Natural Language Processing: Shared Task

Native Language Identification and Proficiency Level Identification

Pauline Claes
Master Digital Text Analysis 2021
Student ID: 20163274
University of Antwerp
pauline.claes@student.uantwerpen.be
10 May 2021

1. Introduction

The current article contributes to a research project for the Natural Language Processing course in the master Digital Text Analysis at the University of Antwerp. This article will explore the performance of traditional machine learning approaches as well as some deep learning approaches in regards to Native Language Identification and Proficiency Level Identification on the TOEFL11 data set (Blanchard et al. 2013). Native Language Identification is the task of automatically identifying the native language of persons based on their writings in the second language. This task is based on the hypothesis that characteristics of L1 will surface and interfere in the production of texts in L2 to the extent that L1 is identifiable (Markov, Nastase, and Strapparava 2020). NLI can be deployed in many fields, such as second language acquisition, but also forensic linguistics, educational applications, advertising, and market research (Chan et al. 2017; Lotfi, Markov, and Daelemans 2020; Blanchard et al. 2013). Proficiency Level Identification is a similar supervised classification task, but differs in that one aims to automatically identify a person's proficiency level and linguistic competence in L2, based on their writings in L2. In what follows, I will first give an overview of some of the related research on Native Language Identification and Proficiency Level Identification in 2. Subsequently, section 3 will elaborate on the used data set, the methods used for each of the tasks, and their respective evaluation and results. Section 4 will discuss the interpretation of the overall results and how they relate to the state-of-the-art research on Native Language Identification. Finally, section 5 will provide a conclusion of this article.

2. Related research

In the 2013 NLI shared task, the winning system achieved the highest accuracy (83.6%) on TOEFL11-TEST by including word, parts-of-speech, and lemma *n*-grams, using an L2-regularized L2-loss Support Vector Machine classifier (Tetreault, Blanchard, and Cahill 2013; Jarvis, Bestgen, and Pepper 2013). However, Ionescu, Popescu, and Cahill (2014) introduced a character-level approach using only character *n*-grams as features in a native language identification task on the ICLE*v*2 (Granger et al. 2009), TOEFL11 (Blanchard et al. 2013) and TOEFL11-Big (Tetreault et al. 2012) corpora. The article

argues that their proposed method has the advantage of being language independent and linguistic theory neutral, while it can also be effectively used for cross-corpus experiments. Using Kernel Ridge Regression and Kernel Discriminant Analysis combined with character-level features and reaching 85.3% accuracy, they have demonstrated that both methods outperform the then state of the art, even though Kernel Ridge Regression performed better than Kernel Discriminant Analysis. Furthermore, the article concludes that the selection of features during the training step by the kernel classifier actually performs better than the usual Natural Language Processing approach of feature selection before training.

In the 2017 NLI shared task, the winning system obtained a classification accuracy of 88.2%, by proposing a stacked sentence-document architecture. More specifically, they stacked two SVM classifiers, each operating on a different feature type. The first classifier is an L1 sentence classifier and its output is used as features by the subsequent L1 document classifier. Their best performing stacked classifier takes into account a number of specific features such as character, lemma, word and CPOS *n*-grams, type-token ratio, and average sentence length (Cimino and Dell'Orletta 2017).

Malmasi et al. (2017) conclude that in the 2017 shared task, multiple classifier systems such as meta-classifiers (or classifier stacking), ensemble combination methods, and multiple kernel learning prove to be most effective. Furthermore, the article argues that lexical *n*-grams such as word and character *n*-grams are the best single feature type, while the addition of syntactic features can provide a slight performance boost. The article also notes that the results from the 2017 shared task confirm the findings from Malmasi et al. (2016) that traditional supervised machine learning models generally still outperform newer deep learning approaches not only in terms of performance, but also taking into account training times.

