, there is no real difference in performance between the old and the new methods. This is at least partly due to the fact that the size of the test set relative to the training set is much larger with the GDI 2018 dataset than with the ILI 2018 dataset.

The additional performance gained using the LM adaptation on the GDI 2017 development data (F1-score rose from 0.890 to 0.903) was much less than in the GDI 2018 development data (F1-score rose from 0.659 to 0.817). This indicates that the training and the development data of the GDI 2017 were already in-domain with each other as opposed to being out-of-domain in the GDI 2018 data. Additionally, the 26% difference in F1-scores between the development portion (0.890) and the test set (0.659) of the GDI 2017 data obtained by the HeLI 2.0 method is considerable. It seems to indicate that the test set contains more out-of-domain material when compared with the partition of the training set we used for development. In order to validate this hypothesis, we divided the test set into two parts. The second part remained to be used for testing in four scenarios with the HeLI 2.0 method. In the scenarios we used different combinations of data for training: the original training set, the training set augmented with the first part of test data, the training set of which a part was replaced by the first part of the test set, and only using the first part of the test set. The results of these experiments support our hypothesis, as can be seen in Table 15. The domain difference between the two sets explains why iterative adaptation performs better with the test set than with the development set. After each iteration, the relative amount of the original training data gets smaller, as the information from the test data is repeatedly added to the LMs.

In the GDI 2018 dataset, there is only a 1.4% difference between the macro F1-scores obtained from the development and the test sets. This indicates that the GDI 2018 development set is in the same way out-of-domain when compared with the training set as the actual test set is.

There is a small difference (7.8%) between the F1-scores attained using the development set and the test set of the ILI 2018 data as well. However, such small differences can be partly due to the fact that the parameters of the identification method have been optimized using the development set.

Though the iterative LM adaptation is computationally costly when compared with the baseline HeLI 2.0 method, it must be noted that the final identifications with 485 epochs on the GDI

[e]The three tasks were GDI, Discriminating between Mainland and Taiwan variation of Mandarin Chinese, and Cuneiform Language Identification (Zampieri *et al.* 2019).

https://doi.org/10.1017/S135132491900038X Published online by Cambridge University Press

Table 15. The macro F1-scores for the second part of test set using different training data combinations

Data used for the language models	Macro F1
Training set	0.656
Training set + first part of test set	0.801
Part of training set replaced with first part of test set	0.803
First part of test set	0.858

Table 16. Time measurements for creating the LMs and predicting the language for different test sets. The measurements give some indication of the computational efficiency but can only be considered rough estimates

Procedure	Time (seconds)	F1
Creating LMs: character *n*-grams 1–8 and words, GDI 2017	4	
Predictions: HeLI 2.0, GDI 2017	3	0.639
Predictions: HeLI 2.0 + LM adapt, GDI 2017	6	0.687
Predictions: HeLI 2.0 + iterative LM adapt, GDI 2017	1194	0.700
Creating LMs: character *n*-grams 1–8 and words, GDI 2018	6	
Predictions: HeLI 2.0, GDI 2018	3	0.650
Predictions: HeLI 2.0 + LM adapt, GDI 2018	12	0.707
Predictions: HeLI 2.0 + iterative LM adapt, GDI 2018	5645	0.696
Creating LMs: character *n*-grams 1–8 and words, ILI 2018	39	
Predictions: HeLI 2.0, ILI 2018	7	0.880
Predictions: HeLI 2.0 + LM adapt, ILI 2018	39	0.955
Predictions: HeLI 2.0 + iterative LM adapt, ILI 2018	557	0.958

2017 test set took only around 20 minutes using one computing core of a modern laptop. We provide the time taken for creating the initial LMs for each dataset as well as the time taken by different methods when calculating the predictions on the test partitions of the different datasets in Table 16.[f]

The time taken by the iterative LM-adaptation is linearly related to the number of epochs used. With the GDI 2017 dataset, one epoch took around 2.45 seconds, which is also near the difference between the basic HeLI 2.0 method and the one with LM adaptation. The reason for one round of LM adaptation taking so much longer (7.64 seconds) with the GDI 2018 test set is that the number of splits used also adds linearly to the time consumed. The test sets were also considerably different in size, and the size of the test set also linearly affects the time used. The ILI 2018 test set had around five times more sentences than that of GDI 2017.

