The Power of Character N-grams in Native Language Identification

Artur Kulmizev, Bo Blankers, Johannes Bjerva, Malvina Nissim,
Gertjan van Noord, Barbara Plank, Martijn Wieling
University of Groningen
Oude Kijk in 't Jatstraat 26
9712 EK Groningen
{a.kulmizev,b.blankers}@student.rug.nl
{j.bjerva,m.nissim,g.j.m.van.noord,b.plank,m.b.wieling}@rug.nl

Abstract

In this paper, we explore the performance of a linear SVM trained on language-independent character features for the NLI Shared Task 2017. Our basic system (GRONINGEN) achieves the best performance (87.56 F1-score) on the evaluation set using only 1-9 character n-grams as features. We compare this against several ensemble and meta-classifiers in order to examine how the linear system fares when combined with other, especially non-linear classifiers. Special emphasis is placed on the topic bias that exists by virtue of the assessment essay prompt distribution.

1 Introduction

Native Language Identification (NLI) is the task of identifying a writer's native language (L1) based on their writings in another language. Typically, low-to-medium proficiency writers exhibit a tendency to "borrow" linguistic constructions from their native language and apply them to the language in which they are communicating. A native Russian speaker, for example, may forego the use of articles such as "the" when writing in English. This phenomenon, widely referred to as Language Transfer, allows for a common set of linguistic features to emerge between speakers of the same native language (Odlin, 1989). NLI is thus concerned with applying machine learning approaches using these features in order to automatically identify the L1 of writers communicating in another language.

There are many practical applications for NLI. Second language (L2) education is a field in which NLI can offer much potential aid. For instance, in identifying the native language of a learner by their writing, it is possible to isolate the linguistic features they employ when communicating. This could subsequently be integrated in language-specific error-correction systems, in which a user receives L1-based suggestions to correct their L2 writing. At a large scale, this could be extended to enhance existing teaching pedagogies and tailor them towards students of a particular L1. NLI is another natural fit for forensic linguistics, where it can be used to detect the native language (and potentially the nationality) of an anonymous writer.

NLI is typically framed as a multi-class classification problem, wherein a classifier is trained on more than two native languages simultaneously. As with many text-classification tasks, Support Vector Machines (SVM) have consistently produced the best results for the task, e.g., (Brooke and Hirst, 2012). However, other classifiers, such as Random Forests and Logistic Regression, have also been explored (Tetreault et al., 2013). Ensemble systems, which combine the predictions of several classifiers and output the most likely class label via voting or probability-averaging, have further been shown to provide a boost in accuracy compared to the single-classifier approach. Such systems, however, are not light-weight. In training several classifiers simultaneously, quick training speeds are typically sacrificed in favor of a (usually marginal) performance gain. This paper is thus concerned with exploring each of these classification methods as they pertain to the NLI Shared Task 2017 (Malmasi et al., 2017).

2 Related Work

In 2005, Koppel et al. (2005) begun exploring methods for NLI by exploring the International Corpus of Learner English (Granger et al., 2009). In this work, they evaluated the effect of several features, including function words, letter n-grams, part-of-speech bigrams and error types.

Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications, pages 382–389
Copenhagen, Denmark, September 8, 2017.

Training an SVM on the combination of these yielded an accuracy score of 80.0%. Several years later, a shared task in NLI was organized by Tetreault et al. (2013). A total of 29 teams participated in this competition, with the winning system implementing a combination of lexeme, lemma, and POS-tagged 1-3grams for their model (Jarvis et al., 2013). This system produced an accuracy of 83.6% discriminating between 11 different native languages. Ionescu et al. (2014) later improved on this result by applying Kernel Ridge Regression and Kernel Discriminant Analysis in order to extract character n-gram features from the NLI Shared Task 2013 data. This approached yielded an 85.3% accuracy score on the 2013 shared task's test set.

