# Native vs. non-native language classification

Zorana, Sanne, Alejandro, Sanne, Hans

Vrije Universiteit Amsterdam

**Abstract**

*In a 2012 preprint Al-'Rfou reports achieving a 74.53% accuracy in classifying native versus non-native speech based on comments extracted from Wikipedia using a linear SVM classifier. The used feature set consists of similarity scores against different k-gram models. This is a non-standard choice as state-of-the-art classification performance in the closely related L1-classification task is achieved by classifying directly based on the Document-Term Matrix (DTM). The main demonstrations of this paper are that incorporation of such information boosts accuracy with ±2%, and accuracy can be increased by another ±2% by using a tuned RBF kernel for the SVM classifier. These increases may seem marginal, but were achieved under limited computational resources (an increase in these may further enhance accuracy). Finally, it should be noted the classification task at hand is inherently complicated by heterogeneity in the mother tongue of writers of "non-native" comments. We demonstrate this explicitly and hypothesize that future increases in accuracy may be achieved by explicitly taking this heterogeneity into account in the classification step.*

## 1. Introduction

Language transfer refers to the phenomenon that in learning new languages, people draw from from knowledge of their first or previously learned languages. The phenomenon means that the non-nativeness of a speaker can reveal itself through spelling mistakes and grammatical errors, but also through word choices and sentence constructions that may seem odd to native speakers of the language [Tsur and Rappoport, 2007]. Existence of such a phenomenon should make it possible to induce the native language of a person (L1) based on text written by that person in a second language (L2). In machine learning/text mining, this task is referred to as L1-classification.

Many of the publications for L1-classification are based on English essays or scientific articles written by non-native speakers [Gebre et al., 2013, Jarvis et al., 2013]. Such documents have a non-conversational writing style, are typically long and typically limited in the number of topics they cover. Therefore, it is unclear whether the classification performances of 80-90% generalize to conversational writing styles and shorter pieces of text. It is also unclear whether these high performances stem from topic bias (e.g. Italian speakers are more likely to write about Italy), and if so, to which extent.

A 2012 paper by Al-Rfou' aimed to address this gap by using comments retrieved from the discussion pages of the English Wikipedia ("Wikitalk data") [Al-Rfou', 2012]. In this study, proper nouns (nouns that refer to a unique entity or event such as Italy, World War II, Brexit) were removed in pre-processing so as to minimize topic bias. The paper reports >74% accuracy in classifying whether the writer of a comment is a native or non-native English speaker compared to a baseline of 50%. This result was reached with both Support Vector Machines (SVMs) and Logistic Regression (LR). A first goal of this report is to reproduce these findings, and to see whether this accuracy may be enhanced through parameter tuning and a different classifier choice.

For classification, Al-'Rfou uses the text in comments in the test/validation to compute similarity scores against a k-gram model that is constructed based on comments in the training set. This choice is unusual in that papers reaching the highest accuracies in the closely related L1-classification task train classifiers directly on k-gram counts, rather than on similarity scores (i.e. they feed the Document-Term Matrix to classifiers). A second goal of this report is therefore to see if we can increase accuracy by training classifiers on the k-gram counts (DTM) directly.

Finally, we wanted to investigate if two recent results in L1-classification generalize to the Wikitalk data. The first is a paper by Kulmizev et al., where L1-classification based on character 9-grams alone performs better than classifiers that include more features [Kulmizev et al., 2017]. The second is a paper by Tetreault et al. which finds that ensemble methods with different classifiers for different k-grams outperform a single classifier for all k-grams alone [Tetreault et al., 2012].

## 2. Structure of this paper

In Section 3 we discuss some basic terminology in the field of text-mining which we use throughout this paper. In Section 4 we demonstrate that we are

nearly able to reproduce a claim by Al-'Rfou that we can get $\pm 74\%$ accuracy in classifying native versus non-native comments extracted from Wikipedia based on similarity scores against k-gram models. In Section 5 we will investigate if we can increase the accuracy of this classification by tuning the SVM parameters (i.e. the soft margin parameter $C$ and the kernel type). In the same section, we will also investigate classification performance with Random Forests. We proceed in Section 6 by evaluating classification performance based on the document-term matrix directly, rather than collapsing all information into a single-dimensional similarity score. In Section 7, the power of higher order k-grams (10-grams) will be investigated. Then in Section 8, it is assessed whether a simple majority vote of combinations of classifiers increases overall performance.