We are not providing an error analysis of the errors made by our system on the test sets. If we would do so, it would make us, and any of the readers, less qualified to use these same datasets for further development of our methods.

[f]The laptop used was a MacBookPro14,3 with 2.9 GHz Intel Core i7 processor. The Java program used only one core during the identification.

https://doi.org/10.1017/S135132491900038X Published online by Cambridge University Press

7. Conclusions

The results indicate that unsupervised LM adaptation should be considered in all language identification tasks, especially in those where the amount of out-of-domain data is significant. If the presence of unseen languages is to be expected, the use of LM adaptation could still be beneficial, but special care must be taken as repeated adaptation in particular could decrease the identification accuracy. We were delighted to see that some of the other participants of the 2019 VarDial Evaluation Campaign (Zampieri *et al.* 2019) had noticed our LM adaptation scheme and used a somewhat similar way of gathering new information from the test sets with their own systems.

8. Future work

We believe that it is possible to apply a similar adaptation scheme with other NLP problems and especially with other classification tasks. We are looking forward to investigating these possibilities in the future.

An experiment left for future work is to test how the amount of test data affects the final language identification accuracy. In the experiment, we would divide the test sets into smaller parts and evaluate how the LM adaptation technique performs in them. Our intuition suggests that the smaller the test set, the less effective the LM adaptation will be. However, if the larger test set consists of texts in several separable domains, it might actually be beneficial to divide the test set to smaller parts.

The adaptation technique presented in this paper could, in theory, be used to annotate a large dataset with an extremely small training set, maybe even with just one sentence. This is perhaps the most interesting avenue for further research.

Acknowledgements. This research was partly conducted with funding from the Kone Foundation Language Programme (Kone Foundation 2012) and from FIN-CLARIN. We thank the anonymous reviewers for their thought-provoking questions.

References

Ali M. (2018a). Character level convolutional neural network for German dialect identification. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Santa Fe, NM, pp. 172–177.

Ali M. (2018b). Character level convolutional neural network for Indo-Aryan language identification. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Santa Fe, NM, pp. 283–287.

Bacchiani M. and Roark B. (2003). Unsupervised language model adaptation. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP 2003)*, pp. 224–227.

Barbaresi A. (2017). Discriminating between similar languages using weighted subword features. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Valencia, Spain, pp. 184–189.

Barbaresi A. (2018). Computationally efficient discrimination between language varieties with large feature vectors and regularized classifiers. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Santa Fe, NM, pp. 164–171.

Benites F., Grubenmann R., von Däniken P., von Grünigen D., Deriu J. and Cieliebak M. (2018). Twist Bytes - German dialect identification with data mining optimization. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Santa Fe, NM, pp. 218–227.

Benites F., von Däniken P. and Cieliebak M. (2019). TwistBytes - identification of Cuneinform languages and German dialects at VarDial 2019. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Minneapolis, USA.

Bergsma S., McNamee P., Bagdouri M., Fink C. and Wilson T. (2012). Language identification for creating language-specific Twitter collections. In *Proceedings of the Second Workshop on Language in Social Media (LSM 2012)*, Montréal, Canada, pp. 65–74.

Bernier-Colborne G., Goutte C. and Léger S. (2019). Improving Cuneiform language identification with BERT. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Minneapolis, USA.

Bestgen Y. (2017). Improving the character Ngram model for the DSL task with BM25 weighting and less frequently used feature sets. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Valencia, Spain, pp. 115–123.

https://doi.org/10.1017/S135132491900038X Published online by Cambridge University Press

Blodgett S.L., Wei J.T.-Z. and O'Connor B. (2017). A dataset and classifier for recognizing social media English. In *Proceedings of the 3rd Workshop on Noisy User-Generated Text*, Copenhagen, Denmark, pp. 56–61.