Markov, Nastase, and Strapparava (2020) provide an investigation of a variety of linguistic phenomena found in two NLI corpora: TOEFL11 (Blanchard et al. 2013) and ICLE*v*2 (Granger et al. 2009). The article aims to fill the gap in performance between a baseline essay representation (part-of-speech tags and function words) and word and character *n*-gram text representations, by adding features that capture specific linguistic phenomena. More specifically, they focus on punctuation usage, emotion expression in language, and formal similarities in native language use through the use of anglicized words, cognates, and other misspellings. In fact, the article concludes that native language interference can be found in (i) the use of punctuation marks to structure written information, (ii) in emotion expression, and (iii) in the use of anglicized words, cognates, and spelling errors. This study demonstrates that there exist patterns of punctuation that are not only shared by language families, but that there are also patterns that are specific to individual languages. Moreover, they illustrate that the interference of punctuation is not influenced by the level of proficiency of the person or author in question. By adding types of sentiments and the frequency of emotion words to the baseline representation (POS-tags and function words) and thus improving its predictions, they conclude that emotion words and characteristics from the native language noticeably interfere in the second language production. Furthermore, the article demonstrates that even though investigating misspelled cognates and anglicized words is useful for native language identification, it should be noted that the frequency of misspellings in general decreases when the proficiency level increases. In conclusion, Markov, Nastase, and Strapparava (2020) have demonstrated that the combination of all investigated phenomena leads to the best performance in their research using the liblinear scikit-learn (Pedregosa et al. 2011) support vector machines (SVM) with OvR (one vs. rest) multi-class strategy.

While it is common to approach Native Language Identification tasks as a supervised multi-class classification problem from a traditional machine learning perspective, Lotfi, Markov, and Daelemans (2020) propose a deep generative language modelling approach using fine-tuned GPT-2 models on ICLE*v*2 (Granger et al. 2009) and TOEFL11 (Blanchard et al. 2013). In fact, instead of training a classifier, they fine-tuned a GPT-2 model per native language (L1) included in each of the data sets, so that each model will have learned the characteristics of a certain L1. Subsequently, an unseen text is fed to all of these models in order to evaluate their LM loss (a measure of likelihood). The model with the lowest loss is then selected to assign the L1 class for an unseen text. The article demonstrates that this proposed method not only outperforms the baseline methods included, but also previous state-of-the-art results, obtaining a classification accuracy of 89% (test set) and 86.8% (10-fold cross-validation) on TOEFL11, and 94.2% on the ICLE data set. Moreover, they argue that their approach is simple, does not require feature engineering, and still achieves the best results on these data sets.

In general, it is clear that Lotfi, Markov, and Daelemans (2020) have improved the state of the art significantly with a fine-tuned GPT-2 model, obtaining 89% classification accuracy on TOEFL11. However, approaching the task of Native Language Identification from a traditional machine learning perspective is still common and allows for comparable state-of-the-art results on the NLI task.

In recent years, there has been an increasing interest in the use of Natural Language Processing technologies for the study of linguistic competence. In that respect, Zarco-Tejada (2019) explored Proficiency Level Identification on the CEFR-levelled English Corpus (CLEC), an English corpus compiled for this research purpose. After having analysed the linguistic features per level in the corpus statistically, this article explores which of those features are the most representative per level and whether the automatic linguistic profiling of CEFR-levelled texts validates the use of NLP tools for proficiency level identification. While the design of this research is relevant for the task of Proficiency Level Identification, it surpasses the scope of this project. In addition, it only discusses the most significant features for the levels A2, B1, and B2, which roughly correspond to the levels *low* and *medium* in the TOEFL11 data set. Markov, Nastase, and Strapparava (2020) conclude that traces of the native language through the use of punctuation marks and emotion words persist in high proficiency levels, and that the ability to choose the desired lexical material for emotion expression increases with the proficiency level. Furthermore, they demonstrate that while the influence of L2-ed words decreases when the proficiency level increases, the influence of cognates increases with the proficiency level, even though the number of L2-ed words remains higher than the number of cognates across all proficiency levels.

3. Experimental set up

This section of the paper will discuss the set up of the included experiments. First, I will discuss the used data set in detail in 3.1. Some features were extracted and added to the data set, to be used in either Native Language Identification or Proficiency Level Identification. These will be discussed in 3.1.1. After that, the two subsections *Native Language Identification* (3.2) and *Proficiency Level Identification* (3.3) will each discuss its evaluation metrics, and the included features and classifiers, as well as their results.

For both tasks, three baselines were used: Dummy Classifier, Support Vector Machine and BERT. BERT is a pre-trained deep bidirectional transformer (Devlin et al. 2018). The Dummy Classifier and Support Vector Machine were each paired with both CountVectorizer and TfidfVectorizer, and instantiated with the default parameters.