In all above-mentioned sections, we develop hypotheses with regards to our classifiers using the validation set. In Section 9 we formally assess these hypotheses using the test set. We end with a brief discussion how classification performance may be improved in the future. For the sake of replicability, all code (Jupyter notebooks and R Markdown) is made available through Github[1].

## 3. Basics of Natural Language Processing (NLP)

Analysis of language is a non-trivial problem. In the context of machine learning, it is essential that we reduce dimensionality of language, and translate the language itself into features comprehensible to a computer. Perhaps the simplest feature one could think of are simple word counts. Word counts as only features represent quite some loss of information, however: not only do we lose the order of the words in the sentence, but we also lose meaning of certain words. A textbook example of such a loss is "New York City", which gives rise to word counts of "New", "York" and "City", none of which refer uniquely to the well-known city.

k-grams are a tool to limit such loss of information by counting combinations of words, rather than words themselves. E.g. "New York City" would lead to the bigrams of ("New","York") and ("York","City"), and the trigram of ("New","York","City"). Construction of these k-grams is not restricted to the domain of words. Al-'Rfou pursues construction of k-grams for word lengths, characters, words, and the Part-of-Speech tags (i.e. a representation of the grammatical structure of a sentence) for k's ranging from 1 to 4. In this paper, we too construct such k-grams and rely on Python's `nltk` package for their extraction.

One way to represent a set of text documents is the document-term matrix (DTM). This sparse representation of text documents DTM has as elements $DTM_{ij} \in \mathbb{N}$ the counts of gram $j$ in document $i$. Note that the DTM matrix consists of $n_{documents} \times n_{terms}$ such that it is typically very large if one has many documents and one does not restrict the number of unique terms/grams. In the Al-'Rfou paper, this problem is circumvented by collapsing gram counts into a one-dimensional similarity score (see next section). As mentioned in the introduction, we aim to use information from the DTM directly as features for classification of native versus non-native English. However, we are limited to computers with 8GB RAM such that we need to limit construction of grams to grams that are not rarely used and to some extent distinctive of native/non-native English.

## 4. Reproduction Al-'Rfou paper

The raw data extracted by Al-'Rfou from the Wikipedia discussion pages was retrieved from BitBucket [2]. With only the raw data available - not similarity scores - data had to be cleansed. The important steps in this pre-processing by Al-'Rfou include:

- Removing any comments consisting of less than 20 words.
- Replacing non-ASCII characters (among which are e.g. Chinese and Japanese characters) by *.
- Removing words identified as proper nouns. This is necessary to prevent classifying native versus non-native speakers based on topic bias. E.g. non-native Germans are more likely to use the word "Berlin". To prevent topic bias from becoming a confounding factor, proper nouns were recognized using SENNA V3.0 [Collbert et al., 2011] and replaced by "NNP".

Conceptually the preprocessing steps are the same as in Al-'Rfou. However, there are slight differences (e.g. we used SENNA V3.0 rather than V2.0 for part-of-speech tagging). For an overview of preprocessing steps, we refer to Github. Since Al-'Rfou reported that in the native vs non-native classification task classes were balanced with a baseline of 50%, the comments written by non-native speakers were down-sampled such that training, test and validation set are all perfectly balanced in terms of the native/non-native divide. Therefore, a baseline accuracy of 50% may be assumed for all accuracies throughout this paper. The size of the total dataset

[1] See: https://github.com/hansdeferrante/PML2

[2] Available via https://bitbucket.org/aboSamoor/wikitalk

was thereby reduced to approximately 323K comments, with a training set of 220K comments (70%), a validation set of 32K comments (10%) and a test set of 63K comments (20%).

**Similarity scores and other features**

Al-'Rfou uses similarity scores against k-gram models as features for the classification. These k-gram models are a fancy name for the empirical distribution of the different types of grams over the different classes (here, native vs. non-native). E.g., the 1-gram word model for native speakers would be the observed word-counts in training data for Native US English speakers. Mathematically, the similarity scores to these models are defined as follows:

$$\text{Sim}(C,k,l) = \frac{1}{||\text{grams}(C,k)||} \sum_{x \in \text{grams}(C,k)} \log_2(\text{count}(x,k,l))$$

where

$$\text{count}(x,k,l) = \begin{cases} \text{gram\_model}(x,k,l) & \text{if} \quad x \in \text{gram\_model}(x,k,l) \\ 1 \end{cases}$$

where $C$ denotes a comment, $k$ corresponds to the observed $k$-grams, $l$ corresponds to the language class ("native", "non-native"), a gram model is the distribution of grams over the language classes and $\text{grams}(C,k)$ is a set of all $k$-grams in a comment $C$.