Breiman L. (2001). Random forests. *Machine Learning* **45**(1), 5–32.

Brown R.D. (2012). Finding and identifying text in 900+ languages. *Digital Investigation* **9**, S34–S43.

Brown R.D. (2013). Selecting and weighting N-grams to identify 1100 languages. In *Proceedings of the 16th International Conference on Text, Speech and Dialogue (TSD 2013)*, Plzeň, Czech Republic, pp. 475–483.

Brown R.D. (2014). Non-linear mapping for improved identification of 1300+ languages. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, Doha, Qatar, pp. 627–632.

Chen Y. and Liu J. (2005). Language model adaptation and confidence measure for robust language identification. In *Proceedings of ISCIT 2005*, vol. 1, 270–273.

Ciobanu A.M., Malmasi S. and Dinu L.P. (2018a). German dialect identification using classifier ensembles. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Santa Fe, NM, pp. 288–294.

Ciobanu A.M., Zampieri M., Malmasi S., Pal S. and Dinu L.P. (2018b). Discriminating between Indo-Aryan languages using SVM ensembles. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Santa Fe, NM, pp. 178–184.

Clematide S. and Makarov P. (2017). CLUZH at VarDial GDI 2017: testing a variety of machine learning tools for the classification of Swiss German dialects. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Valencia, Spain, pp. 170–177.

Çöltekin Ç and Rama T. (2017). Tübingen system in VarDial 2017 shared task: experiments with language identification and cross-lingual parsing. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Valencia, Spain, pp. 146–155.

Çöltekin Ç., Rama T. and Blaschke V. (2018). Tübingen-Oslo team at the VarDial 2018 evaluation campaign: an analysis of n-gram features in language variety identification. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Santa Fe, NM, pp. 55–65.

Gamallo P., Pichel, J.R. and Alegria I. (2017). A perplexity-based method for similar languages discrimination. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Valencia, Spain, pp. 109–114.

Gupta D., Dhakad G., Gupta J. and Singh, A.K. (2018). IIT (BHU) system, for Indo-Aryan language identification (ILI) at VarDial 2018. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Santa Fe, NM, pp. 185–190.

Hanani A., Qaroush A. and Taylor S. (2017). Identifying dialects with textual and acoustic cues. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Valencia, Spain, pp. 93–101.

Hasimu M. and Silamu W. (2018). On hierarchical text language-identification algorithms. *Algorithms* **11**(39).

Hollenstein N. and Aepli N. (2015). A resource for natural language processing of Swiss German dialects. In *Proceedings of the International Conference of the German Society for Computational Linguistics and Language Technology (GSCL 2015)*. Germany: University of Duisburg-Essen, pp. 108–109.

Hosmer D.W., Lemeshow S. and Sturdivant R.X. (2013). *Applied Logistic Regression*, 3rd Edn. *Wiley Series in Probability and Statistics*.Hoboken, NJ: John Wiley & Sons, Inc

Indhuja K., Indu M., Sreejith C. and Reghu Raj P.C. (2014). Text based language identification system for Indian languages following Devanagiri script. *International Journal of Engineering Research and Technology* **3**(4), 327–331.

Ionescu R.T. and Butnaru A. (2017). Learning to identify Arabic and German dialects using multiple kernels. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Valencia, Spain, pp. 200–209.

Ionescu R.T. and Butnaru A.M. (2018). Improving the results of string kernels in sentiment analysis and Arabic dialect identification by adapting them to your test set. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, Brussels, Belgium, pp. 1084–1090.

Jauhiainen T. (2010). Tekstin kielen automaattinen tunnistaminen. University of Helsinki.

Jauhiainen H., Jauhiainen T. and Lindén K. (2015a). The Finno-Ugric languages and the internet project. *Septentrio Conference Series* (2), 87–98.

Jauhiainen T., Jauhiainen H. and Lindén K. (2015b). Discriminating similar languages with token-based backoff. In *Proceedings of the Joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects (LT4VarDial)*, Hissar, Bulgaria, pp. 44–51.