3.1 Data set

The corpus used for this shared task is TOEFL11, which was compiled specifically for the task of native language identification in the 2013 NLI Shared Task (Tetreault, Blanchard, and Cahill 2013), grammatical error detection and correction, and automatic essay scoring. It consists of essays written during a high-stakes college-entrance test (Blanchard et al. 2013). The entire corpus consists of 12,100 essays in total, i.e. 1,100 essays per L1. There are 11 L1s included: Arabic, Chinese, French, German, Hindi, Italian, Japanese, Korean, Spanish, Telugu, and Turkish (Blanchard et al. 2013). In addition, the data set includes the proficiency level (*low, medium, high*) of each essay included. While the L1 labels are balanced, the proficiency levels are not. It consists of 1,330 essays for *low*, 6,568 for *medium*, and 4,202 for *high*.

For this task specifically, a test subset of 100 essays per L1 was kept apart. Consequently, the remaining subset consisted of 1000 essays per L1, and 1,201 essays for *low* proficiency, 5,964 for *medium* proficiency, and 3,835 for *high* proficiency. After that, this subset was distributed into a train and development set using either $train\_test\_split$. For the task of Native Language Identification, this was done in order for the train and development set to consist of respectively 900 and 100 essays per L1, so that the development set used for model selection and evaluation would form an accurate representation of the test set that had been kept apart. For the task of Proficiency Level Identification, the distribution resulted in 1,081 essays for *low*, 5,368 for *medium*, and 3,451 for *high* in the train set, and 120 essays for *low*, 596 for *medium*, and 384 essays for *high*. Table 1 and table 2 summarize the distributions in the respective sets, with "total" representing the subset of 11,000 essays used for this task, not the entire TOEFL11 corpus.

L1	Total	Train set	Dev. set
ARA	1,000	900	100
CHI	1,000	900	100
FRA	1,000	900	100
DEU	1,000	900	100
HIN	1,000	900	100
ITA	1,000	900	100
JPN	1,000	900	100
KOR	1,000	900	100
SPA	1,000	900	100
TEL	1,000	900	100
TUR	1,000	900	100

Table 1
Number of essays per L1 before and after $train\_test\_split$

3.1.1 Feature extraction.

The original train set is constructed by four columns: *Filename*, *text*, *Language*, and *Proficiency*. From the *text* column, some linguistic and syntactic features were extracted to be used either in the Native Language Identification task or in the Proficiency Level Identification task. These will be discussed in this section.

Proficiency	Total	Train set	Dev. set
Low	1,201	1,081	120
Medium	5,964	5,368	596
High	3,835	3,451	384

Table 2
Number of essays per proficiency level before and after $train\_test\_split$

Part-of-speech tags. The TOEFL11 data was tagged using the spaCy module (spaCy Usage Documentation). The POS-tags for each document were added in a separate column. POS *n*-grams are considered to be one of the core features for the task of Native Language Identification, capturing mainly local syntactic patterns of language use. Especially POS *n*-grams of order 1-3 are considered to be useful, with POS trigrams being the best single feature, while sequences of size 4 and greater seem to reduce classification performance (Malmasi and Dras 2017a).

Lemmatized text. The essays were lemmatized using the spaCy module (spaCy Usage Documentation). Lemma *n*-grams are a commonly used surface feature in the task of Native Language Identification (Markov, Nastase, and Strapparava 2020; Malmasi and Dras 2017b). This feature was added in a separate column.

Function words. Function words are a closed class word category consisting of heavily grammaticalized words, and are therefore reliable for textual comparison. In addition, the use of function words is not strongly affected by a topic or genre and it is supposedly less under an author's control during the writing process. That is why these features have proven to be very useful for authorship attribution (Kestemont 2014). Furthermore, alongside POS *n*-grams, function words are one of the core features in the task of Native Language Identification (Malmasi and Dras 2017a). The function words are extracted using NLTK's stopwords corpus (Bird, Klein, and Loper 2009), and added to a separate column.

Text length. Three columns were added for measuring the text length of every essay, containing the text length in number of tokens, number of sentences and number of characters, respectively.

3.2 Native Language Identification

This section will discuss the model selection and evaluation process for the task of Native Language Identification. The performance is evaluated using classification accuracy on the development set. First, in 3.2.1, I will discuss the features included in the best performing model, as well as some features that were examined but did not improve performance. Subsequently, I will discuss in detail the model that performed best on the NLI task, while also discussing other models considered in this research. In order to compare the examined models, three baselines were included. These will be discussed in 3.2.2. An overview of all methods and results can be found in table 3. A confusion matrix of the predictions of the best classifier on the development will be shown and discussed in 4.

