Native Language Identification With Classifier Stacking and Ensembles

Shervin Malmasi
Harvard Medical School
shervin.malmasi@mq.edu.au

Mark Dras
Macquarie University
Department of Computing
mark.dras@mq.edu.au

Ensemble methods using multiple classifiers have proven to be among the most successful approaches for the task of Native Language Identification (NLI), achieving the current state of the art. However, a systematic examination of ensemble methods for NLI has yet to be conducted. Additionally, deeper ensemble architectures such as classifier stacking have not been closely evaluated. We present a set of experiments using three ensemble-based models, testing each with multiple configurations and algorithms. This includes a rigorous application of meta-classification models for NLI, achieving state-of-the-art results on several large data sets, evaluated in both intra-corpus and cross-corpus modes.

1. Introduction

Native Language Identification (NLI) is the task of identifying a writer's native language (L1) based only on their writings in a second language (the L2). NLI works by identifying language use patterns that are common to groups of speakers of the same native language. This process is underpinned by the presupposition that an author's L1 disposes them towards certain language production patterns in their L2, as influenced by their mother tongue. This relates to cross-linguistic influence, a key topic in the field of Second Language Acquisition (SLA), which analyzes transfer effects from the L1 on later learned languages (Ortega 2009).

It has been noted in the linguistics literature since the 1950s that speakers of particular languages have characteristic production patterns when writing in a second language. This language transfer phenomenon has been investigated independently in various fields from different perspectives, including qualitative research in SLA and recently via predictive models in NLP (Jarvis and Crossley 2012). Recently, this has motivated studies in NLI, a subtype of text classification where the goal is to determine the native language of an author, using texts that they have written in a second language (Tetreault, Blanchard, and Cahill 2013).

Submission received: 18 October 2016; Revised version received: 1 August 2017; Accepted for publication: 6 April 2018.

doi:10.1162/COLI_a_00323

© 2018 Association for Computational Linguistics
Published under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0) license

The motivations for NLI are manifold. Such techniques can help SLA researchers identify important L1-specific learning and teaching issues. In turn, the identification of such issues can enable researchers to develop pedagogical material that takes into consideration a learner's L1 and addresses them. It can also be applied in a forensic context, for example, to glean information about the discriminant L1 cues in an anonymous text.

NLI is most commonly framed as a multiclass supervised classification task. Researchers have experimented with a range of machine learning algorithms, with support vector machines (SVMs) having found the most success. However, some of the most successful approaches have made use of classifier ensemble methods to further improve performance on this task. This is a trend that has become apparent in recent work on this task, as we will outline in Section 2. In fact, all recent state-of-the-art systems have relied on some form of multiple classifier system.

However, a thorough examination of ensemble methods for NLI—one empirically comparing different architectures and algorithms—has yet to be conducted. Additionally, more sophisticated ensemble architectures, such as classifier stacking, have not been closely evaluated. This meta-classification approach is an advanced method that has proven to be effective in many classification tasks; its systematic application could improve the state of the art in NLI.

This has links to the idea of adding layers to increase power in neural network–based deep learning, which has come to be an important approach in NLP over the last few years (Manning 2015); Eldan and Shamir (2016, page 907) note that "Overwhelming empirical evidence as well as intuition indicates that having depth in the neural network is indeed important." Deep neural networks can in fact be seen as layered classifiers (Goldberg 2015), and ensemble methods as an alternative way of adding power via additional layers. In this article we look just at ensemble methods: Deep learning has not yet produced state-of-the-art results on related tasks (Malmasi et al. 2016),[1] and our goal is to understand what it is that has made ensemble methods to date in NLI so successful.

The primary focus of the present work is to address this gap by presenting a comprehensive and rigorous examination of how ensemble methods can be applied for NLI. We aim to examine several different ensemble and meta-classification architectures, each of which can utilize different configurations and algorithms.

Furthermore, previous ensemble methods have not been tested on different data sets, making the ability of these models to generalize for NLI unclear. Ideally, the same method should be tested across multiple corpora to assess its validity. To this end, we also apply our methods to multiple data sets, in both English and other languages, to evaluate their generalization capacity. In addition, following the observation of Brooke and Hirst (2012b) that patterns utilized by machine learners can be corpus-specific and influenced by topic, even with apparently topic-neutral features—something that could also be true in our multi-level classifier context—we carry out cross-corpus experiments along the lines of Brooke and Hirst (2012b), Malmasi and Dras (2015), and Ionescu, Popescu, and Cahill (2016).

To summarize, the principal aims of the present study are the following:

1. Apply several advanced ensemble combination methods to NLI and evaluate their performance against previously used ensemble methods.

1 Traditional text classification methods substantially outperformed all deep learning approaches in the 2016 DSL Shared Task.

2. Evaluate the use of meta-classifiers for NLI, applying different feature representations and a range of learning algorithms.
3. Compare the performance of these methods with previous results and assess the methods on, and between, different data sets.

The remainder of this paper is organized as follows. In Section 2 we introduce ensemble classification and recap previous work in NLI. Our data are introduced in Section 3, followed by our experimental set-up in Section 4, and our classification features in Section 5. Our ensemble-based models are detailed in Section 6 and experimental results are reported in Section 7. We then conclude with a discussion in Section 8.

2. Related Work

This work draws on two broad areas of research: ensemble-based classification methods and work in NLI.

2.1 Ensemble Classifiers

Classifier ensembles are a way of combining different classifiers or experts with the goal of improving overall accuracy through enhanced decision making. Instead of relying on decisions by a single expert, they attempt to reach a decision by utilizing the collective input from a committee of experts.

They have been applied to a wide range of real-world problems and have been shown to achieve better results compared with single-classifier methods (Oza and Tumer 2008). Through aggregating the outputs of multiple classifiers in some way, their outputs are generally considered to be more robust. Ensemble methods continue to receive increasing attention from investigators and remain a focus of machine learning research (Kuncheva and Rodríguez 2014; Woźniak, Graña, and Corchado 2014).

Such ensemble-based systems often use a parallel architecture, where the classifiers are run independently and their outputs are aggregated using a fusion method. The specifics of how such systems work will be detailed in Section 6.

They have been applied to various classification tasks with good results. Not surprisingly, researchers have attempted to use them for improving the performance of NLI, as we discuss in the next section.

2.2 Native Language Identification

General Background. Determining the identity of, or properties of, an author of a text is a problem of longstanding interest, and attempts to solve it are similarly longstanding (Koppel, Schler, and Argamon 2009). NLI is the subtype of this task focusing on identifying the L1 of the writer. To our knowledge, the first work tackling this task in a computational manner was that of Tomokiyo and Jones (2001), who worked with speech data recorded for this task. Their main aim was to detect non-native speech, using part-of-speech and lexical features in a naïve Bayes classifier, and to also determine the native language of the non-native speakers. Another early work was that of Jarvis, Castaneda-Jiménez, and Nielsen (2004), who presented an early empirical approach to investigating L1 transfer using lexical style and word choice, which they termed *wordprints*. Working with a written corpus compiled for the task and focusing exclusively on lexical transfer, they collected the 30 most frequent words used by each group of L1 speakers, resulting in a set of 53 words that were used as features in a Linear Discriminant Analysis (LDA) classifier.

The framework that is now most commonly used for NLI was introduced by Koppel, Schler, and Zigdon (2005b). They used essay-style texts drawn from the International Corpus of Learner English (Granger et al. 2009), and a similar multiclass classification paradigm to the earlier work, with an SVM classifier. Their features were broader than the earlier work, and included syntactic, lexical, and stylistic features, such as function words (*e.g., and*, *from*, *which*, *whilst*), character *n*-grams (*i.e.,* sequences of characters of length *n*), and PoS bi-grams, together with some spelling mistakes.

The volume of work then built from further investigation by Wong and Dras (2009). A detailed review of NLI methods is omitted here for reasons of space, but a thorough exposition of work predominantly carried out within the NLP community within the framework set out by Koppel, Schler, and Zigdon (2005a) is presented in the report from the first NLI Shared Task[2] that was held in 2013 (Tetreault, Blanchard, and Cahill 2013). Jarvis and Crossley (2012) present a collection of work from the SLA community, all using LDA classification and features that include word *n*-grams, lexical and semantic indices, and error categories.

Most English NLI work has been done using two corpora. The above-mentioned International Corpus of Learner English (Granger et al. 2009) was widely used until recently, despite its shortcomings[3] being widely noted (Brooke and Hirst 2012a). More recently, TOEFL11, the first corpus designed for NLI, was released (Blanchard et al. 2013). Although it is the largest NLI data set publicly available, it only contains argumentative essays, limiting analyses to this genre. Research has also expanded to use non-English learner corpora (Malmasi and Dras 2014a,c).

Ensemble Methods: The 2013 Shared Task. As mentioned earlier, some of the most successful approaches to NLI have used ensemble learning methods, and investigating ensemble configurations is a key goal of this article. We now therefore present an overview of work in NLI that has used ensembles, focusing on the way in which ensembles were defined and what benefit they brought. Most of this work leads into, is part of, or is positioned with respect to the first NLI shared task of 2013. We discuss briefly after this some observations from the 2017 shared task.

Tetreault et al. (2012) were the first to propose the use of classifier ensembles for NLI, and they performed a comprehensive evaluation of the feature types used until that point. In their study they used an ensemble of logistic regression learners, using a wide range of features that included character and word *n*-grams, function words, parts of speech, spelling errors, and writing quality markers. With regard to syntactic features, they also investigated the use of Tree Substitution Grammars and dependency features extracted using the Stanford parser. Furthermore, they also proposed using language models for this task and in their system used language model perplexity scores based on lexical 5-grams from each language in the corpus. The set of features used here was the largest of any NLI study to date. With this system, the authors reported state-of-the-art accuracies with the ensemble model of 90.1% and 80.9% on the ICLE and TOEFL11 corpora, respectively. The ensemble model improved over a simple concatenation of all feature vectors by 7.5% and 4.9% on the respective corpora. Tetreault et al. also conducted cross-corpus evaluation, using the seven common L1 classes between the

2 https://sites.google.com/site/nlisharedtask2013/home.
3 The issues exist as the corpus was not designed specifically for NLI.

ICLE and TOEFL11 corpora. Training on the ICLE data, they report an accuracy of 26.6%.

The subsequent 2013 NLI Shared Task, attracting entries from 29 teams, saw many approaches that used ensembles.

The winning entry for the shared task did not use an ensemble, but we summarize it to put the other work into context. This system, by Jarvis, Bestgen, and Pepper (2013), had an accuracy of 83.6%, using as features *n*-grams of words, parts-of-speech, as well as lemmas. In addition to normalizing each text to unit length, the authors also applied a log-entropy weighting schema to the normalized values, which clearly improved the accuracy of the model. An L2-regularized SVM classifier was used to create a single-model system. Furthermore, the authors used their own procedure for optimizing the cost parameter (C) of the SVM. Although they did not use a great number of features or introduce any new features for this task, and they did not use an ensemble, we posit that their use of weighting schema and hyperparameter optimization gave their system an edge over their competitors, the majority of whom did not utilize these techniques.

Gyawali, Ramirez, and Solorio (2013) utilized lexical and syntactic features based on *n*-grams of characters, words, and part-of-speech tags (using both the Penn TreeBank and Universal Parts Of Speech tagsets), along with perplexity values of character *n*-grams to build four different models. These models were combined using a voting-based ensemble of SVM classifiers. Features values were weighted using the TF-IDF scheme. In particular, the authors set out to investigate whether a more coarse-grained POS tagset would be useful for NLI. They explored the use of the Universal POS tagset, which has 12 POS categories in the NLI shared task, and compared the results with the fine-grained Penn Treebank (PTB) tagset, which includes 36 POS categories. The highest accuracy of their system in the shared task was 74.8%, achieved by combining all features into an ensemble. The authors found that the use of coarse-grained Universal POS tags as features generalizes the syntactic information and reduces the discriminative power of the feature that comes from the fine granularity of the *n*-grams. For example, the PTB tagset distinguishes verbs into six distinct categories, whereas the Universal POS tagset only has a single category for that grammatical class.

In the system designed by Cimino et al. (2013), the authors used a wide set of general-purpose features, designed to be portable across languages, domains, and tasks. This set included features that are lexical (sentence length, document length, type/token ration, character and word *n*-grams), morpho-syntactic (coarse and fine-grained part-of-speech tag *n*-grams), and syntactic (parse tree and dependency-based features). They found distributional differences across the L1s for many of these features, including average word and sentence lengths. However, we note that many of these differences are not of a large magnitude, and the authors did not run any statistical tests to measure the significance levels of these differences. Using this feature set, they experimented with a single-classifier system as well as classifier ensembles, using SVM and Maximum Entropy classifiers. In their ensemble, they experimented with using a majority voting system as well as a meta-classifier approach. The authors reported that the ensemble methods outperformed all single-classifier systems (by around 2%), and their best performance of 77.9% was provided by the meta-classifier system that used linear SVM and MaxEnt as the component classifiers and combined the results using a polynomial kernel SVM classifier. Although the set of features used in this experiment is not widely different from other reported NLI research, their use of a meta-classifier is an interesting approach that warrants further study.

In their system, Goutte, Léger, and Carpuat (2013) used character, word, and part-of-speech *n*-grams, along with syntactic dependencies. They used an ensemble of

SVM classifiers trained on each feature space, using a majority vote combiner method. To represent the feature values, they used two value normalization methods based on TF-IDF and cosine normalization. Their best entry achieved an accuracy of 81.8%, higher than many systems using the same standard features and more, demonstrating the effectiveness of using ensemble classifiers and appropriate feature value representation. The authors, like many others, also noted that lexical features provided the best performance for a single feature in their system, but that this can be boosted by combining multiple predictors.

The MITRE system (Henderson et al. 2013) was another highly lexicalized system where the primary features used are word, part-of-speech, and character *n*-grams. In this system, these features are used by independent classifiers (logistic regression, Winnow2, and language models) whose output is then combined into a final prediction, using a naïve Bayes model. Their best performing ensemble was 82.6% accurate in the shared task and the authors emphasized the value of ensemble methods that combine independent systems. Furthermore, the authors also optimized the parameters of their naïve Bayes model using a grid search over the development data.

Hladka, Holub, and Kriz (2013) developed an ensemble classifier system, using some standard features (lemma, word, and part-of-speech *n*-grams, word skipgrams) with SVM classifiers. They obtained an accuracy of 72.5% in the shared task. They found that their ensemble, which was based on majority voting, outperformed other methods of combining the features.

Another system that utilized an ensemble is that of Bykh et al. (2013), where they used a probability-based ensemble. They used a set of 16 features, including recurring word-based *n*-grams, recurring Open Class POS *n*-grams, dependencies, trees, and lemmas. To combine the different feature types, they explored combining all features into a single vector and also ensembles of SVM classifiers (each trained on a single feature type). Their best shared task performance of 82.2% was achieved using an ensemble with all of their features; among these features, recurring word-based *n*-grams are the best performing single feature type.