Jauhiainen T., Jauhiainen H. and Lindén K. (2018a). HeLI-based experiments in Swiss German dialect identification. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Santa Fe, NM, pp. 254–262.

Jauhiainen T., Jauhiainen H. and Lindén K. (2018b). HeLI-based experiments in discriminating between Dutch and Flemish subtitles. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Santa Fe, NM, pp. 137–144.

Jauhiainen T., Jauhiainen H. and Lindén K. (2018c). Iterative language model adaptation for Indo-Aryan language identification. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Santa Fe, NM, pp. 66–75.

Jauhiainen T., Jauhiainen H. and Lindén K. (2019). Discriminating between Mandarin Chinese and Swiss-German varieties using adaptive language models. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Minneapolis, USA.

Jauhiainen T., Lindén K. and Jauhiainen H. (2015). Language set identification in noisy synthetic multilingual documents. In *Proceedings of the Computational Linguistics and Intelligent Text Processing 16th International Conference, CICLing 2015*, pp. 633–643. Cairo, Egypt.

Jauhiainen T., Lindén K. and Jauhiainen H. (2016). HeLI, a word-based backoff method for language identification. In *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, Osaka, Japan, pp. 153–162.

Jauhiainen T., Lindén K. and Jauhiainen H. (2017a). Evaluating HeLI with non-linear mappings. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Valencia, Spain, pp. 102–108.

Jauhiainen T., Lindén K. and Jauhiainen H. (2017b). Evaluation of language identification methods using 285 languages. In *Proceedings of the 21st Nordic Conference on Computational Linguistics (NoDaLiDa 2017)*, Gothenburg, Sweden, pp. 183–191.

Jauhiainen T., Lui M., Zampieri M., Baldwin T. and Lindén K. (2018). Automatic language identification in texts: a survey. arXiv preprint arXiv:1804.08186.

Jelinek F., Merialdo B., Roukos S. and Strauss M. (1991). A dynamic language model for speech recognition. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pp. 293–295.

Kestemont M., Tschuggnall M., Stamatatos E., Daeleman W., Specht G., Stein B. and Potthast M. (2018). Overview of the author identification task at PAN-2018. In *Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum*, Avignon, France.

Kone Foundation (2012). The language programme 2012-2016.

Kruengkrai C., Sornlertlamvanich V. and Isahara H. (2006). Language, script, and encoding identification with string kernel classifiers. In *Proceedings of the 1st International Conference on Knowledge, Information and Creativity Support Systems (KICSS 2006)*, Ayutthaya, Thailand.

Kumar R., Lahiri B., Alok D., Ojha A.Kr., Jain M., Basit A. and Dawar Y. (2018). Automatic identification of closely-related Indian languages: resources and experiments. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC)*, Miyazaki, Japan.

Lewis M. P., Simons G. F. and Fennig, C. D. (2013). *Ethnologue: Languages of the World*, 17th Edn. Dallas, Texas: SIL International.

Li Y., Baldwin T. and Cohn T. (2018). What's in a Domain? Learning domain-robust text representations using adversarial training. arXiv preprint arXiv:1805.06088v1.

Malmasi S. and Zampieri M. (2017). German dialect identification in interview transcriptions. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Valencia, Spain, pp. 164–169.

Malmasi S., Zampieri M., Ljubešić N., Nakov P., Ali A. and Tiedemann J. (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.

Medvedeva M., Kroon M. and Plank B. (2017). When sparse traditional models outperform dense neural networks: the curious case of discriminating between similar languages. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects*, Valencia, Spain, pp. 156–163.

Murthy K.N. and Kumar G.B. (2006). Language identification from small text samples. *Journal of Quantitative Linguistics* **13**(1), 57–80.

Mustonen S. (1965). Multiple discriminant analysis in linguistic problems. *Statistical Methods in Linguistics* **4**, 37–44.

Pla F. and Hurtado L.-F. (2017). Language identification of multilingual posts from Twitter: a case study. *Knowledge and Information Systems* **51**(3), 965–989.