3.2.1 Included features.

For the task of Native Language Identification, the following features were examined. The features included in the best performing model are marked with an asterisk.

- Token *n*-grams of the order 1-4
- Lemma *n*-grams of the order 1-4 *
- Text length in number of tokens
- Character *n*-grams of the order 1-3
- Function word uni- and bigrams

3.2.2 Baselines. As mentioned above, three baselines were used: Dummy Classifier, Support Vector Machine and BERT. Dummy Classifier and SVM were both paired with CountVectorizer and TfidfVectorizer. It was clear from the SVM baseline that TfidfVectorizer performed best, yielding a difference of 12% in classification accuracy. This is why all considered classifiers were built with TfidfVectorizer.

3.2.3 Optimized stacked classifier. The classifier obtaining the highest classification accuracy on the development set (83.4%) is a stacked classifier that was fit on lemma *n*-grams. The classifier was built using scikit-learn's *StackingClassifier* module. It consists of three consecutive optimized pipelines, each made up of a TfidfVectorizer and the respective classifier. The three classifiers included are scikit-learn's implementation of Logistic Regression, Stochastic Gradient Descent Classifier, and LinearSVC (Pedregosa et al. 2011). These were first added to separate pipelines in combination with TfidfVectorizer, using the default parameters. Using scikit-learn's *RandomizedSearchCV* module on each of the pipelines, it was possible to find the optimal hyperparameters for TfidfVectorizer as well as for the respective classifier in all three of the combinations. The stacked classifier was then constructed of three consecutive pipelines, each consisting of an optimized TfidfVectorizer and an optimized classifier. That way, every classifier is defined with its optimal parameters as well as with its own vectorizer containing the parameters that have been proven to fit best for that classifier. Depending on the vectorizer, the lemma *n*-grams ranged from unigrams to fourgrams. While this stacked classifier performed best when fit on the lemmatized text column, its performance on the text column is comparable.

Every vectorizer is instantiated with NLTK's word tokenizing module (Bird, Klein, and Loper 2009), and an *'l2'* norm. The optimized parameters of the pipelines constituting the stacked classifier are the following:

- Pipeline 1.
  - TfidfVectorizer. Vectorizes lemma unigrams with a document frequency higher than 0.01% (0.001 as a float) and lower than 80%, without taking into account lowercase.
  - Logistic Regression using the 'one vs. rest' multi-class strategy, a C value of 1.0 (the inverse regularization strength), with maximally 2500 iterations and a random state of 1.
- Pipeline 2.
  - TfidfVectorizer. Lowercases the input and takes into account lemma *n*-grams of the order 1-4 occurring minimally in 10% of documents.

  - Stochastic Gradient Descent Classifier instantiated with *'l1'* penalty, 'hinge' loss function, maximally 1000 iterations, and a random state of 1.

- Pipeline 3.
  - TfidfVectorizer. Analyzes lemma uni-, bi-, and trigrams occurring minimally in 0.01% and maximally in 80% of documents, without lowercasing.
  - LinearSVC with its default parameters and a random state of 1.

The final estimator in the stacked classifier is a multinomial Logistic Regression with the maximum number of iterations set to 1000.

3.2.4 Discarded classifiers. Apart from the individual pipelines that were built for the optimized stacked classifier yielding the best results on the development set (discussed above), a Logistic Regression classifier was used in two different pipelines with a combination of features. More specifically, token uni- and bigrams, character uni-, bi- and trigrams, and the text length in number of tokens were included in the first pipeline. The second pipeline took into account lemma uni- and bigrams, character uni-, bi- and trigrams, and function word uni- and bigrams. Furthermore, another stacked classifier was used in combination with different features. This stacked classifier consists of the same classifiers with the same parameters as the stacked classifier discussed above, and was able to reach a comparable classification accuracy. The features included in this classifier are text length in number of tokens, token uni- and bigrams, character uni-, bi-, and trigrams, and lemma uni- and bigrams.