In the years between the initial NLI shared task and the current one, teams have continued to produce new state-of-the-art systems. Most recently, Malmasi and Dras (2017), presented an exhaustive survey of potentially relevant features for NLI. These included character, word, lemma, and POS n-grams, function words, context-free grammar production rules, and dependency tags, among others. Separate SVMs were trained on each of these features and their outputs were fed into a mean probability ensemble. A meta Linear Discriminant Analysis (LDA) classifier was then trained on the probability distributions generated by the ensemble, yielding an accuracy of 87.1%.

3 Methodology and Data

3.1 Data

The provided data set consists of 13,200 English-language essays submitted for a standardized assessment of English proficiency for academic purposes. The essays are equally divided into 11 native languages (L1s), totalling 1,200 essays per language. The languages represented therein are as follows: Arabic (ARA), Chinese (CHI), French (FRE), German (GER), Italian (ITA), Hindi (HIN), Japanese (JPN), Korean (KOR), Telugu (TEL), and Turkish (TUR). The full data set is divided into three parts, with 11,000, 1,100, and 1,100 essays constituting the training, development, and test set, respectively. The number of words per essay varies between 300 and 400.

In participating in the assessment, all participants were instructed to write their essay about a specific prompt topic. The data set is thus divided over 8 different prompts as well as L1s. While the L1s are evenly distributed over the data, the distribution of the prompts is skewed by both language and overall, as shown in Table 1. The table represents the distribution of the prompts for the training and the development set combined, since this constitutes all of the data that our final system is trained on. Noteworthy is the fact that Hindi and Telugu have similar distributions over all the prompts, which is a different distribution than the the other languages. It is also interesting that only a small portion (12 essays) are written about P1 for Italian, which could cause a discrepancy in the later classification of the language.

Due to these factors, we pay particular attention to the prompt distribution during our analysis.

3.2 Features

Our main classifier is a Linear Support Vector Machine, which has been shown to perform well in prior NLI tasks. We performed a grid search over the C, loss, and penalty parameters of the Linear SVM in order to obtain the best-performing variant. However, tuning these parameters failed to produce any noteworthy results and we thus opted for a non-parametric SVM. We also evaluated the performance of the classifier with an RBF kernel in order to examine whether a non-linear approach would generalize better over the data. Ultimately, this resulted in significantly longer training times as well as much lower performance accuracy and was discarded in favor of the linear alternative. The non-parametric classifier is based on several combinations or stand-alone models of the features described below.

3.2.1 Character n-grams

Large ranges of character n-grams contain characteristic information about the writing style of an author. Compared to word n-grams, which only capture the identity of a word and its possible neighbors, character n-grams are additionally capable of detecting the morphological makeup of a word. In a task such as NLI, where many words are likely to be misspelled, character n-grams are especially powerful at detecting patterns in such misspellings, and substantially less sparse than word n-grams. In this paper, we experimented with several ranges of character n-grams. Even though prior research in NLI has largely focused on character n-grams of up to 5 characters, this range did not perform well in this task. Instead, increasing the upper bound of the range to between

Table 1: Overview of prompt distribution per language for the data set

	ARA		CHI		FRE		GER		HIN		ITA		JPN		KOR		SPA		TEL		TUR	
	#	%	#	%	#	%	#	%	#	%	#	%	#	%	#	%	#	%	#	%	#	%
P0	139	12.6%	140	12.7%	156	14.2%	151	13.7%	86	7.8%	187	17.0%	138	12.5%	128	11.6%	159	14.5%	55	5.0%	170	15.5%
P1	133	12.1%	141	12.8%	68	6.2%	28	2.5%	53	4.8%	12	1.1%	142	12.9%	142	12.9%	157	14.3%	41	3.7%	43	3.9%
P2	141	12.8%	139	12.6%	160	14.5%	153	13.9%	161	14.6%	141	12.8%	143	13.0%	143	13.0%	162	14.7%	171	15.5%	169	15.4%
P3	138	12.5%	139	12.6%	151	13.7%	152	13.8%	158	14.4%	173	15.7%	141	12.8%	141	12.8%	160	14.5%	166	15.1%	167	15.2%
P4	138	12.5%	140	12.7%	158	14.4%	155	14.1%	161	14.6%	173	15.7%	116	10.5%	140	12.7%	141	12.8%	165	15.0%	169	15.4%
P5	136	12.4%	134	12.2%	160	14.5%	150	13.6%	156	14.2%	187	17.0%	138	12.5%	137	12.5%	134	12.2%	169	15.4%	147	13.4%
P6	138	12.5%	126	11.5%	87	7.9%	157	14.3%	163	14.8%	138	12.5%	140	12.7%	136	12.4%	54	4.9%	167	15.2%	90	8.2%
P7	137	12.5%	141	12.8%	160	14.5%	154	14.0%	162	14.7%	89	8.1%	142	12.9%	133	12.1%	133	12.1%	166	15.1%	145	13.2%
Total:	1100		1100		1100		1100		1100		1100		1100		1100		1100		1100		1100	