Priya R., Ojha A.Kr. and Jha G.N. (2018). Automatic language identification system for Hindi and Magahi. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan.

Samardžić T., Scherrer Y. and Glaser E. (2016). ArchiMob–a corpus of spoken Swiss German. In *Proceedings of the Language Resources and Evaluation (LREC)*, Portoroz, Slovenia, pp. 4061–4066.

Scherrer Y. and Rambow O. (2010). Word-based dialect identification with georeferenced rules. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*. Massachusetts, USA: Association for Computational Linguistics, pp. 1151–1161.

Sibun P. and Reynar J.C. (1996). Language identification: examining the issues. In *Proceedings of the 5th Annual Symposium on Document Analysis and Information Retrieval (SDAIR-96)*, Las Vegas, USA, pp. 125–135.

Sreejith C., Indu M. and Reghu Raj P.C. (2013). N-gram based algorithm for distinguishing between Hindi and Sanskrit texts. In *Proceedings of the Fourth IEEE International Conference on Computing, Communication and Networking Technologies*, Tiruchengode, India.

Tan L., Zampieri M., Ljubešic N. and Tiedemann J. (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*, Reykjavik, Iceland, pp. 11–15.

Tiedemann J. and Ljubešić N. (2012). Efficient discrimination between closely related languages. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, Mumbai, India, pp. 2619–2634.

Vaid J. and Gupta A. (2002). Exploring word recognition in a semi-alphabetic script: the case of Devanagari. *Brain and Language* **81**, 679–690.

Vatanen T., Väyrynen J.J. and Virpioja S. (2010). Language identification of short text segments with N-gram models. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010)*, Valletta, Malta, pp. 3423–3430.

Wu N., DeMattos E., So K.H., Chen P.-z. and Çöltekin Ç. (2019). Language discrimination and transfer learning for similar languages: experiments with feature combinations and adaptation. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Minneapolis, USA.

Zampieri M., Malmasi S., Ljubešić N., Nakov P., Ali A., Tiedemann J., Scherrer Y. and Aepli N. (2017). Findings of the VarDial evaluation campaign 2017. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Valencia, Spain.

Zampieri M., Malmasi S., Nakov P., Ali A., Shon S., Glass J., Scherrer Y., Samardžić T., Ljubešić N., Tiedemann J., van der Lee C., Grondelaers S., Oostdijk N., van den Bosch A., Kumar R., Lahiri B. and Jain M. (2018). Language identification and morphosyntactic tagging: the second VarDial evaluation campaign. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Santa Fe, NM.

Zampieri M., Malmasi S., Scherrer Y., Samardžić T., Tyers F., Silfverberg M., Klyueva N., Pan T.-L., Huang C.-R., Ionescu R.T., Butnaru A. and Jauhiainen T. (2019). A report on the third VarDial evaluation campaign. In *Proceedings of the Sixth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Minneapolis, USA.

Zampieri M., Tan L., Ljubešić N. and Tiedemann J. (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)*, Dublin, Ireland, pp. 58–67.

Zampieri M., Tan L., Ljubešić N., Tiedemann J. and Nakov P. (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)*, Hissar, Bulgaria, pp. 1–9.

Zavaliagkos G. and Colthurst T. (1998). Utilizing untranscribed training data to improve performance. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pp. 301–305.

Zhong S., Chen Y., Zhu C. and Liu J. (2007). Confidence measure based incremental adaptation for online language identification. In *Proceedings of International Conference on Human-Computer Interaction (HCI 2007)*, pp. 535–543.

Zlatkova D., Kopev D., Mitov K., Atanasov A., Hardalov M., Koychev I. and Preslav N. (2018). An ensemble-rich multi-aspect approach for robust style change detection - notebook for PAN at CLEF-2018. In *Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum,* Avignon, France.

Cite this article: Jauhiainen T, Lindén K and Jauhiainen H. Language model adaptation for language and dialect identification of text. *Natural Language Engineering* **25**, 561–583. https://doi.org/10.1017/S135132491900038X