3.2.5 Results. In general, it is clear that all considered models easily outperformed the Dummy Classifier, BERT, and SVM with CountVectorizer baselines. While most classifiers also outperformed the SVM with TfidfVectorizer, Logistic Regression in combination with token unigrams (71.9%), and Logistic Regression trained on lemma uni- and bigrams, function word uni- and bigrams, and character uni-, bi-, and trigrams (70.5%) were not able to.

The stacked classifier as described in 3.2.3 yielded the highest classification accuracy on the development set, and is therefore the best model considered in this research. Consisting of three consecutive optimized pipelines and taking into account lemma *n*-grams of order 1-4 depending on the vectorizer included, this stacked classifier reaches 83.4% classification accuracy on the development set. The same stacked classifier was used on token *n*-grams of order 1-4 depending on the vectorizer, and was able to reach a similar but slightly inferior classification accuracy of 82.9%.

3.3 Proficiency Level Identification

This section on Proficiency Level Identification will touch on the model selection and evaluation process for this task. The performance of the models is measured in macro-average f1-score obtained on the development set. In 3.3.1, I will review the features included in the best performing classifier, as well as the features that were considered but did not improve performance. After that, I will discuss in detail the best performing model for this task, as well as the other models that were included in this research. As for the task of Native Language Identification, three baselines were included and will be briefly discussed in 3.3.2. A detailed overview of all methods and results can be

	Method	(%)*
Baselines	Dummy Classifier (CountVectorizer) *Token unigrams*	8.4
	Dummy Classifier (TfidfVectorizer) *Token unigrams*	8.4
	Support Vector Machine (CountVectorizer) *Token unigrams*	60.4
	Support Vector Machine (TfidfVectorizer) *Token unigrams*	72.3
	BERT *Text column*	48.2
Single Classifiers**	Logistic Regression *Token unigrams*	71.9
	Stochastic Gradient Descent Classifier *Token unigrams*	75.5
	LinearSVC *Token unigrams*	76.3
	Logistic Regression *Token n-grams (1-2), number of tokens, character n-grams (1-3)*	73.3
	Logistic Regression *Lemma n-grams (1-2), function word n-grams (1-2), character n-grams (1-3)*	70.5
Stacked Classifiers**	Logistic Regression, SGDClassifier, LinearSVC *Token n-grams (1-4) (depending on vectorizer)*	82.9
	Logistic Regression, SGDClassifier, LinearSVC *Lemma n-grams (1-4) (depending on vectorizer)*	**83.4**
	Logistic Regression, SGDClassifier, LinearSVC *Token n-grams (1-2), lemma n-grams (1-2), character n-grams (1-3), number of tokens*	81.5
(*) Classification accuracy on development set (**) All with TfidfVectorizer		

Table 3
An overview of classifiers and results for the task of Native Language Identification.

found in table 4. A confusion matrix of the best classifier on the development set will be discussed in section 4.

As mentioned in 3.1, the distribution of the proficiency levels in the data set is not balanced. It is important to take this into account when building and training a classifier. For example, if a certain class in the data set comprises 70% of all instances, a classifier would report an optimistic classification accuracy of 70% by simply predicting (classifying) every instance as the majority class. There are several ways to anticipate this. One would be to set all instances per proficiency level to the amount of the level with the least amount of instances. That way, we would manipulate a balanced data set and every class would have the exact same amount of training examples. While this technique might be very useful, I decided not to do this as the data set is already quite limited and setting all classes equal to the class with the least amount of instances (*low*), means that 7,397 training instances would have to be discarded, which is more than half

of the data set. Instead, all classifiers were instantiated with the class weight parameter set to "balanced". This makes sure that the weights of each class are automatically adjusted inversely proportional to the class frequencies in the input data (Pedregosa et al. 2011).

3.3.1 Included features. For Proficiency Level Identification, the following features were examined. The features included in the best performing model are marked with an asterisk. All classifiers except for two baselines were instantiated with CountVectorizer using NLTK's word tokenizing module (Bird, Klein, and Loper 2009).

- Token unigrams
- Token uni-, bi-, and trigrams *
- Number of tokens *
- POS-tag uni-, bi-, and trigrams *
- Function word uni-, bi-, and trigrams

3.3.2 Baselines. Three baselines were used for performance comparison: Dummy Classifier, Support Vector Machine and BERT. The first two classifiers were each paired with CountVectorizer and TfidfVectorizer. First, it became clear that an SVM in combination with CountVectorizer, both instantiated with their default parameters, was able to reach a considerable macro-average f1-score (71.8%). Therefore, all subsequent models were built with CountVectorizer.