8 and 10 characters yielded very encouraging results. In training a classifier on character n-grams ranging from 1 to 10, this effectively models the effect of word unigrams, bigrams, and, in cases of very short words, trigrams. Indeed, the most informative features in these cases are 7-9 character n-grams in which the author discusses his or her own country (Japanese people talk about Japan, etc.). When combined with the morphological insight of the lower-range character n-grams, this approach proved to be simultaneously very powerful and simple. Though the performance difference between ranges of 1-8, 1-9, 1-10, 2-8, etc. is marginal, we choose the 1-9 range for our system as it provided the best results for Hindi and Telugu on the dev set - the languages which were most often confused in our case. It is important to note that we also employ a binary-counting approach, where a feature is either present (if it appears at least once in a document) or absent (if it did not appear). These counts are normalized via term-frequency, inverse-document frequency (tf-idf) as implemented by the scikit-learn[1] machine learning package for Python.

3.2.2 Part-of-speech tags

Part-of-speech tags provide information about how ESL learners approach English at a morpho-syntactic level. Intuitively, then, it is likely that the native distribution and usage of POS tags might affect a learner's production of English. For example, native speakers of Turkish are known for the difficulty they experience with appropriately inserting definite articles in English. This difficulty could thus surface in terms of observed POS sequences or distributions in Turkish-native production of English texts. Furthermore, this could be notably different from what is observed in the production of speakers whose native language features a usage of determiners that is more similar to English.

[1] http://scikit-learn.org/stable/

Although countless POS-taggers exist, one major problem in acquiring reliable tags is intrinsic to the non-native nature of the texts we deal with. As POS models for English are trained on native English data, it is not granted that they will perform as well on non-native writing as they do on canonical texts. For this reason, we trained a POS tagger (Plank et al., 2016) on POS-tagged English learner data obtained from the Universal Dependencies project (Nivre et al., 2016), i.e., `English-ESL` (78k training tokens), tuning its parameters on the corresponding UD dev data. We experimented with several combinations of POS n-grams on our data, and found that a range of 1–4 yielded the best results on the development set.

3.2.3 Prompt word extraction

In an attempt to remove topic bias, we extracted words that appear to be typical of each prompt topic. This was done with the intuition that such words would ultimately confuse the classifier towards modeling prompt instead of native language. As such, we concatenated every essay per topic (resulting in 8 large prompt documents) as well as every essay in total. Each of the prompt documents was then passed to a sparse additive generative model (SAGE) (Eisenstein et al., 2011), with the concatenated corpus acting as a comparison corpus. This allowed us to identify the top keywords per topic (P0: "advertiser"; P1: "tourist", etc.). We then combined these prompt lists into a single keyword list (ranked by SAGE-score), which we considered to be representative of the total number of prompt words written in the corpus. In training our classifier, we then replaced the top 100 of such words (if correctly spelled) in the individual essays with a dummy token (*). We retained misspelled prompt words, as our intuition is that these represent (incorrect) information about the writers' understanding of English morphology. Using this approach resulted in a slight drop in performance accuracy (∼2%) for every

model it was tested with, thereby confirming our intuition that our systems were indeed modeling topics, too.