Following the shared task, Bykh and Meurers (2014) further explored the use of lexicalized and non-lexicalized phrase structure rules for NLI. They showed that the inclusion of lexicalized production rules (i.e., preterminal nodes and terminals) provides improved results. In addition to the standard normalized frequency and binary feature representations, they also proposed two new representations based on a "variationist sociolinguistic" perspective. Although they showed that these representations outperformed the normalized frequency approach, they did not compare this to other representations that have been shown to improve NLI accuracy, such as TF-IDF. They combined their lexicalized production rules feature with additional surface *n*-gram features in a tuned and optimized ensemble, reporting an accuracy of 84.82% on the TOEFL11-TEST set.

Ionescu, Popescu, and Cahill (2014) extended the previous work of Popescu and Ionescu (2013), which used string kernels to perform NLI, using only character *n*-gram features. One improvement here is that several string kernels are combined through multiple kernel learning. Although this approach is not based on the types of ensembles we use here, it is similar in the sense that it attempts to combine multiple learners. The authors also performed parameter tuning to select the optimal settings for their system. They reported an accuracy of 85.3% on the TOEFL11-TEST set, 1.7% higher than the winning shared task system. Recently, they expanded their approach with additional experiments (Ionescu, Popescu, and Cahill 2016), although they did not achieve further improvements on TOEFL11-TEST.

Ensemble Methods: The 2017 Shared Task. The 2017 NLI shared task saw a similar level of interest as the 2013 one, with 19 teams participating. The framework was broader than for 2013, looking at NLI based on both text and speech, with three tracks in the task: essay-only, speech-only, and fusion. The essay-only track, which we focus on here, was very similar to the 2013 task, and also based on TOEFL essays. This time, the training set consisted of the TOEFL11 training and development sets, the development set was the TOEFL11 test set, and there was a new test set. (Consequently, the accuracies for systems that took part in this task are not directly comparable with those discussed earlier.) Ranking of systems also involved the use of McNemar's test for statistical significance, which determined five bands of performance that were statistically indistinguishable.

Again, several systems used classifier stacking or ensembles of classifiers, including the top system, and we discuss these briefly. The 2017 shared task took place after the release of the first version of this paper,[4] and the systems that used classifier ensembles or stacking for the most part cited this work as the source of the architectural framework used. In the top band of seven systems (accuracies 86.5% to 88.2%), three of the best systems from each team (Cimino and Dell'Orletta 2017; Goutte and Léger 2017; Li and Zou 2017) used the architectures described later in this paper, with the same kinds of base machine learners, but incorporating some variations: for instance, Cimino and Dell'Orletta (2017) additionally used a sentence-level classifier. In the second band of three systems (accuracies 85.4% to 86.4%), all used ensembles (Chan et al. 2017; Ircing et al. 2017; Oh et al. 2017). Interestingly, this band included the highest-ranked system to incorporate deep learning (Oh et al. 2017), supporting our earlier observation that a state-of-the-art deep learning architecture for this kind of classification task has not yet been identified. Two systems that explored deep learning architectures (Bjerva et al. 2017; Sari, Rifqi Fatchurrahman, and Dwiastuti 2017) were ranked in the third band.

One new observation that can be made from the systems in this 2017 task is that although some systems found that more features can be helpful (Cimino and Dell'Orletta 2017; Goutte and Léger 2017; Li and Zou 2017; Markov et al. 2017), some found that fewer were better (Ionescu and Popescu 2017; Kulmizev et al. 2017; Rama and Çöltekin 2017).[5] In one case (Rama and Çöltekin 2017) this was specific to the fusion task, rather than the essay task, but in one other (Kulmizev et al. 2017) a single SVM with only character *n*-grams performed better than some ensemble architectures motivated by the previous version of the present paper (Malmasi and Dras 2017b). The motivation of a systematic exploration of ensemble architectures that is the goal of the present paper is thus still of broad interest.

In sum, we can see that ensemble-based approaches have yielded some of the most successful results in this field. However, we also believe that it is possible to use ensemble models that are even more sophisticated, leading to improved results. This is the key research question we investigate in the present study.

3. Data

We now introduce the data sets used in this study.

One of the goals of this study is to assess the generalization capability of the methods and results across data sets, so we consequently use two corpora that have been used in previous NLI work.

4 We made this available at `https://arxiv.org/abs/1703.06541`, as Malmasi and Dras (2017b).
5 We thank an anonymous reviewer for this observation.

3.1 The TOEFL11 Corpus

The TOEFL11 corpus (Blanchard et al. 2013)—also known as the Educational Testing Service Corpus of Non-Native Written English—is the first data set designed specifically for the task of NLI and developed with the aim of addressing the above-mentioned deficiencies of other previously used corpora. By providing a common set of L1s and evaluation standards, the authors set out to facilitate the direct comparison of approaches and methodologies.

Furthermore, as all of the texts were collected through the Educational Testing Service's electronic test delivery system, this ensures that all of the data files are encoded and stored in a consistent manner.[6] The corpus is available through the Linguistic Data Consortium.[7]

It consists of 12,100 learner texts from speakers of 11 different languages. The texts are independent task essays written in response to eight different prompts, and were collected in the process of administering the Test of English as a Foreign Language (TOEFL) between 2006 and 2007. The texts are divided into specific training (TOEFL11-TRAIN), development (TOEFL11-DEV), and test (TOEFL11-TEST) sets. It is also common to combine the training and development sets for training, which we refer to as TOEFL11-TRAINDEV.

The 11 L1s are Arabic, Chinese, French, German, Hindi, Italian, Japanese, Korean, Spanish, Telugu, and Turkish. This selection ensures that there are L1s from diverse language families, but also several from within certain families. The L1s and their language families are shown in Figure 1.

This data set was designed specifically for NLI and the authors attempted to balance the texts by topic and native language. There are a total of eight essay prompts in the corpus, with the prompts setting each essay's topic or theme. Although they were not able to create a perfectly balanced corpus, the distribution of topics across L1s is very even.[8] This distribution of essay prompts by L1 is shown in Figure 2.

3.2 The EFCAMDAT Corpus

The EF Cambridge Open Language Database (EFCAMDAT) is an English L2 corpus that was released in 2013 (Geertzen, Alexopoulou, and Korhonen 2013) and used for NLI in Malmasi and Dras (2015).

EFCAMDAT is composed of texts submitted to *Englishtown*, an online school used by thousands of learners daily. This corpus is a large one, containing some 550k texts from numerous nationalities. Whereas TOEFL11 is made of argumentative essays, EFCAMDAT has a much wider range of genres including writing e-mails, descriptions, letters, reviews, instructions, and more.

Malmasi and Dras (2015) used this corpus for two purposes: to apply NLI to another large-scale corpus for L2 English that differ in the characteristics (e.g., genre) of its constituent texts; and to carry out a large-scale cross-corpus analysis, following earlier

6 The essays are distributed as UTF-8 encoded text files.
7 `https://catalog.ldc.upenn.edu/LDC2014T06`.
8 Kulmizev et al. (2017) observed that this does not perfectly prevent an NLI from improving performance via topic modeling. They replaced the top 100 keywords per prompt in TOEFL11, determined using a sparse additive generative model, with a dummy string, and observed that performance dropped by around 2%. Nevertheless, the even distribution of topics in TOEFL11 does mitigate this to some extent, compared with the earlier ICLE corpus.

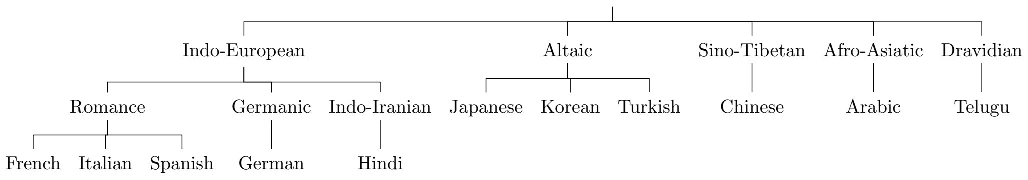


Figure 1
Language families in the TOEFL11 corpus. The languages were selected to represent different families, but to also have several from within the same families. Diagram reproduced from Blanchard et al. (2013) with permission.

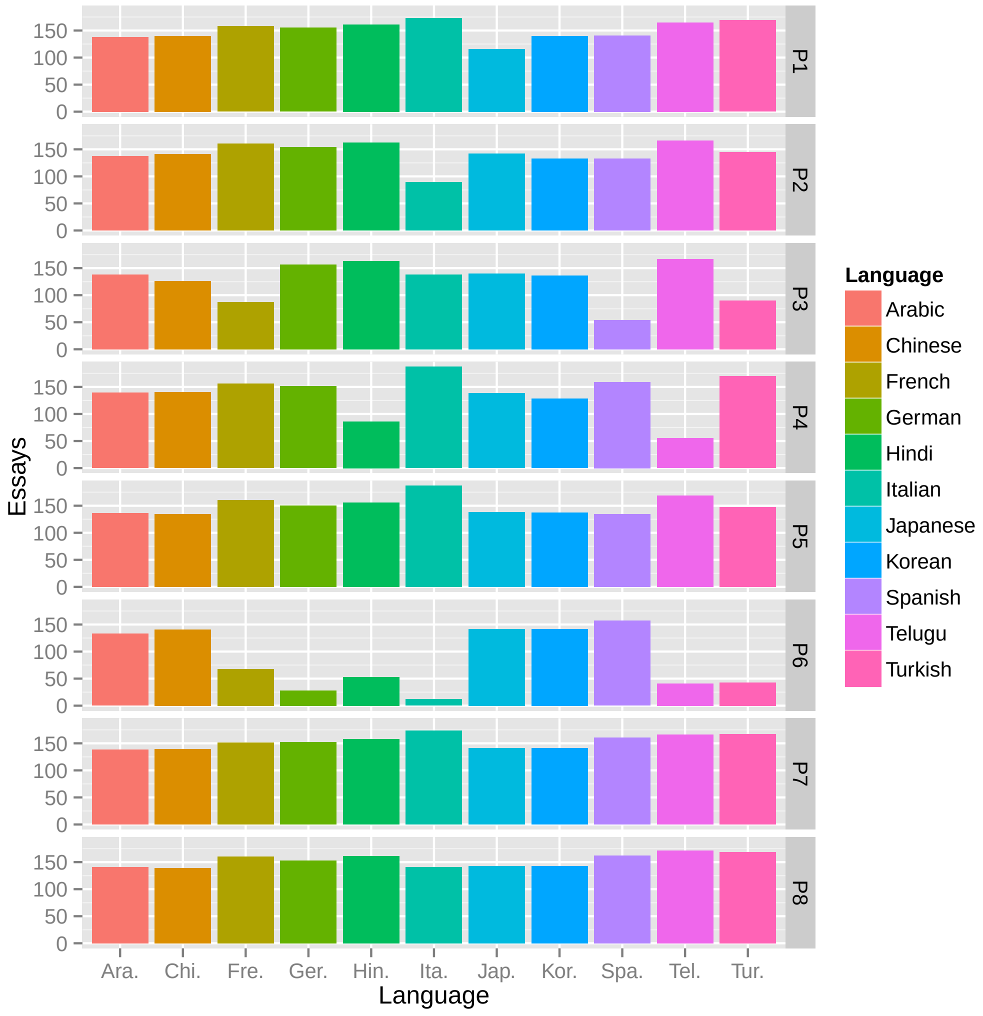


Figure 2
A plot of how the eight topic prompts are distributed across the L1 groups in the TOEFL11 corpus. Prompts are labeled P1–P8. Figure reproduced from Blanchard et al. (2013) with permission.

Table 1
The 11 L1 classes extracted from the EFCAMDAT corpus, compared with the TOEFL11 corpus. The first 9 classes are common to both.

Common	Arabic, Chinese, French, German, Italian, Japanese, Korean, Spanish, Turkish
EFCAMDAT	Portuguese, Russian
TOEFL11	Hindi, Telugu

work by Brooke and Hirst (2012b) and Tetreault et al. (2012) on other, smaller corpora. Brooke and Hirst (2012b, page 392) note that patterns utilized by machine learners can be corpus-specific, and can be influenced by topic, even with apparently topic-neutral features; but that "any variation that is consistent across very distinct corpora is likely to be a true indicator of L1." Following Malmasi and Dras (2015), we use EFCAMDAT for both within-corpus and cross-corpus evaluations—training on TOEFL11 and testing on EFCAMDAT, and vice versa—and apply the same set-up for our experiments with the data set, which we recap here.

As some of the texts in EFCAMDAT can be short, Malmasi and Dras (2015) followed the methodology of Brooke and Hirst (2011) to concatenate and create texts with at least 300 tokens, producing texts similar in size to TOEFL11. From the data 850 texts were chosen from each of the top 11 nationalities, the same number of L1s as in TOEFL11 to allow a more direct comparison of classifier accuracies (see Table 1 for the languages). This subset of EFCAMDAT thus consists of 9,350 documents totalling approximately 3.2m tokens. This is an average of 337 tokens per text, close to the 348 tokens per text in TOEFL11. The within-corpus experiments use this whole subset, whereas the cross-corpus experiments use just the nine languages in common between EFCAMDAT and TOEFL11. There is no separate test set for this corpus.

3.3 The ASK Corpus

In this study we use data from the ASK Corpus (Andrespråkskorpus [Second Language Corpus]). The ASK Corpus (Tenfjord, Meurer, and Ragnhildstveit 2013; Tenfjord, Johansen, and Hagen 2006; Tenfjord, Meurer, and Hofland 2006) is a learner corpus composed of the writings of learners of Norwegian. These texts are essays written as part of a test of Norwegian as a second language. Each text also includes additional metadata about the author such as age or native language. A positive feature of this corpus is that all the texts have been collected under the same conditions and time limits. The corpus also contains a control subcorpus of texts written by native Norwegians under the same test conditions. The corpus also includes error codes and corrections, although we do not make use of this information here.

There are a total of about 1,700 essays[9] written by learners of Norwegian as a second language with ten different first languages: German, Dutch, English, Spanish, Russian, Polish, Bosnian-Croatian-Serbian, Albanian, Vietnamese, and Somali. The essays are written on a number of different topics, but these topics are not balanced across the L1s.

9 Tenfjord, Meurer, and Ragnhildstveit (2013), in the description of the corpus, remark that it consists of 1,700 essays. However, we received 1,736 essays from the corpus maintainers.

Detailed word-level annotations (lemma, POS tag, and grammatical function) have been first obtained automatically, using the Oslo-Bergen tagger. These annotations have then been manually post-edited by human annotators because the tagger's performance can be substantially degraded due to orthographic, syntactic, and morphological learner errors. These manual corrections can deal with issues such as unknown vocabulary or wrongly disambiguated words.