Note that $\log_2(1) = 0$ such that unique grams can be discarded. In our reproduction, we encountered the problem that derivation of the complete k-gram models is computationally too expensive and intractable with 8GB RAM, even if unique grams are discarded. Hence, throughout this paper we k-gram models are limited to grams that are present at least 10 times in the entire training dataset. This means that we use

$$\tilde{\text{count}}(x,k,l) = \begin{cases} \text{gm}(x,k,l) & \text{if} \quad x \in \text{gm}(x,k,l), x \geq 10 \\ 1 & \text{otherwise} \end{cases}$$

rather than $\text{count}(x,k,l)$ as in Al-'Rfou. We expect the impact of this restriction to be limited as grams with less than 10 occurrences in the training dataset expected to occur even less often in the test and validation set (20/7 and 10/7 times, resp.)

Al-'Rfou uses some features in addition to these similarity scores. These include "the average size of words, the size of the comments and the average number of sentences" [Al-Rfou', 2012]. In a similar vein, we use the number of words, number of sentences per comment and the average word length per comment. However, some features Al-'Rfou claims using are not unambiguously defined (e.g. "relative frequency of stopwords", "size of comment"). Therefore, there are slight differences in feature sets used here and in Al-'Rfou.

**Linear SVM classifier**

Al-'Rfou reports that the linear support vector machine classifier with default parameters for the native versus non-native classification task reaches an accuracy of 74.53%. Training the linear SVM on all training data and evaluating its performance on the validation set here yields an accuracy of 72.70%. These accuracies are not quite the same, which is to be expected as we have taken a shortcut in computing the similarity scores to keep things tractable and are not able to construct exactly the same feature set. With accuracies in the same league, however, we do believe that this accuracy demonstrates Al-'Rfou's results are in principle reproducible, even if we cannot reproduce them exactly here.

## 5. Parameter tuning and classifier choice

The classifier yielding the best performance in the Al-'Rfou paper is the linear SVM classifier. It has been argued shortly after the introduction of SVM that SVM is both theoretically appropriate for text classification and practically yielding state-of-the-art performance [Gebre et al., 2013]. An influential 1998 paper by Joachims *et al.* argued SVM is most appropriate for text classification as text has few irrelevant features but a high-dimensional input space. However, in the replication of the Al-'Rfou paper classification is based not on the document-term matrix itself, but on similarity scores against k-gram models. This means that the input space is no longer extremely high-dimensional. Therefore, it does not automatically follow SVM is most appropriate, and comparing SVM to other classifiers such as the Random Forest classifier is of interest.

The SVM classifier used in Al-'Rfou has a linear kernel and uses default parameters (i.e. the soft-margin parameter $C$ set to 1). A first idea to improve accuracy of classification is to tune $C$. Use of a linear kernel for the SVM classifier implicitly assumes that the classes are linearly separable (see e.g. [Hastie et al., 2001, p. 422]). Joachims et al. mention that for text classification problems the linear kernel is typically appropriate as texts are often linearly separable [Joachims, 1998]. Since it does not automatically follow linear separability of the DTM implies linear separability in terms of the similarity scores, we will investigate performance with a non-linear kernel as well.

### Parameter tuning linear SVM

The linear SVM was constructed with different values for parameter $C$. The considered values ranged from $2^{-5}$ to $2^{15}$. Accuracy assessed on the validation set showed that tuning $C$ had little effect on accuracy: accuracies for all considered values of $C$ were in the range of 72.67% to 72.77%. This suggests tuning of the SVM with a linear kernel is a fruitless exercise.

### Parameter tuning with RBF kernel and bagging

Fitting the SVM classifier with an RBF kernel has complexity $O(n_{features} * n^2_{observations})$, meaning the training time scales quadratically in the number of observations. This leads to very long training times for the data at hand, as we have >200K training comments. To reduce this training time, we use bagging, i.e. training SVM classifiers on subsets of the training data. Classification is then based on a majority vote of all classifiers. This idea, originally proposed by Leo Breiman in 1996 , is usually used to reduce variance of the classifier. However, here we motivate its use by the resulting reduction in complexity; training 20 estimators on 1/20th of the training comments reduces complexity per SVM classifier by a factor $(1/20)^2$, leading to a total reduction in complexity of a factor 20 compared to a single RBF kernel SVM classifier.