3.2.4 P7 Omission

In examining the prompt distribution for the test data, we noticed that Prompt 7 was excluded entirely from the provided essays. As such, we hypothesized that omitting P7 from the training data would remove a degree of confusion introduced by the P7 prompt lexicon. To check this assumption, we first removed all essays with P7 as a prompt from the dev data (143 documents). This resulted in a slight drop in F1-score (from 84% to 83%) on the dev set. We then repeated this experiment with the P7 essays also omitted from the training data (1,419 documents). Doing so reproduced the initial f1-score of 84%, albeit with different (less) training and dev data. We replicated this for the test submission, removing P7 from the concatenated train and dev data (1,562 documents) in order to balance our system in terms of prompt with respect to the actual test set.

3.3 Meta-classification and Ensembles

Meta-classification is the process of training a classifier on the probability distributions output by another classifier. Doing so has the effect of revealing the classification patterns of the latter classifier, including cases where it experienced the most confusion in assigning a label. In seeing enough of these patterns, a meta classifier can effectively learn from the label probability distributions and correct the decisions of the main classifier. We experimented with an SVM meta classifier trained on both the output of the character n-gram classifier, as well as the output of a combined character n-gram and a simple neural network (CBOW, see Section 3.3.1). We performed 5-fold cross-validation on the training set in order to obtain the label probabilities for the documents in the training data. Though this approach improves upon the performance of both classifiers we evaluated, it is important to note that it may lead to over-fitting (Thornton et al., 2013).

We separately trained an ensemble and meta-classifier on the probability distributions output by several systems as features. The goal in doing so was to examine how the aforementioned linear SVM fares when combined with other, non-linear classifiers. The classifier employed is an ensemble linear SVM, which is trained on the predicted probability distributions of a randomly chosen 60% of the dev set and tuned on the remaining 40%. We use the standard hyperparameters of the SVM implementation in scikit-learn, without any tuning. The maximum of the predicted probability distribution on the test set is then used as the system's label prediction.

We evaluated the performance of three ensembles. The first two ensembles included the character 1-9-gram system, the CBOW system, and a CNN system (see Section 3.3.2). The CNN systems in the two runs differed in the input representations used. In first case (CNN1), the CNN used word unigrams and character 4-5grams, whereas in the second case (CNN2), it used word unigrams and character 6-grams. The third ensemble (Submission #8) concatenated the probability distributions generated by both character 1-9-gram and CBOW models (i.e. `ARA_CBOW: 0.1234; ARA_CHAR: 0.1432; etc.`) and trained a meta-classifier on these probabilities.

Each of the systems included in the ensembles (excluding the character 1-9-gram SVM) as well as the meta-classifier are described below.

3.3.1 CBOW system

We incorporated a simple neural baseline that combines word embeddings with a feedforward neural architecture similar to the *continous bag-of-words* (CBOW) model introduced in (Mikolov et al., 2013). This system represents each document as the average embedding of all words in the document. We used a shallow model (no deep layers), with a single dropout layer followed by the softmax output layer. The parameters of this model were set based on the dev set: 512 input dimensions, 0.1 dropout, 20 epochs, trained with the `adam` optimization algorithm (Kingma and Ba, 2014) for 20 iterations with a batch size of 50.

3.3.2 Deep Residual Networks

Deep residual networks (resnets) are a class of convolutional neural networks (CNNs), which consist of several convolutional blocks with skip connections in between (He et al., 2016). Such skip connections facilitate error propagation to earlier layers in the network, which allows for building deeper networks. Resnets have been shown to be useful for NLP tasks, such as text classification (Conneau et al., 2016), and sequence labelling (Bjerva et al., 2016). We applied resnets with four residual blocks in our en-

semble experiments, each containing two successive one-dimensional convolutions. Each such block is followed by an average pooling layer and dropout ($p = 0.5$, Srivastava et al. (2014)). The resnets were applied to several input representations: word unigrams, and character 4-6-grams. The outputs of each resnet are concatenated before passing through two fully connected layers. We trained the resnet over 50 epochs with `adam`, using the model with the lowest validation loss. In addition to dropout, we used weight decay for regularization ($\epsilon = 10^{-4}$).