We use two different derived data sets for Norwegian, which we term "raw" and "generated."

Raw Data Set. This consists of just the full set of original documents.

Generated Data Set. Unlike TOEFL11, the ASK corpus is not designed for NLI. Consequently, we cannot expect that the essays will be balanced for topic or proficiency, nor for length.[10]

As one way of deriving a corpus consisting of documents that are more homogeneous with respect to these properties, we generate artificial essays from the raw documents.[11] Manufacturing documents in this manner has a number of effects. First, it ensures that all documents are similar and comparable in length. If the data are being used to classify documents from another source, rather than for cross-validation, the generation parameters could be changed so that the training set is similar to the test set in terms of length. Second, the random sampling used here means that the texts created for each class are a mix of different authorship styles, proficiencies, and topics.

In this work we extracted 750k tokens of text from the ASK corpus in the form of individual sentences. Following a similar methodology to that of Brooke and Hirst (2011), we randomly selected and combined the sentences from the same L1 to generate texts of approximately 300 tokens on average, creating a set of documents suitable for NLI.

More specifically, the data set composed of artificial documents is generated as follows. For each class, all the available texts are processed and the individual sentences from these texts are placed into a single pool. Once this pool has been created, we begin the process of generating artificial documents.

For each artificial text to be generated, its required minimum length is first determined by randomly picking a value within a prespecified range $[M, N]$. This chosen value represents the minimum number of tokens or characters that are required to create a new document. By specifying this range parameter, rather than a single fixed value, we can create an artificial data set where there is still some reasonable (and controlled) amount of variation in length between texts.

Sentences from the pool are then randomly allocated to the document until its length exceeds the required minimum value. The document is then considered complete; it is added to the new data set and we proceed to generate another. It should also be noted that the document length may exceed the upper bound of the range parameter, depending on the length of the final sentence that crosses the minimum threshold. The sampling of sentences from the pool is done without replacement.

10 Blanchard et al. (2013) remark that there is some length variability in TOEFL11, as high-proficiency texts tend to be longer and low-proficiency ones shorter, but that most essays were in the middle of the length distribution.

11 We repeat the earlier observation that, even with a corpus balanced across topic, a classifier is not completely prevented from taking advantage of this information. This approach is again only intended to mitigate the effect.

Table 2
The 10 L1 classes included in the Norwegian NLI data set and the numbers of raw documents and documents we generated for each class.

Native Language	Documents	
	raw	generated
Albanian	124	121
Dutch	200	254
English	200	273
German	200	280
Polish	200	281
Russian	200	257
Serbian	200	259
Somali	107	90
Spanish	200	243
Vietnamese	105	100
Total	1,736	2,158

This process continues until there are insufficient sentences to create any more documents. The sentences remaining in the pool are then discarded. This procedure is performed for every class in the original data set and yields a new data set of artificial documents.

The 10 native languages and the number of texts generated per class are listed in Table 2. In addition to these we also generate 250 control texts written by native

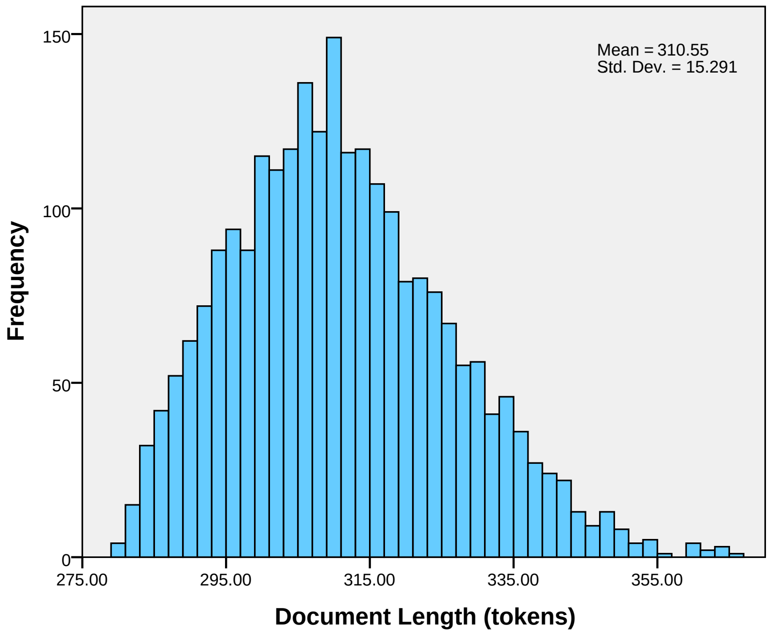


Figure 3
A histogram of the number of tokens per document in the data set that we generated.

speakers. A histogram of the number of tokens per document is shown in Figure 3. The documents have an average length of 311 tokens with a standard deviation of 15 tokens.

Part-of-Speech Tagset. The ASK corpus uses the Oslo-Bergen tagset,[12] which has been developed based on the Norwegian Reference Grammar (Faarlund, Lie, and Vannebo 1997).

Here each POS tag is composed of a set of constituent morphosyntactic tags. For example, the tag `subst-appell-mask-ub-fl` signifies that the token has the categories "noun common masculine indefinite plural." Similarly, the tags `verb-imp` and `verb-pres` refer to imperative and present tense verbs, respectively.

Given its many morphosyntactic markers and detailed categories, the ASK data set has a rich tagset with over 300 unique tags.

3.4 The Jinan Chinese Learner Corpus

Growing interest has led to the recent development of the Jinan Chinese Learner Corpus (JCLC; Wang, Malmasi, and Huang 2015), the first large-scale corpus of L2 Chinese consisting of university student essays. Learners from 59 countries are represented and proficiency levels are sampled representatively across beginner, intermediate, and advanced levels. Texts by learners from other Asian countries are disproportionately represented, with this likely being due to geographical proximity and links to China. Again, we constructed raw and generated data sets from this corpus, the former as a standard set-up and the latter to try to mitigate the effects of topic bias, length heterogeneity, and so forth.[13]

Raw. To construct the raw JCLC data set, for the smaller L1s, we kept all of the documents, and for the larger ones we randomly sampled from among the original documents, in a manner such that the rank order of each language was preserved (e.g., Indonesian was still the most common L1).

Generated. For the generated JCLC data set, we extracted 3.75 million tokens of text from the JCLC in the form of individual sentences. Following the methodology described in the previous section, we combine the sentences from the same L1 to generate texts of 600 tokens on average,[14] creating a set of documents suitable for NLI.

Although there are over 50 L1s available in the corpus, we choose the top 11 languages, shown in Table 3, to use in our experiments. This is based on two considerations. First, although many L1s are represented in the corpus, most have relatively few texts. Choosing the top 11 classes allows us to have a large number of classes and also to ensure that there are sufficient data per-class. Second, this is the same number of classes used in the NLI 2013 shared task, enabling us to draw cross-language comparisons with the shared task results.

12 `http://tekstlab.uio.no/obt-ny/english/tagset.html`.

13 We note that this corpus does not provide topic information at a document level in the metadata, so there is no possibility of balancing raw documents by topic. We also note that in the description by Wang, Malmasi, and Huang (2015) of the JCLC corpus, there is huge variance in the mean text length per class, ranging from 401 to 1,135 tokens. This is reflected across the whole data set, as shown in Figure 1 of Wang, Malmasi, and Huang (2015).

14 A single Chinese character is considered a token.

Table 3
The 11 L1 classes included in the Chinese NLI data set, and the numbers of raw documents and documents we generated for each class.

Native Language	Documents	
	raw	generated
Burmese	408	349
Filipino	293	415
Indonesian	807	402
Japanese	270	180
Khmer	329	294
Korean	566	330
Laotian	398	366
Mongolian	119	101
Spanish	198	112
Thai	806	400
Vietnamese	806	267
Total	5,000	3,216

4. Experimental Set-up

In this study we use a supervised multiclass classification approach. The learner texts are organized into classes according to the author's L1 and these documents are used for training and testing in our experiments. In this section we describe our experimental methodology, including evaluation and the algorithms we use. Our classification features will be described in Section 5; and the three classification models we create using these algorithms and features are then described in Section 6.

4.1 Evaluation

In the same manner as many previous NLI studies and also the NLI 2013 shared task, we report our results as classification accuracy under k-fold cross-validation, with $k = 10$. In recent years this has become a de facto standard for reporting NLI results. For creating our folds, we used stratified cross-validation, which aims to ensure that the proportion of classes within each partition is equal (Kohavi 1995). For TOEFL11 we also test on the standard test set, which we call TOEFL11-TEST.

We use three baselines to provide lower bounds for accuracy:

- Random—uniform random over classes.
- Majority Class (where this differs from Random).
- Single Vector—a linear SVM (Section 4.3) with a single feature vector consisting of all features used in the data set, as a direct comparator for using an ensemble vs. not using one (as per Tetreault et al. [2012]).

We use additional baselines depending on the corpus, where comparable results by other systems are available.

Oracles, which we describe in Section 6.1.8, will also be used to estimate a potential upper bound for accuracy. We use them to assess how close our models are to achieving optimal performance. Additionally, we also compare our results against those reported in previous work. These were described earlier in Section 2.

4.2 Statistical Significance Testing

Statistical significance testing in NLP is not universal, and there is not a standard approach to it even among purely classification tasks. To our knowledge the level of significance of differences in various NLI systems has not been previously carried out, so in this section we just note the test we will use to compare the best result from our various approaches against other top results from the literature.

Because we are evaluating the classifier outputs for the same samples, a test for paired nominal data is suitable. We use McNemar's test (McNemar 1947), a non-parametric method to test for significant differences in proportions for paired nominal data; it has been used widely elsewhere in the machine learning literature (West 2000; Aue and Gamon 2005, for example), and a description of its use in classification tasks is available in Foody (2004).

We note that it does require the entire set of predictions from each system to be compared. Predictions made by all 144 submissions in the 2013 shared task were made available as part of Malmasi, Tetreault, and Dras (2015).[15] For the work described in this article, Ionescu, Popescu, and Cahill (2014) also kindly provided us with the predictions from their state-of-the-art system, which we also make available.[16] As part of this work, we also make available the predictions of our best system on the TOEFL11-TEST set.

4.3 Classification Methods

In this work we use a range of classifiers, all of which have been used for text classification and most of which are standard, although we make some remarks on a couple of them here. All of the learners listed will be used as meta-classifiers, but only SVMs and Logistic Regression will be considered as base learners, as we note subsequently. The learners to be used are:

- linear SVMs (Joachims 1998);
- SVMs with a Radial Basis Function (RBF) kernel (Joachims 1998);
- multinomial logistic regression (Genkin, Lewis, and Madigan 2007);
- perceptrons (Rosenblatt 1958);
- ridge regression (Zhang and Oles 2001);
- decision trees (Quinlan 1993);
- Linear Discriminant Analysis (LDA, not to be confused with Latent Dirichlet Allocation) (Fisher 1936);
- Quadratic Discriminant Analysis (QDA) (Hastie, Tibshirani, and Friedman 2009);

15 Available from `http://web.science.mq.edu.au/~smalmasi/resources/nli2013`.
16 These two sets of predictions are available at `http://web.science.mq.edu.au/~smalmasi/resources/nli-predictions`.

- *k*-nearest neighbors (Cover and Hart 1967); and
- nearest centroid (NC) (Tibshirani et al. 2002).

Multiclass Classification. Some methods such as multinomial logistic regression are inherently multiclass. Others, such as SVMs, however, are inherently binary classifiers. A common way to adapt these binary classifiers for multiclass problems is through a one-vs.-all (OVA) approach, also known as a one-vs.-rest (OVR) approach. Another common alternative is a one-vs.-one (OVO) method that builds $\frac{N(N-1)}{2}$ binary classifiers for all pairwise combinations. For SVMs in the context of NLI, it has been found that the OVR approach works best (Brooke and Hirst 2012b), and our experiments confirmed this. We therefore adopt this approach to multiclassing for all methods that are not already multiclass.

Classifier Output Representation. As we will describe in Section 6, our meta-classifiers are trained on the outputs generated by individual base classifiers, which can be either discrete labels or continuous values.

All classifiers produce discrete label output, which is represented as a one-hot vector of size K, the number of classes in the data set. Additionally, some classifiers such as Logistic Regression naturally provide continuous values associated with each of the possible K class labels; for some of these the continuous values represent probabilities of the labels, or can be interpreted as some other measure of confidence in the labels. Where available, this output information can also be used to form a vector for classification. For each input, this would result in a K-element vector where each element is the continuous output associated with a class label. Where it is not available by default, it is sometimes possible to derive a score associated with each label.

SVMs and Logistic Regression. Linear SVM classifiers are a highly robust supervised classification method that has proven to be very effective for text classification (Joachims 1998). Strong theoretical and empirical arguments have been made for the utility of SVMs for text classification, showing that their capabilities are well suited for several properties of text data, including extremely large feature spaces, very high sparsity, and few irrelevant features.

With regard to NLI, post hoc analysis of the 2013 shared task results revealed that many of the top systems, including the winning entry, followed two broad patterns: they used SVM classifiers along with frequency-based feature values. We thus consider this as one possibility for our base classifiers.

We note that SVM is a margin-based classifier and does not output probability estimates for each class label as a part of the classification. However, there are methods to map the outputs to probabilities based on the signed distance to the separating hyperplane (Platt 2000); we use this technique in this article.

We also use RBF-kernel SVMs, but only for the meta-classifier. Hsu et al. (2003, Appendix C) have examined the way in which this type of kernel may work poorly for large feature spaces, which makes it unsuitable for the base classifier, but the meta-classifier will use many fewer features.

Logistic regression is a type of linear regression model where the dependent variables are categorical. Although high-dimensional input poses a challenge for these models (Genkin, Lewis, and Madigan 2007), this issue can be addressed to some degree using regularization methods (Zhu and Hastie 2004).

Logistic regression was also popular in the shared task, so we consider that as an alternative to linear SVMs for our base classifier, with both L1 and L2 regularization.

This algorithm is inherently multiclass, meaning that OVA and OVO approaches are not required. The logistic regression classifier is also probabilistic and provides continuous probability estimates for each class label.

LDA/QDA. Unlike the other classifiers, LDA and QDA are not widely used in NLP. However, we investigate them in this work because they have been successfully used for classification elsewhere—Liu and Wechsler (2002), for example, in vision—but more particularly because the body of work that came out of the field of SLA (Jarvis and Crossley 2012) used it extensively.