In tuning the RBF kernel parameters we follow the widely cited "Practical Guide to Support Vector Classification" by Hsu et al. [2008]. They propose an optimal $(C, \gamma)$ may be selected using grid search. The paper suggests for $C$ the range used too for tuning the linear kernel and the uneven powers ranging from $2^{-15}$ and $2^1$ for $\gamma$. Results of this grid search are shown in Figure 1. The reportedly optimal $(C, \gamma)$ are $(C*, \gamma*) = (8, 2^{-11})$. With these parameter settings and bagging, accuracy jumps to 74.62%. Bagging with a linear SVM kernel does not show such an increase, indicating that it is indeed the non-linear kernel that accounts for the increase (see Github).

### Random Forests

Although Random Forests are not typically used in text-classification settings, the feature set used in the Al-'Rfou paper is not extremely sparse/high-dimensional, such that two important objections against using this classifier no longer apply. Random Forests, introduced in 2001 by Leo Breiman , are claimed to be able to deal with moderate to a substantial number of features, to have a tendency to converge quickly and to not overfit. In addition to these attractive features, they are computationally much more efficient than SVM, which is our main motivation to use them here.

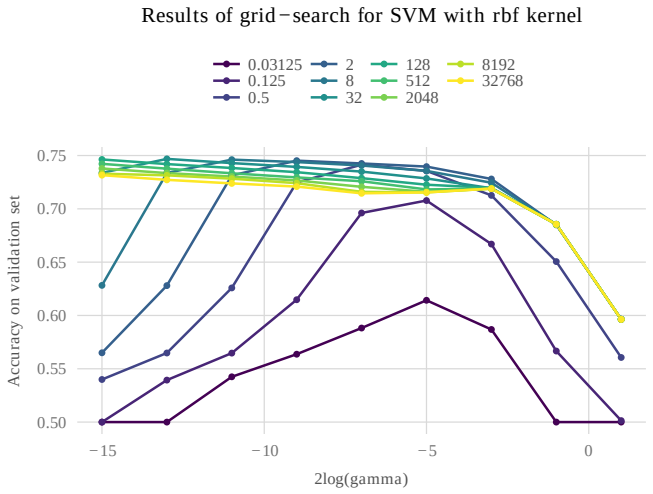


**Figure 1:** *Grid-search for the parameters of the RBF kernel of SVM. The grid was constructed following advise by Hsu et al.*

An implementation with somewhat arbitrary parameters for the Random Forest classifier reaches an accuracy of 73.05%. Since it is less clear how to tune Random Forests in a text classification setting, we wanted to try out a broad range of parameters for tuning. Unfortunately, a grid-search based over all possible parameters for the random forest classifier would quickly become intractable. In a 2012 paper, Bergstra and Bengio formally demonstrate that in situations where it is unknown which hyperparameter settings are important, random search performs better than grid search for a given computational budget . Hence, we have chosen to tune the random forests using random search.
Selecting parameters based on a randomized search with cross-validation (implemented with `sklearn`'s `RandomSearchCV`) shows that the optimal parameters increase the accuracy only marginally to 73.12%. Based on these results, we can conclude that the Random Forests classifier works well, but fails to yield a performance competitive to the tuned RBF kernel (even though it performs better than the Linear SVC classifier!).

## 6. Classification based on DTM

Use of similarity scores as features is attractive in that it reduces the dimensionality of the document-term matrix per type of gram to a single dimension. However, classification in the L1-classification task is usually directly based on the DTM. In this section, classification based on the DTM itself will be assessed. To keep computations tractable, filtering of grams included in the DTM is necessary. Ideally,

such a filtering yields grams that are both prevalent across comments and distinctive of native/non-native text. Here, a particular filtering will be discussed and classification performance under this filtering will be assessed.

### Filtering of grams in DTM

Odds-ratios are a statistic used to assess how strong the outcome of a binary property A is associated with another binary property B. Here, we employ these odds-ratios to select grams distinctive of native/non-native text by how strong the presence (i.e. count) of a particular gram in the non-native language (property A) is associated with the non-native language vis-à-vis the English language (property B). Using the notation presented in Table 1, this ratio can be calculated as $\frac{(a/b)}{(c/d)} = \frac{ad}{bc}$. If the odds ratio for a particular gram is larger than 1, it is associated with being in the language X. If it is smaller than 1, it is associated with not being in the language.