3.4 Discarded features

All previous features and systems have been used for the final submissions and will be discussed in the results section of this paper. However, it is noteworthy to mention which features were evaluated but nonetheless failed to provide a performance improvement. These were tested on the development set of the data as standalone features as well as in combination with others. However, in none of the cases were these features able to improve the performance of any of the submitted systems.

3.4.1 Word, lemma, and POS n-grams

Given the relative success of prior work in NLI, such as (Jarvis et al., 2013), we decided to experiment with traditional n-gram features. In these experiments, we employed the spaCy[2] NLP toolkit in order to generate the lemma and POS representations of words tokens. Several combinations of these features were evaluated, such as `WORD + LEMMA + POS` bigrams/trigrams, `WORD/LEMMA` unigrams + `POS` bigrams/trigrams, etc. Combinations of binary features and frequency-based features were evaluated for all aforementioned feature types. The inclusion of any of the features, however, decreased the performance of our system by at least 2% and they were therefore excluded from any of our final submissions.

3.4.2 Skipgrams

Skipgrams are a relatively new approach in NLP, most notable for their effectiveness in approximating word meaning in vector space models (Mikolov et al., 2013). In addition to calculating the n consecutive units in a sequence, skipgrams introduce another parameter, k, which calculates n-grams of units separated by a distance of k. For example, the character bigram $k = 1$ representation of `apple` would thus be: `(a, p), (p, l), (p, e)`. As such, we experimented with skipgrams for several of our systems. Most notably, we evaluated character 2-9-grams with skips of 2 and 3. These results, however, were largely identical to our simpler 1-9-gram system and were thus discarded due to significantly longer training times and exceedingly sparse feature matrices.

3.4.3 IPA representation

Due to the success of the character n-gram models in capturing morphological details, we tested a feature that transcribed every essay into its phonetic representation. Even though we knew this would largely reproduce the same information captured by the raw text of the essays, we nonetheless hypothesized that an IPA-transcription would reveal further insights about how learners impose the morpho-phonetic features of their native language onto their spelling in English. For example, while dipthongization is not represented by the orthography of a word, a phonetic transcription is able to capture it. Thus, we employed the eSpeak text-to-speech software[3], which reproduced words according to how an English speaker would pronounce them. When tested against our best-performing character-level system, this approach produced a slight drop in F1-score ($\sim 1\%$). This factor, combined with very long training time, led us to ultimately discard the feature.

3.4.4 Misspellings

In examining the essays in the training set, we observed a large number of misspelled words. As such, we experimented with incorporating word misspellings into our system. These words were identified via the PyEnchant Python library[4] and replaced with a dummy token ($\star$). We posited that this would have the effect of identifying misspellings and capturing their distributions per language. The character-level features of misspelled words were also retained by the combined character 1-9-n-gram model. We attempted various feature representations for this method, including 1-9 character n-grams as well as word unigrams combined with non-filtered 1-9 character n-grams. None of these results were noteworthy, however.

[2] https://spacy.io/

[3] http://espeak.sourceforge.net/

[4] http://pythonhosted.org/pyenchant/

4 Results

Table 2 provides an overview of all submissions with results obtained both on the development and on the final evaluation sets.

The results show that our best-performing system was the character 1-9-gram system trained on the concatenation of training and development data. This is a notable improvement on the development set, which was 84%. The system for the first submission is the same as for the second submission. However the first submission was only trained on the training data set and the second also included the dev set. This resulted in more documents for training, which improved the performance of the system. This resulted in our overall best system. The confusion matrix for this system is given in Figure 1.

Here, the most confused cases are Hindi and Telugu, replicating what we observed (at a higher rate) during the development of our system. This discrepancy has also been reported by various prior studies, including Jarvis et al. (2013), whose system was trained and evaluated on different data than ours. The other noteworthy cases are Turkish and Arabic, which are confused at least once for all but one and two languages, respectively. Even though Korean is mislabeled as Japanese eight times, the same does not apply for the reverse situation: Japanese is the second most accurately-labeled class, with only German faring better. Interestingly, the meta classifier over the character 1-9-gram probabilities fares slightly worse than the standalone system. This, however, could likely be due to overfitting the system on the training data, which is a risk posed by any meta-classification approach.