LDA is a classic learning algorithm, dating to Fisher (1936); an accessible explanation is given in Hastie, Tibshirani, and Friedman (2009). Like naive Bayes, LDA and QDA are based on models for the class-conditional densities; whereas naive Bayes uses non-parametric class density estimates that are the products of marginal densities (i.e., assuming inputs are conditionally independent with respect to class), LDA and QDA use multivariate Gaussian densities. Where classes are assumed to have a common covariance matrix, this gives the special case of LDA, where the decision boundary becomes a hyperplane. QDA is the case where the assumption of common covariance across classes is not made, resulting in a quadratic decision surface. Unlike LDA, however, it makes no assumption about equal class covariances, allowing them to be class-specific. Both LDA and QDA are inherently multiclass. Hastie, Tibshirani, and Friedman (2009, page 111) note that they have performed well on a range of classification tasks and comment that "whatever exotic tools are the rage of the day, we should always have available these two simple tools."

5. Features

As its focus is on classifier architecture, this study just utilizes a standard set of NLI features used in previous work, as noted in, for example, the overview of the systems in the NLI shared task of 2013 (Tetreault, Blanchard, and Cahill 2013). We combine these various feature types because it has been shown that both lexical and syntactic features each capture diverse types of information that are complementary (Malmasi and Cahill 2015). Different feature types are extracted for each of our data sets, as shown in Table 4.

Table 4
An overview of the features used for each data set.

Feature	TOEFL11	EFCAMDAT	Chinese	Norwegian
Word/Lemma *n*-grams	X			
Character *n*-grams	X			
Function word unigrams	X	X	X	X
Function word bigrams	X	X	X	X
POS *n*-grams	X	X	X	X
Dependencies	X		X	
CFG Rules	X	X	X	
Adaptor Grammars	X			
TSG Fragments	X			

The feature types for each data set were chosen based on properties of the data set and the availability of NLP resources for the L2.

For stylistic classification tasks like NLI, these content-based features can only be used if the training data are balanced for topic. Otherwise, topic balance may impact the results and artificially inflate accuracy (Malmasi and Dras 2017a). Accordingly, we only use these features for our experiments with TOEFL11, which is balanced for topic. The NLP tools used to extract adaptor grammar and Tree Substitution Grammar (TSG) fragment features are only available for English. The paucity of NLP tools for Norwegian and the lack of topic balance in the data also limited our features to those manually annotated in the data set.

We remark here that the key goal of this article is to systematically evaluate ensemble architectures, not features: We will be comparing the performance of our models *within* a particular experimental set-up (a single data set for within-corpus, two data sets for cross-corpus) and not *across* these.

The remainder of this section describes the feature types listed in Table 4.

Word, Lemma, and Character n*-grams*. For TOEFL11, we extract content word unigrams and bigrams, lemma unigrams and bigrams, and character uni/bi/trigrams. For stylistic classification tasks like NLI, these content-based features can artificially boost performance through detecting topic-related, rather than NLI-related, clues (Koppel, Schler, and Argamon 2009): As noted in Section 2.2 and Section 3.1, TOEFL11 was constructed as the first corpus used for NLI that was topic-balanced to minimize this issue. Accordingly, we only use these features for our experiments with TOEFL11.

As discussed in more detail in Malmasi and Dras (2015), we do not use these features for EFCAMDAT, aiming to minimize topic-related clues: They noted that, for a classifier using word unigrams, the topic 15 features contained strong content-based clues (e.g., the words *Germany*, *Berlin*, *Hamburg*, *Frankfurt*, and *Munich* for German; and *Saudi*, *Arabia*, *Arabic*, *Mohammed*, and *Ali* for Arabic). Although it is not possible to eliminate the topic bias,[17] omitting these features reduces it. And as Brooke and Hirst (2011) note, the use of cross-corpus evaluation is another technique for minimizing the effect of topical clues.

Function Words. In contrast to content words, function words (e.g., articles, determiners, conjunctions and auxiliary verbs) do not have any thematic meaning themselves, but rather can be seen as indicating the grammatical relations between other words; consequently, they have been widely used in stylistic classification tasks. In this work, the English word list was obtained from the Onix Text Retrieval Toolkit.[18] For Norwegian we used a list of 176 function words obtained from the distribution of the Apache Lucene search engine software.[19] This list includes stop words for the Bokmål variant of the language and contains entries such as *hvis* [whose], *ikke* [not], *jeg* [I], *så* [so], and *hjå* [at]. We also make this list available on our Web site.[20] For Chinese, we utilize the function word list described in Malmasi and Dras (2014b).

17 It is true that other features can still detect topic bias: Brooke and Hirst (2011) showed that the ICLE corpus suffered from this, as, for example, the fact that French texts were more about philosophy and religion, whereas Japanese texts were more about personal experiences, such as language learning and travel, had related register differences that could be detected by non-content features.

18 `http://www.lextek.com/manuals/onix/stopwords1.html`.

19 `https://github.com/apache/lucene-solr`.

20 `http://web.science.mq.edu.au/~smalmasi/data/norwegian-funcwords.txt`.

In addition to single function words, we also extract function word bigrams, as described by Malmasi, Wong, and Dras (2013). Function word bigrams are a type of word *n*-gram where content words are skipped: They are thus a specific subtype of the skipgrams discussed by Guthrie et al. (2006). For example, the sentence "*We should all start taking the bus*" would be reduced to "*we should all the*," from which we would extract the *n*-grams.

Part-of-Speech n-*grams*. The TOEFL11 data was tagged using two different tagsets: the Penn Treebank tagset (PTB) and the CLAWS tagset, which has been shown to perform well for NLI (Malmasi, Wong, and Dras 2013). The tagging was performed using the Stanford CoreNLP software[21] (Manning et al. 2014) and the Robust Accurate Statistical Parsing (RASP)[22] system (Briscoe, Carroll, and Watson 2006), respectively. Detailed information and comparison between the tagsets, including a discussion of their effects on NLI accuracy, can be found in Malmasi and Dras (2017a, Section 9). The EFC data were tagged using only the PTB tagset in order to replicate the baseline system against which we evaluate our results.

We also used Stanford CoreNLP for Chinese. We did not use any NLP tools for Norwegian as the corpus we use is already annotated with POS tags.

We extracted POS *n*-grams of order 1–3, which have been shown to be useful for NLI (Malmasi, Wong, and Dras 2013). Previous work and our experiments showed that sequences of size 4 or greater achieve lower accuracy, possibly because of data sparsity, so we do not include them.

Adaptor Grammar Collocations. Following Wong, Dras, and Johnson (2012), for the TOEFL11 data we include arbitrary length *n*-gram collocations discovered by an adaptor grammar. We use the specific features obtained by Wong, Dras, and Johnson (2012), both the pure POS *n*-grams as well as the better-performing mixtures of POS and function words.

Stanford Dependencies. For English and Chinese, we use Stanford dependencies as a syntactic feature: For each text we extract all the basic dependencies returned by the Stanford parser (de Marneffe, Maccartney, and Manning 2006). We then generate all the variations for each of the dependencies (grammatical relations) by substituting each lemma with its corresponding POS tag. For instance, a grammatical relation of `det(knowledge, the)` yields the following variations: `det(NN, the)`, `det(knowledge, DT)`, and `det(NN, DT)`.

CFG Rules. For English and Chinese, we obtain a constituent-based parse using the Stanford parser, and extract the production rules that constitute those trees to use as features.

Tree Substitution Grammar Fragments. TSG fragments were proposed by Swanson and Charniak (2012) as another type of syntactic feature for NLI that captures a broader syntactic context than the single-level fragments of phrase-structure trees that constitute the CFG rules. We only extract TSG fragments for the TOEFL11 data as they include lexical terminal nodes.

21 `http://nlp.stanford.edu/software/corenlp.shtml`.
22 `http://users.sussex.ac.uk/~johnca/rasp/`.

6. Classification Models

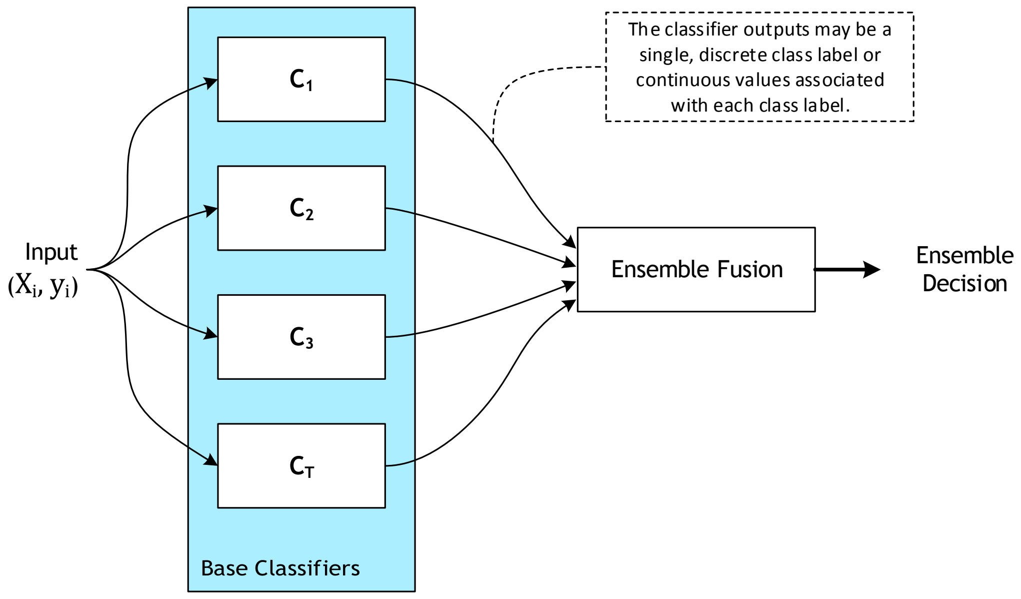
We conduct a set of three experiments, each based on different ensemble structures that we describe in this section. The first model is based on a traditional parallel ensemble structure and the second model examines meta-classification using classifier stacking. The third and final model is a hybrid approach, building an ensemble of meta-classifiers.

6.1 Ensemble Classifiers

The most common ensemble structure, as described earlier in Section 2.1, relies on a set of base classifiers whose decisions are combined using some predefined method. This is the approach for our first model.

Such systems often use a parallel architecture, as illustrated in Figure 4, where the classifiers are run independently and their outputs are aggregated using a fusion method. The first part of creating an ensemble is generating the individual classifiers. Various methods for creating these ensemble elements have been proposed. These involve using different algorithms, parameters, or feature types; applying different pre-processing or feature scaling methods; and varying (e.g., distorting or resampling) the training data. (See Dietterich [2000] for a more detailed discussion of several methods.)

For example, **Bagging** (bootstrap aggregating) is a commonly used method for ensemble generation (Breiman 1996) that can create multiple base classifiers. It works by creating multiple bootstrap training sets from the original training data and a separate classifier is trained from each set. The generated classifiers are said to be diverse because



The classifier outputs may be a single, discrete class label or continuous values associated with each class label.
C₁
C₂
C₃
C_T
Input (Xi, yi)
Ensemble Fusion
Ensemble Decision
Base Classifiers

Figure 4
An example of a parallel ensemble classifier architecture where T independent classifiers provide predictions, which are then fused using a rule-based ensemble combination method. The class labels for the input, y_i, are only available during training or cross-validation. In this work the input vector X_i for each classifier is a distinct vector based on a single feature type.

each training set is created by sampling with replacement and contains a random subset of the original data. **Boosting** (e.g., with the AdaBoost algorithm) is another method where the base models are created with different weight distributions over the training data with the aim of assigning higher weights to training instances that are misclassified (Freund and Schapire 1996).

In our first model we follow the same approach as previous NLI research and train each classifier on a different feature type.[23] However, our other models make use of the boosting and bagging techniques, which will be discussed in later sections. As noted in Section 4.3, we consider linear SVMs and logistic regression for these base classifiers.

The second part of ensemble design is choosing a combination or fusion rule to aggregate the outputs from the various learners; this is discussed in the next section. Most NLI research to date has not compared different types of such combiners and we aim to evaluate a number of different strategies.

6.1.1 Ensemble Fusion Methods. Once it has been decided how the set of base classifiers will be generated, selecting the classifier combination method is the next fundamental design question in ensemble construction.

The answer to this question depends on what output is available from the individual classifiers. The two different output types were discussed earlier in Section 4.3. Some combination methods are designed to work with class labels, assuming that each learner outputs a single class label prediction for each data point. Other methods are designed to work with class-based continuous output, requiring that for each instance every classifier provide a measure of confidence[24] for each class label. These outputs may correspond to probabilities for each class and consequently sum to 1 over all the classes. If an algorithm can provide both types of output, then all the methods can be tested. This is the case for the SVM and logistic regression classifiers we will work with.

These methods are usually based on some predefined rule or logic and cannot be trained. This can be considered an advantage, allowing them to be implemented and used without additional training of domain-specific combination models. On the other hand, they may not be able to exploit domain-specific trends and patterns in the input data.

Although a number of different fusion methods have been proposed and tested, there is no single dominant method (Polikar 2006). The performance of these methods is influenced by the nature of the problem and available training data, the size of the ensemble, the base classifiers used, and the diversity between their outputs. This is an important motivation for comparatively assessing these methods on NLI data.

The selection of this method is often done empirically. Many researchers have compared and contrasted the performance of combiners on different problems, and most of these studies—both empirical and theoretical—do not reach a definitive conclusion (Kuncheva 2014, page 178).

In the same spirit, we experiment with several classifier fusion methods that have been widely applied and discussed in the machine learning literature. Our selected methods are described in the following paragraphs; a variety of other methods exist and the interested reader can refer to the thorough exposition by Polikar (2006).

23 An alternative is to train each classifier on a subspace of the entire feature set that includes all types.
24 For example, an estimate of the posterior probability for the class label.

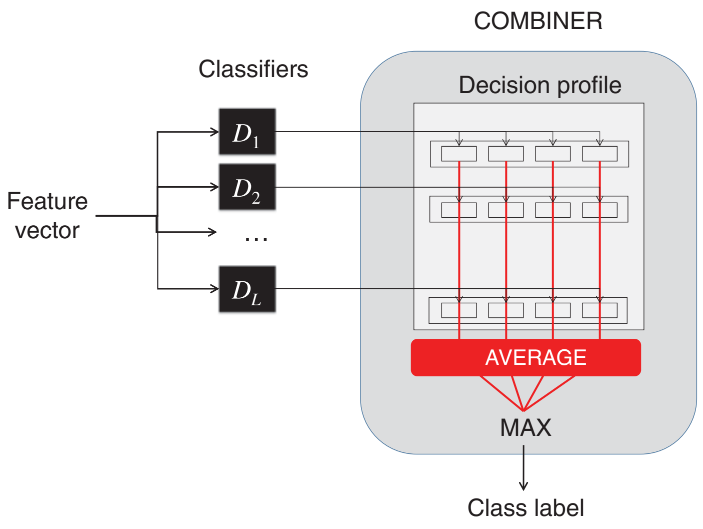


Figure 5
An example of a mean probability combiner. The feature vector for a sample is input to L different classifiers, each of which output a vector of confidence probabilities for each possible class label. These vectors are combined to form the decision profile for the instance that is used to calculate the average support given to each label. The label with the maximum support is then chosen as the prediction. Image reproduced from Kuncheva (2014, Fig. 5.5) with permission.