**Table 1:** *Table illustrating how odds-ratios are calculated. a through d represent counts of the grams. The odds-ratio is then defined as* $\frac{(a/b)}{(c/d)} = \frac{ad}{bc}$ *and can be seen as a measure of association of a gram with the language of interest.*

| | Gram | All other grams |
|---|---|---|
| In language X | a | b |
| In English | c | d |

To obtain grams distinctive of native/non-native text, one should thus select grams with odds ratios different from 1. To limit included grams to prevalent grams, a lower threshold can be specified for $a$. With this in mind, the following filtering was applied to obtain grams that we consider for the document-term matrix:

- Construct for all gram-types and $k = 1, 2, 3, 4$ k-gram models for each of the twenty languages contained in the dataset.
- For the nineteen non-English languages, compute the odds-ratios against the English language.
- Obtain the unique list of grams with odds-ratios >6/5 and <5/6 and $a \geq 5$.

This filtering scheme led to a unique list of 130,249 word-grams, 80,360 character grams, and 60,487 structure-grams. These lists - derived exclusively from the training data - were subsequently used to construct three different sparse document-term matrices for the training data, i.e. DTMs for character grams, word grams and structure grams, respectively. The DTMs were too constructed for validation data. Note that each of these DTMs features 1- up to 4-grams.

### Classification accuracy with linear SVM

As discussed in Section 5, linear SVM is considered an excellent classifier for classification based on the document-term matrix. Therefore, attention is restricted to the SVM classifier with a linear kernel within this section. In particular, `sklearn`'s `SGDClassifier` is used which employs stochastic gradient descent to introduce sparseness into a the linear SVM model.
For each of the DTM types, a classifier was trained on the DTM corresponding to the training data. Subsequently, labels for the DTM corresponding to the validation data were predicted and performance was assessed in terms of accuracy. Table 3 shows these accuracies for the different types of DTMs.

**Table 2:** *Table reporting accuracies the Stochastic Gradient Descent (linear SVM) classifier (`SGDClassifier`) for the validation set.*

| Type | Words | Characters | Structure |
|---|---|---|---|
| Total # grams | 130,249 | 80,360 | 60,487 |
| SVM accuracy | 71.99% | 71.56% | 60.40% |

Note that the accuracies of classifying based on word-grams and character-grams in particular show these classifiers are competitive with the classifier of Al-'Rfou that is trained on all similarity scores (their accuracy is just $\pm 1$ % lower).

## 7. Classification based on 10-char grams

As mentioned in the introduction, Kulmizev et al. reported in a 2017 paper that an SVM classifier trained on character 9/10-gram DTMs outperforms classifiers taking other information into account. However, a concern the authors carry is that the classifier might be overfitting on the writing prompts of the essay and would thereby fail to generalize towards essays . They substantiate this concern by showing that omission of a single of the essay writing prompts shows that the classifiers performance drops 3-20%, indicating that it indeed classifies (partially) on topics. The Al-'Rfou dataset, with proper nouns censored using SENNA, can be expected not to suffer from much topic bias and may be a better way to assess the power of k-grams in native language identification.

**Filtering**

To assess this, we embarked on construction of a DTM matrix for character 10-grams. However, a more aggressive type of filtering had to be applied in comparison to the filtering applied in Section 6 to keep things computationally feasible. In particular:

- A 10-gram character model was constructed on 85% of the training data. k-grams were included in a gram-list if they (i) had asymptotic lower bounds exceeding $\frac{11}{10}$ or asymptotic upper bounds of $\frac{10}{11}$ at $\alpha = 0.01$, (ii) occured in language "X" at least $100/k$ times, and (iii) were not "superfluous". Loosely said, $k$-grams that have a near one-to-one relationship with $k-1$-grams are removed. E.g. "ncyclop" will be deleted as it uniquely leads to "ncyclo" (from "encyclopedia"). For a proper definition of "superfluous" and the asymptotic lower bounds, we refer to the Jupyter notebook.
- The filtered 10-char gram-list is used to construct a DTM matrix for the 15% held-out training data. A Random Forest is trained on this held-out data and the features with importance of at least $10^{-6}$ are selected.

This first filtering step reduces the approximately $\pm 20$ million grams contained in the language distribution to a list of about 300K grams. The second step reduces this further to a list 41,790 grams.

**Classification performance with Linear SVM**

The 40K grams obtained after filtering were used to construct DTM matrices for 60% training set and validation set (full training set was intractable on 8GB RAM computer). Classification performance was assessed for the Multinomial Naive Bayes and SGDClassifier (see also Section 6 by training on the DTM for 60% of the training data and predicting the labels for validation data.