3.3.3 Best classifier. For Proficiency Level Identification, there are actually two classifiers who performed equally as well, reaching a similar macro-average f1-score of 73.2% on the development set. Both of them use Logistic Regression as a classifier, using 'one vs. rest' multi-class strategy, a balanced class weight, 5000 maximum iterations and a random state of 1.

The first best classifier is trained on token uni-, bi-, and trigrams, POS-tag uni-, bi-, and trigrams and the text length in number of tokens. The second best classifier is trained on the same features, but also takes into account function word uni-, bi-, and trigrams. While the addition of this feature might be useful, it does not improve the performance of the classifier. Following the principle of parsimony, the first best classifier is selected as the final best estimator and will be applied on the test set predictions.

3.3.4 Discarded classifiers. Alongside the best classifiers for this task, a LinearSVC, Logistic Regression, and Stochastic Gradient Descent Classifier were trained on token uni-, bi-, and trigrams. Each of those were instantiated with a balanced class weight, the maximum number of iterations set to 5000, and a random state of 1. In addition, LinearSVC was instantiated using the 'one vs. rest' multi-class strategy, and the fine-tuned Stochastic Gradient Descent Classifier was trained using a logistic regression loss function.

In addition, two Logistic Regression classifiers instantiated with balanced class weight, the maximum number of iterations set to 5000, and a random state of 1 were trained respectively on token uni-, bi-, and trigrams and text length in number of tokens and on lemma uni-, bi-, and trigrams and text length in number of tokens. While both were able to obtain satisfactory macro-average f1-scores, the latter was not

able to improve upon the best baseline classifier. Therefore, lemma *n*-grams were not included in the best classifier in this research. From all conducted experiments, it is clear that Logistic Regression performed best, consistently reaching macro-average f1-scores higher than 70%.

3.3.5 Results. As mentioned in 3.3.2, the baseline consisting of Support Vector Machine and CountVectorizer performed notably well, reaching a macro-average f1-score of 71.8%. Unfortunately, only a few of the considered classifiers were able to outperform that baseline. However, the other baselines (Dummy Classifier with CountVectorizer and TfidfVectorizer, SVM with TfidfVectorizer, and BERT) were all outperformed. First of all, the two best performing models discussed in 3.3.3 were both able to reach 73.2%. Secondly, a Logistic Regression classifier trained on token uni-, bi-, and trigrams obtained 72.2%, and lastly, a Logistic Regression classifier trained on token uni-, bi-, and trigrams and the text length in number of tokens, yielded a macro-average f1-score of 72.4% on the development set.

In general, it is clear that the best performing classifier improved upon all of the baselines. However, it should be noted that the difference only consists of 1.4% in macro-average f1-score. Still, the text length in number of tokens and the addition of POS-tag uni-, bi-, and trigrams is useful for the identification of proficiency level in the second language.

4. Discussion

In general, it is clear from this research that the included classifiers were not able to outperform the current state of the art, while the selected best classifiers for both of the classification tasks still performed relatively well on the data set, reaching a classification accuracy of 83.4% for the task of Native Language Identification and 73.2% for Proficiency Level Identification. Confusion matrices for each of the tasks can be found in figures 1 and 2 respectively.

The results of this research confirm the findings from Malmasi et al. (2016), that traditional supervised machine learning models generally still outperform the newer deep learning approaches such as BERT, both in terms of general performance, but also taking into account training times. However, it should be noted that these findings can only be confirmed to the extent that deep learning approaches were used in this research, as the current state of the art was not outperformed by the models considered.

For the task of Native Language Identification, the best performing classifier reached a classification accuracy of 83.4%. While the classifier performed considerably well on all language categories, there are some outliers to consider. First of all, German appears to be the easiest to predict, obtaining the highest recall score (93%). Italian, Chinese, French, and Turkish also demonstrate a high recall score of 86% and more. Hindi and Telugu seem to be harder to predict, and are most often misclassified as each other, even though Hindi is more often classified as Telugu than the other way around. This is not surprising, considering both languages are spoken in India and therefore might influence each other, even though they have both originated from two different language families (Sengupta and Saha 2015). In addition, Spanish is often confused with Italian, French, Arabic and Turkish. Korean, Japanese, and Chinese are often misclassified as each other as well. We can hypothesize that these native languages presumably demonstrate some morphological similarities that persist and remain noticeable in the second language performance, as this classifier is only trained on lemma *n*-grams, and not on grammatical or syntactic features.