6.1.2 Plurality Voting. Each classifier votes for a single class label. The votes are tallied and the label with the highest number of votes wins.[25] Ties are broken arbitrarily. This method is simple and does not have any parameters to tune. An extensive analysis of the method and its theoretical underpinnings can be found in Kuncheva (2004, page 112).

6.1.3 Mean Probability Rule. The probability estimates for each class, provided by each individual classifier, are summed and the class label with the highest average probability is the winner. This is illustrated in Figure 5. This is equivalent to the probability sum combiner, which does not require calculating the average for each class. An important aspect of using probability outputs in this way is that a classifier's support for the true class label is taken into account, even when it is not the predicted label (e.g., it could have the second highest probability). This method has been shown to work well on a wide range of problems and, in general, it is considered to be simple, intuitive, stable (Kuncheva 2014, page 155), and resilient to estimation errors (Kittler et al. 1998), making it one of the more robust combiners discussed in the literature.

6.1.4 Median Probability Rule. Given that the mean probability used in the mean probability rule is sensitive to outliers, an alternative is to use the median as a more robust estimate of the central value (Kittler et al. 1998). Under this rule, each class label's estimates are sorted and the median value is selected as the final score for that label. The label with the highest median value is picked as the winner. As with the mean

25 This differs with a *majority* voting combiner, where a label must obtain over 50% of the votes to win. However, the names are sometimes used interchangeably.

combiner, this method measures the central tendency of support for each label as a means of reaching a consensus decision.

6.1.5 Product Rule. For each class label, all of the probability estimates are multiplied together to create the label's final estimate (Polikar 2006, page 37). The label with the highest estimate is selected. This rule can theoretically provide the best overall estimate of posterior probability for a label, assuming that the individual estimates are accurate. A trade-off here is that this method is very sensitive to low probabilities: A single low score for a label from any classifier will essentially eliminate that class label.

6.1.6 Highest Confidence. In this simple method, the class label that receives the vote with the largest degree of confidence is selected as the final prediction (Kuncheva 2014, page 150). In contrast to the previous methods, this combiner disregards the consensus opinion and instead picks the prediction of the expert with the highest degree of confidence.

6.1.7 Borda Count. This method works by using each classifier's confidence estimates to create a ranked list of the class labels in order of preference, with the predicted label at rank 1. The winning label is then selected using the Borda count[26] algorithm (Ho, Hull, and Srihari 1994). The algorithm works by assigning points to labels based on their ranks. If there are N different labels, then each classifier's preferences are assigned points as follows: the top-ranked label receives N points, the second place label receives $N - 1$ points, third place receives $N - 2$ points, and so on, with the last preference receiving a single point. These points are then tallied to select the winner with the highest score.

The most obvious advantage of this method is that it takes into account all of each classifier's preferences, making it possible for a label to win even if another label received the majority of the first preference votes.

6.1.8 Oracle Combiners. Our final set of combiners are designed to assist with assessing the potential performance upper bound that could be achieved by a system, given a set of classifiers. As such, they are primarily used for evaluation in the same manner that baselines help determine the lower bounds for performance. These combiners cannot be used to make predictions on unlabeled data.

One possible approach to estimating an upper-bound for classification accuracy, and one that we use here, is the use of an "oracle" combiner. This method has previously been used to analyze the limits of majority vote classifier combination (Kuncheva, Bezdek, and Duin 2001). An oracle is a type of multiple classifier fusion method that can be used to combine the results of an ensemble of classifiers that are all used to classify a data set.

The oracle will assign the correct class label for an instance if at least one of the constituent classifiers in the system produces the correct label for that data point. Some example oracle results for an ensemble of three classifiers are shown in Table 5. The probability of correct classification of a data point by the oracle is:

$$P_{\text{Oracle}} = 1 - P(\text{All Classifiers Incorrect})$$

26 This method is generally attributed to Jean-Charles de Borda (1733–1799), but evidence suggests that it was also proposed by Ramon Llull (1232–1315).

Table 5
Example oracle results for an ensemble of three classifiers.

Instance	True Label	Classifier Output C_1	C_2	C_3	Oracle
18354.txt	ARA	TUR	ARA	ARA	Correct
15398.txt	CHI	JPN	JPN	KOR	Incorrect
22754.txt	HIN	GER	TEL	HIN	Correct
10459.txt	SPA	SPA	SPA	SPA	Correct
11567.txt	ITA	FRE	GER	SPA	Incorrect

Oracles are usually used in comparative experiments and to gauge the performance and diversity of the classifiers chosen for an ensemble (Kuncheva 2002; Kuncheva et al. 2003). They can help us quantify the *potential* upper limit of an ensemble's performance on the given data and how this performance varies with different ensemble configurations and combinations.

To account for the possibility that a classifier may predict the correct label essentially by chance (with a probability determined by the random baseline) and thus exaggerate the oracle score, we also use an Accuracy@N combiner. This method is inspired by the "Precision at k" metric from Information Retrieval (Manning, Raghavan, and Schütze 2008), which measures precision at fixed low levels of results (e.g., the top 10 results). Here, it is an extension of the Plurality vote combiner, where instead of selecting the label with the highest votes, the labels are ranked by their vote counts and an instance is correctly classified if the true label is in the top N ranked candidates.[27] Another way to view it is as a more restricted version of the Oracle combiner that is limited to the top N ranked candidates in order to minimize the influence of a single classifier having chosen the correct label by chance. In this study we experiment with $N = 2$ and 3. We also note that setting $N = 1$ is equivalent to the Plurality voting method and setting N to the number of class labels is equivalent to the Oracle combiner.

6.2 Meta-Classifiers

While the combination methods in our first model are not trainable, other more sophisticated ensemble methods that rely on meta-learning use a stacked architecture where the output from a first layer of classifiers is fed into a second-level meta-classifier, and so on. For our second model we expand our methodology to such a meta-classifier, also referred to as **classifier stacking** (Wolpert 1992).

A meta-classifier architecture is composed of an ensemble of base classifiers, just as in our first model. A key difference is that instead of using a rule-based fusion method, the individual classifier outputs, along with the training labels, are used to train a second-level meta-classifier. This second meta-learner serves to predict the final decision for an input, given the decisions of the base classifiers. This set-up is illustrated in Figure 6.

27 In case of ties, we choose randomly from the labels with the same number of votes.

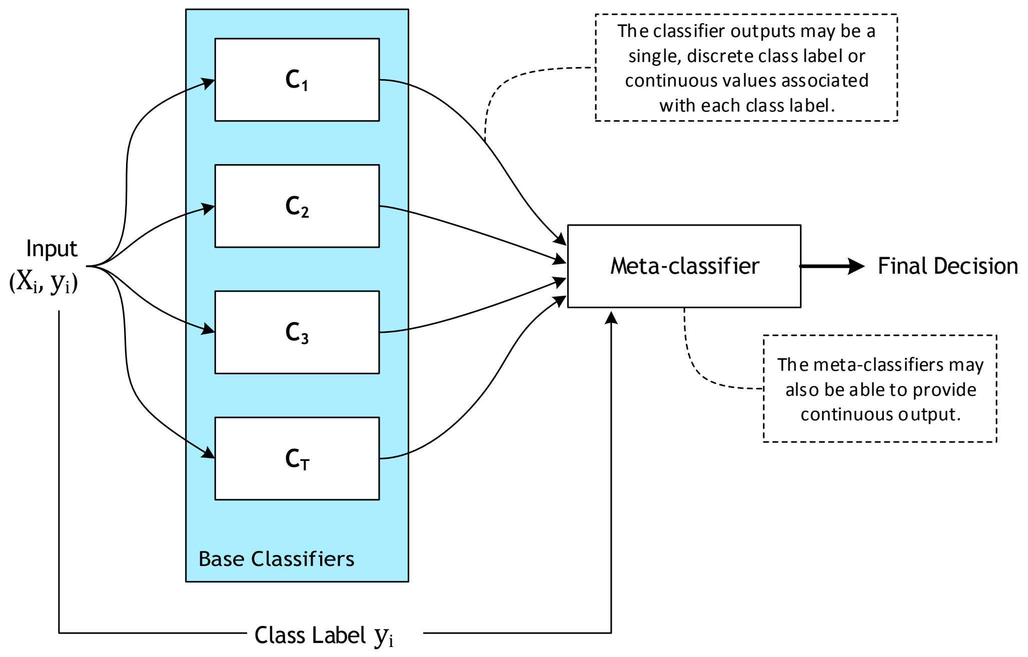

C1
C2
C3
CT
Base Classifiers
Input
(Xi, yi)
Meta-classifier
Final Decision
Class Label yi
The classifier outputs may be a single, discrete class label or continuous values associated with each class label.
The meta-classifiers may also be able to provide continuous output.

Figure 6
An example architecture for a meta-classifier. The outputs generated by the set of *T* base classifiers, along with the labels of the training data, are used to train a meta-learner that can predict the final decision for an input. This meta-classifier attempts to learn from the collective knowledge represented by the ensemble of base classifiers and may be able to learn and exploit regularities in their output. The class labels for the input are only available during training or cross-validation. In this work the input vector X_i for each base classifier is a distinct vector based on a single feature type.

This meta-classifier attempts to learn from the collective knowledge represented by the ensemble of local classifiers and may be able to learn and exploit patterns and regularities in their output (Polikar 2006, Section 3.6). For example, it may be the case that a certain ensemble element is more accurate at classifying instances of a certain class or there may be interactions or correlations between the outputs of certain elements that could help improve results over a simple fusion method. Thus the meta-classifier may learn how the base classifiers commit errors and attempt to correct their biases.

Just as there are different fusion methods for ensemble combination, different learning algorithms can be used for the meta-classifier element. In this study we experiment with all of the learning algorithms listed earlier in Section 4.3. Additionally, we also test each learner, using both the continuous and discrete output data in order to comparatively assess their performance. This approach allows us to evaluate whether one method performs better, and whether certain algorithms are better suited for some specific input formats.

Similar to the base learners, the meta-classifier can generate both continuous and discrete output. In this model we take the discrete label output and use it as the final decision to be used in evaluating the model.

For training, the input for the meta-classifier can be obtained from the outputs of the base classifiers under cross-validation. That is to say, the classifier outputs from

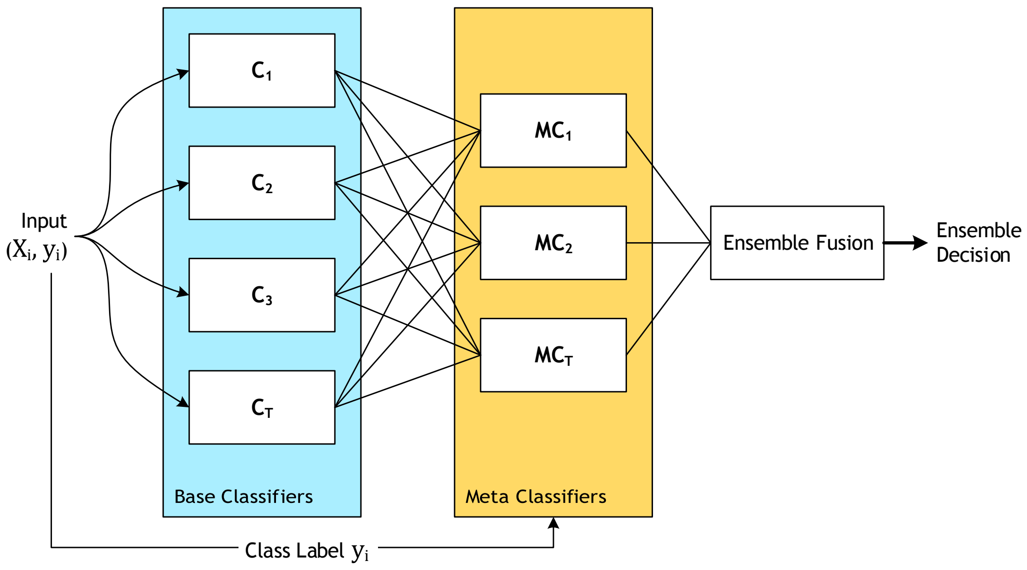


Figure 7
An illustration of our meta-classifier ensemble. The outputs from the ensemble of base classifiers is used to train an ensemble of meta-classifiers, the output of which is processed using a decision fusion method. This is a hybrid approach that attempts to combine both ensemble fusion and classifier stacking. In this work the input vector X_i for each base classifier is a distinct vector based on a single feature type.

each test fold are paired with the original gold-standard label and this is used to train the meta-classifier.[28]

6.3 Meta-Classifier Ensembles

The two models described thus far have relied on multiple classifier combination and meta-learning. Although they both have their advantages, would it be possible to combine both approaches?

To this end, our final model is a hybrid of the previous two approaches that attempts to answer this question. Results from the set of base classifiers are provided to an ensemble of meta-classifiers, instead of a single one. The outputs from the meta-classifiers are then combined using a fusion method to reach a final verdict. The layout for this model is illustrated in Figure 7. This approach, although adding substantial complexity to the model, could potentially combine the benefits of stacking and ensemble combination.

Additionally, this approach also requires a method for generating the meta-classifier ensemble itself. While the first level ensemble is generated using a different feature type per classifier, that method cannot be applied here because we are using classifier outputs. For that purpose we experiment with boosting and bagging (as described in Section 6.1). These methods have been widely used for creating decision tree ensembles (Geurts, Ernst, and Wehenkel 2006); we will experiment with random forests, extra

28 "Test fold" refers to the cross-validation division of the training data, not the test data.

trees, and the AdaBoost algorithm. We will also experiment with bagging, which can be applied to any learner that can be applied for meta-classification, such as SVMs.

7. Experiments and Results

We divide our experiments into three parts: comprehensive experiments on the English TOEFL11 data (Section 7.1); comparative experiments using the EFCAMDAT data set, in both intra-corpus and cross-corpus modes (Section 7.2); and experiments on our non-English L2 corpora (Section 7.3). Details of system configurations and parameter settings are available in Appendix A.

7.1 Experiments on TOEFL11

As additional baselines for the TOEFL11 experiments, beyond the ones common to all experiments described in Section 4.1, we also compare our ensemble results against the winning system from the 2013 NLI shared task (Jarvis, Bestgen, and Pepper 2013) and two systems by Bykh and Meurers (2014) and Ionescu, Popescu, and Cahill (2014), which presented state-of-the-art results following the task. They were all previously described in Section 2.