**Table 3:** *Table reporting accuracies of the multinomial Naive Bayes classifier (`MultinomialNB`) and the linear SVM classifier (`SGDClassifier`) for the validation set. Classifiers were trained on DTM constructed for 10-char grams.*

| Type | Characters |
|---|---|
| # grams | 41,790 |
| NB Accuracy | 64,72% |
| SVM accuracy | 68.17% |

Note that classification based on the character 10-gram DTM actually performs worse than based on the 4-gram DTM (see section 6) on the validation set. We should mention that it is not entirely fair to compare these accuracies as (i) a larger set of grams was considered for the 4-gram DTM, (ii) grams were selected in a different way, and (iii) more training documents were used for the 4-gram DTM. This implies that we should be wary of making strong statements about the value of higher order grams. However, we do think that this result is more supportive of the idea of higher order character grams overfitting on topic bias rather than them picking up some latent information in non-native/native text.

## 8. Classification by majority vote

So far, different classifiers have been constructed and tuned. By tuning the SVM parameters, accuracy of the SVM classifier based on similarity scores could be improved from 72.7% to 74.6% on the validation set. Features for this classifier were similarity scores against 24 k-gram models. In Section 6, a different approach was taken, i.e. classify based on the document-term matrices directly rather than similarity scores. The terms for these DTMs were selected for prevalence among non-native comments and for being distinctive of non-native/native comments. SVM classification based on the word- and character-DTMs alone were competitive with the classifier of Al-'Rfou, at 71.99% and 71.56% accuracy, respectively. Classification based on structure had lower accuracy at 60.40%.

Since the classifiers use different types of information for classification, it would be interesting to see how they would perform if we combine them. To keep things simple, we assessed the accuracy of the (unweighted) majority vote by two different groups of classifiers. Table 4 shows that a majority vote for both groups indeed increases classification performance, which indicates these classifiers indeed pick up different types of information.

**Table 4:** *Table reporting accuracies of individual classifiers and a simple unweighted majority vote by combinations of 3 classifiers. Accuracy is evaluated on validation set.*

| Classifier | Accuracy |
|---|---|
| Sim. RBF | 74.62 |
| Char DTM | 71.56% |
| Word DTM | 71.99% |
| Struc DTM | 60.40% |
| Sim+Char+Word | **75.56%** |
| Char+Word+Struc | 73.23% |

## 9. Hypothesis testing

All classifiers thus far were trained on the training set and evaluated on the validation set. All deliberations above should thus be considered hypotheses,

not conclusions. Here, we will evaluate the performance of classifiers on the independent test set to see if our observations generalize beyond the validation set. The most interesting findings in the above are that parameter tuning of the SVM kernel increases performance of the classifier and that a simple majority vote of the three different classifiers outperforms classification by the separate classifiers in terms of the accuracy. Let us pose these as our hypotheses and evaluate them here.

**Hypothesis 1: Use of the RBF kernel for the SVM with tuned parameters increases classification performance.**

To evaluate this hypothesis, we assess the accuracy of both the linear SVM classifier with default parameters and the SVM classifier with the RBF kernel parameters determined by grid-search in Section 5. As features for the test set, similarity scores are computed against the k-gram models derived from training data. The test accuracy for the linear SVM classifier is 71.94%. This accuracy improves to 73.88% if we use the RBF kernel. Although these accuracies are both somewhat lower than the accuracy for the validation set, we still see an increase in accuracy of approximately 2.0%, supporting the hypothesis that the RBF kernel works better for similarity scores than an untuned linear SVM kernel.

**Hypothesis 2: The simple majority vote of the "Sim+Char+Word" classifier outperforms performance of individual classifiers.**

Here, we first construct the DTMs for test observations based on the list of grams extracted from training data. We evaluate accuracy for each of the classifiers and for the joint classifier. An overview of the test set accuracies is given in Table 5. These accuracies again support our hypotheses that the weighted majority vote of the classifiers outperforms the separate classifiers.