	Methods	(%)*
Baselines	Dummy Classifier (CountVectorizer) *Token unigrams*	32.5
	Dummy Classifier (TfidfVectorizer) *Token unigrams*	32.5
	Support Vector Machine (CountVectorizer) *Token unigrams*	71.8
	Support Vector Machine (TfidfVectorizer) *Token Unigrams*	58.9
	BERT *Text column*	NA
Single classifiers**	LinearSVC *Token uni-, bi-, trigrams*	66.6
	Logistic Regression *Token uni-, bi-, trigrams*	72.2
	Stochastic Gradient Descent Classifier *Token uni-, bi-, trigrams*	60.9
	Logistic Regression *Token n-grams (1-3), number of tokens*	72.4
	Logistic Regression *Lemma n-grams (1-3), number of tokens*	70.8
	Logistic Regression *Token n-grams (1-3), number of tokens,* *POS-tag n-grams (1-3)*	**73.2**
	Logistic Regression *Token n-grams (1-3), number of tokens,* *POS-tag n-grams (1-3), function word n-grams (1-3)*	73.2
(*) Macro-average f1-score on development set (**) All with CountVectorizer		

Table 4
An overview of classifiers and results for the task of Proficiency Level Identification.

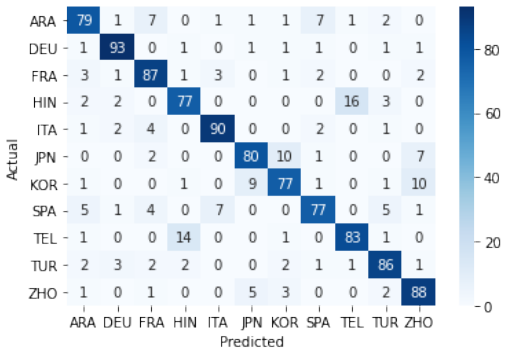

Figure 1
Confusion matrix of the best classifier for the task of Native Language Identification

For the task of Proficiency Level Identification, it is clear that the classification was notably more difficult for the *low* class than for the *medium* and *high* class. This is not surprising, as there were significantly fewer training instances for the *low* class (1,081) than for the other two classes (*medium*: 5,368 ; *high*: 3,451). In addition, the best classifier takes into account text length in number of tokens, alongside token uni-, bi-, and trigrams as well as POS-tag uni-, bi, and trigrams. It is remarkable that the mean number of tokens differs noticeably per proficiency level. The *high* level authors use on average 363 tokens per essay, whereas *medium* authors use 306 and the *low* authors use 206 tokens per essay on average. However, it is clear that the inclusion of the morphosyntactic feature of POS-tag uni-, bi-, and trigrams results in an improvement of classification performance.

To summarize, the selected best classifiers for the task of Native Language Identification and Proficiency Level Identification performed relatively well on the data set, without improving the current state of the art. However, this research confirms the findings from Malmasi et al. (2016) that traditional supervised machine learning models generally still outperform deep learning approaches to the extent that these were used in this research.

Future research could improve upon the techniques discussed in this article by including more useful morphosyntactic and grammatical features such as POS-tags, function words, punctuation usage, and the use of emotion words in combination with the proposed stacked classifier.

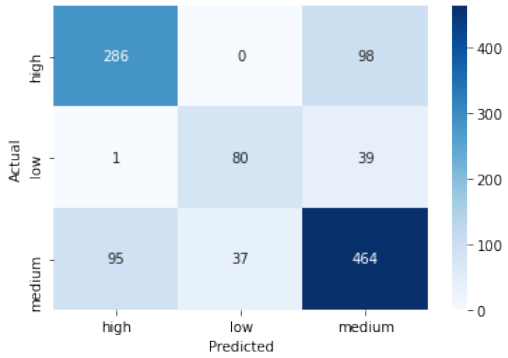

Figure 2
Confusion matrix of the best classifier for the task of Proficiency Level Identification.

5. Conclusion

This article contributes to the shared task for the Natural Language Processing course and explored the performance of traditional machine learning approaches as well as some deep learning approaches for Native Language Identification and Proficiency Level Identification on the TOEFL11 data set.