Both of the prompt-based systems produced largely similar results. In the case of prompt-word omission, the confusion between Hindi and Telugu is slightly reduced, but also moved to other classes. The omission of P7 documents from training also resulted in a larger drop in accuracy from the character 1-9-n-gram system, which was not observed during development. Of course a drop would be expected, but not as large as it would likely be if a prompt topic written about in the test data had been omitted from the training data instead. Also, it is possible that the omission of 1,562 total documents from training is responsible for this result, prompt-effect notwithstanding.

Each of the ensemble methods failed to match the performance of the character 1-9-n-gram system. Though the intuition in assembling the ensemble classifiers was that they would provide extra insight for the main classifier by virtue of being non-linear, this is not reflected in terms of accuracy. It is noteworthy, however, that the system including the CNN trained on character 4-5-n-grams and word unigrams (CNN1) improved much on Arabic (88% F1-score) and Turkish (83% F1-score), suggesting that further refinement of this system (perhaps extending the character n-gram ranges) may be fruitful. Unfortunately, due to time constraints, we did not have a chance to explore other configurations, as CNNs take a considerably long time to train.

Finally, we note that the native-trained POS-approach did not produce encouraging results. Unlike the ensemble, which improved the classifier's performance on some classes despite yielding a lower F1-score, the POS tags failed to provide any notable insight relative to the character classifier. However, this is not to say that this approach should be entirely discarded. Rather, it would be interesting to combine the POS features with other feature types, such as word or lemma n-grams, as opposed to character n-grams.

5 Discussion

Our first submission (i.e., standalone 1-9 character n-grams), which was trained on both the training and development data yielded the best test-set performance out of all our submitted systems. As we received the results of our primary submissions from the organizers during the testing period, it was confirmed that the 1-9 character n-gram features were very powerful when evaluated on the test set. We thus continued to include these features in subsequent system submissions. The insight regarding the performance of these initial systems against the test set has certainly impacted the decisions we made about which features to include later. It must be noted, however, that our best performing system was submitted before we had received any such feedback, since it was one of the first systems we submitted. Therefore we did not tweak any aspect of the system for the test set. The system we developed initially without any knowledge of the test set performed best and also proved to be the most compact system.

As mentioned in the features section, topic bias was one of our major concerns during the sys-

Submission	Char. 1-9-grams	Char. 1-10-grams	POS tags	PW omitted	Meta	CBOW	CNN1	CNN2	P7 Omitted	Dev F1	Test F1
Random baseline:										0.0700	0.0909
Essay baseline:										0.6907	0.7104
1	x									0.8374	0.8684
2	x									0.8374	**0.8756**
3	x			x						0.8165	0.8682
4	x				x					**0.8459**	0.8737
5	x					x		x		-	0.8515
6	x					x	x			-	0.8616
7	x								x	0.8410	0.8613
8	x				x	x				0.8321	(0.5302)
9	x		x							0.8212	0.8414
10	x	x								0.8385	0.8720

Table 2: Overview of submissions to the NLI Shared Task 2017. Test scores were received during the test phase. As CNN1 and CNN2 were evaluated against 30% of the development set, their results are excluded from the performance on the development set. (The low test performance for Submission 8 suggests that something went wrong with uploading the correct system.)