**Table 5:** *Table reporting accuracies of individual classifiers and a simple unweighted majority vote by combinations of 3 classifiers. Accuracy is evaluated on test set.*

| Classifier | Test Accuracy |
|---|---|
| Sim. RBF | 73.88% |
| Char DTM | 71.82% |
| Word DTM | 72.24% |
| Struc DTM | 60.45% |
| Sim+Char+Word | **75.53%** |
| Char+Word+Struc | 73.13%% |

## 10. Conclusions

Within this paper, we set out to reproduce the paper of Al-'Rfou, and increase native versus non-native classification performance by (i) parameter tuning and (ii) incorporating information contained in the Document-Term Matrix directly, following ideas found within the closely related task L1-classification. Apart from using information from the DTM directly, these ideas included (a) including higher order k-grams and (b) constructing a classifier separately on the features.
Unfortunately, a thorough investigation of (a) and (b) was found to be too computationally expensive to run on a standard 8GB RAM computer. With regards to (a), no better performance was found for higher-order character grams, but aggressive filtering had to be applied to make inclusion of such grams for classification feasible. With regards to (b), construction of a joint DTM for character grams, word grams, and structure grams was computationally infeasible. Hence, although the accuracy of a simple ensemble of classifiers could be assessed (see e.g. Table 5, the accuracy of a classifier trained on all DTMs could not be assessed.
The work succeeds, however, in demonstrating that (i) the performance can be by approximately 2% increased by using a non-linear RBF kernel tuned using grid-search, and (ii) including features from the DTM can increase performance by an additional 2%; this supports the idea that dimensional reduction of similarity scores results in loss of information.

## 11. Discussion and future work

Throughout this paper, we have worked with a single training set, single validation set and single test set. Parameters were tuned based on the training/validation set and cross-validation was not pursued. A motivation not to pursue cross-validation is that an extremely large dataset with >200K observations was available, such that (i) we hypothesized estimated accuracy would be reasonably robust and (ii) cross-validation becomes computationally very expensive. The accuracies of the different classifiers on the validation set and test set are quite close (see Table 4 and 5) and thereby support (i). In future work, one could focus on how robust these results are w.r.t. the training/validation/test set split.
With regards to the results, we succeed in increasing accuracy in classifying whether a comment was written by a native or non-native speaker. However, one could argue that the increase of approximately 3-4% is only marginal. We think this marginal increase is

the result of the classification task being complicated by the fact that the non-native comments are highly heterogeneous. For example, comments are of different lengths, writers of non-native comments have a different mother tongues, and English proficiency differs per user. Future work could focus explicitly on what differentiates non-native comments that are hard-to-classify, for example by boosting or taking this into account explicitly in training.

To establish that this heterogeneity truly poses a problem, a logistic regression was performed of the false positives on the nationality of the user who wrote the comment, the self-reported level of English and the length of the comment. This regression show that this heterogeneity indeed has profound effects on the classification performance (see Appendix A). A few findings from this regression:

- Users who report a higher level of proficiency in English are harder to classify. Those who do not report a proficiency level are harder to classify than people who report a level of 4 (on a five-point scale).
- Comments in the first quantile of word-lengths (i.e. comments consisting of 7-32 words) are hardest to classify. The longer the comment, the easier it is to classify correctly.
- The coefficient estimates indicate comments written by native Chinese and native Japanese speakers are harder to classify than some European languages.

Especially the last finding is surprising in light of English belonging to the Pan-European language family (Chinese and Japanese obviously do not belong to this language family). An explanation for this finding may be that comments written by a native Chinese and native Japanese speakers each represent just 1% of the comments by non-native speakers. This may make it difficult to pick up signals that differentiate comments written by native Chinese and native Japanese speakers from comments written by native speakers.. We explicitly tried to take into account the heterogeneity in native tongues by selecting grams for the DTM based on the odds ratios for each language versus English. However, the classifiers we have trained assume classes are binary and may have difficulties picking up such if they are only distinctive of 1% of the observations.

We think future work could and should focus on the imbalance between different non-native languages in the classification to further increase performance. One way to do this could be multi-label classification. Another interesting matter would be to approach this problem as a $L1$-classification task, respecting that how people use the English language crucially depends on their first language.

## References


R. Al-Rfou'. Detecting English Writing Styles For Non Native Speakers. 2012. URL https://arxiv.org/pdf/1211.0498.pdf.

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, Feb. 2012. ISSN 1532-4435. URL http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf.

L. Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, 1996. ISSN 08856125. doi: 10.1023/A:1018054314350. URL http://link.springer.com/10.1023/A:1018054314350.

L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 08856125. doi: 10.1023/A:1010933404324. URL http://link.springer.com/10.1023/A:1010933404324.

R. Collbert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011. ISSN 1532-4435. doi: 10.1145/2347736.2347755. URL http://dl.acm.org/citation.cfm?id=2078186.