Base Classifiers. As described in Section 6, we create our ensembles from a set of models where each is trained on a different feature type. We first test our individual classifiers that form this first layer of our three models; this can inform us about their individual performance and the single best feature type. This is done by training the models on the combined TOEFL11-TRAIN and TOEFL11-DEV data, which we refer to as TOEFL11-TRAINDEV, and testing against the TOEFL11-TEST set. As noted in Section 4.3, we considered linear SVMs and logistic regression for these base classifiers. Table 6 presents the results of these. SVMs are clearly the best in this evaluation; there are only two instances (for POS bigrams, with PTB or RASP tags) where they are beaten by the logistic regression classifier with L2 regularization, and only by small amounts (1%). We therefore use linear SVMs as our base classifiers in the following experimental work.

These SVM results are shown graphically in Figure 8. We observe that there is a range of performance, and some features achieve similar accuracy. We also note that the best single-model performance is approximately 78%. Having shown that our base classifiers achieve good results on their own, we now apply our three ensemble models.

Ensembles. We begin by applying the six ensemble combination methods discussed in Section 6.1 as part of our first model. We do this using both cross-validation within TOEFL11-TRAINDEV and by training our models on TOEFL11-TRAINDEV and testing on TOEFL11-TEST. The results for all fusion methods are shown in Table 7.

The mean probability combiner which uses continuous values for each class (Section 6.1.3) achieves the best performance for both of our test sets. This is followed by the plurality vote and median probability combiners, both of which have similar performance. Although plurality voting achieves good results, the Borda Count method performs worse.

In our cross-validation experiments, some 2.7% of the instances resulted in voting ties that were broken arbitrarily. This randomness leads to some variance in the results for voting-based fusion methods; running the combiner on the same input can produce slightly different results.

Table 6
Comparing as potential base classifiers SVMs and Logistic Regression with both L1 and L2 regularization. The classifiers were trained on TOEFL11-TRAINDEV data and the accuracy in the table is evaluated on the TOEFL11-TEST set.

Features	**Accuracy (%)**		
	SVM	LogRegr (L1)	LogRegr (L2)
AG	71.1	57.8	66.6
CFG	54.5	48.2	50.8
CHAR1	37.0	32.9	28.8
CHAR2	57.9	51.3	49.0
CHAR3	69.7	61.5	63.9
FUNCW1	55.2	55.2	54.4
FUNCW2	64.5	51.1	61.9
LEMMA1	76.2	66.9	70.4
LEMMA2	78.3	55.6	71.7
PTB1	37.9	37.4	34.6
PTB2	50.8	52.3	51.8
PTB3	56.4	51.8	55.7
RASP1	48.5	46.0	46.5
RASP2	63.5	56.8	64.6
RASP3	65.3	49.3	63.3
S. DEPS	75.9	49.6	64.7
TSG FRAGS	75.2	59.0	64.5
WORD1	77.5	66.3	71.7
WORD2	78.3	54.2	72.5

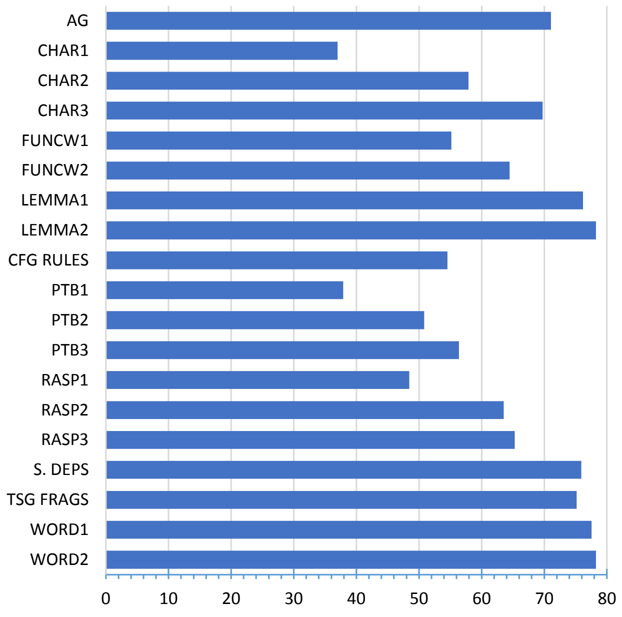


Figure 8
NLI accuracy per feature type on the TOEFL11 test set.

Table 7
Comparing different ensemble classifiers against our baselines and oracles. The CV column lists cross-validation within the TOEFL11-TRAINDEV data and the Test column is the TOEFL11-TEST set. Best ensemble result per column represented in **bold**.

	Method	Accuracy (%) CV	Test
Baselines	Random Baseline	9.1	9.1
	Single Vector Baseline	78.2	77.5
	2013 Shared Task Winner	84.5	83.0
	Bykh and Meurers (2014)	—	84.8
	Ionescu, Popescu, and Cahill (2014)	84.1	85.3
Oracles	Accuracy@2	91.8	92.0
	Accuracy@3	94.5	94.6
	Oracle	96.1	96.0
Ensembles	Plurality Voting	82.6	82.5
	Borda Count	81.2	81.5
	Mean Probability	**82.6**	**83.3**
	Median Probability	82.4	82.7
	Product Rule	80.3	80.6
	Highest Confidence	80.1	80.4

Results from the highest confidence and product rule combiners are the poorest among the set, and by a substantial margin. We hypothesize that this is because they are both highly sensitive to outputs from all classifiers. A single outlier or poor prediction can adversely affect the results. They should generally be used in circumstances where the base classifiers are known to be extremely accurate, which is not the case here. Accordingly, we do not experiment with these any further.

These results from this first model comport with previous research reporting that ensembles outperform single-vector approaches (see Section 2); our best ensemble result is some 5% higher than our best feature, and 4–6% higher than the single vector baseline, the direct comparator for using the same features without an ensemble.

Meta-classifiers. We next apply our meta-classifier (Section 6.2) to both the discrete and continuous outputs generated by the base classifiers. Although the base classifiers remain the same, we train a meta-classifier, using each of the machine learning algorithms we listed earlier in Section 4.3. This results in 15 meta-classification models. Each model is tested using both discrete and continuous input, using both cross-validation and the TOEFL11-TEST set. The results for all of these experiments are shown in Table 8.

Broadly, we observe two important trends here: The meta-classification results are substantially better than the ensemble combination methods from Table 7, and the meta-classifiers trained on continuous output perform better than their discrete label counterparts. This last pattern is not all that surprising because we already observed that the probability-based ensemble combiners outperformed the voting-based combiners. Using continuous values associated with each label provides the meta-learner with more information than a single label, likely helping it make better decisions.

Although most of our algorithms perform well, the LDA meta-classifier yields the best results across both input types and test conditions. These results, 85.2% under

Table 8
Results from our 15 meta-classifiers applied to TOEFL11, using both discrete and continuous outputs from the base classifiers. The best result in each column is in **bold**. The results can be compared with the baselines in Table 7.

Meta-classifier	**Discrete**		**Continuous**	
	CV	Test	CV	Test
Random Baseline	9.1	9.1	9.1	9.1
Linear SVM	84.3	84.4	84.5	85.2
RBF-Kernel SVM	84.2	84.7	84.6	85.1
Logistic Regression	83.9	83.8	84.3	84.8
Ridge Regression	84.3	84.8	84.2	84.5
Perceptron	78.5	81.5	80.9	81.7
Decision Tree	77.6	78.3	75.1	75.2
QDA	56.8	57.3	67.4	67.9
LDA	**84.3**	**84.7**	**85.2**	**86.8**
Nearest Centroid	83.6	83.5	83.1	83.5
5-NN	83.2	82.5	81.6	82.0
10-NN	83.5	83.0	83.0	84.5
15-NN	83.6	83.4	83.2	84.3
20-NN	83.6	83.6	83.3	84.7
50-NN	83.4	83.6	83.7	84.2
100-NN	83.1	83.3	83.6	84.1

cross-validation and 86.8% on TOEFL11-TEST, are already higher than the current state of the art on this data and well exceed the baselines listed in Table 7. It is also important to note that the same classifier achieves the best performance across all four testing conditions.

The linear and RBF SVMs also achieve competitive results here. Instance-based *k*-NN and Nearest Centroid classifiers also do well. On the other hand, decision trees, QDA, and the Perceptron algorithm have the poorest performance across both discrete and continuous inputs.

Ensemble of Meta-classifiers. We have thus far shown that ensembles outperform a single-vector approach, and that meta-classifiers achieve state-of-the-art results. Our final TOEFL11 experiment involves applying our hybrid ensemble of meta-classifiers (Section 6.3) to determine if we can further improve these results. Given the results from the previous model, we only test using continuous classifier outputs.

We experiment with two general methods for creating ensembles of meta-classifiers: boosting and bagging. Although a single decision tree was not a good meta-classifier, it has been shown that ensembles of trees can perform very well (Banfield et al. 2007). We experiment with random forests, extra trees, and AdaBoost for creating such tree-based ensembles. We also apply bagging to several of the best meta-classifiers from Table 8: SVMs, Logistic Regression, Ridge Regression, and LDA. To combine the ensemble of meta-classifiers we use the mean probability combiner, given its better performance among the combiners listed in Table 7.

Results from these models are shown in Table 9. As expected, we observe that the tree-based methods receive a substantial performance increase compared with the

Table 9
Results for our ensembles of meta-classifiers applied to TOEFL11. Best result per column in **bold,** best result per row grouping is underlined.

	Method	Accuracy (%) CV	Accuracy (%) Test
Baselines	Random Baseline	9.1	9.1
	Single Vector Baseline	78.2	77.5
	2013 Shared Task Winner	84.5	83.0
	Bykh and Meurers (2014)	—	84.8
	Ionescu, Popescu, and Cahill (2014)	84.1	85.3
	Our LDA Meta-classifier (continuous)	<u>85.2</u>	<u>86.8</u>
Decision Tree Ensembles	Random Forest	<u>84.2</u>	<u>84.6</u>
	Extra Trees	81.0	82.7
	AdaBoost	75.3	76.2
Bagging	Linear SVM	84.5	85.2
	Logistic Regression	84.4	84.9
	Ridge Regression	84.5	85.2
	LDA	**85.3**	**87.1**

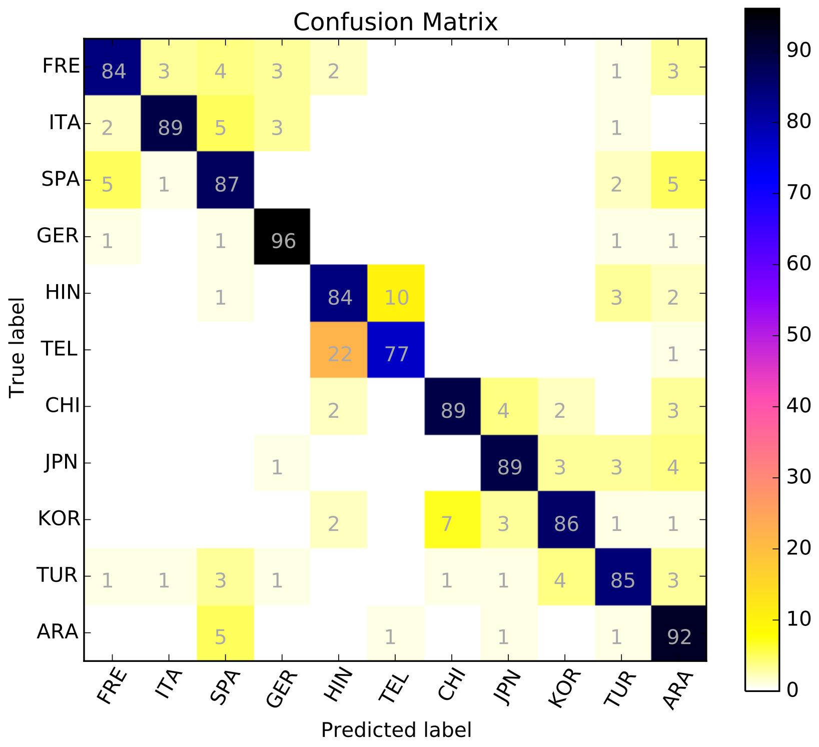
single decision tree meta-classifier from the previous model. Random forests provide the biggest boost, improving performance by almost 10%. However, this is still lower than our LDA meta-classifier.

Applying bagging to our discriminative meta-classifiers, we observe that we gain a small improvement over the previous model. The LDA-based method again outperforms the others, and while the improvement is not huge, it sets a new upper bound for TOEFL11. In fact, this result is only 9% lower than the oracle accuracy of 96%. Given that bagging involves some randomness, we calculate the mean and standard deviation of our top-performing LDA bagging approach to quantify its variability. Running it 50 times, the mean is the value reported in Table 9, and the standard deviation is 0.11, which is relatively small, suggesting a genuine difference.

It is interesting to consider why LDA performs so well here, especially in light of it not being a standard classification method in NLP. Hastie, Tibshirani, and Friedman (2009) speculate as to its good performance across a range of classification problems, and note that it is *not* likely to be because those problems have innately near-Gaussian data that fits the models' assumptions with respect to class-conditional densities. They comment, in an observation that is likely to be applicable to our data:

> More likely a reason is that the data can only support simple decision boundaries such as linear or quadratic, and the estimates provided via the Gaussian models are stable. This is a bias-variance tradeoff — we can put up with the bias of a linear decision boundary because it can be estimated with much lower variance than more exotic alternatives. (page 111)

The performance of the other meta-classifiers supports the suggestion that our data are susceptible to poor quality decision boundaries when these are permitted. Linear SVMs and logistic regression, for example, have quite smooth simple boundaries, and perform well here; decision trees, by contrast, can have very complex boundaries, and



	FRE	ITA	SPA	GER	HIN	TEL	CHI	JPN	KOR	TUR	ARA
FRE	84	3	4	3	2					1	3
ITA	2	89	5	3						1	
SPA	5	1	87							2	5
GER	1		1	96						1	1
HIN			1		84	10				3	2
TEL					22	77					1
CHI					2		89	4	2		3
JPN				1				89	3	3	4
KOR					2		7	3	86	1	1
TUR	1	1	3	1			1	1	4	85	3
ARA			5			1		1		1	92

Figure 9
Confusion matrix of our best-performing NLI system, achieving 87.1% accuracy on the TOEFL11-TEST set.

perform poorly. The very poor performance of QDA suggests that there is a particular issue in permitting class-specific covariances for this type of technique.

In designing this set-up we were initially concerned that the addition of further layers could lead to the addition of errors in the deeper classifiers, resulting in performance degradation. However, this was not the case and accuracy increased, if only slightly.