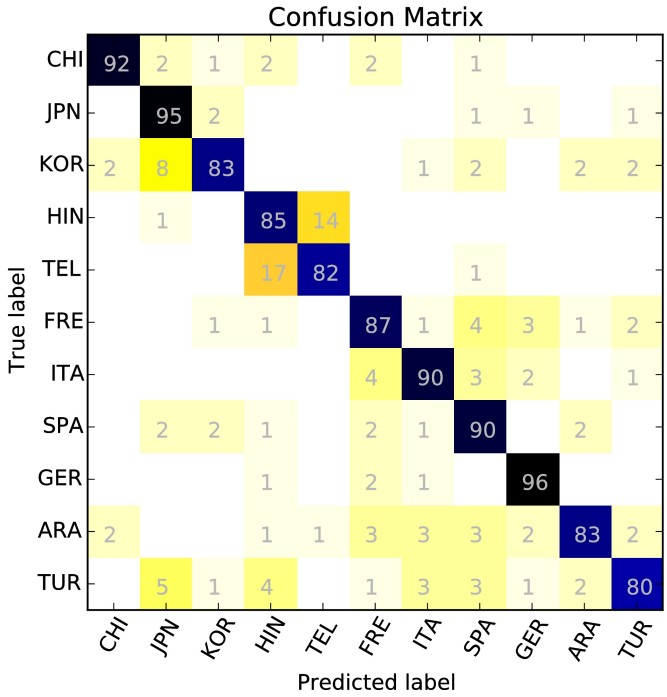

Figure 1: Confusion matrix for character 1-9-grams trained on train+dev

tem evaluation process. Our experiments with the prompt-word feature revealed that our system was indeed modeling prompt topic in addition to native language. In order to further validate this, we ran a series of experiments in which we omitted a prompt from the training set in a leave-one-prompt out scenario. We then fit a classifier on this truncated training data and evaluated it against the entire dev set (i.e., with all prompt information retained). The performance of the classifier varied greatly depending on which prompt was omitted, dropping in accuracy between 3% and 20%.

Interestingly, our experiments with omitting prompt information from the test set (both dev and test, in separate instances) did not reproduce such drastic drops in performance. Instead, the system's accuracy declined only slightly (as in the case of P7 omission), if at all. This suggests that, in evaluating the assessment submissions, the evaluation data can consist of a smaller prompt distribution than the training data, with only minimal prompt-overfitting observed for the latter. Conversely, this also means that a system must be trained on at least the same prompts that the data against which it is evaluated. Otherwise, the drop in performance may be unpredictable.

These prompt-omission experiments led us to conclude that, while it is possible to build a state-

of-the-art model, the fact that it is trained and tested against the same prompt topics likely renders it unable to generalize towards other, potentially unseen future prompts. Furthermore, it is improbable that a system trained on one year's assessments will come close to replicating similar results when tested against essays from other years, due to the discrepancy in potential prompts. Certainly, this is to say that, in order to obtain a true metric of how well any of the submitted systems would fare in practical scenarios (i.e. NLI on future year's TOEFL essays), it is vital that they be tested against a data set that contains different and unseen prompts.

References

Johannes Bjerva, Barbara Plank, and Johan Bos. 2016. Semantic tagging with deep residual networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, pages 3531–3541. http://aclweb.org/anthology/C16-1333.

Julian Brooke and Graeme Hirst. 2012. Robust, lexicalized native language identification. In *Proceedings of COLING 2012*.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for natural language processing. *arXiv preprint arXiv:1606.01781* .

Jacob Eisenstein, Amr Ahmed, and Eric P Xing. 2011. Sparse additive generative models of text .

Sylviane Granger, Estelle Dagneaux, Fanny Meunier, and Magali Paquot. 2009. International corpus of learner english.

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

Radu Tudor Ionescu, 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)*. Association for Computational Linguistics.

Scott Jarvis, Yves Bestgen, and Steve Pepper. 2013. Maximizing classification accuracy in native language identification. In *BEA@ NAACL-HLT*. pages 111–118.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Moshe Koppel, Jonathan Schler, and Kfir Zigdon. 2005. Determining an author's native language by mining a text for errors. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, pages 624–628.

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

Shervin Malmasi, 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 Building Educational Applications Using NLP*. Association for Computational Linguistics, Copenhagen, Denmark.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan T McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *LREC*.

Terence Odlin. 1989. *Language transfer: Cross-linguistic influence in language learning*. Cambridge University Press.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *ACL 2016, arXiv preprint arXiv:1604.05529*. http://arxiv.org/abs/1604.05529.

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

Joel R Tetreault, Daniel Blanchard, and Aoife Cahill. 2013. A report on the first native language identification shared task. In *BEA@ NAACL-HLT*. pages 48–57.

Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2013. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 847–855.