The best classifier for NLI was a stacked classifier containing three consecutive optimized pipelines and was able to reach 83.4% classification accuracy on the development set, and trained on lemma *n*-grams of the order 1-4. It performed notably well on most language categories, although some outliers were observable. While German appeared the easiest to predict, Hindi and Telugu are often confused, as well as Korean, Japanese,

and Chinese. We can hypothesize that the languages that often get confused arguably demonstrate morphological similarities that surface in the second language production.

In Proficiency Level Identification, the best classifier is a single Logistic Regression classifier trained on token uni-, bi-, and trigrams, POS-tag uni, bi-, and trigrams and the text length in number of tokens, obtaining a macro-average f1-score of 73.2% on the development set.

In conclusion, the models considered in this research have not outperformed the current state of the art, but nevertheless performed relatively well on the TOEFL11 data set, both for the task of Native Language Identification as for the task of Proficiency Level Identification, even though the former has proven to yield more satisfactory results on the development set.

References

Bird, Steven, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Blanchard, Daniel, Joel Tetreault, Derrick Higgins, Aoife Cahill, and Martin Chodorow. 2013. Toefl11: A corpus of non-native english. *ETS Research Report Series*, 2013(2):i–15.

Chan, Sophia, Maryam Honari Jahromi, Benjamin Benetti, Aazim Lakhani, and Alona Fyshe. 2017. Ensemble methods for native language identification. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 217–223.

Cimino, Andrea and Felice Dell'Orletta. 2017. Stacked sentence-document classifier approach for improving native language identification. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 430–437.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Granger, Sylviane, Estelle Dagneaux, Fanny Meunier, and Magali Paquot. 2009. The international corpus of learner english. version 2. handbook and cd-rom.

Ionescu, Radu Tudor, Marius Popescu, and Aoife Cahill. 2014. Can characters reveal your native language? a language-independent approach to native language identification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1363–1373.

Jarvis, Scott, Yves Bestgen, and Steve Pepper. 2013. Maximizing classification accuracy in native language identification. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 111–118.

Kestemont, Mike. 2014. Function words in authorship attribution. from black magic to theory? In *Proceedings of the 3rd Workshop on Computational Linguistics for Literature (CLFL)*, pages 59–66.

Lotfi, Ehsan, Ilia Markov, and Walter Daelemans. 2020. A deep generative approach to native language identification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1778–1783, International Committee on Computational Linguistics, Barcelona, Spain (Online).

Malmasi, Shervin and Mark Dras. 2017a. Multilingual native language identification. *Natural Language Engineering*, 23(2):163–215.

Malmasi, Shervin and Mark Dras. 2017b. Native language identification using stacked generalization. *arXiv preprint arXiv:1703.06541*.

Malmasi, Shervin, Keelan Evanini, Aoife Cahill, Joel Tetreault, Robert Pugh, Christopher Hamill, Diane Napolitano, and Yao Qian. 2017. 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*, pages 62–75.

Malmasi, Shervin, Marcos Zampieri, Nikola Ljubešić, Preslav Nakov, Ahmed Ali, and Jörg Tiedemann. 2016. Discriminating between similar languages and arabic dialect identification: A report on the third dsl shared task. In *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, pages 1–14.

Markov, Ilia, Vivi Nastase, and Carlo Strapparava. 2020. Exploiting native language interference for native language identification. *Natural Language Engineering*, page 1–31.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Sengupta, Debapriya and Goutam Saha. 2015. Study on similarity among indian languages using language verification framework. *Advances in Artificial Intelligence*, 2015.

Tetreault, Joel, Daniel Blanchard, and Aoife Cahill. 2013. A report on the first native language identification shared task. In *Proceedings of the eighth workshop on innovative use of NLP for building educational applications*, pages 48–57.

Tetreault, Joel, Daniel Blanchard, Aoife Cahill, and Martin Chodorow. 2012. Native tongues, lost and found: Resources and empirical evaluations in native language identification. In *Proceedings of COLING 2012*, pages 2585–2602.

spaCy Usage Documentation. Linguistic features. Accessed: 2021-04-26.

Zarco-Tejada, Maria Angeles. 2019. Automatic profiling of l2-simplified texts: Identifying discriminate features of linguistic proficiency. *Digital Scholarship in the Humanities*, 34(3):661–675.