A confusion matrix of our best system's predictions on the TOEFL11-TEST set is presented in Figure 9. The labels in the matrix have been ordered in a way similar to Figure 1 in order to group similar languages together. We achieve our best performance on German texts, with only four misclassifications. In the top left corner we also observe some confusion between the Romance languages. We also observe the asymmetric confusion between Hindi and Telugu, as discussed in previous research (Malmasi, Tetreault, and Dras 2015). Another interesting observation is that Arabic, which has poor precision, receives misclassifications from every other class, except Italian. This trend can be observed in the last column of the matrix.

We also assessed per-class performance using precision, recall, and the F1-score, with results listed in Table 10. As shown in the confusion matrix, Hindi and Telugu have the worst performance. Recalculating the values without those two classes, the average F1-score improves to 0.89.

Comparison with State-of-the-Art Systems. We now evaluate the performance of our top model against that of the two previous state-of-the-art systems that were used as

Table 10
Per-class performance breakdown of our top system's results on the TOEFL11-TEST set. Best result per column in **bold**. Best and worst performances per column are also highlighted in green/yellow.

Class	Precision	Recall	F1-score
ARA	0.80	0.92	0.86
CHI	0.92	0.89	0.90
FRE	0.90	0.84	0.87
GER	0.92	**0.96**	**0.94**
HIN	0.75	0.84	0.79
ITA	**0.95**	0.89	0.92
JPN	0.91	0.89	0.90
KOR	0.91	0.86	0.88
SPA	0.82	0.87	0.84
TEL	0.88	0.77	0.82
TUR	0.87	0.85	0.86
Average	0.87	0.87	0.87

Table 11
Results for statistical significance testing between our top system and two previous state-of-the-art systems. The p values from McNemar's test are reported. * = significant at the 0.001 level, *** = significant at the 0.05 level.

	Jarvis et al.	**Ionescu et al.**	**Our Method**
Jarvis, Bestgen, and Pepper (2013)	—	0.1082	0.0001*
Ionescu, Popescu, and Cahill (2014)	—	—	0.0314***
Our Method	—	—	—

baselines in our experiments, using the aforementioned prediction data. We report the pairwise p values for the test, as listed in Table 11. They show that the improvement in our results is significantly better than both of the baselines.

7.2 Cross-Corpus Experiments

The second set of our experiments focuses on investigating the extent to which our findings so far generalize to another corpus. The result patterns observed on TOEFL11 have been stable across the training and test set, but we now apply them to another large data set, both corpus-internally and in a cross-corpus setting.

We present results for the top performing models tested in Section 7.1: four ensemble combiners, four bagging-based meta-classifiers, and four ensembles of meta-classifiers. The results for the models trained and tested on EFCAMDAT are in Table 12; those trained on TOEFL11 and tested on EFCAMDAT in Table 13; and those trained on TOEFL11 and tested on EFCAMDAT in Table 14. The baselines (except Single Vector) and oracle scores in all three tables are reproduced from Malmasi and Dras (2015).

Table 12
Results for our three models on the EFCAMDAT data set, using continuous classifier outputs. Best result per column in **bold**, best result per row grouping is underlined.

	Feature	Accuracy (%)
Baselines	Random Baseline	9.1
	Majority Class Baseline	9.1
	Single Vector Baseline	58.5
	Current Best (Malmasi and Dras 2015)	65.0
Oracles	Accuracy@2	74.6
	Accuracy@3	82.0
	Oracle	86.8
Ensembles	Plurality Voting	58.8
	Borda Count	60.9
	Mean Probability	65.0
	Median Probability	62.2
Meta-classifier	Linear SVM	66.8
	Logistic Regression	66.7
	Ridge Regression	66.1
	LDA	67.3
Meta-classifier Bagging	Linear SVM	67.0
	Logistic Regression	66.8
	Ridge Regression	66.4
	LDA	**67.8**

Table 13
Results for our three models trained on the TOEFL11 data set and tested on EFCAMDAT, using continuous classifier outputs. Best result per column in **bold**, best result per row grouping is underlined.

	Feature	Accuracy (%)
Baselines	Random Baseline	11.1
	Majority Class Baseline	11.1
	Single Vector Baseline	24.7
	Current Best (Malmasi and Dras 2015)	28.4
Oracles	Accuracy@2	42.0
	Accuracy@3	53.6
	Oracle	62.4
Ensembles	Plurality Voting	25.5
	Borda Count	26.9
	Mean Probability	28.4
	Median Probability	27.2
Meta-classifier	Linear SVM	31.9
	Logistic Regression	31.9
	Ridge Regression	31.4
	LDA	32.3
Meta-classifier Bagging	Linear SVM	32.0
	Logistic Regression	32.0
	Ridge Regression	31.8
	LDA	**32.4**

Table 14
Results for our three models trained on the EFCAMDAT data set and tested on TOEFL11, using continuous classifier outputs. Best result per column in **bold**, best result per row grouping is underlined.

	Feature	Accuracy (%)
Baselines	Random Baseline	11.1
	Majority Class Baseline	11.1
	Single Vector Baseline	32.3
	Current Best (Malmasi and Dras 2015)	33.5
Oracles	Accuracy@2	49.9
	Accuracy@3	59.2
	Oracle	64.9
Ensembles	Plurality Voting	32.7
	Borda Count	32.6
	Mean Probability	<u>33.5</u>
	Median Probability	33.0
Meta-classifier	Linear SVM	42.7
	Logistic Regression	42.7
	Ridge Regression	42.0
	LDA	<u>43.3</u>
Meta-classifier Bagging	Linear SVM	43.1
	Logistic Regression	43.0
	Ridge Regression	42.6
	LDA	**43.7**

The primary point to note is that all results, both intra-corpus and cross-corpus, follow the same pattern as the results on TOEFL11 in Section 7.1.

The Current Best Result baseline (Malmasi and Dras 2015) was an ensemble using mean probability combination, which is reproduced in the Ensembles row; the other ensemble models produce slightly lower scores, as in TOEFL11. Also for TOEFL11, all ensembles are better than the Single Vector baseline (although only marginally in the case of plurality voting).

The meta-classifier again provides a strong improvement over the plain ensembles, smaller in the intra-corpus results (2.3%, comparing the best ensemble against the best meta-classifier in Table 12) and more substantial in the cross-corpus case (3.9% for Table 13 and a much larger 9.8% for Table 14), where this is in the context of lower absolute accuracies.

In all three cases, the ensemble of meta-classifiers yields additional improvement over the single meta-classifier model. This gap is a smaller one than seen between plain ensembles and single meta-classifiers, but is still a consistent one that leads to our best results, which as with TOEFL11 occur with LDA-based meta-classification. Again, the largest improvements are seen in the cross-corpus case: The ultimate difference between the Best Current Result baseline and our best result in the intra-corpus case (Table 12) is 2.8%, compared with 4.0% (Table 13) and 10.2% (Table 14).

Table 15
Results for our three models on the Chinese data sets (one consisting of raw documents and the other of generated documents), using continuous classifier outputs. Best result per column in **bold**, best result per row grouping is underlined.

	Feature	**Accuracy (%)** raw	generated
Baselines	Random Baseline	9.1	9.1
	Majority Class Baseline	16.1	12.9
	Single Vector	44.7	70.6
	Current Best Result	n/a	70.6
Oracles	Oracle	67.7	92.2
	Accuracy@2	57.1	76.9
	Accuracy@3	65.3	84.7
Ensembles	Plurality Voting	43.5	68.5
	Borda Count	43.2	66.4
	Mean Probability	<u>45.4</u>	<u>71.1</u>
	Median Probability	43.4	66.1
Meta-classifier	Linear SVM	49.5	75.4
	Logistic Regression	49.5	74.6
	Ridge Regression	48.3	71.4
	LDA	<u>50.8</u>	<u>75.9</u>
Meta-classifier Bagging	Linear SVM	49.6	75.5
	Logistic Regression	49.5	75.1
	Ridge Regression	48.4	71.4
	LDA	**51.2**	**76.5**

7.3 Experiments on Other Languages

The final set of our experiments looks at another kind of generalizability. The result patterns observed on TOEFL11 have also held in a cross-corpus analysis with EFCAMDAT; we now look at non-English L2s.

The experiments in this section are conducted on the Chinese and Norwegian data sets, described in Section 3. As these data sets do not have a predefined test set like TOEFL11, these experiments were performed using stratified cross-validation, as discussed in Section 4.1. Previous experiments on these corpora have also been conducted using cross-validation only.

We again utilize the top performing models tested in Section 7.1: four ensemble combiners, four bagging-based meta-classifiers, and four ensembles of meta-classifiers. These selected models, and their results, are listed in Tables 15 and 16 along with baselines for both raw and generated data sets for each language. Only a generated data set Current Best baseline is available for each language.[29]

29 Ionescu, Popescu, and Cahill (2016) do carry out experiments on raw documents from the Norwegian ASK corpus, and reach accuracies up to 68%. However, their set-up is rather different, following Pepper (2012). They work with seven L1s, and results are over varying subsets of five L1s. Their string kernel approach improves by around 15% over the LDA baseline of Pepper (2012), which is greater than the improvement we see for our best ensemble model over our SVM baseline, but this is not directly comparable to the numbers in Tables 15 and 16.

Table 16
Results for our three models on the Norwegian data sets (one consisting of raw documents and the other of generated documents), using continuous classifier outputs. Best result per column in **bold**, best result per row grouping is underlined.

	Feature	**Accuracy (%)** raw	generated
Baselines	Random Baseline	10.0	10.0
	Majority Class Baseline	11.5	13.0
	Single Vector Baseline	48.8	78.6
	Current Best Result	n/a	78.6
Oracles	Oracle	82.0	94.5
	Accuracy@2	68.1	89.3
	Accuracy@3	76.7	92.5
Ensembles	Plurality Voting	51.1	75.7
	Borda Count	50.5	76.7
	Mean Probability	<u>51.5</u>	<u>77.9</u>
	Median Probability	51.0	77.3
Meta-classifier	Linear SVM	52.8	78.6
	Logistic Regression	52.5	79.6
	Ridge Regression	51.6	78.6
	LDA	<u>53.3</u>	<u>81.0</u>
Meta-classifier Bagging	Linear SVM	53.1	78.7
	Logistic Regression	52.8	80.1
	Ridge Regression	51.9	78.8
	LDA	**54.2**	**81.8**

Immediately apparent is that the magnitude of results differs between raw and generated data sets, by around 20%: Prediction is easier on the generated set. This may be connected to greater length consistency (so that all documents are of reasonable length)[30] outweighing the extra difficulty of topic and proficiency clues being removed by the homogenization process.

Abstracting away from that and focusing on the effect of ensembles, the patterns are the same across both L2s and raw and generated data sets.

The oracle values are quite high for the generated data sets at over 90%, similar to TOEFL11 (which was listed in Table 7). As with all other results in these tables, the oracles for the raw data sets are much lower. The ensemble model does well, beating the previously reported best result for Chinese and coming close for Norwegian (generated data set). Just like our previous experiments, the mean probability combiner yields the best performance. A little differently from the English L2 data sets, not all ensembles perform better than the Single Vector baseline for Norwegian.

The meta-classifier model achieves a new state of the art for both (generated) data sets, just as it did for TOEFL11 and EFCAMDAT. Also consistent with the previous experiment, LDA achieves the best results for all data sets. Finally, the ensemble of

30 We have observed, working with some TOEFL11 development data, that for much shorter texts accuracy of prediction can drop from a typical 80% level to something close to 30%.

meta-classifiers yields additional improvement over the single meta-classifier model, achieving our best results. All meta-classifier models and all ensembles of meta-classifiers perform better than the Single Vector baseline. In terms of the generated data sets, we achieve best accuracies of 76.5% on the Chinese data and 81.8% on the Norwegian data, both substantial improvements over previous work. These results show that these classification models are applicable to data sets over other L2s besides English. The results followed the same pattern across all data sets, with LDA-based meta-classification yielding top results.

8. Discussion

We have presented the first systematic study of meta-classification techniques for NLI, achieving state-of-the-art accuracy on several large data sets for the task, under both intra-corpus and cross-corpus settings.

We applied many different methods from the armamentarium of machine learning algorithms, and the observed consistency was an important facet of our results. The performance patterns of our models were similar across different data sets, with the same model configurations achieving the best results across different test sets and corpora.

The application of these methods is not limited to cross-validation studies and we have attempted to apply them elsewhere. During the course of developing these methods we evaluated them under test conditions by using them to compete in several shared tasks in different tasks. Although a detailed exposition exceeds the scope of the present work, we briefly mention our results. The ensemble classifier was used to participate in the 2015 Discriminating Similar Language shared task, and was the winning entry among the 10 participating teams. The ensemble was also used to train a system to participate in the Complex Word Identification task at SemEval 2016 (Track 11), with our systems ranking in second and third place. Finally, the meta-classifier ensemble approach described here was the basis of an entry in the 2016 Computational Linguistics and Clinical Psychology (CLPsych) shared task, where it also ranked in first place among 60 systems. We believe that these results, in conjunction with the state-of-the-art NLI performance reported in the present article, highlight the utility of the classification models we described here for various NLP tasks.

Future work can be directed towards answering some of the following questions. *Why does LDA outperform other meta-classification methods?* While we have outlined a possible reason for the unexpectedly good performance of LDA, given its infrequent use in NLP, an in-depth visualization and examination of the trained models—something beyond the scope of this work—may reveal interesting clues about what precise characteristics of the space of outputs of base classifiers make LDA the best of the alternatives.

How does the amount of training data affect meta-classifier performance? This analysis, along with further evaluation of the models' learning curves, could inform us about training data requirements as well as bias-variance and overfitting issues.

Appendix A. Implementation Details and Parameters

Feature Representation. The raw counts for all base classifier features were normalized using TF-IDF weighting. They can also be normalized to unit length with similar results, but the TF-IDF scheme yielded better accuracy. No feature selection method was used.

Bagging. Each bootstrap aggregating model consists of 200 training sets that are randomly sampled with replacement. Prediction is performed by running samples against all sets and selecting the outcome by voting.

The value 200 was chosen experimentally. Bagging models with training sets in the range [2, 5, 10, 20, 50, 100, 200, 300, 500, 1000] were tested under cross validation and we saw no substantial improvement beyond 200 sets. Each training set contained 80% of the samples in the full data.

SVM. For the RBF and linear kernel SVM models, the value of the C parameter was set to 1.0 in all experiments. It would be possible to tune this for each feature type, but this was not done here.

Logistic Regression. For the LR models, the value of the C parameter was set to 1.0 in all experiments. For assessing base classifiers, both L1 and L2 regularization were used. Subsequently, based on these results, only L2 regularization was used.

Perceptron. The learning rate was set to 1.0 and training was performed using 10 epochs/iterations over the data.