B. G. Gebre, M. Zampieri, P. Wittenburg, and T. Heskes. Improving Native Language Identification with TF-IDF Weighting. In *Proceedings of the 8th NAACL Workshop on Innovative Use of NLP for Building Educational Applications*, pages 216–223, 2013. doi: 10.1.1/jpb001.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A Practical Guide to Support Vector Classification. *BJU international*, 101(1):1396–400, 2008. ISSN 1464-410X. doi: 10.1177/02632760022050997. URL http://www.csie.ntu.edu.tw/{~}cjlin/papers/guide/guide.pdf.

S. Jarvis, Y. Bestgen, and S. Pepper. Maximizing classification accuracy in native language identification. *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, (1):111–118, 2013.

T. Joachims. 1 Introduction 2 Text Categorization 3 Support Vector Machines. *Machine Learning*, 1398(LS-8 Report 23):137–142, 1998. ISSN 03436993. doi: 10.1007/BFb0026683. URL http://www.springerlink.com/index/drhq581108850171.pdf.

A. Kulmizev, B. Blankers, J. Bjerva, M. Nissim, G. van Noord, B. Plank, and M. Wieling. *The Power of Character N-grams in Native Language Identification*, pages 382–389. Association for Computational Linguistics (ACL), 2017. ISBN 978-1-945626-85-2. doi: 10.18653/v1/W17-5043.

J. Tetreault, D. Blanchard, A. Cahill, and M. Chodorow. Native Tongues, Lost and Found: Resources and Empirical Evaluations in Native Language Identification. In *Proceedings of COLING2012*, volume 2, pages 2585–2602, 2012. URL www.aclweb.org/anthology/C12-1158.

O. Tsur and A. Rappoport. Using Classifier Features for Studying the Effect of Native Language on the Choice of Written Second Language Words. *Computational Linguistics*, (June):9–16, 2007. doi: 10.3115/1629795.1629797. URL http://eprints.pascal-network.org/archive/00003485/.

## A. Logistic Regression (False-Positives) on factors

To investigate how heterogeneity of the non-native group affects classification performance, we did a logistic regression of all False Positives on native languages, word length quantiles and English levels using `glm.fit` in R. The summary of this regression is shown below.

```
Call:
glm(formula = FP ~ level_english + native_lang + numwordquantile,
    family = binomial, data = df.joined)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.35329  -0.66822  -0.00006  -0.00005   2.36982

Coefficients: (1 not defined because of singularities)
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)       -1.50075    0.23602  -6.358 2.04e-10 ***
level_english2     0.36321    0.22276   1.631 0.102989
level_english3     0.37325    0.21834   1.709 0.087365 .
level_english4     0.67029    0.21856   3.067 0.002163 **
level_english5     1.10851    0.22006   5.037 4.72e-07 ***
level_englishN   -18.80111   99.56203  -0.189 0.850220
level_englishUNK   0.79552    0.22413   3.549 0.000386 ***
native_langDA      0.13030    0.10951   1.190 0.234108
native_langDE      0.06953    0.09497   0.732 0.464058
native_langEN           NA         NA      NA       NA
native_langES      0.13264    0.09676   1.371 0.170413
native_langFI      0.25191    0.11796   2.136 0.032710 *
native_langFR      0.22772    0.09920   2.296 0.021701 *
native_langHU     -0.36552    0.14185  -2.577 0.009972 **
native_langIT      0.06422    0.12236   0.525 0.599666
native_langJA      0.48422    0.16533   2.929 0.003403 **
native_langKO      0.14205    0.27862   0.510 0.610176
native_langNL     -0.02471    0.09780  -0.253 0.800557
native_langNO      0.31961    0.11225   2.847 0.004408 **
native_langPL     -0.57787    0.11249  -5.137 2.79e-07 ***
native_langPT      0.03496    0.11497   0.304 0.761073
native_langRU     -0.33559    0.10145  -3.308 0.000940 ***
native_langSV      0.15896    0.10325   1.539 0.123683
native_langTR     -0.52399    0.16741  -3.130 0.001748 **
native_langYUE     0.79670    0.14264   5.586 2.33e-08 ***
native_langZH      0.50290    0.11330   4.439 9.06e-06 ***
numwordquantileQ2 -0.14720    0.03637  -4.047 5.19e-05 ***
numwordquantileQ3 -0.32776    0.03665  -8.943  < 2e-16 ***
numwordquantileQ4 -0.66718    0.03877 -17.210  < 2e-16 ***
```