Ridge Regression. Singular Value Decomposition of the training data was used to compute the Ridge coefficients. The weights for all classes was set to 1.

Software Used. The various models available in the LIBSVM, LIBLINEAR, and scikit-learn open-source libraries were used in this study. Syntactic features were extracted using the Stanford CoreNLP system. The RASP[31] system (Briscoe, Carroll, and Watson 2006) was used to perform POS tagging using the CLAWS tag set.

Acknowledgments

The ASK corpus (with persistent identifier hdl:11495/DA23-DEB6-9EE5-2) was provided by CLARINO.

We would also like to express our gratitude to Anne Golden, Paul Meurer, Silje Karin Ragnhildstveit, Koenraad De Smedt, and Kari Tenfjord for their help in making the ASK corpus available to us.

References

Aue, Anthony and Michael Gamon. 2005. Customizing sentiment classifiers to new domains: A case study. Technical Report, Microsoft Research. `https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/new_domain_sentiment.pdf`.

Banfield, Robert E., Lawrence O. Hall, Kevin W. Bowyer, and W. Philip Kegelmeyer. 2007. A comparison of decision tree ensemble creation techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):173–180.

Bjerva, Johannes, Gintare Grigonyte, Robert Östling, and Barbara Plank. 2017. Neural networks and spelling features for native language identification. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 235–239.

Blanchard, Daniel, Joel Tetreault, Derrick Higgins, Aoife Cahill, and Martin Chodorow. 2013. TOEFL11: A corpus of non-native English. Technical report ETSRR-13-14, Educational Testing Service.

Breiman, Leo. 1996. Bagging predictors. In *Machine Learning*, 24, 123–140.

Briscoe, Ted, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the COLING/ACL on Interactive Presentation sessions*, COLING-ACL 2006, pages 77–80.

Brooke, Julian and Graeme Hirst. 2011. Native language detection with "cheap"

31 `http://users.sussex.ac.uk/~johnca/rasp/`.

learner corpora. In *Proceedings of the First Learner Corpus Research Conference*, pages 37–47.

Brooke, Julian and Graeme Hirst. 2012a. Measuring interlanguage: Native language identification with L1-influence metrics. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*, pages 779–784, Istanbul.

Brooke, Julian and Graeme Hirst. 2012b. Robust, lexicalized native language identification. In *Proceedings of COLING 2012*, pages 391–408, Mumbai.

Bykh, Serhiy and Detmar Meurers. 2014. Exploring syntactic features for native language identification: A variationist perspective on feature encoding and ensemble optimization. In *Proceedings of the 25th International Conference on Computational Linguistics*, pages 1962–1973, Dublin.

Bykh, Serhiy, Sowmya Vajjala, Julia Krivanek, and Detmar Meurers. 2013. Combining shallow and linguistically motivated features in native language identification. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 197–206, Atlanta, GA.

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.

Cimino, Andrea, Felice Dell'Orletta, Giulia Venturi, and Simonetta Montemagni. 2013. Linguistic profiling based on general–purpose features and native language identification. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 207–215, Atlanta, GA.

Cover, Thomas M. and Peter E. Hart. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

Dietterich, Thomas G. 2000. Ensemble methods in machine learning. *Multiple Classifier Systems*, number 1857 in Lecture Notes in Computer Science. Springer, pages 1–15.

Eldan, Ronen and Ohad Shamir. 2016. The power of depth for feedforward neural networks. *JMLR: Workshop and Conference Proceedings*, 49:1–34.

Faarlund, Jan Terje, Svein Lie, and Kjell Ivar Vannebo. 1997. *Norsk Referansegrammatikk*. Columbia University Press.

Fisher, Ronald A. 1936. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188.

Foody, Giles M. 2004. Thematic map comparison. *Photogrammetric Engineering & Remote Sensing*, 70(5):627–633.

Freund, Yoav and Robert E. Schapire. 1996. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, pages 148–156.

Geertzen, Jeroen, Theodora Alexopoulou, and Anna Korhonen. 2013. Automatic linguistic annotation of large scale L2 databases: The EF-Cambridge Open Language Database (EFCAMDAT). In *Proceedings of the 31st Second Language Research Forum*, pages 240–254.

Genkin, Alexander, David D. Lewis, and David Madigan. 2007. Large-scale Bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304.

Geurts, Pierre, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine Learning*, 63(1):3–42.

Goldberg, Yoav. 2015. A primer on neural network models for natural language processing. *The Computing Research Repository (CoRR)*, abs/.1510.00726.

Goutte, Cyril and Serge Léger. 2017. Exploring Optimal Voting in Native Language Identification. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 367–373.

Goutte, Cyril, Serge Léger, and Marine Carpuat. 2013. Feature space selection and combination for native language identification. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 96–100, Atlanta, GA.

Granger, Sylviane, Estelle Dagneaux, Fanny Meunier, and Magali Paquot. 2009. *International Corpus of Learner English (Version 2)*. Presses Universitaires de Louvain, Louvian-la-Neuve, Belgium.

Guthrie, David, Ben Allison, Wei Liu, Louise Guthrie, and Yorick Wilks. 2006. A closer look at skip-gram modelling. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 1222–1225, Genoa.

Gyawali, Binod, Gabriela Ramirez, and Thamar Solorio. 2013. Native language identification: A simple n-gram based approach. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 224–231, Atlanta, GA.

Hastie, Trevor, Robert Tibshirani, and Jerome H. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Henderson, John, Guido Zarrella, Craig Pfeifer, and John D. Burger. 2013. Discriminating non-native English with 350 words. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 101–110, Atlanta, GA.

Hladka, Barbora, Martin Holub, and Vincent Kriz. 2013. Feature engineering in the NLI shared task 2013: Charles University submission report. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 232–241, Atlanta, GA.

Ho, Tin Kam, Jonathan J. Hull, and Sargur N. Srihari. 1994. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):66–75.

Hsu, Chih Wei, Chih-Chung Chang, Chih-Jen Lin et al. 2003. A practical guide to support vector classification.

Ionescu, Radu Tudor and Marius Popescu. 2017. Can string kernels pass the test of time in native language identification? In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 224–234.

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*, pages 1363–1373, Doha.

Ionescu, Radu Tudor, Marius Popescu, and Aoife Cahill. 2016. String kernels for native language identification: Insights from behind the curtains. *Computational Linguistics*, 42(3):491–525.

Ircing, Pavel, Jan Svec, Zbynek Zajic, Barbora Hladka, and Martin Holub. 2017. Combining textual and speech features in the NLI task using state-of-the-art machine learning techniques. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 198–209.

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, Atlanta, GA.

Jarvis, Scott, Gabriela Castaneda-Jiménez, and Rasmus Nielsen. 2004. Investigating L1 lexical transfer through learners' wordprints. Paper presented at Second Language Research Forum (SLRF).

Jarvis, Scott and Scott Crossley, editors. 2012. *Approaching Language Transfer Through Text Classification: Explorations in the Detection-based Approach*. Multilingual Matters, Bristol, UK.

Joachims, Thorsten. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142.

Kittler, Josef, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas. 1998. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239.

Kohavi, Ron. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence*, pages 1137–1145.

Koppel, Moshe, Jonathan Schler, and Shlomo Argamon. 2009. Computational methods in authorship attribution. *Journal of the American Society for Information Science and Technology*, 60(1):9–26.

Koppel, Moshe, Jonathan Schler, and Kfir Zigdon. 2005a. Automatically determining an anonymous author's native language. In *Intelligence and Security Informatics*, volume 3495 of Lecture Notes in Computer Science. Springer-Verlag, pages 209–217.

Koppel, Moshe, Jonathan Schler, and Kfir Zigdon. 2005b. 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*, pages 624–628, Chicago, IL.

Kulmizev, Artur, Bo Blankers, Johannes Bjerva, Malvina Nissim, Gertjan van Noord, Barbara Plank, and Martijn Wieling. 2017. The power of character *n*-grams in native language identification. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 382–389.

Kuncheva, Ludmila I. 2002. A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):281–286.

Kuncheva, Ludmila I. 2004. *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley & Sons.

Kuncheva, Ludmila I. 2014. *Combining Pattern Classifiers: Methods and Algorithms*, second edition. Wiley.

Kuncheva, Ludmila I., James C. Bezdek, and Robert P. W. Duin. 2001. Decision templates for multiple classifier fusion: An experimental comparison. *Pattern Recognition*, 34(2):299–314.

Kuncheva, Ludmila I. and Juan J. Rodríguez. 2014. A weighted voting framework for classifiers ensembles. *Knowledge and Information Systems*, 38(2):259–275.

Kuncheva, Ludmila I., Christopher J. Whitaker, Catherine A. Shipp, and Robert P. W. Duin. 2003. Limits on the majority vote accuracy in classifier fusion. *Pattern Analysis & Applications*, 6(1):22–31.

Li, Wen and Liang Zou. 2017. Classifier stacking for native language Identification. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 390–397.

Liu, Chengjun and Harry Wechsler. 2002. Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition. *IEEE Transactions on Image Processing*, 11(4):467–476.

Malmasi, Shervin and Aoife Cahill. 2015. Measuring feature diversity in native language identification. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 49–55, Denver, CO.

Malmasi, Shervin and Mark Dras. 2014a. Arabic native language identification. In *Proceedings of the Arabic Natural Language Processing Workshop*, pages 180–186, Doha.

Malmasi, Shervin and Mark Dras. 2014b. Chinese native language identification. *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 95–99.

Malmasi, Shervin and Mark Dras. 2014c. Finnish native language identification. In *Australasian Language Technology Association Workshop 2014*, pages 139–144.

Malmasi, Shervin and Mark Dras. 2015. Large-scale native language identification with cross-corpus evaluation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1403–1409, Denver, CO.

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. *CoRR*, abs/1703.06541.

Malmasi, Shervin, Joel Tetreault, and Mark Dras. 2015. Oracle and human baselines for native language identification. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 172–178, Denver, CO.

Malmasi, Shervin, Sze-Meng Jojo Wong, and Mark Dras. 2013. NLI shared task 2013: MQ submission. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 124–133, Atlanta, GA.

Malmasi, Shervin, Marcos Zampieri, Nikola Ljubešić, Preslav Nakov, Ahmed Ali, Liling Tan, and Jörg Tiedemann. 2016. Discriminating between similar languages and Arabic dialect identification: A report on the third DSL shared task. In *Proceedings of the Joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects*, pages 1–14, Osaka.

Manning, Christopher D. 2015. Computational Linguistics and Deep Learning. *Computational Linguistics*, 41(4):701–707.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. 2008. Evaluation in information retrieval. In *Introduction to Information Retrieval*, Cambridge University Press, pages 151–175.

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Markov, Ilia, Lingzhen Chen, Carlo Strapparava, and Grigori Sidorov. 2017. CIC-FBK approach to native language identification. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 374–381.

de Marneffe, Marie Catherine, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 449–454, Genoa.

McNemar, Quinn. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Oh, Yoo Rhee, Hyung-Bae Jeon, Hwa Jeon Song, Yun-Kyung Lee, Jeon-Gue Park, and Yun-Keun Lee. 2017. A deep-learning based native-language classification by using a latent semantic analysis for the NLI shared task 2017. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 413–422.

Ortega, Lourdes. 2009. *Understanding Second Language Acquisition*. Hodder Education, Oxford, UK.

Oza, Nikunj C. and Kagan Tumer. 2008. Classifier ensembles: Select real-world applications. *Information Fusion*, 9(1):4–20.

Pepper, Steve. 2012. Lexical transfer in Norwegian interlanguage: A detection-based approach. Master's thesis, University of Oslo.

Platt, John. 2000. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors. *Advances in Large Margin Classifiers*, MIT Press, pages 61–74.

Polikar, Robi. 2006. Ensemble based systems in decision making. *Circuits and Systems Magazine, IEEE*, 6(3):21–45.

Popescu, Marius and Radu Tudor Ionescu. 2013. The story of the characters, the DNA and the native language. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 270–278, Atlanta, GA.

Quinlan, J Ross. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc.

Rama, Taraka and Çağrı Çöltekin. 2017. Fewer features perform well at native language identification task. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 255–260.

Rosenblatt, Frank. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

Sari, Yunita, Muhammad Rifqi Fatchurrahman, and Meisyarah Dwiastuti. 2017. A shallow neural network for native language identification with character n-grams. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 249–254.

Swanson, Benjamin and Eugene Charniak. 2012. Native language detection with tree substitution grammars. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 193–197, Jeju Island.

Tenfjord, Kari, Hilde Johansen, and Jon Erik Hagen. 2006. The "hows" and the "whys" of coding categories in a learner corpus (or "How and why an error-tagged learner corpus is not ipso facto one big comparative fallacy"). *Rivista di psicolinguistica applicata*, 6(3):1000–1016.

Tenfjord, Kari, Paul Meurer, and Knut Hofland. 2006. The ASK corpus: A language learner corpus of Norwegian as a second language. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 1821–1824.

Tenfjord, Kari, Paul Meurer, and Silje Ragnhildstveit. 2013. Norsk andrespråkskorpus — A corpus of Norwegian as a second language. In *Learner Corpus Research Conference*.

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, Atlanta, GA.

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 the International Conference on Computational Linguistics*, pages 2585–2602, Mumbai.

Tibshirani, Robert, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu. 2002. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences*, 99(10):6567–6572.

Tomokiyo, Laura Mayfield and Rosie Jones. 2001. You're not from round here, are you? Naive Bayes detection of non-native utterance text. In *Proceedings of the Second North American Chapter of the*

Association for Computational Linguistics, pages 239–246.

Wang, Maolin, Shervin Malmasi, and Mingxuan Huang. 2015. The Jinan Chinese learner corpus. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 118–123, Denver, CO.

West, David. 2000. Neural network credit scoring models. *Computers & Operations Research*, 27(11):1131–1152.

Wolpert, David H. 1992. Stacked generalization. *Neural Networks*, 5(2):241–259.

Wong, Sze Meng Jojo and Mark Dras. 2009. Contrastive analysis and native language identification. In *Proceedings of the Australasian Language Technology Association Workshop*, pages 53–61, Sydney.

Wong, Sze Meng Jojo, Mark Dras, and Mark Johnson. 2012. Exploring adaptor grammars for native language identification. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 699–709, Jeju Island.

Woźniak, Michał, Manuel Graña, and Emilio Corchado. 2014. A survey of multiple classifier systems as hybrid systems. *Information Fusion*, 16:3–17.

Zhang, Tong and Frank J. Oles. 2001. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4(1):5–31.

Zhu, Ji and Trevor Hastie. 2004. Classification of gene microarrays by penalized logistic regression. *Biostatistics*, 5(3):427–443.

This article has been cited by:

1. Shervin Malmasi, Iria del Río, Marcos Zampieri. Portuguese Native Language Identification 115-124. [Crossref]